MES-CoBraD

Multidisciplinary Expert System
for the Assessment & Management
of Complex Brain Disorders

D5.1

Data integration, harmonisation, sharing and security – v1

March 2022

The MES-CoBraD project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under grant agreement No 965422

www.mes-cobrad.eu

PREFACE

The Multidisciplinary Expert System for the Assessment & Management of Complex Brain Disorders (MES-CoBraD) is an interdisciplinary project combining Real-World Data (RWD) from multiple clinical and consumer sources through comprehensive, cost-efficient, and fast protocols towards improving diagnostic accuracy and therapeutic outcomes in people with Complex Brain Disorders (CoBraD), as reflected in Neurocognitive (Dementia), Sleep, and Seizure (Epilepsy) disorders and their interdependence.

1. It brings together internationally recognized experts in medicine, engineering, computer science, social health science, law, and marketing and communication from across Europe, and combines clinical information and scientific research in CoBraD with technical innovation in secure data-sharing platforms, artificial intelligence algorithms, and expert systems of precision and personalized care, with a primary focus on improving the quality of life of patients, their caregivers, and the society at large.

2. It leverages RWD from diverse CoBraD populations across cultural, socioeconomic, educational, and health system backgrounds, with special attention on including vulnerable populations and minorities in an equitable manner and engaging key stakeholders to maximize project impact.

3. The project will deliver a rigorous and self-standing methodology that will drive the MES-CoBraD implementation and define its operational principles; The MES-CoBraD solution, through the implementation of novel, self-standing AI based components and their integration under a common platform for scientific exploitation will assist focusing on the evaluation and validation of the solution, the spread of the excellence gained, the expansion of its ecosystem and the real-life sustainability.

CONSORTIUM

NTUA	National Technical University of Athens	EL
NIA	Neurological Institute of Athens	EL
HSCSP	Fundació privada Institut de Recerca de l'Hospital de la Santa Creu i Sant Pau	IR
VUB - LSTS	VRIJE UNIVERSITEIT BRUSSEL	BE
UU	UPPSALA UNIVERSITY	SE
CHS-RMC	Clalit Health Services- Rabin Medical Center Cognitive Neurology and Epilepsy Clinics	IL
KCL	King's College London	UK
HOLISTIC	HOLISTIC P.C.	EL
SIMAVI	Software Imagination & Vision	RO
LIBER	STICHTING LIBER	NL
ENG	Engineering - Ingegneria Informatica S.p.A.	EN
EVOLUTION	MICHOPOULOS I. & CH. G.P.	EL
UE	University of Edinburgh	UK
CEL	CyberEthics Lab	IT

The MES-CoBraD project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under grant agreement No 965422

MULTIDISCIPLINARY EXPERT SYSTEM FOR THE ASSESSMENT & MANAGEMENT OF COMPLEX BRAIN DISORDERS

GA#: 965422 **Start Date:** 01/04/2021
Topic: SC1-DTH-12-2020 **Duration:** 36 months
Type of Action: RIA **Coordinator:** NTUA

Deliverable Number	D5.1
Deliverable Title	Data integration, harmonisation, sharing and security – v1
Work Package Number	WP5
Task Number	T5.1, T5.2, T5.3
Date of Delivery	31/03/2022
Dissemination Level	Confidential
Work Package Leader	ENG
Lead Beneficiary	ENG
Authors	ENG
Contributors	NIA, VUB
Reviewers	NTUA, SIMAVI

Disclaimer

Copyright Message

Document History

Version	Date	Author (Partner)	Remarks/Changes
0.10	28/01/2022	Antonino Sirchia (ENG)	ToC
0.11	31/01/2022	Christos Ntanos (NTUA)	ToC Review
0.20	08/03/2022	Antonino Sirchia (ENG), Danilo Trombino (ENG), Igor Molnar (ENG)	Partner contributions
0.31	15/03/2022	George Doukas (NTUA)	Peer Review
0.32	15/03/2022	Andrei Ogrezeanu (SIMAVI)	Peer Review
0.40	22/03/2022	Antonino Sirchia (ENG)	Review feedbacks resolution
0.50	23/03/2022	Michael Kontoulis (NTUA)	Quality Control
1.00	30/03/2022	Christos Ntanos (NTUA)	Final Version to be Submitted

The MES-CoBraD project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under grant agreement No 965422

Executive Summary

This deliverable (D5.1) is a technical document that has the purpose of presenting the design and the implementation of the MES-CoBraD platform components intended for the data management.

The document starts with an analysis of the data management in the healthcare and medical research landscape, then continues with a comparison among several already available IT solutions that could cover the requirements necessary for the purposes of the MES-CoBraD project.

The platform components discussed in this document are:

- Edge module
- Anonymisation module
- Data collection component
- Data Lake
- Data harmonisation component

For each of the up-mentioned tools, the entire design process is described across the document, starting from the requirements analysis, the planned API specifications and even the intended UI mock-ups. Once described in detail the desired design, the technological implementation is analysed along with the integration among the components.

Finally, a plan for the User acceptance tests for each component is presented.

The document is intended as reference for all the stakeholders, for what concern the technical specification of the components belonging to the data management layer of the MES-CoBraD platform

The MES-CoBraD project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under grant agreement No 965422

Table of Contents

The MES-CoBraD project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under grant agreement No 965422

The MES-CoBraD project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under grant agreement No 965422

The MES-CoBraD project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under grant agreement No 965422

LIST OF FIGURES

The MES-CoBraD project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under grant agreement No 965422

LIST OF TABLES

The MES-CoBraD project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under grant agreement No 965422

LIST OF ABBREVIATIONS

Abbreviation / Acronym	Description
Dx.y	Deliverable number y belonging to WP x
Tx.y	Task number y belonging to WP x
GA	General Assembly
ToC	Table of Contents
WP	Work Package
MES-CoBraD	Multidisciplinary Expert System for the Assessment & Management of Complex Brain Disorders
DoA	Description of Action
WP	Work Package
WPL	Work Package Leader
UAT	User Acceptance Test
DE	Digital Enabler
DME	Data Mashup Editor
REST	Representational State Transfer
API	Application Programming Interface

The MES-CoBraD project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under grant agreement No 965422

1 INTRODUCTION

1.1 PURPOSE AND SCOPE

The MES-CoBraD project has the goal of creating a clinical research platform that relies on Real World Data. The platform planned data lifecycle is composed by several steps: a non-exhaustive list encompasses the collection, the integration, the harmonisation and the sharing of them. Since most of such data is patients' clinical information, it definitely falls into the category of personal data and, as such, must be managed through the top-of-range security patterns and practices, following a *security by design* approach.

The purpose of this deliverable is to present and discuss the data management strategy within the MES-CoBraD platform. The requirements that have been defined for each of the up-mentioned steps and how they are implemented.

1.2 STRUCTURE OF THE DELIVERABLE

This document, through the various chapters, will incrementally describe the rationale that steered the implementation of the data management components.

The Chapter 2 presents a data management landscape analysis with a particular focus on the data management in healthcare and in medical research. This chapter has the purpose of presenting the domain the platform components will work over. Then, the Chapter 3 presents the technological state-of-the-art for the high-level requirements of the platform.

From the Chapter 4 on, each of the planned technological components has a dedicated sub-paragraph for a detailed analysis from the specific point of view of each chapter; the list of components discussed in this document are:

- Edge module (also called "Local back end")
- De-personalisation module
- Data Collection Component
- Data Lake
- Data harmonisation component
- API Gateway

In particular, the Chapter 4 presents the elaboration of the requirements, originally listed in the *D7.1 – System requirements and architecture*, along with some new requirement described here in order to deep dive a little bit in more details for all the components.

Once clarified which are the requirements for each tool, the Chapter 5 presents the designed architecture; it presents the overall architecture – in order to describe how all the components interact with each-other – and also with the internal architecture of each single tool.

Since the components present both a UI and (all of them) at least one API, Chapters 6 and 7 are intended to discuss these aspects. Chapter 6 is about the indented user experience: it presents some mock-ups, to give a clearer idea about the functionality of the tools and, ultimately, of the platform; Chapter 7, instead is about the APIs and their specifications.

Chapter 8 describes the specific implementation of each component, putting together all the requirements and specifications discussed until then. The chosen technologies and the rationale behind them are presented in a fine-grained way.

The MES-CoBraD project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under grant agreement No 965422

How the tools interact with each other, from a technological point of view, is the subject of the discussion of Chapter 9. The chapter is divided into two sections: the first one that investigates the interaction between the data management components, and the second that presents the integration with other platform components, not related to data management, like data analytics and data visualisation.

Finally, Chapter 10 presents a first list of user acceptance tests for the components.

1.3 RELATION TO OTHER DELIVERABLES/WPS/TASKS

This document presents the following relations with other MES-CoBraD deliverables:

Work package 1:

- **D1.1 – Project Handbook**: the project handbook presents the methodological guidelines that the contributors respected during the writing of the current document. In particular, the template used for the document has been defined Annex I of the handbook along with the quality assurance plan that the current document will follow before being submitted to the EC.
- **D1.2 - Scientific, Technical and Innovation Roadmap**: the roadmaps described in D1.2 are the guidelines followed in the definition of the contents of this document. The goals (both scientific, societal, and technological) were always kept in mind while describing the component requirements and even the challenges have been analysed while defining the architecture and the implementations.

Work package 2:

- **D2.3 – Ethical and regulatory framework**: the ethical aspects, along with the compliance ones, are key point across the entire project execution. D2.3 and this document have several cross references, especially in the sections related to the data collection, de-personalisation and persistence.

Work package 6:

- **D6.1 – MES-CoBraD Advanced Analytics Prototype**: the Advanced Analytics Prototype is one of the consumers of the data collected on the platform. The data sharing functionality is the most relevant one for this integration; so, all the sections related to the Data Lake and the API Gateway are particularly relevant for Advanced Analytics purposes. Finally, the integration between the two parts is described in Chapter 9.

Work package 7:

- **D7.1 – System requirements and architecture**: the requirements and the overall architecture, that tracked the roadmap for the entire project – and, consequently, also for the components described in this document – are described in D7.1 that is, then, frequently referenced throughout the current document.

Work package 8:

- **D8.2 – Validation framework**: the software produced on top of what is written in this document will be validated according to the framework described in the D8.2. In particular, Chapter 10 is about the testing of the components and picks up what has been presented and discussed in D8.2; also, the template used for the definition of the UAT is described in the Annex I of the same deliverable.

The MES-CoBraD project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under grant agreement No 965422

2 DATA MANAGEMENT LANDSCAPE ANALYSIS

In the recent years the healthcare management around the world has changed from disease-centred to a patient-centred model. The outdated data management systems are not powerful enough to analyse big data as variety and volume of data sources have increased in the past two decades [12]. Such "Big Data" are generated from clinical and research historical activities and has significant effects on the medical industry. For instance, the processing and analysis of such big data can assist in planning treatment paths for patients, give support to clinical decisions, and improve healthcare technologies and systems [12]. Big data in healthcare involves collecting large collections of data from various healthcare data sources followed by storing, managing, analysing, visualizing, and delivering information for effective decision making. Such volume of data responds to the "Big Data Six Vs" [12].

1. ***Volume*** – Today's healthcare data are in the order of exabytes and are constantly growing.
2. ***Variety*** – Data comes in different data formats and protocols from different data sources.
3. ***Velocity*** – Wearables and sensors generate data with high frequency. Indeed, due to the rising of number of patients data rate is increasing 55-60% every year.
4. ***Veracity*** – Big data can never be 100% accurate and it is difficult to validate them.
5. ***Variability*** – Data usually fluctuates during their lifecycle; this has impact to patterns and hidden characteristics.
6. ***Value*** – Extracting value from such big data is the main goal of data analytics.

Modern data management platforms need to handle, validate, and maintain scattered data that comes from heterogenous data silos. Some of them are:

- Electronic Health Records (EHRs) – Within such a platform, data comes from physicians' notes, lab reports, ECGs, Scans, X-RAYs, health sensor devices, medical prescriptions etc. They represent the core of the patient's medical data, including any basic demographic information, like name, profession, addresses, age, gender etc.
- Laboratory results – Laboratory tests include many biochemical tests, such haematological tests. Their results can help in monitoring the progress of a disease or evaluating new treatments.
- Medical sensors – Medical devices that measure specific patient's bio-signals.
- Wearables – Such IoT devices are usually low-cost, and their aim is continuous monitoring of a patient's health condition for home rehabilitation purposes, as well as the secure transfer of individual-related health information directly to a central data repository for clinical evaluation and further analysis.
- Historical clinical activities – Historical data is often used as a reference to start new clinical trials, analyse the evolution of a disease and as a reference for side effects and treatments.
- Patient Registries (PRs) – Research-oriented collections of health information from patients for clinical or scientific purposes in the form of an observational study
- Cohort studies – A clinical cohort study comprises data from group of people who share common disease occurrences and medical conditions and are useful for measuring the disease occurrence and progress.
- Open data – Health open data portals are important as starting points for research studies and for developing new treatments and cures. Moreover, metadata of clinical studies are publicly available and can be used as additional data sources.
- Social media – Data coming from social media like Facebook, Instagram and Twitter is usually

The MES-CoBraD project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under grant agreement No 965422

used for analysing disease spreading/ transmission and to predict future illness.

Data coming from such heterogenous data sources is often difficult to process due to the variety of its format. Moreover, some of those datasets, such as doctors' notes and handwritten medication prescriptions, need to be digitalised and imported to medical data platforms before the harmonisation processes. Others, like bio-images, bio-signals or cohort studies results must be treated with advanced tools like statistical models, mathematical operations, or AI-algorithms.

Such preliminary operations are some of the pre-requisites for the data harmonisation phase.

It is possible to identify a sequence of steps that are applied in health data harmonisation. Such steps can be summarised as follow:

1. Data curation
2. Metadata extraction
3. Data model identification
4. Data integration

Data curation refers to the techniques applied to enhance the overall data quality. Such techniques can be categorised into the fields of data normalisation, data standardisation, missing data imputation and outlier treatments. Formally, missing data imputation is only part of the medical research processes, since it's common to have missing values in cohort studies or clinical trials. Moreover, also outlier detection is a practice mainly used in medical research, where researchers decide if and how to handle such values or collect them to conduct specific studies.

The implementation of a standardized metadata template for medical data can make their content fairer (dealing with heterogeneous and vendor-dependent metadata templates brings complexity at computational level and make the metadata difficult to understand and analyse), expand their audience, and extend their applications. Indexing such data transforms otherwise isolated observations into interpretable patterns that can be used to identify new diseases, treatments or other fields of research.

Usually, Heterogeneous data is difficult to process and analyse. In such situations data models can provide high quality, consistent and structured data for running business processes and to allow scientists and researchers to explore data widely. In this phase data harmonisation involves several mechanisms including dataset transformation to a common format, dataset description, similarity detection, terminology detection, and alignment. The mapping between the field coming from raw data and the ones that are part of the data model (also called field matching) is usually made using two different approaches: semantic matching and the lexical matching.

Data integration can reveal valuable clinical information regarding a disease's onset and progress over time. Indeed, the integration of harmonised data from multiple data sources can yield more powerful patient stratification models which are able to precisely identify groups of people that are more prone to the development of a disease outcome.

Once harmonised and integrated data must satisfy several medical standards to ensure interoperability between health systems [19]. Some of them are:

- Health Level 7 (HL7)
- Continuity of Care Record (CCR)
- Continuity of Care Document (CCD)
- Controlled Medical Vocabulary (CMV)
- Computerized Provider Order Entry (CPOE)

Such sensitive information needs to be extracted, processed, and analysed using a set of policies, rules and processes that define the Data Governance (GC) of those health data. One of the most important

The MES-CoBraD project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under grant agreement No 965422

aspects of the GC is the anonymisation and pseudo-anonymisation. The anonymisation process involves the complete removal of any information that can lead to the identification of the individual, whereas pseudonymization involves the partial removal of the individual data with an additional storage of information that can indirectly lead to the identification of the individual. These steps ensure to have systems compliant with the different data protection laws such as Health Insurance Portability and Accountability Act (HIPAA) and General Data Protection Regulation (GDPR).

Healthcare and medical research share almost all the up mentioned data management practices. They differ in the goals and in how such data is analysed and used.

2.1 In Healthcare

Data management in healthcare puts the patient as the main actor of all the processes, decisions and initiatives that are taken for medical purposes. This patient-centric approach uses the outputs coming from medical research as source of knowledge to pre-identify diseases, give better medications and pursuing prevention initiatives. Moreover, experts' opinions provide guidelines based on research and their own experience to help patients with severe issues and situations that are difficult to manage. Indeed, the available information is used to apply the evidence-based medicine principles (EBM). The practice of EBM involves five essential steps [20]:

1. converting information needs into answerable questions
2. finding the best evidence with which to answer the questions
3. critically appraising the evidence for its validity and usefulness
4. applying the results of the appraisal into clinical practice
5. evaluating performance

In healthcare patients' condition can change rapidly depending on the severity of the disease, it's nature and the medical treatments provided to the patients. Doctors use the adaptive reasoning to iteratively 'adapt' their decision based on cause-effect reactions triggered by new medications, patient's conditions change etc. The whole process generates multi-dimensional information that are used to answer specific questions or hypothesis.

2.2 In Medical Research

Unlike healthcare, the aim in medical research is not to treat a specific patient, instead it is to discover new information, processes or mechanism that can bring new solutions to patients' problems. The data strategies' goal in such scenario is to discover what is true, by confirming or rejecting hypotheses. Such hypotheses, usually first being theorised, are then tested by developing experiments, where data is collected through trials and cohort studies conducted on a part of the population selected by using specific sampling methods. The process generates a high-volume scattered data that needs to be managed as described in the previous section. Once harmonised, such data is analysed using statistical tools like correlations, regression models and classification algorithms, summarise the results and bring new value to the health community.

The MES-CoBraD project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under grant agreement No 965422

3 STATE-OF-THE-ART ANALYSIS

With the main interest being in processing medical data there are a lot of tools to consider. One of the first requirements is to have a large database where all of the data can be stored and most suitable for this purpose are Data Lakes.

After securely storing the data, it should be properly formatted for further usage. This requires a data harmonisation tool.

All this must be done while taking in consideration the security of the data, as well as the need to share curated data and make it available for data analytics.

In this chapter we will identify, analyse, and compare the theory and approaches behind these tools.

3.1 DATA LAKE – COMPARATIVE ANALYSIS

A **Data Lake** is best described as a highly scalable environment, rather than a giant collection of data, which may contain structured, unstructured or semi-structured data from various sources organised in a logically centralised and well architected unit.

The scalability of the data lake comes from the fact that it is almost always based on a cloud solution, which also offers the benefit of the pay-as-you-go prices.

The ability to store heterogenous types of data is supported by a different approach as opposed to data warehouses, where data is inserted using the ETL (Extract, Transform, Load) technique. This means that when using data warehouses, data needs to be transformed upon the insert phase, which requires a specific organisational structure of the data warehouse itself. Instead, data lakes rely on the ELT (Extract, Load, Transform) technique, which means that the data is inserted in its raw form and transformed only upon being used.

Data lake storage can be divided into four zones:

- Raw data zone – data extracted from various sources without modification
- Cleansed data zone – data that went through a process of treatment and harmonization
- Data model zone – data grouped and restructured to be used in long-term analytics
- Experimental zone – data for short-term analytics from any of the other zones

3.1.1 TOOLS FOR DATA LAKES

It is important to remember that, although logically centralised, Data Lakes are physically distributed among numerous servers which serve as data storage. The tools used for this kind of data management are based on two different approaches:

1. Distributed File Systems (Hadoop)
2. Object Storage Based Systems (Amazon S3, MinIO, Azure Data Lake Storage, Google Cloud Storage)

3.1.1.1 Distributed File Systems

A **Distributed File System** is a file system that uses servers to store data which could be accessed and transformed in the same way as if it were present on a local client machine.

The best-known example of a distributed file system is the **Hadoop Distributed File System** (HDFS). It is highly fault-tolerant and is designed to be deployed on low-cost hardware while providing high throughput access to application data and being suitable for applications that have large data sets [1].

HDFS allows simultaneous processing of data which is broken into segments and distributed through

The MES-CoBraD project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under grant agreement No 965422

different nodes in a cluster.

Distributed file systems offer high scalability at a low cost and can work with various data types with a high fault tolerance. On the other hand, they face various security problems which must be alleviated with additional tools and due to the fact that they work with large size data, it is hard to provide results with low latency and without processing overhead.

Benefits:

- has good file availability and access time
- can easily change the size of data
- enables multiple users to access date at once

Limitations:

- hard to maintain the database
- possible problems with the transfer of data between nodes

Object Storage Based Systems

An **Object Storage System** is one which manages data as objects, instead of using a file hierarchy. These objects act as data containers and may include other information such as metadata or identifiers. There are several implementations of the object storage system, but the most notable are **Amazon Simple Storage Service** (S3), **Microsoft Azure Data Lake Storage** (ADLS), **Google Cloud Storage** (GCS) and **MinIO**.

As with distributed file systems, object storage systems also provide great scalability at a low cost.

Additionally, as opposed to file systems which use fixed metadata, object storage systems use custom metadata, which enables them to leverage better indexing capabilities, to easily move data between storage tiers and to centralise data between multiple data clusters.

Benefits:

- simple API adaptable for all programming languages
- highly scalable
- low cost
- data loss resistant
- easily integrated with IoT devices

Limitations:

- not recommended for backing up traditional databases
- not ideal for very frequent data changes

3.2 Data Harmonisation – Comparative Analysis

Usage of data lakes consequently leads to a heap of raw data that may or may not be useful. In order to extract useful data from the data lake, a type of data harmonization must be used.

Data harmonisation encompasses all methods that gather data from multiple diverse sources and as a result provide a cohesive dataset.

Harmonisation involves several mechanisms including dataset transformation to a common format, dataset description, similarity detection, terminology detection, and alignment [2].

The MES-CoBraD project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under grant agreement No 965422

3.2.1 TOOLS FOR DATA HARMONISATION

In cases where it is necessary to process huge amounts of data, it is impossible to accomplish data harmonization manually. The solution is to use already existing automated workflows and processors. There are a couple of different approaches to this solution:

1. Workflow management
2. Batch data processing
3. Stream data processing

3.2.1.1 Workflow management

To completely leverage the power of automation using data harmonization and to establish that data is not corrupted or incomplete, it is of the essence to ensure that every scheduled task is successful.

This can be accomplished by using **Workflow management**. Workflows can be set up for a specific task or process and they can be automated in a way that one task can depend on another, but also to run a task only if the previous task has been successful or if all prerequired tasks have been finished.

An example of this approach can be found in tools like **Airflow**, **Luigi**, **Pinball** or **Chronos**.

Airflow: a platform that can programmatically create, schedule and manage workflows. The management of workflows is done using Directed Acyclic Graphs (DAGs). Each DAG can be viewed as a group of tasks to run, which is organized in a pipeline. Airflow is able to manage data pipelines from multiple sources and export the result data for later consumption by machine learning models or creating visualizations. The tasks it performs can be scheduled and usually depend one on another in order to develop fine-tuned data models.

Benefits:

- completely free to use
- has a growing community with more than 1000 contributors
- has a great extensibility due to the fact it can use different custom plugins that can cover almost any use case. It is also constantly extended by the community
- since it is written in Python, anything that can be done in Python can be done in Airflow too

Limitations:

- requires extensive knowledge of Python

Luigi: a python package for building complex data pipelines. Its main goal is to tackle long-running batch processes by orchestrating a group of tasks. Each task has a target because of computation and oversees computing and consuming targets made by other tasks.

Benefits:

- good Hadoop support
- open-source
- modular code (easy to update)
- python-based

Limitations

- user-interface hard to navigate
- scalability issues

The MES-CoBraD project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under grant agreement No 965422

Pinball: created by Pinterest to address their workflow management needs. It is based on a master-worker paradigm. The workers are in charge of periodically contacting master in order to perform allocated tasks.

Benefits:

- python-based
- ability to retry failed jobs, abort workflows or resume execution
- dynamic resizing (workers can be added or removed)

Limitations:

- runs on python 2
- challenging user interface

3.2.1.2 Batch data processing

To process a very large amount of data at once in a repetitive manner, it is common to use an approach called **Batch data processing**. It is completely automated and can be scheduled as well.

However, one of its drawbacks in case of an error, is the needed effort to debug the process, which usually can be overseen only by an IT expert [3].

Most notable tool that uses this approach is **Talend**.

Benefits:

- can work fast with large amounts of data
- easy to setup environment

Limitations:

- not adequate for integration with machine learning processes
- hard to debug

3.2.1.3 Stream data processing

Contrary to batch data processing, **Stream data processing** can work with continuous data and produce results almost instantly, which makes it the most appropriate choice for analysing real-time data. This kind of approach is most efficient in cases where events emerge in short time frames and it is necessary to react as soon as the results are visible (e.g., cybersecurity).

The main challenge it faces is the amount of data that should be stored and handled, as well as the speed of the data processing, which must be at least as fast as the data input [3].

One of the tools that uses stream data processing is **NiFi**.

Benefits:

- supports clustering
- able to fetch data from remote machines via SFTP
- has security policies on user level
- able to use custom plugins

Limitations:

- not robust when handling data

The MES-CoBraD project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under grant agreement No 965422

3.3 DATA SECURITY – COMPARATIVE ANALYSIS

When medical data is being processed, data security is of utmost importance since personal data can be exposed. By personal data, it is referred to any type of physical information which could be used to unequivocally identify a specific individual (name, date of birth etc.). Another often used term is personal sensitive data, which refers to any type of physical information which might be used to directly identify an individual (credit card number, phone number etc.).

Other than following all the known regulations, there are numerous technical challenges to secure this kind of data which include effective de-identification mechanisms, reducing needed information or use of the encrypted communication mechanisms to collect data [2].

3.3.1 TOOLS FOR DATA SECURITY

Implementing data security in healthcare is a tedious issue. However, during the years a few methods have emerged as useful to protect sensitive information from data breaches [4]. In summary they are:

1. Authentication and Authorisation
2. Access control
3. Encryption
4. Data masking

3.3.1.1 Authentication and Authorisation

Authentication, or direct identification of the user trying to access a resource, must not be confused with authorisation, or decision whether the user who tried to access the resource has a permission to perform a specific action. By resource, it is referred to any data that could be accessed from the web.

To authenticate a user, an application must confirm its identity. This includes the registration of a user, which in turn enables them to access the application by signing in. Once signed in, the user creates a session, which lasts while user is signed in and expires after user signs out or after a certain amount of time. Then, the user must sign in again in order to access the application.

However, even when the users are signed in, they are not able to access every part of the application, as they must first be authorised to do so. Authorisation of users is usually done through their user accounts. Depending on the type and structure of resources they want to access to, users can be identified as resource owners. An owner of a resource may edit or delete their resource, but is not authorised to edit or delete a resource that does not belong to them. These restrictions are most often implemented using roles or policies.

Suffice to say, today there are many tools that handle authentication and authorisation and there is no need for any kind of development on that field. They are called Identity Access Managers (IAM). However, it is necessary to choose the tool which most adequately suits the needs of an organization.

Most popular identity access managers include:

- Cloud services IAM (e.g. AWS IAM, Azure AD, Google IAM)
- WSO2 Identity Server
- Auth0
- Keycloak

3.3.1.2 Cloud services IAM

Although there are different implementations, all cloud identity managers rely on more or less the same principles. They all have a single access point interface for all cloud platforms where it is possible to handle users, roles, policies and permissions. Also, they all support the Single Sign-On (SSO) mechanism

The MES-CoBraD project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under grant agreement No 965422

and Multifactor Authentication (MFA). Examples of cloud services IAM are **AWS IAM**, **Azure AD** and **Google IAM**.

Benefits:

- pre-integrated with the rest of the platform services
- high level of reliability and scalability
- easy to patch and upgrade

Limitations:

- optimised for specific platforms

3.3.1.3 WSO2 Identity Server

Besides cloud identity managers, there are others that are very popular in the community. One of them is **WSO2 Identity server**, an open-source identity manager based on open standards such as SAML, OIDC and OAuth. WSO2 is perfect for providing identity management for enterprise web applications, services or APIs. It is easily customizable and extendable and has a useful management console.

Benefits:

- uses role-based access control (RBAC)
- supports SSO and MFA
- identity federation

Limitations:

- complex integration

3.3.1.4 Auth0

Auth0 is another tool that offers identity management with high customization and is easy to implement. As other tools, it provides a good SSO and MFA support. What makes it different is its centralised authentication server, which immediately notifies users if their credentials have been compromised and prevents them from logging in until they change their password.

Benefits:

- easy to implement and use
- centralised authentication server

Limitations:

- inadequate search feature
- not good for multitenancy

3.3.1.5 Keycloak

Keycloak is an open-source identity and access management solution that supports OpenID and SAML 2.0 open protocol standards and is able to secure applications easily with little to no code.

It supports multiple features including SSO, MFA, identity and social brokering, user federation etc.

At the moment, Keycloak has a growing community and is being improved often with new features. It can be easily integrated with many social logins including Google, GitHub, Facebook and others. Also, it can be adapted to any database.

Apart from its security features, it is worth noting that Keycloak is also natively supported by Kubernetes,

The MES-CoBraD project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under grant agreement No 965422

meaning it can be used as an operator.

Benefits:

- easy to integrate
- adaptable and customizable
- highly scalable
- good with multitenancy

Limitations:

- unreliable documentation
- theme customization takes some practice

3.3.1.6 Access control

Although authenticated, not all users should have permission to all available information. Access control is a suitable way to grant specific permissions based on privilege to some users who are allowed to handle certain data.

Most often used solutions are **Role-based access control** (RBAC) and **Attribute-based access control** (ABAC). The main difference between them is how they grant access to the users.

For RBAC, the user is granted access using roles. Roles define what the user is able to see, add or delete, how long he is able to use the system etc.

On the other hand, ABAC grants access by appointing resource attributes. Attributes could be related to the user information, type of resource they are accessing or from which environment.

RBAC rules are easier to use and faster to execute, while ABAC rules provide greater granularity without increasing complexity.

Access control has not proved as reliable when used on its own, so it is recommended to combine it with other data security solutions.

3.3.1.7 Encryption

Encryption is a reliable way to protect sensitive data from data breaches because it protects it through its whole lifecycle, either when data is in rest or in transit.

By **data at rest**, we refer to the data that is stored in memory (hard drive, flash drive etc.).

On the other side, **data in transit** is the one that is actively moving through network (e.g., from local storage to cloud storage).

Data requires protection in both of these cases since unprotected data is vulnerable to potential attacks of the hackers.

When protecting data at rest, it is common to encrypt the sensitive data before storing it and/or encrypt the storage itself.

Protection of data in transit is often done by encrypting data before moving it and/or through encrypted connection, for example through **Transport Layer Security** (TLS) protocol. TLS is a cryptographic protocol most widely known for its use in securing web through HTTPS.

Encryption algorithms are usually separated in two groups: symmetric and asymmetric. In the symmetric algorithms, the same key is used for both encryption and decryption and each pair must be changed often to avoid attacks. The latter use different keys, public for encryption and private for decryption. Public key is available to everyone, while the private key is available only to the receiver of the encrypted message.

Three best known and widely used encryption algorithms are **RSA**, **DES**, and **AES.**

The MES-CoBraD project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under grant agreement No 965422

RSA is a strong, asymmetric type encryption, most often used to verify digital signatures.

DES was one of the first algorithms and was used in many different financial and business organisations, but is considered a bit slow for modern standards.

AES is considered to be the most secure encryption type available and is often used by governments and security organizations.

3.3.1.8 Data masking

Regardless of the goal of collecting medical data, usually there are legal or ethical limitations to its dissemination.

Sometimes it is not enough to only remove obvious identifiers such as name or address because of metadata which may point to geographical details or some other data that could be used to identify the individual.

The main goal of data masking is to obscure sensitive data by replacing them with other unidentifiable information. This must be done in a way that the statistical value of the information remains intact.

Contrary to encryption, with data masking it is not possible to extract the value which was replaced. This is one of the most preferred methods of data anonymisation as it is also the least expensive.

The organisation can mask data from the original source data in two fundamentally distinct ways [5]:

1. Disclosure-limiting masks. Modifications of the data that will be used to create results that pose an acceptable level of risk of exposure while sustaining the usefulness of data
2. Synthetic data. Estimated statistical model that is based on the source data and used to create simulated data

The main difference between masked data and synthetic data is that the latter has an additional step of creating a statistically estimated probability model, which is then used to simulate the data.

3.3.1.8.1 Disclosure-limiting masks

It can be valuable to partition data attributes into those that may be considered key variables and those that can be considered sensitive. The key variables present the connection to an external database and as such pose a threat to be the connection to sensitive variables. To best maintain the usefulness of data, it is recommended to mask the key variables.

Depending on the method of covering up the data variables, there are four typical approaches [5]:

- Suppression: decrease the possibility of confidentiality leak by suppressing an amount of data (deleting records or specific variables such as date of birth)
- Perturbation: intentional distortion of values in data records by replacing a part of data with some substitute values. The main goal is to generate similar results as using the original data, but to reduce the possibility that an outside source can link the information
- Sampling: includes creating a data product from a probability sample
- Aggregation: includes merging of attributes for some records, mixing the values of multiple attributes or the combination of both

3.3.1.8.2 Synthetic data

The concept of using synthetic data involves the creation of a model that fits the original data and the usage of samples from the generated model.

The advantage of synthetic data as opposed to masked data is that it is easy to analyse, but may also produce less accurate results in some cases.

The MES-CoBraD project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under grant agreement No 965422

3.4 DATA SHARING – COMPARATIVE ANALYSIS

When it comes to data sharing, the most important thing is to ensure that the data is in compliance with the corresponding data regulations, as well as that it is useful in terms of quality and completeness.

In order to achieve this, data should undergo a number of processes, after which it can be exposed in a manner useful for other analytical purposes.

3.4.1 TOOLS FOR DATA SHARING

Exposing data may be performed in many different ways, including REST (Representational state transfer) APIs (Application programming interfaces), message and context brokers, as well as software development kits (SDK), depending mainly on the format of data as a result of data harmonisation.

3.4.2 REST APIS

REST web services are the most often used method of retrieving data in the form of objects (JSON or XML) using HTTP requests. This is especially useful when developing applications as it is not important which programming language is used to process the data.

In essence, REST APIs represent a collection of methods used to retrieve, create, update or delete data. This collection can be created as user-friendly using tools like **Kong, Apigee, Traefik, Istio.**

Kong

Kong is an open-source tool based on Nginx's HTTP proxy server and written in the Lua programming language. It includes many features such as load balancing and service discovery. Kong is comprised of two layers, public and private. Public layer is responsible for exposing APIs to the final users and for routing requests to the backend, while the private layer manages the configuration of APIs and plugins in the backend.

One of its greatest abilities is flexibility and possibility to extend it through different modules or plugins, which is why it is often used to decentralise applications into microservices.

Benefits:

- flexible and customisable with plugins
- native integration with Kubernetes
- good scalability
- good community support
- has GUI

Limitations:

- requires some Lua language experience
- relies on third-party components

Apigee

Apigee is a gateway management tool that is robust and easy to use. It is written in Java and it was primarily made to deal with the legacy monolithic systems, while its focus on microservices is very low. It has a very complex architecture and requires a lot of planning before being deployed. Since it was acquired by Google, they also introduced the cloud hosted solution.

Benefits:

- full API gateway solution

The MES-CoBraD project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under grant agreement No 965422

- possible cloud hosting

Limitations:

- complex architecture
- not open-source

Traefik

Traefik is another open-source HTTP reverse proxy and load balancer with a focus on microservices. It is easy to integrate and has a good support for Docker and Kubernetes.

Benefits:

- good integration with Kubernetes
- several possible backends to use

Limitations:

- complicated setup
- not very good performance

Istio

Istio is an open platform based on service mesh architecture that is able to maintain microservices' traffic on a cluster management platform such as Kubernetes. Inside a service mesh, each microservice has its own individual proxy called "envoy" and each of them communicates with one another, forming their own mesh network. This type of architecture simplifies configuration and makes it easier to perform tests or debug problems.

Benefits:

- good integration with Kubernetes
- decentralized architecture

Limitations:

- complex setup
- still in early development

3.4.3 STREAMING SERVICES

To straighten the shortcomings of REST APIs, it is common to use streaming services, which enable users to create a sort of a data pipeline and make data analysis more efficient. In other words, with streaming services it is possible to deliver the data in real-time. A typical medical example would be to monitor the state of a patient and being able to respond immediately with adequate treatment in case of emergency.

Streaming services operate using events in a publish/subscribe paradigm. One entity is publishing events and the other is listening to the published events. These events can be organised and stored until it is confirmed that they have been received by the listening entity.

One of the most notable examples of a streaming service is **Apache Kafka**, but other solutions are available like: RabbitMQ, HornetQ, Cloud message queues (AWS SQS, Google PubSub, Azure Service Bus).

Apache Kafka

Kafka is a publish/subscribe tool based on message queue mechanism, developed by Apache. It is an open-source software and it is widely used for the exposure of streaming data. Due to its ability to use

The MES-CoBraD project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under grant agreement No 965422

multiple nodes, it is highly scalable and it can process large amounts of real-time data.

Benefits:

- high scalability
- works with real-time data
- robustness
- low latency and high throughput

Limitations:

- compression and decompression of data may reduce performance
- incomplete monitoring tools

3.4.4 HUMAN-TO-MACHINE READABLE DATA

There are cases in which exposing data via API is unacceptable. However, it is possible to provide data in a format which is easily readable by machines, and it is called CSV (comma separated values).

CSV is a format commonly used to present tabular data and is made up of lines and columns, where each line represents a data record.

One of its disadvantages is that it must be in a precisely defined format in order to be readable.

The MES-CoBraD project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under grant agreement No 965422

4 REQUIREMENTS' ELABORATION

4.1 EDGE MODULE

Table 1 Edge module requirements' elaboration

Code	LBE-1
Title	Pluggable system
User Facing	Yes
User Roles	Scientist; Clinician
Priority	Must have
Description and Rationale	The edge module (also called *"Local back end"*) is a lightweight system that allows to plug-in, configure and remove modules in fast and easy way.

Code	LBE-2
Title	Data workflow definition
User Facing	Yes
User Roles	Scientist; Clinician
Priority	Must have
Description and Rationale	The edge module should allow to define workflows over the incoming data. Each step of a workflow should be supported by a plugged module.

Code	LBE-3
Title	Workflow assignment to input data
User Facing	Yes
User Roles	Scientist; Clinician
Priority	Must have
Description and Rationale	The user is able to assign one or more workflow to the data.

4.1.1 ANONYMISATION MODULE

Code	AM-1
Title	Local RWD anonymisation
User Facing	No
User Roles	-

The MES-CoBraD project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under grant agreement No 965422

Priority	Must have
Description and Rationale	The Anonymisation module is one of the modules pluggable into the Local back end. It will allow to use a local script/app/executable for automatically anonymising/defacing/etc. local RWD before they are uploaded in the system.

Code	AM-1
Title	Anonymised data validation
User Facing	Yes
User Roles	Scientist; Clinician
Priority	Must have
Description and Rationale	The module will prompt back to the user the anonymised data for validation purposes.

Code	AM-3
Title	Anonymised data sharing
User Facing	No
User Roles	-
Priority	Must have
Description and Rationale	Once anonymised, the module will send the data to the cloud platform for analytical purposes.

4.2 DATA COLLECTION COMPONENT

Table 2 Data collection module requirements' elaboration

Code	DCO-1
Title	Data collection
User Facing	No
User Roles	None
Priority	Must have
Description and Rationale	The system shall collect and integrate data from various legacy and external systems using a common and standard based data framework.

Code	DCO-2
Title	Data collection configuration
User Facing	Yes

The MES-CoBraD project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under grant agreement No 965422

User Roles	System admin; Science coordinator; Patients
Priority	Should have
Description and Rationale	The system shall allow users to configure the data being collected.

Code	DCO-3
Title	Manage DAG
User Facing	Yes
User Roles	System admin
Priority	Must have
Description and Rationale	The system allows the user to create, delete and edit a DAG (a DAG is a Direct Acyclic Graph that represents a data flow with ingestion purposes). The business logic of a mashup is defined entirely in a low-code way, according to a python-based language.

Code	DCO-4
Title	Monitor a DAG
User Facing	Yes
User Roles	System admin
Priority	Must have
Description and Rationale	The system allows the user to track the log of each mashup execution and the status of each step of the data flow

Code	DCO-5
Title	Developer tool – new RWD type
User Facing	Yes
User Roles	Clinician-scientist; Platform developer
Priority	Would be nice
Description and Rationale	The system shall allow to introduce new RWD types (e.g, cardiac Holter). This type of data can also be used in a flexible platform.

Code	DCO-6
Title	Developer tool – recurrent executions
User Facing	Yes
User Roles	System admin; Platform developer

The MES-CoBraD project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under grant agreement No 965422

Priority	Would be nice
Description and Rationale	The system allows to schedule DAGs by specifying time frequencies. The system will automatically trigger the specific DAGs keeping track of their execution.

Code	DCO-7
Title	Developer tool – new data sources
User Facing	Yes
User Roles	System admin; Platform developer
Priority	Would be nice
Description and Rationale	The system shall allow to introduce new Data source connectors (e.g., RedCap, MySQL databases). Such connectors can be used then to build or enrich DAGs.

Code	DCO-8
Title	Activate/Deactivate DAG
User Facing	Yes
User Roles	System admin; Platform developer
Priority	Must have
Description and Rationale	The system allows the user suspend or restore the scheduling of a specific DAG. If a DAG is suspended it will not be executed or automatically trigger anymore.

Code	DCO-9
Title	Handle sensitive configuration
User Facing	Yes
User Roles	System admin; Platform developer
Priority	Must have
Description and Rationale	The system allows to securely handle sensitive configurations like password or access tokens. Such configuration will not be shown in the UI and can be only accessed by the DAGs.

Code	DCO-10
Title	Developer tool – Connection pools
User Facing	Yes
User Roles	System admin; Platform developer

The MES-CoBraD project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under grant agreement No 965422

Priority	Must have
Description and Rationale	The system allows the user to limit the number of active connections toward a specific data source, by reducing his workload and reducing its response time.

4.3 DATA LAKE

Table 3 Data lake module requirements' elaboration

Code	DLK-1
Title	Data storage
User Facing	No
User Roles	None
Priority	Must have
Description and Rationale	The system shall store data collected from heterogenous outside data sources (RWD) and from outputs of the system in Data Lake solution.

Code	DLK-2
Title	Management of stored data
User Facing	Yes
User Roles	System admin
Priority	Must have
Description and Rationale	The system shall allow users to manage data stored from heterogeneous external data sources (RWD).

Code	DLK-3
Title	Push data
User Facing	Yes
User Roles	System admin; Components
Priority	Must have
Description and Rationale	The system shall create a new file in a bucket.

Code	DLK-4
Title	Get data
User Facing	Yes
User Roles	System admin; Components
Priority	Must have
Description and Rationale	The system shall get an already existing file in a

The MES-CoBraD project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under grant agreement No 965422

	bucket.

Code	DLK-5
Title	Delete data
User Facing	Yes
User Roles	System admin; Components
Priority	Must have
Description and Rationale	The system shall remove a file the user owns.

Code	DLK-6
Title	Update data
User Facing	Yes
User Roles	System admin; Components
Priority	Must have
Description and Rationale	The system shall update a file with a newer version.

Code	DLK-7
Title	Create credentials for programmatic access
User Facing	Yes
User Roles	System admin
Priority	Would be nice
Description and Rationale	The system allows the user to create credentials with specific roles on different levels (bucket/folder/file) that will enable programmatic access on specific areas of the data lake.

4.4 DATA HARMONISATION COMPONENT

Table 4 Data Harmonisation module requirements' elaboration

Code	DHA-1
Title	Data Harmonisation
User Facing	No
User Roles	None
Priority	Must have
Description and Rationale	The system shall harmonise and transform the collected data into a standard format, using an internal data model that will be defined during the project.

The MES-CoBraD project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under grant agreement No 965422

Code	DHA-2
Title	Data Harmonisation Configuration
User Facing	Yes
User Roles	System admin; Science coordinator
Priority	Should have
Description and Rationale	The system shall provide users with an interface to configure and view process of harmonisation.

Code	DHA-3
Title	Manage a Mashup
User Facing	Yes
User Roles	System admin; Science coordinator; Scientist
Priority	Must have
Description and Rationale	The system shall allow the user to create, delete and edit a mashup (a mashup is a process that manipulates data with harmonisation purposes). The business logic of a mashup is defined entirely in a no-code way; it can be done by simply dragging and dropping operators on a working area.

Code	DHA-4
Title	Monitor a Mashup
User Facing	Yes
User Roles	System admin; Science coordinator; Scientist
Priority	Would be nice
Description and Rationale	The system shall allow the user to track the log of each mashup execution.

Code	DHA-5
Title	Manage datasources
User Facing	Yes
User Roles	System admin; Science coordinator; Scientist
Priority	Must have
Description and Rationale	The system shall allow the user to create, delete and edit a data source to use it in a mashup. (e.g. a rest api to be invoked during the mashup execution flow).

Code	DHA-6
Title	Create a mashup trigger

The MES-CoBraD project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under grant agreement No 965422

User Facing	Yes
User Roles	System admin; Science coordinator; Scientist
Priority	Must have
Description and Rationale	The system shall allow the user to define which are the possible triggers for a mashup. The possibilities are: schedule interval, http call, mqtt message, kafka message.

Code	DHA-7
Title	Create a mashup sink
User Facing	Yes
User Roles	System admin; Science coordinator; Scientist
Priority	Would be nice
Description and Rationale	The system shall allow the user to define where the mashup result should be published. The possibilities are: HTTP endpoint, MQTT Topic, KAFKA Topic, FIWARE Orion Context Broker.

Code	DHA-8
Title	Common contextual data framework
User Facing	No
User Roles	None
Priority	Must have
Description and Rationale	The system shall design and develop in a common contextual framework the collected, harmonised and RWD data, integrating and connecting them efficiently to ensure the interoperability of the data.

Code	DHA-9
Title	Missing data treatment
User Facing	Yes
User Roles	Scientist/Clinician
Priority	Should have
Description and Rationale	The system shall provide functions to impute missing data.

4.5 API GATEWAY

Table 5 API Gateway module requirements' elaboration

Code	APG-1

Title	API Gateway
User Facing	No
User Roles	None
Priority	Must have
Description and Rationale	The system shall make available an API Gateway according to OpenAPI specifications, exposing project information and allowing data collection in a standard and simple way to facilitate access.

The MES-CoBraD project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under grant agreement No 965422

5 DATA MANAGEMENT ARCHITECTURE

5.1 OVERALL ARCHITECTURE

The data management architecture in MES-CoBraD platform is depicted in the following image.

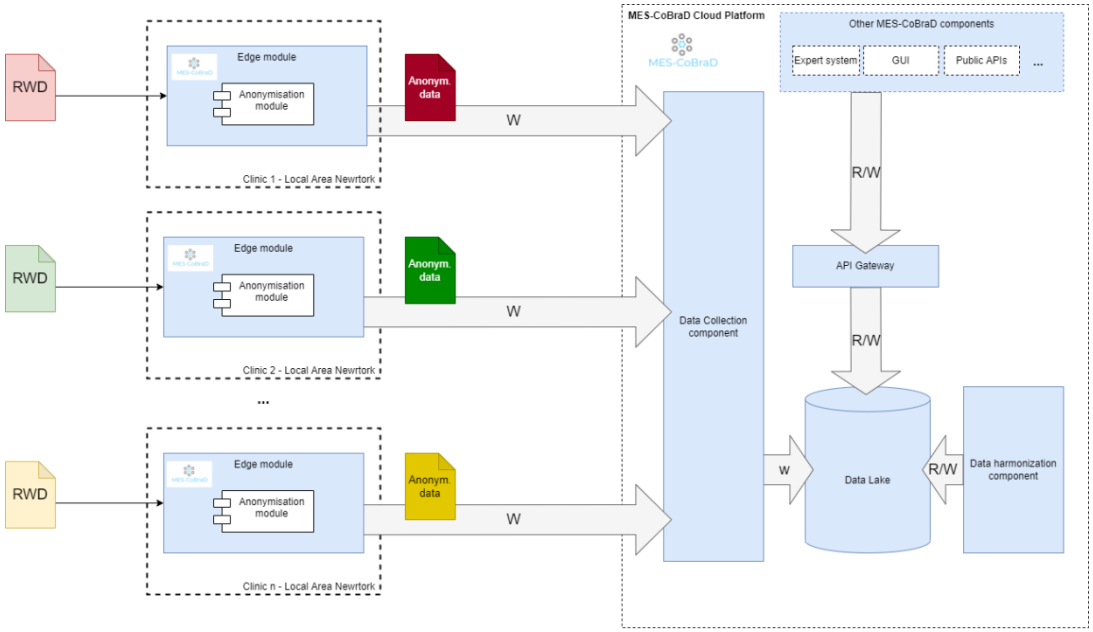

Figure 1 - MES-CoBraD data management architecture

The image shows the data lifecycle in the MES-CoBraD platform, from the ingestion to the sharing towards other components both internal to the MES-CoBraD platform and external ones, going through anonymisation, persistence and harmonisation steps.

The **edge module** (a.k.a. Local Back end) is the main entry point for the real-world data (RWD). It's a pluggable system that, out of the box, contains a mandatory plugin in charge for de-personalise the input data. The de-personalisation process is described in more details in the upcoming paragraphs and, even better, in the deliverable *Ethical and Normative Framework* for MES-CoBraD project.

As clearly depicted in the image above, each clinic has his own local instance of such an edge module to guarantee the restriction of the personal data availability only within the clinic Local Area Network. This would allow also to retrieve RWD from other local systems that, otherwise would not be available on the public internet.

The output of the edge module (i.e. the anonymised version of the data) is then written, across the internet, on the cloud. The cloud-side entry point for the data is the **Data Collection Component** that is in charge of receiving the data from all the local back ends and store them in the proper portion of the Data Lake.

The **Data Lake** is the focal point of the entire data management architecture since it stores both the original anonymised version of the RWD as well as their harmonised counterpart. The harmonised version of the data is created through the **Data harmonisation component** that allows to manipulate whichever original format of the data to convert them into a data-model known by all the other platform components. This tool both reads the raw data from the Data Lake and writes the harmonised version back.

All the data available in the platform, both raw and harmonised, as well as historical and real-time ones, is shared with the other platform components through the **API Gateway.** Such a gateway allows to read data from the Data Lake, for example for analytics and visualization purposes, and also to write back data onto the Data Lake, for example to track logs and relevant events for GDPR compliance purposes.

The MES-CoBraD project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under grant agreement No 965422

In the upcoming section, the internal architecture of each of the just mentioned components will be described in more detail.

5.2 COMPONENTS

5.2.1 EDGE MODULE

The idea behind the Edge module is to create a pluggable system, available only within the LAN of the hospital, that allows the scientists to install autonomously software modules in order to process the patients' data according to different business logics.

The up-mentioned software modules are, actually, plugins that must be installed in the system in a fast and easy way. Hence, the internal architecture of the edge module should be able to deal with the management of several plugins and their lifecyle. A **Plugin Manager** is in charge of importing and installing plugins in the local system, getting the plugin reference metadata from a remote (cloud) **Plugin Catalogue**.

The publication of plugins in the catalogue is ruled and validated by the MES-CoBraD community that ensures that all the plugins available in the catalogue are safe, performant, and not malicious at all. The plugin manager is also in charge of the entire lifecyle of the installed plugins: it takes in charge the updates, the trustfulness and the, eventual, deletion of the plugins.

Once installed, a plugin could be part of one or more workflows. A workflow is intended as a set of steps each of which manipulates the input data according to the internal implementation of the associated plugin. The workflows are managed through a **Workflow Manager** that is in charge of dealing with the creation, configuration and deletion of workflows.

Through the **API** layer, it's possible to access and inspect all the available edge modules assets (i.e. workflows and plugins) and also trigger specific workflows, submitting the necessary input data.

In order to deal with the limited resources that, most probably, the underneath hardware will have, the edge module is also enriched with a **Resource Negotiator** that is in charge for the assignment of resources to the workflows and the planning of the executions in order to both maximise the performance and minimise the delays of each run.

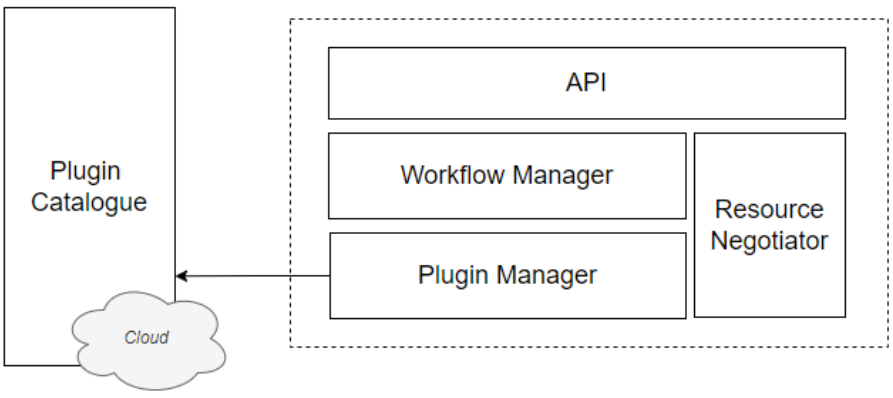

Figure 2 - Edge module internal architecture

5.2.2 ANONYMISATION MODULE

The anonymisation module, is the component responsible for anonymising patients' personal data before to be submitted to the MES-CoBraD platform. It will be used by scientists and clinicians as a tool to prepare the patients' data they collect (or produce) by running the anonymisation process accordingly to the existent configuration. The anonymisation will ensure to remove any patient-related information from the raw data (anonymisation phase) to avoid users' identification in case of data leaks or undesired data

The MES-CoBraD project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under grant agreement No 965422

access. The following diagram shows the high-level flow:

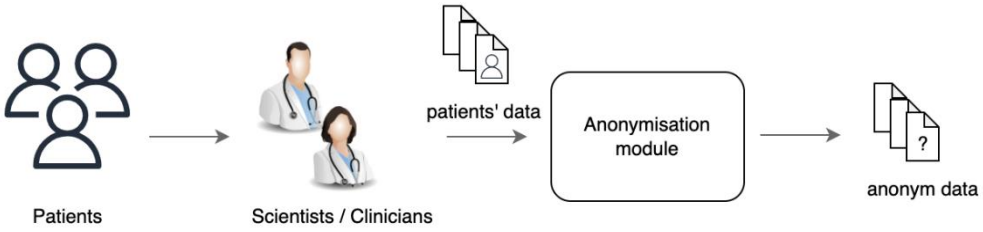

Figure 3 - Basic anonymisation flow

Once anonymised the output data will be verified by the original submitter. After the final approval the data could be uploaded within the MES-CoBraD platform to be anonymously available to other end-users and other components of the platform. The anonymisation module will be extendible through plug-and-play plugins that could be made available by other scientists and developers. The plugins enhance the power of the anonymisation component by bringing additional functionalities such as advanced anonymisation algorithms, support for different data formats and new protocols.

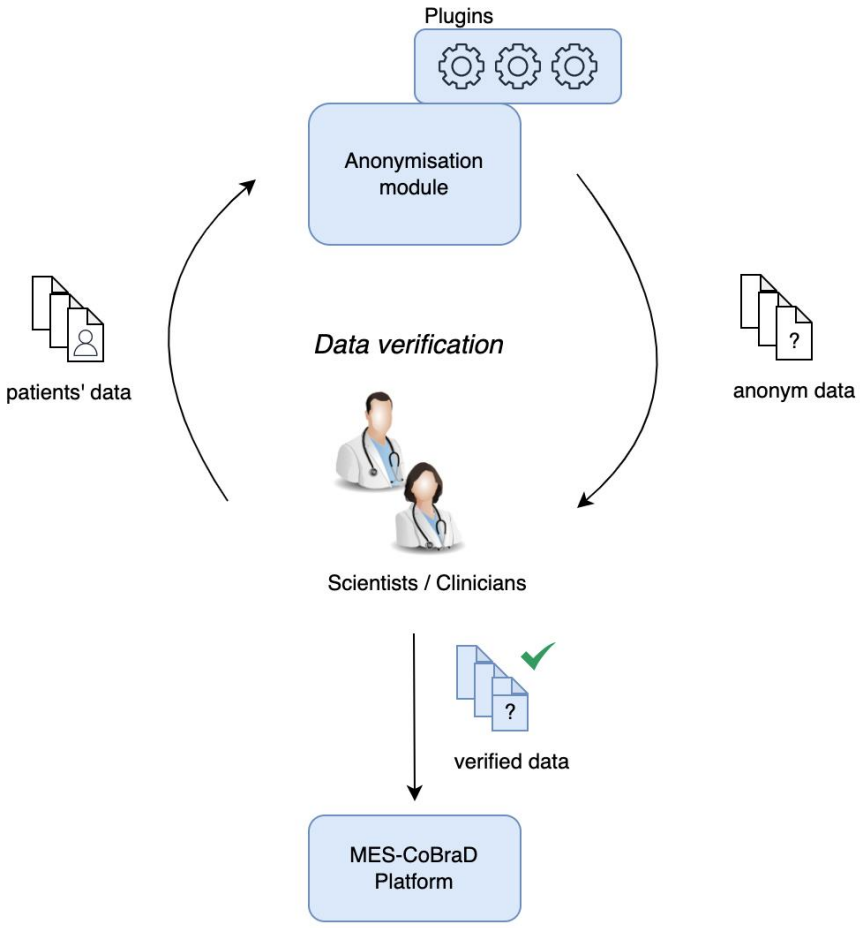

Figure 4 - Anonym data verification is an iterative process

Security is one of the most important aspects of the component and, in general, of the entire MES-CoBraD platform. Patients' information are intrinsically sensible data, for this reason the anonymity must be ensured. At the same time the submitter might want to access the original information with all the patients' data back (e.g. to update the file with a newer version or to simply check the patients' health status). Only the authorised user(s) will be able to see raw data containing the sensitive information,

The MES-CoBraD project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under grant agreement No 965422

making the whole system compliant with several security and privacy standards.

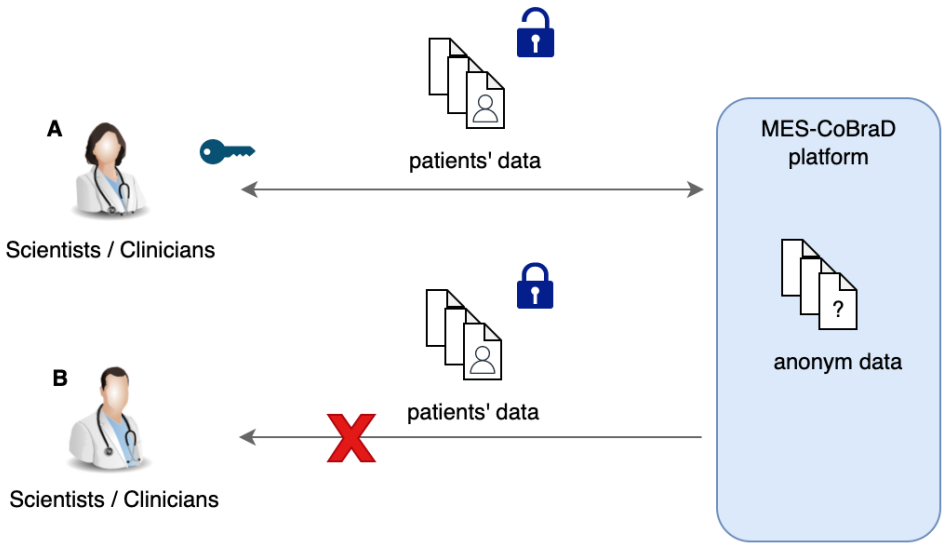

Figure 5 - Original submitters are the only users able to recover raw data with sensitive data

The overall process flow is described as follow:

1. Anonymisation module configuration – The anonymisation module is available locally and could be extended with plugins provided by developers and scientists. Scientists and clinicians can configure the anonymisation module by using the available plugins.
2. Patients' data collection – Scientists and Clinicians prepare patients' data to be submitted to the anonymisation module.
3. Data anonymisation through the module – Patients' data along with the module configuration are the input for the component.
4. Manual verification of the anonymised data – Once data are anonymised, they are manually verified by the submitter that will determine if the data has been correctly anonymised accordingly with the requirements.
5. Verified data submission – Verified data could be uploaded within the platform to make them available to other components and users.

Moreover, as described in the previous steps anonymised data can be de-anonymised only by the user who originally submitted the data within the MES-CoBraD platform.

The MES-CoBraD project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under grant agreement No 965422

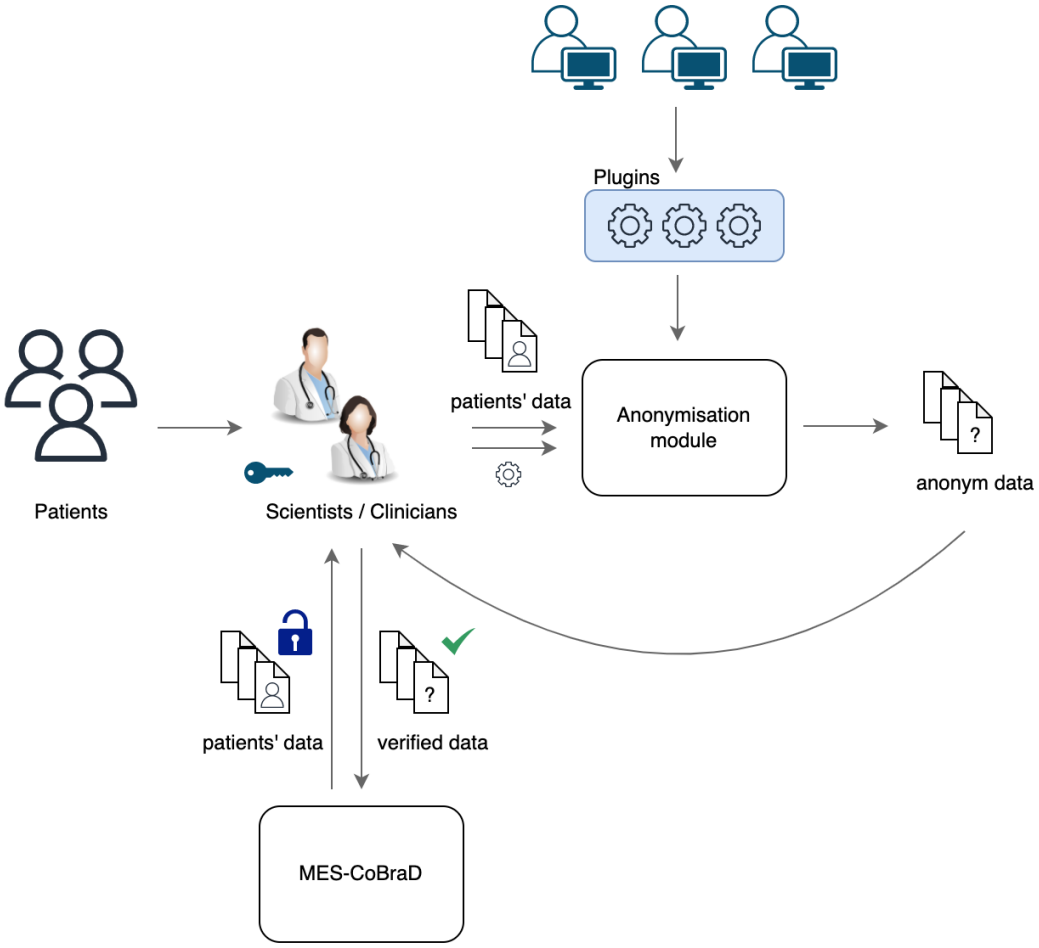

Figure 6 - Overall anonymisation flow

5.2.3 DATA COLLECTION COMPONENT

The data sources that contain all the information needed to make the platform working usually use different technology stacks. Dealing with them imply having several ways of connecting with the correct protocol, limiting the number of connections, correctly handling parallelisms and being able to efficiently query them. The data collection component is the tool responsible to extract information from heterogeneous data sources independently from their technology stack, perform operations from the extracted data and store the result in a set of target systems, ready to be analysed and used. Its main goal is to provide a set of building block to create ETL pipeline easily, without taking care of some low-level technical details like performance optimisations and connections handling. Such component will act as workflow orchestrator, where users can define a (complex) sequence of steps, like connections to databases, processing operations and script executions, their order and, if necessary, a schedule interval to automatically trigger the pipeline based on datetime or events. Such executions will be monitored at different level: for each DAG execution will be possible to monitor its status, the execution time, and check the artifacts produced within the staging area (DAG level monitoring). Since each DAG could be composed by several tasks they can be monitored as well (Task level monitoring). Indeed, it will be possible to access tasks' status and logs in order to monitor its behaviour or to debug anomalies. The component will also be able to scale the workload across several instances to reduce execution time of DAG and to efficiently allocate resources among the underlying hardware. To accomplish such goal the data collection component will run on top of latest cloud technologies, to ensure high-availability, isolation and security.

The core concept within the component is the Directed Acyclic Graph (or DAG) that defines a workflow.

The MES-CoBraD project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under grant agreement No 965422

Each DAG is composed by "tasks" that can interact with external systems and run some logic on top of data (e.g., run a script, perform joins, clean data, …). A typical DAG looks like the following:

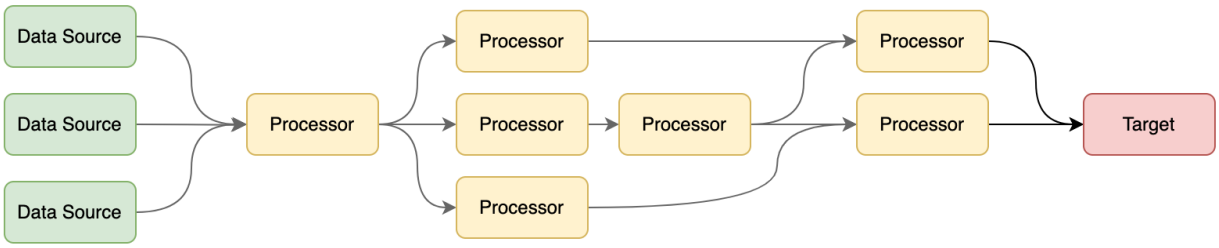

Figure 7 - Directed Acyclic Graph (DAG)

Each task is an instance of an operator, a provided building block that can perform different types of operations. There are mainly five kinds of operations:

- Sensors – Such operators wait for some events before to move forward to next tasks. The events could be an HTTP request, a new file within a remote folder, a notification or any alert that can notify the component.
- Extract operators – These operators use the underlying connectors (provided or custom) to interact with external sources in order to extract data from.
- Migration operators – Like the extract operators also the migration ones use the connectors to communicate with external sources. Here the sources are usually two and the operator allow the data migration from sources A to source B, allowing users to specify configurations for both the external systems.
- Processors – Such operators implement the real business logic of the workflow. They can be written in whatever programming language giving teams the flexibility to choose the right technology for the right purpose.
- Utilities – All the operators that help in the maintenance, the organisation and the proper execution of the pipelines are considered utility operators and are available too as building blocks for composing the workflows.

Each DAG can be run manually or scheduled to be executed on specific time interval, e.g., one day, one hour, twice a day, etc… The scheduler is the component responsible for checking the schedule interval of each DAG and eventually start it. Once a DAG status is set to *running* the "executor" create the processes among the underlying hardware. Specifically, each process run on a worker, that is a special process that can create, manage and delete operator instances.

The MES-CoBraD project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under grant agreement No 965422

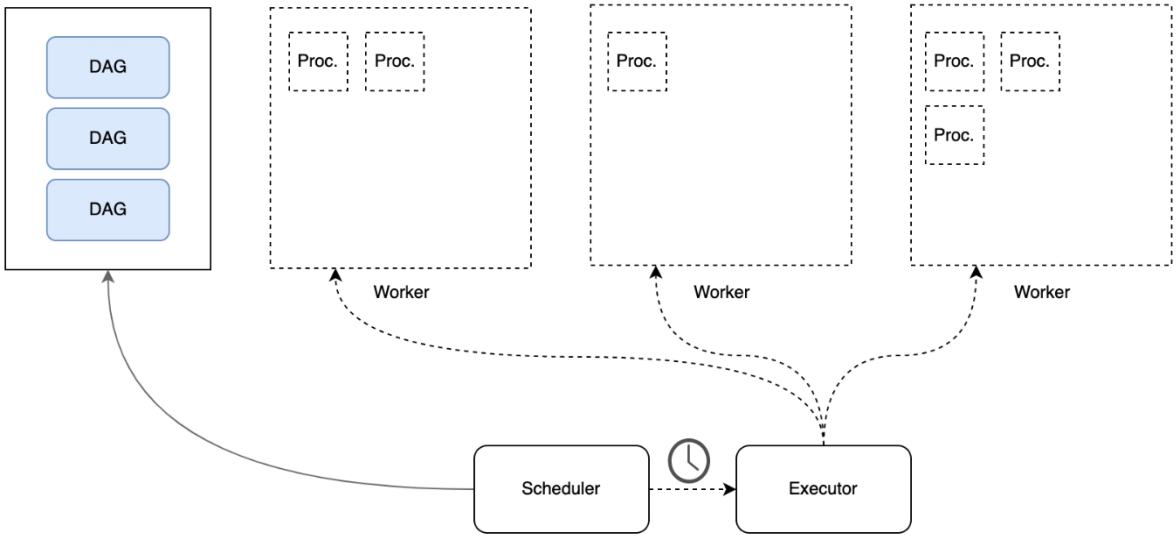

Figure 8 - Scheduler architecture

Workers can run inside ad-hoc hardware or inside a cluster if the data collection component is deployed using cloud technologies. They are strictly related to the type of executor the component is configured with. If the component is deployed on a single server, then the executer will be a *local* executor that will spawn process in the same machine. Within the MES-CoBraD project the data collection component will be deployed in a cloud environment composed by multiple nodes, where each node is a machine that can run software and that is connected to the other nodes to act as a single entity. In such scenario the executor is strictly related to the cloud technology, that is powered by Kubernetes.

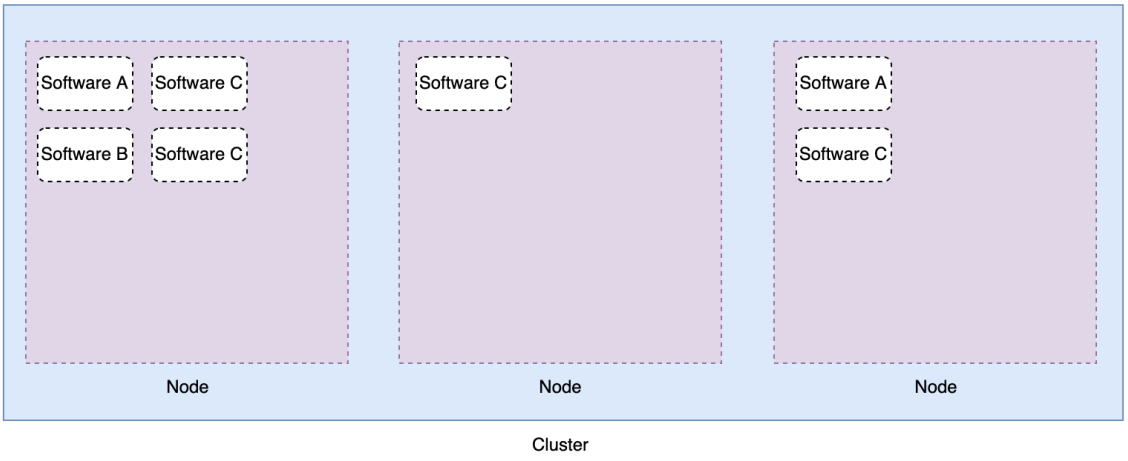

Figure 9 - A cluster is a set of logically connected nodes

The overall component's architecture includes a web server that serves a UI to the end user to check DAGs execution, manually trigger DAGs, monitoring workflows and managing connections and configurations. All these information is stored within a Metadata DB, that is accessed also by the scheduler, the actual executor (In case of cloud deployment it will be a Kubernetes executor) and the processes. The complete architecture is shown in the next diagram:

The MES-CoBraD project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under grant agreement No 965422

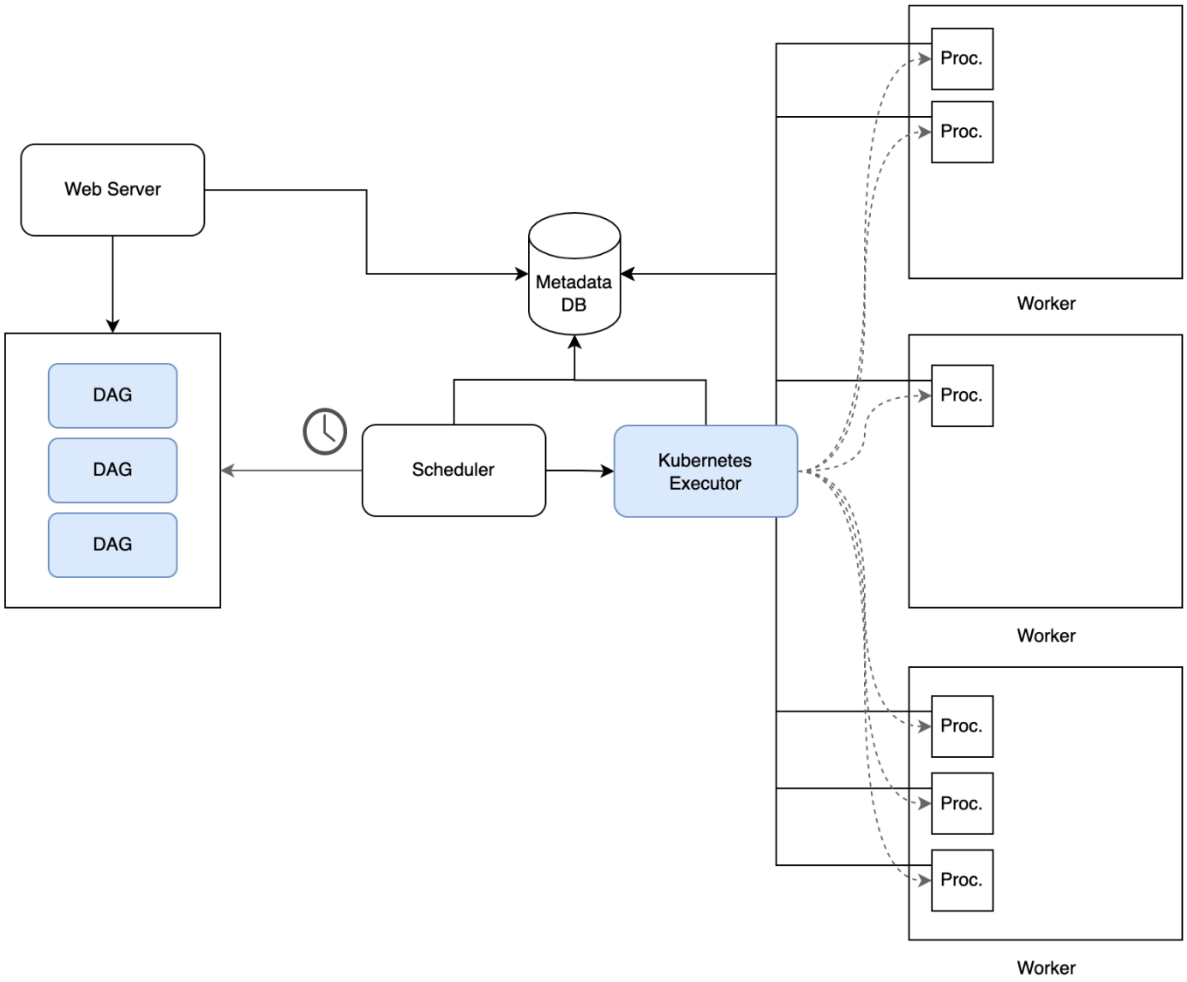

Figure 10 - Data collection component architecture

5.2.4 DATA LAKE

Dealing with real-world data implies managing high-volume heterogeneous data. As described in section 2 health data are intrinsically Big Data and must be managed with the correct techniques and tools. Within the MES-CoBraD platform the persistence layer will be composed by a Data Lake, where the different actors of the platform can push and extract information or generate analytical reports.

The data lake architecture is composed by three types of storage:

- Object Storage – It allows to manage whatever object (e.g., documents, videos, audios and binary files) by allowing versioning and encryption of each file uploaded within the object storage. Files can be associated with metadata, that can be used to query data, group them and perform analytics based them. Such storage layer will be used to store anonymised patients' data and the outcomes of the harmonisation process. Data can be organised in folders that relies on same the "bucket", where a backet is a cloud resources that usually reflect a specific use case or business sector. The tool allows also to define access policies to specific files, folders and buckets by defining the king of permitted/denied actions. In order to ensure high-availability and replicability the object storage is replicated across the cluster and can be scaled elastically if needed.
- Relational Databases – Such databases will contain all the necessary data to make the platform working, and to allow analytical components to be able to efficiently query data to

The MES-CoBraD project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under grant agreement No 965422

create reports, machine learning models and to identify patterns and insights. Relational databases usually are partitioned into multiple database instances to ensure to have fast performance on read/write operations and to better handle replicability and security.

- NoSQL Databases – Those are used to store non-relational data like health devices measures, asynchronous events and logs. This layer is composed by two different databases: a time series database to store time-dependent information and a logging database to collect information about platform usage and to allow better monitoring/debugging of the MES-CoBraD platform's components. Depending on the workload NoSQL databases can scale up and down elastically, by using the capabilities of the underlying cloud technology.

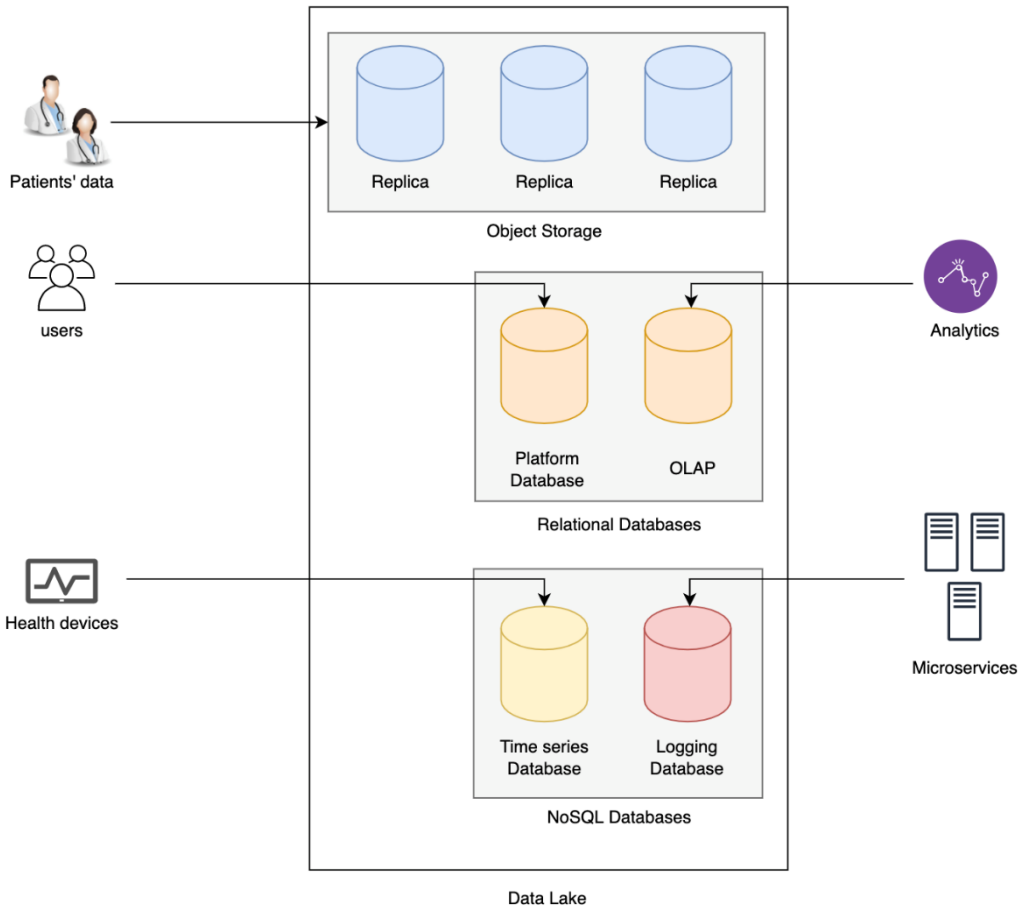

Figure 11 - Data lake components

5.2.5 DATA HARMONISATION COMPONENT

Data Harmonisation tool is the one which is used to extract and transform data from the data lake, preparing it for further processing.

It consists of three main layers: core, shared and tenant.

The core layer contains the authentication management, key-value storage and message broker. The authentication is used when accessed through the frontend UI. Key-value storage is used to store shared configuration and the message broker is used for a communication between layers.

The shared layer contains the backend API, database, scheduler, workflow engine, logger and frontend UI. Backend API is the microservice containing all of the exposed server APIs. Database is used to store

The MES-CoBraD project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under grant agreement No 965422

workflow information, as well as logs from all layers. Scheduler is used to manage deployment and execution of workflows in tenant namespaces through the message broker. Workflow engine performs the execution. It uses the received JSON descriptor to return transformed data in JSON format.

Core and shared layers are common to all the tenants, while the tenant layer should be unique for a specific tenant.

On tenant layer, main component is the controller, which manages the deployment of workflows, using the Kubernetes API to deploy and stop Kubernetes pods. These pods can be triggered or targeted from the outside using HTTP, MQTT, KAFKA or ORION. In case of HTTP, the request must go through a proxy which exposes a single ingress to all workflow pods, and must contain an execution flow id which is forwarded to a specific pod.

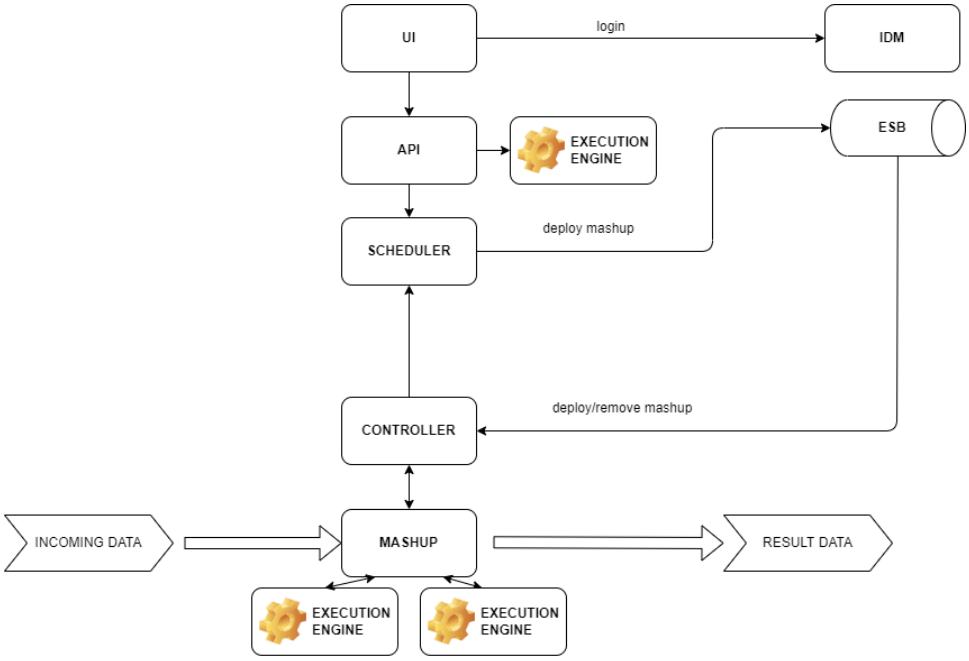

Figure 12 - Architecture of the Data harmonisation tool

5.2.6 API GATEWAY

An **API gateway** is a management tool that collects all calls to the application, invokes all necessary microservices and returns an appropriate result, behaving like an abstraction layer between the clients and the server.

Usage of API gateways is becoming more popular since the rise of the microservices approach. It is not a best practice to enable direct communication between clients and microservices because the API for each microservice would need to be exposed and sometimes it can be hard to understand the complex network of services behind the server. By using an API gateway, we ensure that there is only one entry point for the consumers of the application.

The MES-CoBraD project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under grant agreement No 965422

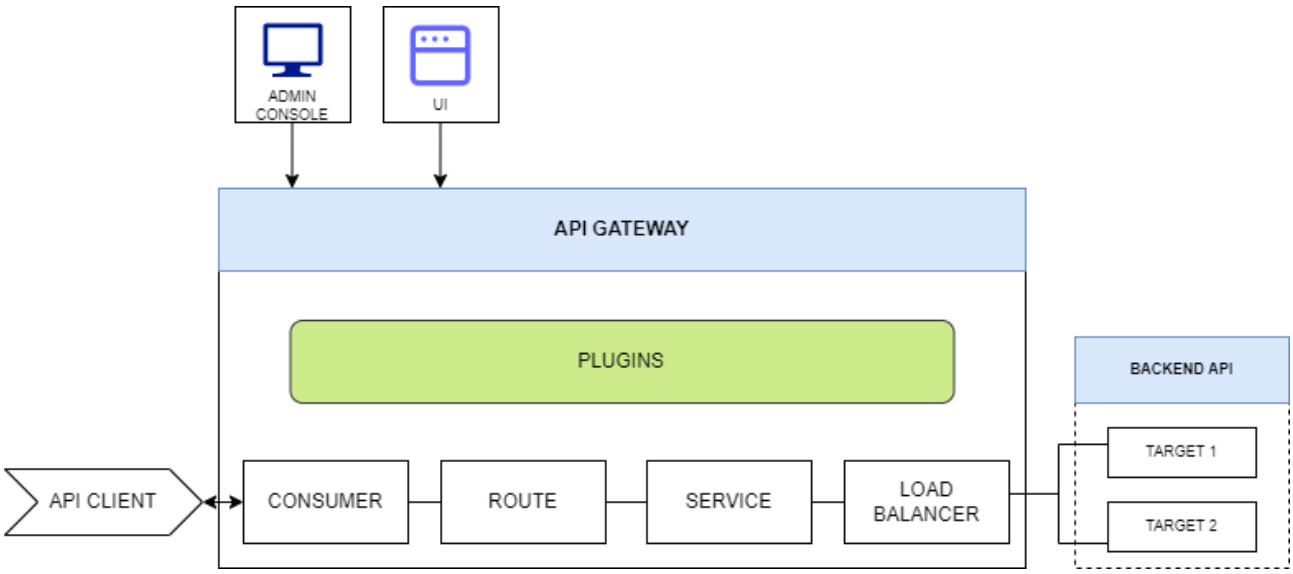

Figure 13 - Architecture of the API gateway

The MES-CoBraD project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under grant agreement No 965422

6 DATA MANAGEMENT MOCK-UPS

6.1 USER INTERFACE

6.1.1 EDGE MODULE

The edge module presents a web UI where the plugin management is possible. The following picture presents a detailed wireframe of the plugin management page.

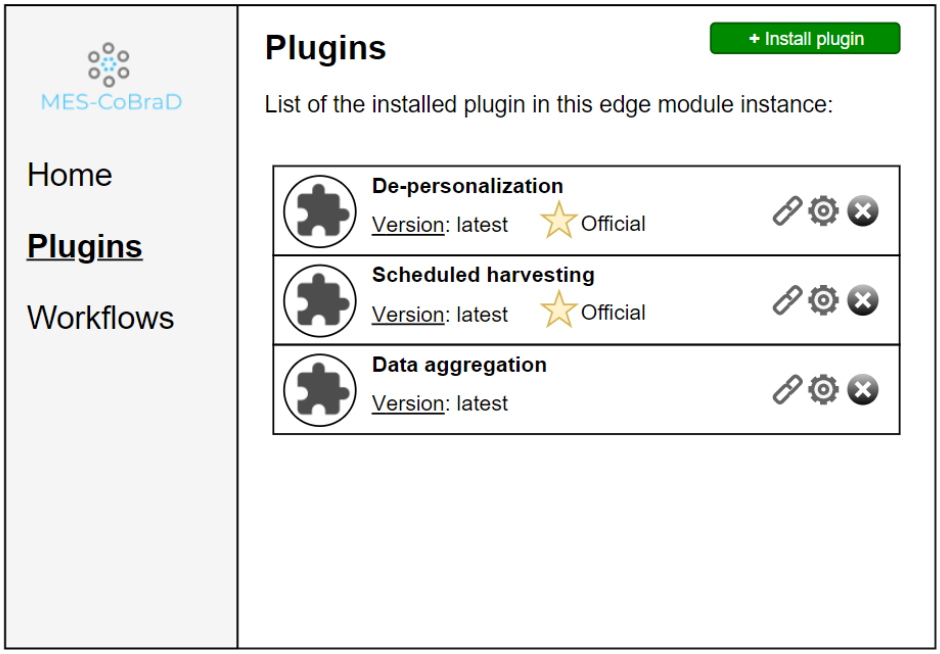

Figure 14 – Mock-up of the edge module (I)

The plugin management page presents the list of the plugins already installed in the system. For each plugin are clearly visible:

- the name of the plugin
- the installed version
- a star badge to highlight the official plugins, released directly by the MES-CoBraD project consortium
- the link to the documentation
- the link to the configuration page
- the button to uninstall the plugin from the system

The button *"Install plugin"* will open a new page, shown hereafter, to install a new plugin into the system.

The MES-CoBraD project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under grant agreement No 965422

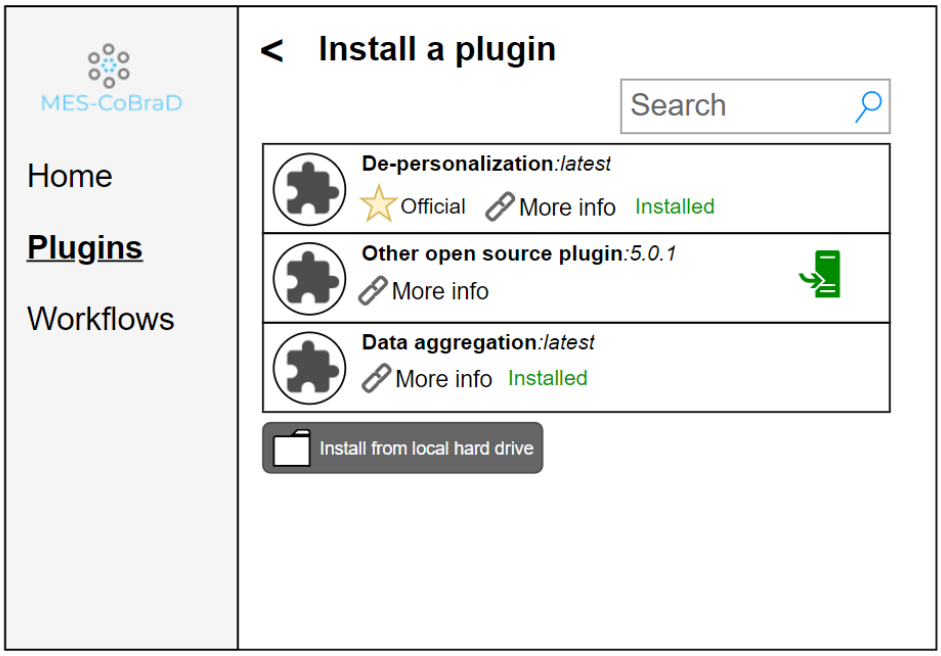

Figure 15 – Mock-up of the edge module (II)

The installation page is populated from a central repository of available plugins. A search box allows to search over the, potentially several, available plugins.

For each plugin available in the repository, the corresponding list item shows:

- the name and the version
- the star badge to highlight the official plugins
- the link to the documentation
- Either the “installed” label – for the already installed plugins – or the “Install” icon to trigger the installation of the plugin in the current system

In case the user wants to install an unofficial plugin in the system there is also the possibility to upload the plugin from the local hard drive of the user, through a dedicated button that will open an “upload” prompt window.

Once installed a plugin can be incorporated into a workflow.

The MES-CoBraD project has received funding from the European Union’s Horizon 2020 Research and Innovation Programme under grant agreement No 965422

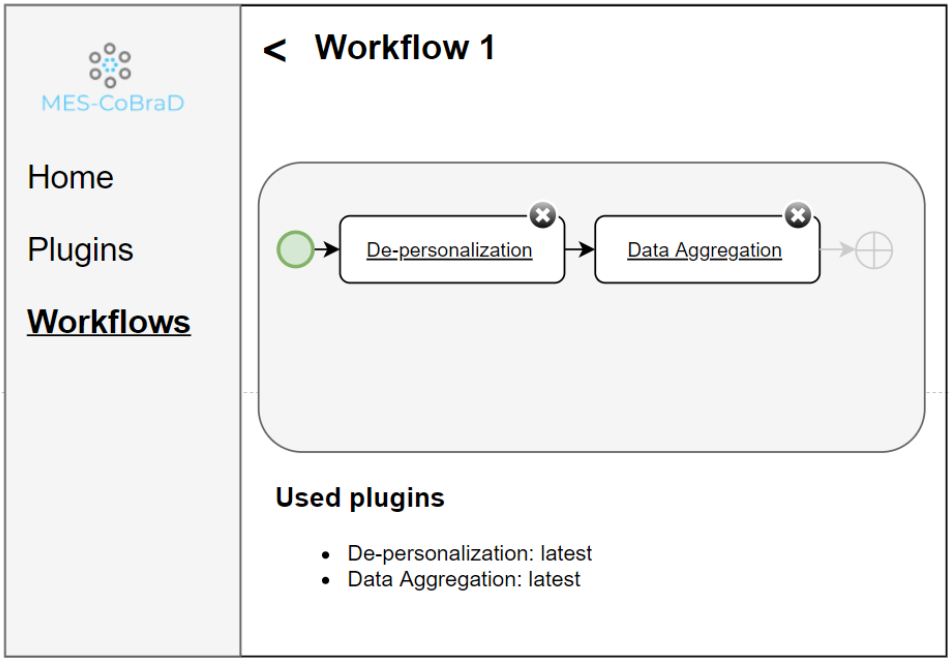

Figure 16 – Mock-up of the edge module (III)

The workflow details page shows the end-to-end workflow elaboration for the input data. An activity diagram will show the steps of the data processing. It's possible to both configure and remove each of the step already in the workflow and even add new steps.

For documentation purposes, especially for complex and log workflows, the bottom part of the page shows a list of the plugins used in the current workflow.

6.1.2 ANONYMISATION MODULE

This component will not have a dedicated User interface since it will be a backend module.

6.1.3 DATA COLLECTION COMPONENT

The data collection component allows end-users to exploit its functionalities using both APIs and a user interface. The focus of this section is to describe the component's UI focusing on the functionalities of each the application.

The component UI is divided into three different sections:

- DAGs
- Connections
- Configurations

The following picture represent a detailed wireframe of the DAG section:

The MES-CoBraD project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under grant agreement No 965422

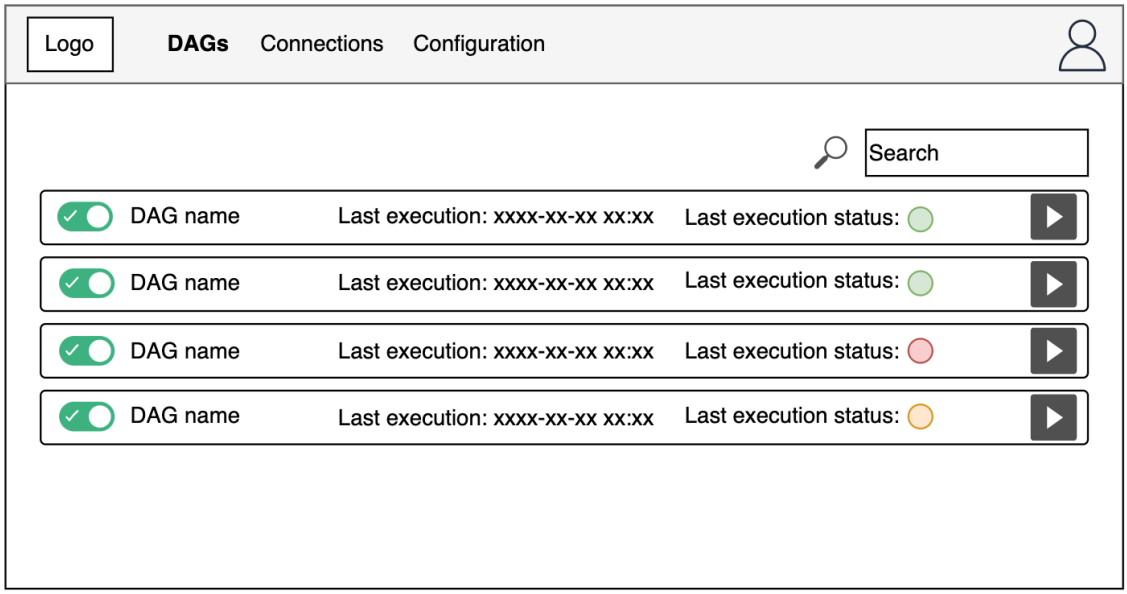

Figure 17 - DAG View

The main section shows the list of the DAGs that has been published within the component. Each row contains information like DAG's name, its latest execution (Date time) and the last execution status. The Execution status can assume the following values:

- Light green (Running) – The DAG is currently running.
- Green (Done) – The DAG has been executed correctly.
- Yellow (Warning) – Some of the tasks within a DAG could have been skipped for some reason.
- Red (Error) – One of the tasks failed during its execution.

DAG can be searchable by using a search bar that can filter them using part of the name or tags (if any). From the same page DAGs can be enabled/disabled and triggered manually.

The DAG workflow is accessible by clicking one of the rows in the main page. It contains three different sub-sections:

- Workflow
- Execution time
- Logs

The workflow view shows the DAG diagram with the interconnections between tasks. Each task is highlighted using a colour-scheme that reflects the one described above for the DAG status. Tasks can be searchable through a search bar by using their name or the operator's name.

The MES-CoBraD project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under grant agreement No 965422

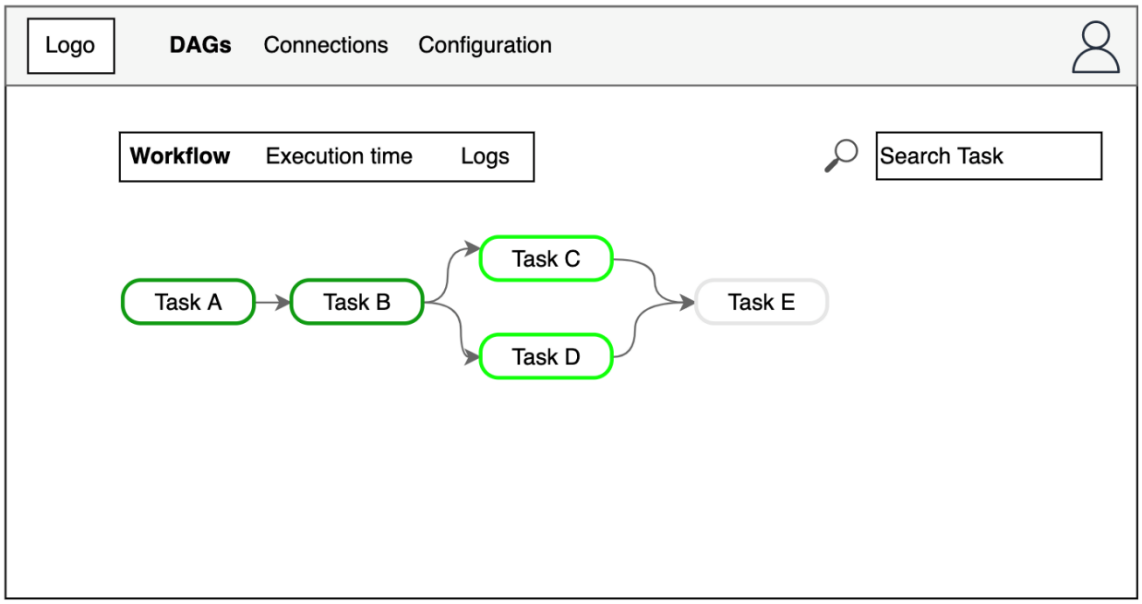

Figure 18 - Workflow details

The execution time section shows a diagram where tasks' execution time are compared. This section allows developers to rapidly identify the tasks that act as bottleneck for the entire DAG to focus the attention for further optimisations.

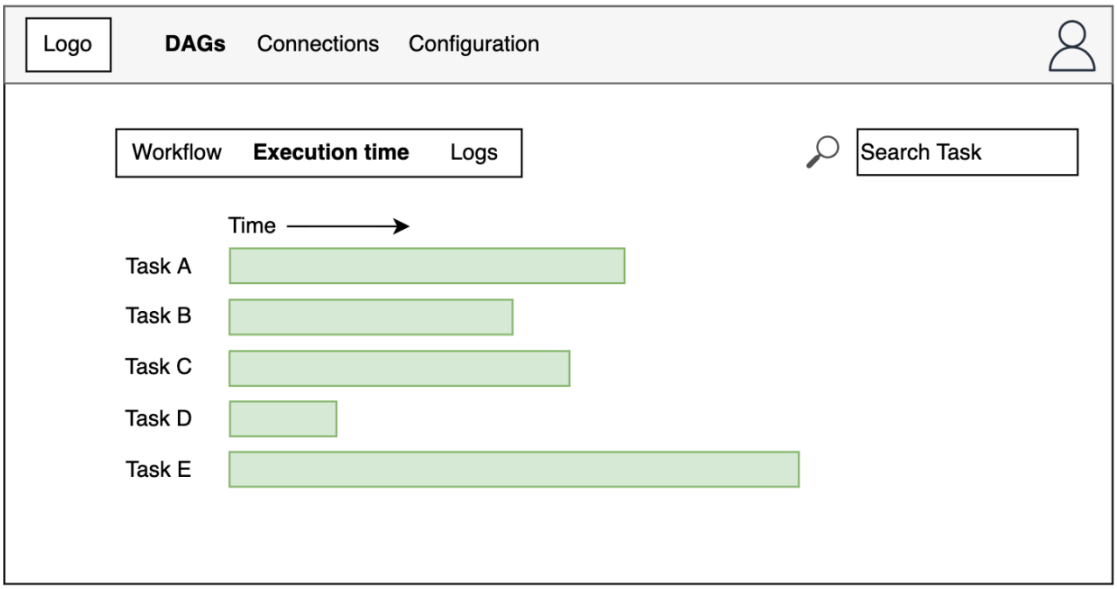

Figure 19 - Data collection component allows to compare execution times

The log sub-section shows the DAG execution's details at different grains. It can help in debugging not-working tasks or to identify unwanted behaviours.

The MES-CoBraD project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under grant agreement No 965422

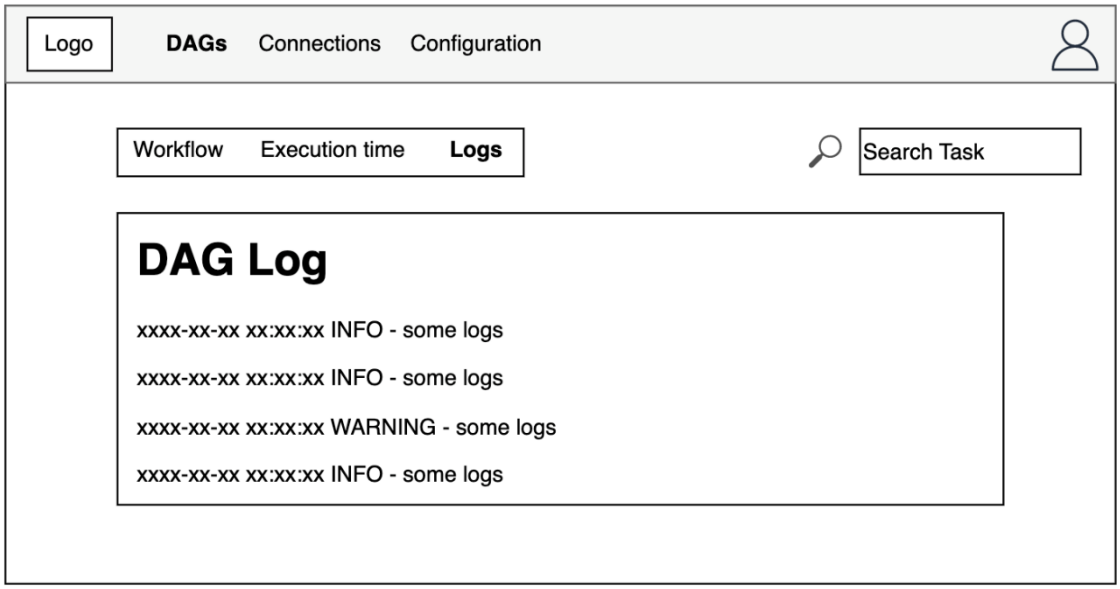

Figure 20 - DAG logs

The second component's section shows the information about connections with the registered external systems. As described in section 4.2 the component is able to connect with several data sources. New source can be installed within the component in order to let system admin to create connections based on such source. This page allows CRUD operations on connections and to search them by name or by kind of connection (Database, Http endpoint, FTP server, etc.).

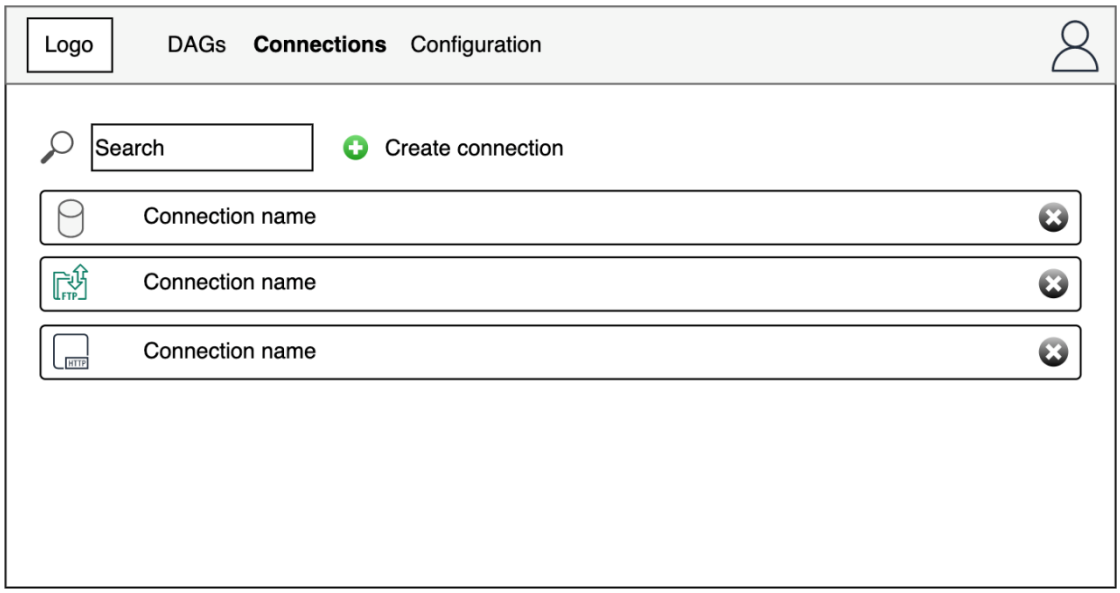

Figure 21 - Data collection component allows to define connections by using the user interface

The configuration section of the component is the configuration manager. Here users can define configurations that will be used within the DAGs. The page shows the list of available configurations against their value. Configurations can be searched using the search bar or consulted for modifications by clicking on them.

If system admin needs to define configurations that contains sensitive information like password, tokens or keys he can define secrets using this section. Secrets are hidden in the UI and can be accessed only by DAGs without showing their content within the logs.

The MES-CoBraD project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under grant agreement No 965422

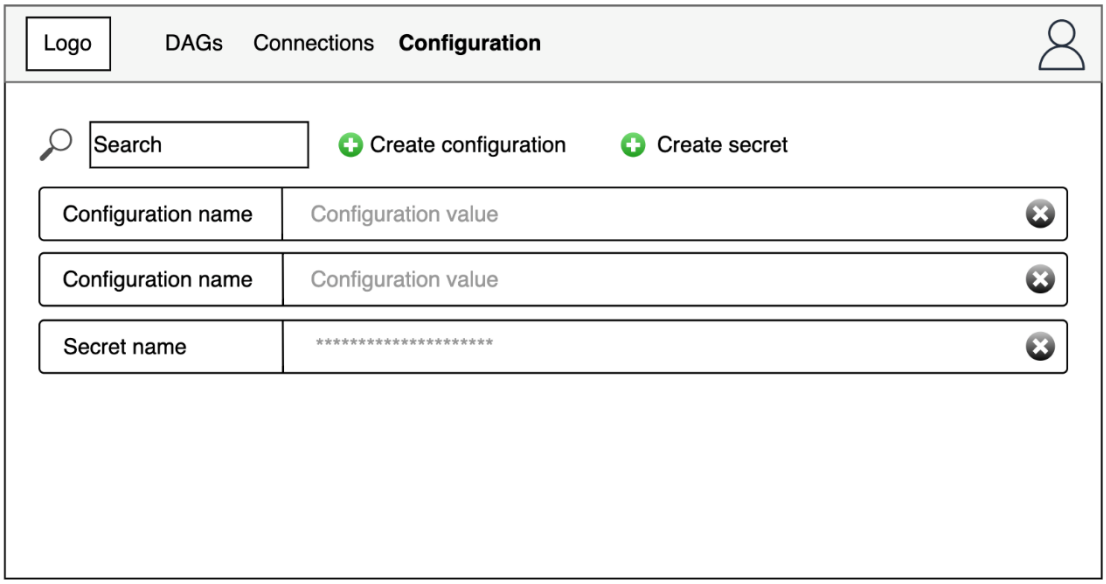

Figure 22 - Configuration management

6.1.4 DATA LAKE

Data lake is mainly accessed programmatically by the components that need to read and write file within specific areas of the storage. The tool comes with a bundled-in UI that enables end-users to access files and folders using a web browser. The UI contains two sections:

- Bucket view
- Settings

The bucket view allows user to manage the object storage at bucket level. It shows the list of available buckets, to create new buckets and to remove existing ones.

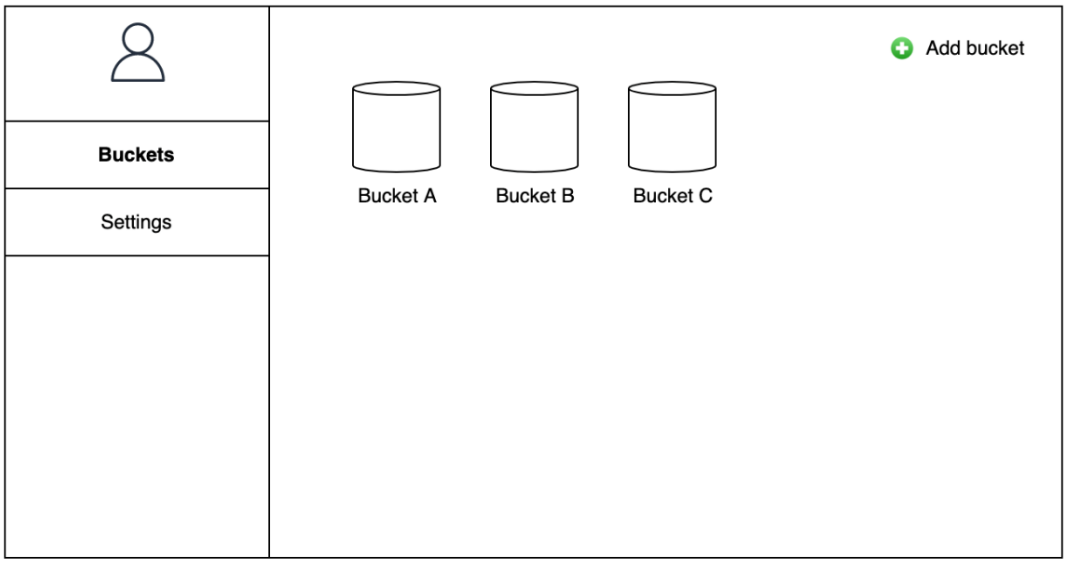

Figure 23 - Bucket view

Buckets contain a tree structure composed by folders (nodes) and files (leaves). Elements within a bucket can be accessed also through the UI by clicking in the file/folder of interest. Moreover, The UI

The MES-CoBraD project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under grant agreement No 965422

allows also to create folders and upload files using a drag-and-drop approach.

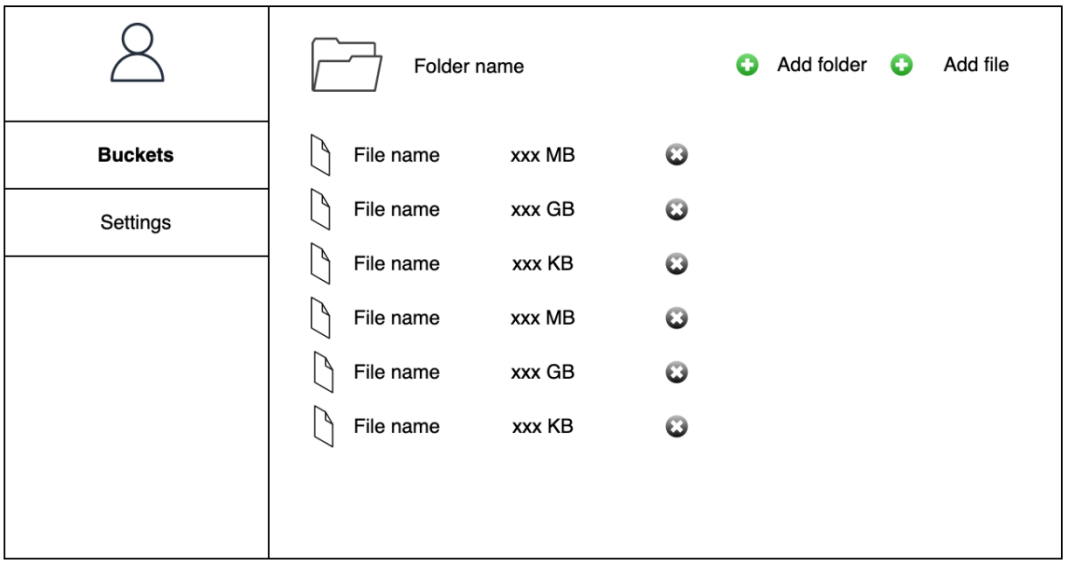

Figure 24 - Folder structure

CRUD Operations (at different grains) can be performed only by users who are authorised by system admin. Such authorisation can be also created for programmatic access of the object storage by defining specific credentials. The setting section of the UI allows user to manage such credentials by doing CRD operations on them. Indeed, this section allows to list the existing credentials, remove them and to create new ones.

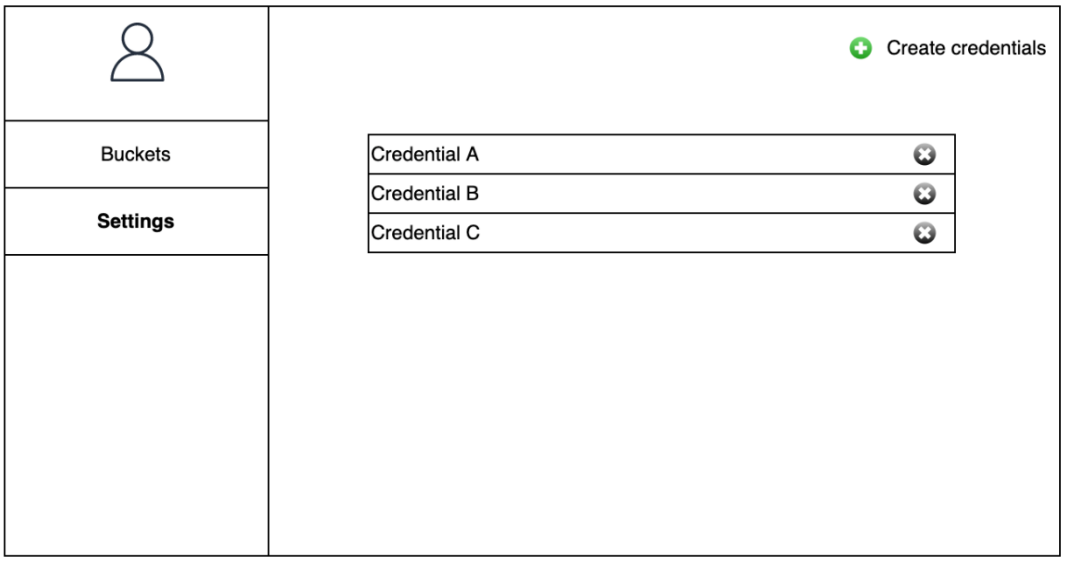

Figure 25 - Object storage allows to define credentials for programmatic access

Credentials can be created by specifying a set of permission using a meta-language easy to understand. The picture below is a wireframe showing the credential creation view:

The MES-CoBraD project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under grant agreement No 965422

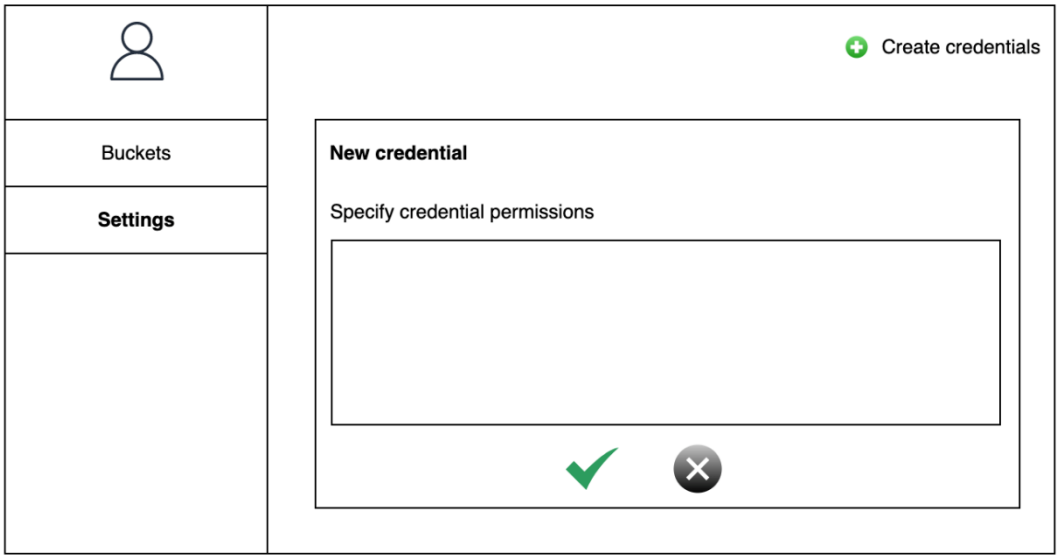

Figure 26 - Policy form

6.1.5 DATA HARMONISATION COMPONENT

Data harmonisation tools consist of a palette of transformation tools on the left side and a dashboard in which these tools can be used. The tools can be dragged and dropped into the dashboard and then connected to transform the data into the desired format. After each connection, the user should choose the data which should be transformed.

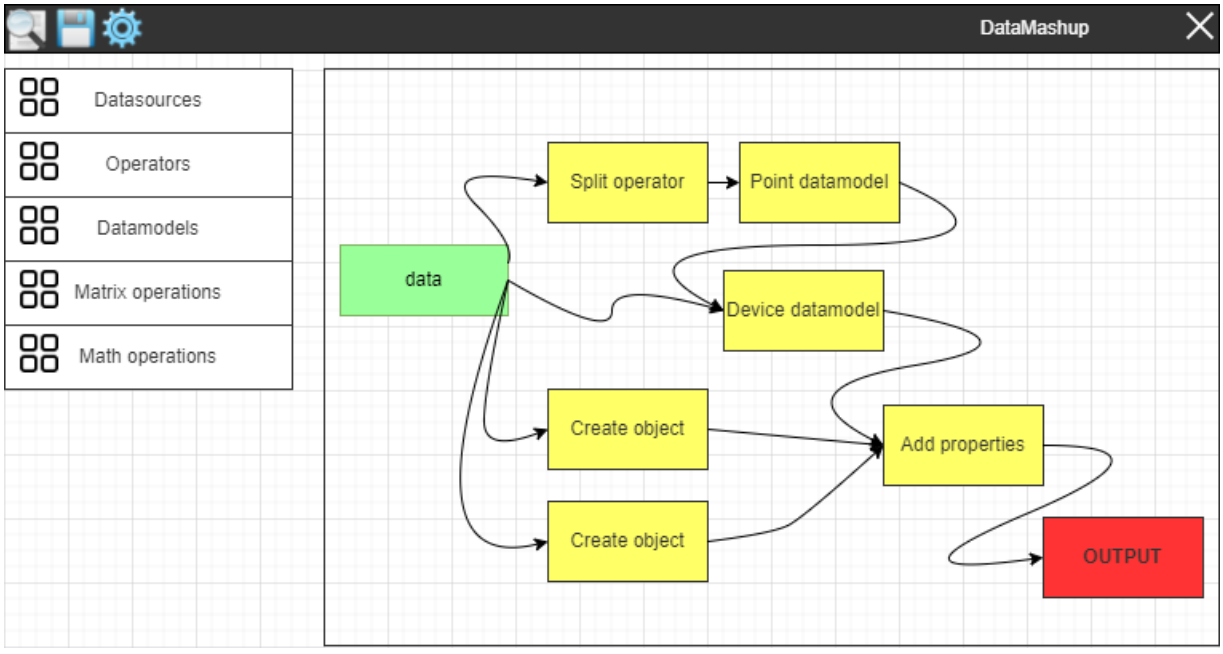

Figure 27 - Mock-up of the Data harmonisation tool

6.1.6 API GATEWAY

The API gateway is a backend tool that, then, will not have any GUI.

The MES-CoBraD project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under grant agreement No 965422

7 APIS' SPECIFICATION

7.1 EDGE MODULE PLUGINS

GET	/plugins	
Path params	-	
Query params	limit	The numbers of items to return.
	Offset	The number of items to skip before starting to collect the result set.
	Order_by	The name of the field to order the results by. Prefix a field name with "-" to reverse the sort order.
Body	-	
Headers	Authorization	Bearer <token>
Success Response	Code	200; 401
	Schema	[{ "id": "string", "name": "string", "version": "string", "official": true | false, "docs": "string" }, ...]
List all the installed plugins		

POST	/plugins	
Path params	-	

Body	{
	"id": "string",
	"version": "string",
	"configuration": object
	}	

Headers	Authorization	Bearer <token>
Success Response	Code	201; 401; 400
	Schema	-
Install a new plugin		

GET	/plugins/{id}	
Path params	id	Identifier of the installed plugin to inspect

The MES-CoBraD project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under grant agreement No 965422

Body	-	
Headers	Authorization	Bearer <token>
Success Response	Code	200; 401; 404

	Schema	{
		"id": "string",
		"name": "string",
		"version": "string",
		"official": true | false,
		"docs": "string",
		"configuration": object
		}

Get details of an installed plugin. The configuration field is a generic object that depends on the plugin.		

DELETE	/plugins/{id}	
Path params	id	Identifier of the installed plugin to inspect
Body	-	
Headers	Authorization	Bearer <token>
Success Response	Code	204; 401; 404
	Schema	-
Uninstall a plugin from the system		

GET	/workflows	
Path params	-	
Query params	limit	The numbers of items to return.
	Offset	The number of items to skip before starting to collect the result set.
	Order_by	The name of the field to order the results by. Prefix a field name with "-" to reverse the sort order.
Body	-	
Headers	Authorization	Bearer <token>
Success Response	Code	200; 401
	Schema	[{ "id": "string", "name": "string" }, ...]

The MES-CoBraD project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under grant agreement No 965422

List all the available workflows

POST	/workflows	
Path params	-	

Body	<pre>{
	“id”: “string”,
	“name”: “string”,
	“steps”: [{
	“plugin_id”: “string”,
	“configuration”: object
	}, …]
	}</pre>	

Headers	Authorization	Bearer <token>
Success Response	Code	201; 401; 400
	Schema	-
Create a new workflow.		

GET	/workflows/{id}	
Path params	id	Identifier of the workflow to inspect
Body	-	
Headers	Authorization	Bearer <token>
Success Response	Code	200; 401; 404

	Schema	<pre>{
		“id”: “string”,
		“name”: “string”,
		“steps”: [{
		“plugin_id”: “string”,
		“configuration”: object
		}, …]
		}</pre>

Get details of an available workflow. The configuration field is a generic object that depends on the plugin.		

The MES-CoBraD project has received funding from the European Union’s Horizon 2020 Research and Innovation Programme under grant agreement No 965422

DELETE	/workflows/{id}	
Path params	id	Identifier of the workflow to inspect
Body	-	
Headers	Authorization	Bearer <token>
Success Response	Code	204; 401; 404
	Schema	-
Delete a workflow.		

POST	/workflows/{id}/run	
Path params	id	Identifier of the workflow to launch
Body	*`Object` (workflow specific)*	
Headers	Authorization	Bearer <token>
Success Response	Code	200; 401; 404
	Schema	*`Object` (workflow specific)*
Trigger a workflow run via API. Both the input and the output are workflow specific.		

7.2 DATA COLLECTION COMPONENT

GET	/dags	
Path params	-	
Query params	limit	The numbers of items to return.
	Offset	The number of items to skip before starting to collect the result set.
	Order_by	The name of the field to order the results by. Prefix a field name with "-" to reverse the sort order.
	Tags	List of tags to filter results.
	Only_active	Only return active DAGs.
Body	-	
Headers	Authorization	Basic B64(username:password)
Success Response	Code	200; 401
	Schema	{ "dags": [{

The MES-CoBraD project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under grant agreement No 965422

<table>
<tr><td></td><td></td><td><pre>
"dag_id": "string",
"root_dag_id": "string",
"is_paused": true,
"is_active": true,
"is_subdag": true,
"fileloc": "string",
"file_token": "string",
"owners": [
  "string"
],
"description": "string",
"schedule_interval": {
  "__type": "string",
  "days": 0,
  "seconds": 0,
  "microseconds": 0
},
"tags": [
  {
    "name": "string"
  }
]
}
],
"total_entries": 0
}
</pre></td></tr>
<tr><td colspan="3">List all the DAGs</td></tr>
</table>

GET	/dags/{dagId}	
Path params	dagId	Id of the specified DAG
Query params	-	
Body	-	
Headers	Authorization	Basic B64(username:password)
	Code	200; 401;403;404

Success Response	Schema	<pre>{
		"dag_id": "string",
		"root_dag_id": "string",
		"is_paused": true,
		"is_active": true,
		"is_subdag": true,
		"fileloc": "string",
		"file_token": "string",
		"owners": [
		"string"
		],</pre>

		"description": "string",
		"schedule_interval": {
		"__type": "string",
		"days": 0,
		"seconds": 0,
		"microseconds": 0
		},
		"tags": [
		{
		"name": "string"
		}
		]
		}

Get basic information about the specified DAG		

GET	/dags/{dagId}/tasks	
Path params	dagId	Id of the specified DAG
Query params	order_by	The name of the field to order the results by. Prefix a field name with – to reverse the sort order.
Body	-	
Headers	Authorization	Basic B64(username:password)
	Code	200; 401;403;404

Success Response	Schema	{
		"tasks": [
		{
		"class_ref": {
		"module_path": "string",
		"class_name": "string"
		},
		"task_id": "string",
		"owner": "string",
		"start_date": "2019-08-24T14:15:22Z",
		"end_date": "2019-08-24T14:15:22Z",
		"trigger_rule": "all_success",
		"extra_links": [
		{
		"class_ref": {
		"module_path": "string",
		"class_name": "string"
		}
		}
		],
		"depends_on_past": true,
		"wait_for_downstream": true,
		"retries": 0,
		"queue": "string",

```
"pool": "string",
"pool_slots": 0,
"execution_timeout": {
  "__type": "string",
  "days": 0,
  "seconds": 0,
  "microseconds": 0
},
"retry_delay": {
  "__type": "string",
  "days": 0,
  "seconds": 0,
  "microseconds": 0
},
"retry_exponential_backoff": true,
"priority_weight": 0,
"weight_rule": "downstream",
"ui_color": "string",
"ui_fgcolor": "string",
"template_fields": [
  "string"
],
"sub_dag": {
  "dag_id": "string",
  "root_dag_id": "string",
  "is_paused": true,
  "is_active": true,
  "is_subdag": true,
  "fileloc": "string",
  "file_token": "string",
  "owners": [
    "string"
  ],
  "description": "string",
  "schedule_interval": {
    "__type": "string",
    "days": 0,
    "seconds": 0,
    "microseconds": 0
  },
  "tags": [
    {
      "name": "string"
    }
  ]
},
"downstream_task_ids": [
  "string"
]
```

		}] }
List all the tasks within the specified DAG		

GET	/dags/{dagId}/tasks/{taskId}	
Path params	dagId	Id of the specified DAG
	taskId	Id of the specific task
Query params	-	
Body	-	
Headers	Authorization	Basic B64(username:password)
	Code	200; 401;403;404
Success Response	Schema	{ "class_ref": { "module_path": "string", "class_name": "string" }, "task_id": "string", "owner": "string", "start_date": "2019-08-24T14:15:22Z", "end_date": "2019-08-24T14:15:22Z", "trigger_rule": "all_success", "extra_links": [{ "class_ref": { "module_path": "string", "class_name": "string" } }], "depends_on_past": true, "wait_for_downstream": true, "retries": 0, "queue": "string", "pool": "string", "pool_slots": 0, "execution_timeout": { "__type": "string", "days": 0, "seconds": 0, "microseconds": 0 }, "retry_delay": { "__type": "string", "days": 0,

```
    "seconds": 0,
    "microseconds": 0
  },
  "retry_exponential_backoff": true,
  "priority_weight": 0,
  "weight_rule": "downstream",
  "ui_color": "string",
  "ui_fgcolor": "string",
  "template_fields": [
    "string"
  ],
  "sub_dag": {
    "dag_id": "string",
    "root_dag_id": "string",
    "is_paused": true,
    "is_active": true,
    "is_subdag": true,
    "fileloc": "string",
    "file_token": "string",
    "owners": [
      "string"
    ],
    "description": "string",
    "schedule_interval": {
      "__type": "string",
      "days": 0,
      "seconds": 0,
      "microseconds": 0
    },
    "tags": [
      {
        "name": "string"
      }
    ]
  },
  "downstream_task_ids": [
    "string"
  ]
}
```


Get basic information about a specific task within the specified DAG

GET	/dags/{dagId}/dagRuns	
Path params	dagId	Id of the specified DAG
Query params	limit	The numbers of items to return.
	Offset	The number of items to skip before starting to collect the result set.

The MES-CoBraD project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under grant agreement No 965422

Body	Execution_date_gte	Returns objects greater or equal to the specified date. This can be combined with execution_date_lte parameter to receive only the selected period.
	Execution_date_lte	Returns objects less than or equal to the specified date. This can be combined with execution_date_gte parameter to receive only the selected period.
	Start_date_gte	Returns objects greater or equal the specified date. This can be combined with start_date_lte parameter to receive only the selected period.
	Start_date_lte	Returns objects less or equal the specified date. This can be combined with start_date_gte parameter to receive only the selected period.
	End_date_gte	Returns objects greater or equal the specified date. This can be combined with start_date_lte parameter to receive only the selected period.
	End_date_lte	Returns objects less than or equal to the specified date. This can be combined with start_date_gte parameter to receive only the selected period.
	Order_by	The name of the field to order the results by. Prefix a field name with "-" to reverse the sort order.
Headers	-	
	Authorization	Basic B64(username:password)
Success Response	Code	200; 401

	Schema	{
		"dag_runs": [
		{
		"dag_run_id": "string",
		"dag_id": "string",
		"logical_date": "2019-08-24T14:15:22Z",
		"execution_date": "2019-08-24T14:15:22Z",
		"start_date": "2019-08-24T14:15:22Z",
		"end_date": "2019-08-24T14:15:22Z",
		"state": "queued",
		"external_trigger": true,
		"conf": {}
		}
		],
		"total_entries": 0
		}

List all the runs for the specified DAG		

GET	/dags/{dagId}/dagRuns/{dagRunId}	
Path params	dagId	Id of the specified DAG

Query params	dagRunId	Id of the specific DAG run
	-	
Body	-	
Headers	Authorization	Basic B64(username:password)
	Code	200; 401;403;404
Success Response	Schema	{ "dag_run_id": "string", "dag_id": "string", "logical_date": "2019-08-24T14:15:22Z", "execution_date": "2019-08-24T14:15:22Z", "start_date": "2019-08-24T14:15:22Z", "end_date": "2019-08-24T14:15:22Z", "state": "queued", "external_trigger": true, "conf": {} }
List all the runs for the specified DAG		

POST	/dags/{dagId}/dagRuns	
Path params	dagId	Id of the specified DAG
Query params	-	
Body	{ "dag_run_id": "string", "logical_date": "2019-08-24T14:15:22Z", "execution_date": "2019-08-24T14:15:22Z", "state": "queued", "conf": {} }	
Headers	Authorization	Basic B64(username:password)
	Code	200; 400;401;403;404;409
Success Response	Schema	{ "dag_run_id": "string", "dag_id": "string", "logical_date": "2019-08-24T14:15:22Z", "execution_date": "2019-08-24T14:15:22Z", "start_date": "2019-08-24T14:15:22Z", "end_date": "2019-08-24T14:15:22Z", "state": "queued", "external_trigger": true, "conf": {}

The MES-CoBraD project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under grant agreement No 965422

		}
Trigger a new run for the specified DAG		

PATCH	/dags/{dagId}/dagRuns/{dagRunId}	
Path params	dagId	Id of the specified DAG
	dagRunId	Id of the specific DAG run
Query params	-	

Body	{
	"state": "success"
	}	

Headers	Authorization	Basic B64(username:password)
	Code	200; 400;401;403;404

Success Response	Schema	{
		"dag_run_id": "string",
		"dag_id": "string",
		"logical_date": "2019-08-24T14:15:22Z",
		"execution_date": "2019-08-24T14:15:22Z",
		"start_date": "2019-08-24T14:15:22Z",
		"end_date": "2019-08-24T14:15:22Z",
		"state": "queued",
		"external_trigger": true,
		"conf": {}
		}

Edit an existent DAG run		

DELETE	/dags/{dagId}/dagRuns/{dagRunId}	
Path params	dagId	Id of the specified DAG
	dagRunId	Id of the specific DAG run
Query params	-	
Body	-	
Headers	Authorization	Basic B64(username:password)
	Code	200;400;401;403;404

Success Response	Schema	{
		"type": "string",
		"title": "string",
		"status": 0,
		"detail": "string",
		"instance": "string"
		}

Delete an existent DAG run		

The MES-CoBraD project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under grant agreement No 965422

PATCH	/dags/{dagId}	
Path params	dagId	Id of the specified DAG
Query params	-	

Body	{
	“is_paused”: true
	}	

Headers	Authorization	Basic B64(username:password)
	Code	200;400;401;403;404

Success Response	Schema	{
		“type”: “string”,
		“title”: “string”,
		“status”: 0,
		“detail”: “string”,
		“instance”: “string”
		}

Update an existent DAG run		

DELETE	/dags/{dagId}	
Path params	dagId	Id of the specified DAG
Query params	-	
Body	-	
Headers	Authorization	Basic B64(username:password)
	Code	200;400;401;403;404

Success Response	Schema	{
		“type”: “string”,
		“title”: “string”,
		“status”: 0,
		“detail”: “string”,
		“instance”: “string”
		}

Delete an existent DAG run		

7.3 DATA LAKE

GET	/
Path params	-
Query params	-
Body	-

The MES-CoBraD project has received funding from the European Union’s Horizon 2020 Research and Innovation Programme under grant agreement No 965422

Headers	Authorization	HMAC Authorization-String
Success Response	Code	200
	Schema	<?xml version=”1.0” encoding=”UTF-8”?> <ListAllMyBucketsResult> <Buckets> <Bucket> <CreationDate>timestamp</CreationDate> <Name>string</Name> </Bucket> </Buckets> <Owner> <DisplayName>string</DisplayName> <ID>string</ID> </Owner> </ListAllMyBucketsResult>
List all the buckets		

PUT	/bucket_name	
Path params	-	
Query params	-	
Body	<CreateBucketConfiguration> <LocationConstraint>string</LocationConstraint> </CreateBucketConfiguration>	
Headers	Authorization	HMAC Authorization-String
Success Response	Code	200
	Schema	-
Create a *bucket_name* bucket		

DELETE	/bucket_name	
Path params	-	
Query params	-	
Body	-	
Headers	Authorization	HMAC Authorization-String
Success Response	Code	204
	Schema	-
Delete a *bucket_name* bucket		

The MES-CoBraD project has received funding from the European Union’s Horizon 2020 Research and Innovation Programme under grant agreement No 965422

PUT	/bucket_name	
Path params	-	
Query params	versioning	
Body	<VersioningConfiguration> <MfaDelete>Enabled|Disabled</MfaDelete> <Status>Enabled|Suspended</Status> </VersioningConfiguration>	
Headers	Authorization	HMAC Authorization-String
Success Response	Code	200
	Schema	-
Set versioning configuration for the specified bucket		

GET	/bucket_name/full_file_path	
Path params	-	
Query params	-	
Body	-	
Headers	Authorization	HMAC Authorization-String
Success Response	Code	200
	Schema	File content
Return the latest version of the selected file		

GET	/bucket_name	
Path params	-	
Query params	versions	
	prefix	File prefix
Body	-	
Headers	Authorization	HMAC Authorization-String
	Code	200
Success Response	Schema	<?xml version=”1.0” encoding=”UTF-8”?> <ListVersionsResult> <IsTruncated>72nteres</IsTruncated> <KeyMarker>string</KeyMarker> <VersionIdMarker>string</VersionIdMarker> <NextKeyMarker>string</NextKeyMarker> <NextVersionIdMarker>string</NextVersionIdMarker> <Version>

The MES-CoBraD project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under grant agreement No 965422

```
<ChecksumAlgorithm>string</ChecksumAlgorithm>
      ...
      <Etag>string</Etag>
      <IsLatest>73nteres</IsLatest>
      <Key>string</Key>
      <LastModified>timestamp</LastModified>
      <Owner>
         <DisplayName>string</DisplayName>
         <ID>string</ID>
      </Owner>
      <Size>integer</Size>
      <StorageClass>string</StorageClass>
      <VersionId>string</VersionId>
   </Version>
   ...
   <DeleteMarker>
      <IsLatest>73nteres</IsLatest>
      <Key>string</Key>
      <LastModified>timestamp</LastModified>
      <Owner>
         <DisplayName>string</DisplayName>
         <ID>string</ID>
      </Owner>
      <VersionId>string</VersionId>
   </DeleteMarker>
   ...
   <Name>string</Name>
   <Prefix>string</Prefix>
   <Delimiter>string</Delimiter>
   <MaxKeys>integer</MaxKeys>
   <CommonPrefixes>
      <Prefix>string</Prefix>
   </CommonPrefixes>
   ...
   <EncodingType>string</EncodingType>
</ListVersionsResult>
```

Return all the versions of the files that start with the provided prefix

GET	/bucket_name/full_file_path	
Path params	-	
Query params	versionId	version_id
Body	-	
Headers	Authorization	HMAC Authorization-String

Success Response	Code	200
	Schema	File content
Return the specific version of the selected file		

GET	/bucket_name/full_file_path	
Path params	-	
Query params	attributes	
Body	-	
Headers	Authorization	HMAC Authorization-String
Success Response	Code	200
	Schema	<GetObjectAttributesOutput> <Etag>string</Etag> <Checksum> <ChecksumCRC32>string</ChecksumCRC32> <ChecksumCRC32C>string</ChecksumCRC32C> <ChecksumSHA1>string</ChecksumSHA1> <ChecksumSHA256>string</ChecksumSHA256> </Checksum> <ObjectParts> <IsTruncated>74nteres</IsTruncated> <MaxParts>integer</MaxParts> <NextPartNumberMarker>integer</NextPartNumberMarker> <PartNumberMarker>integer</PartNumberMarker> <Part> <ChecksumCRC32>string</ChecksumCRC32> <ChecksumCRC32C>string</ChecksumCRC32C> <ChecksumSHA1>string</ChecksumSHA1> <ChecksumSHA256>string</ChecksumSHA256> <PartNumber>integer</PartNumber> <Size>integer</Size> </Part> ... <PartsCount>integer</PartsCount> </ObjectParts> <StorageClass>string</StorageClass> <ObjectSize>long</ObjectSize> </GetObjectAttributesOutput>
Return all the metadata associated with the selected file without returning it		

PUT	/bucket_name/full_file_path
Path params	-

Query params	-	
Body	File content	
Headers	Authorization	HMAC Authorization-String
	..Metadata	Metadata value
Success Response	Code	200
	Schema	Valid XML
Put a file within the selected bucket with *full_file_path* name. If for the specified bucket the versioning is enabled than the object storage will generate a new version id every time the same file is put within the same bucket.		

DELETE	/bucket_name/full_file_path	
Path params	-	
Query params	-	
Body	-	
Headers	Authorization	HMAC Authorization-String
Success Response	Code	200
	Schema	Valid XML
Delete a file within the selected bucket with *full_file_path* name		

DELETE	/bucket_name/full_file_path	
Path params	-	
Query params	versionId	version_id
Body	-	
Headers	Authorization	HMAC Authorization-String
Success Response	Code	200
	Schema	Valid XML
Delete the specific version of the selected file		

7.4 DATA HARMONISATION COMPONENT

POST	/api/{tenantId}/projects/{projectId}/versions
Path params	tenantId, projectId
Query params	-
Body	{ "name": "string", "project": "string", "description": "string",

The MES-CoBraD project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under grant agreement No 965422

<table>
<tr><td rowspan="5">Headers

Success Response</td><td colspan="2"><pre>
"inputs": [
  {
    "name": "string",
    "alias": "string",
    "type": "string",
    "fixed": false,
    "defaultValue": {}
  }
],
"dependencies": {},
"outputSchema": {},
"released": false,
"createdAt": "string",
"updatedAt": "string",
"releasedAt": "string"
}
</pre></td></tr>
<tr><td>Authorization</td><td>Bearer token to authenticate and authorize the requestor</td></tr>
<tr><td>Content-Type</td><td>application/json</td></tr>
<tr><td>Code</td><td>200</td></tr>
<tr><td>Schema</td><td>Success</td></tr>
<tr><td colspan="3">Create a new mashup</td></tr>
</table>

<table>
<tr><td>GET</td><td colspan="2">/api/{tenantId}/projects/{projectId}/versions/{versionId}/draw</td></tr>
<tr><td>Path params</td><td colspan="2">tenantId, projectId, versionId</td></tr>
<tr><td>Query params</td><td colspan="2">-</td></tr>
<tr><td>Body</td><td colspan="2">-</td></tr>
<tr><td rowspan="3">Headers

Success Response</td><td>Authorization</td><td>Bearer token to authenticate and authorize the requestor</td></tr>
<tr><td>Code</td><td>200</td></tr>
<tr><td>Schema</td><td>{}</td></tr>
<tr><td colspan="3">Get mashup graphical flow</td></tr>
</table>

<table>
<tr><td>POST</td><td colspan="2">/api/{tenantId}/projects/{projectId}/versions/{versionId}/draw</td></tr>
<tr><td>Path params</td><td colspan="2">tenantId, projectId, versionId</td></tr>
<tr><td>Query params</td><td colspan="2">-</td></tr>
<tr><td>Body</td><td colspan="2">-</td></tr>
<tr><td rowspan="3">Headers

Success Response</td><td>Authorization</td><td>Bearer token to authenticate and authorize the requestor</td></tr>
<tr><td>Code</td><td>200</td></tr>
<tr><td>Schema</td><td>Success</td></tr>
<tr><td colspan="3">Update mashup graphical flow</td></tr>
</table>

The MES-CoBraD project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under grant agreement No 965422

POST	/api/{tenantId}/projects/{projectId}/versions/{versionId}/run	
Path params	tenantId, projectId, versionId	
Query params	-	

Body	{
	"inputs": {},
	"target": "string"
	}	

Headers	Authorization	Bearer token to authenticate and authorize the requestor
	Content-Type	application/json
Success Response	Code	200
	Schema	Success
Execute the mashup from the editor		

POST	/mashups/{execflowId}	
Path params	execflowId	
Query params	-	
Body	-	
Headers	Authorization	Bearer token to authenticate and authorize the requestor
Success Response	Code	200
	Schema	Success
Trigger a mashup using HTTP request		

7.5 API GATEWAY

OpenAPI specification (OAS) is a standard interface for presenting RESTful APIs in a manner that allows both humans and computers to easily understand the logic behind the services.

By using document generation tools, it is possible to automatize the creation of the API display page.

Main definitions of the OpenAPI include one or more OpenAPI documents that describe the APIs, path templates, media types, as well as http status codes.

By path templating, it is referred to the template expressions, most often represented by curly braces "{}", which serve to point out that this part of the URL can be replaced with a path parameter.

Media types definitions should be in accordance with the RFC6838.

To indicate the response status of the executed request, HTTP status codes are used and the available ones are all defined in RFC7231.

In regards to versioning, OpenAPI is using Semantic Versioning 2.0.0 (semver).

By format, OpenAPI document is a .json or a .yaml file which contains the following information: OpenAPI version, API metadata, server information, path definition, security configuration etc. It is also possible to extend this list of fields with custom properties [6].

The MES-CoBraD project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under grant agreement No 965422

8 IMPLEMENTATION

8.1 EDGE MODULE

The edge module is a key point for the entire MES-CoBraD platform. It ensures the compliance with the GDPR and gives the scientists the possibility to exploit different plugins, also provided by third parties to further manipulate the patients' data.

The plugin mechanism is the focal point of the overall edge module implementation since it gives the possibility to scale the functionalities as desired and required by the specific scientists' needs.

The most suitable technology to implement such a mechanism is python that allows to extend the capabilities of the system at runtime.

Python is also a good choice from the performance perspective because ensures to keep the system as lightweight as possible without the overhead of languages like Java, Javascript and others.

In particular, the Plugin Manager will be implemented exploiting the capabilities of the **pip** package manager that will allow to install packages (i.e. plugins) at runtime.

For the workflow manager the library **workflow** will be used that provides a method of running Finite State Machines with memory.

Several python libraries are available for the implementation of a resource negotiator, but the chosen one is **multiprocessing** that allows to spawn processes in parallel according to a predefined pool of resources that can be configured dynamically basing on the system resource availability.

For the API implementation in python, the most spread framework is **Flask**. It ensures to be able to create RESTful APIs in easy and safe way.

The following picture shows the association between each architectural component and the corresponding technical implementation.

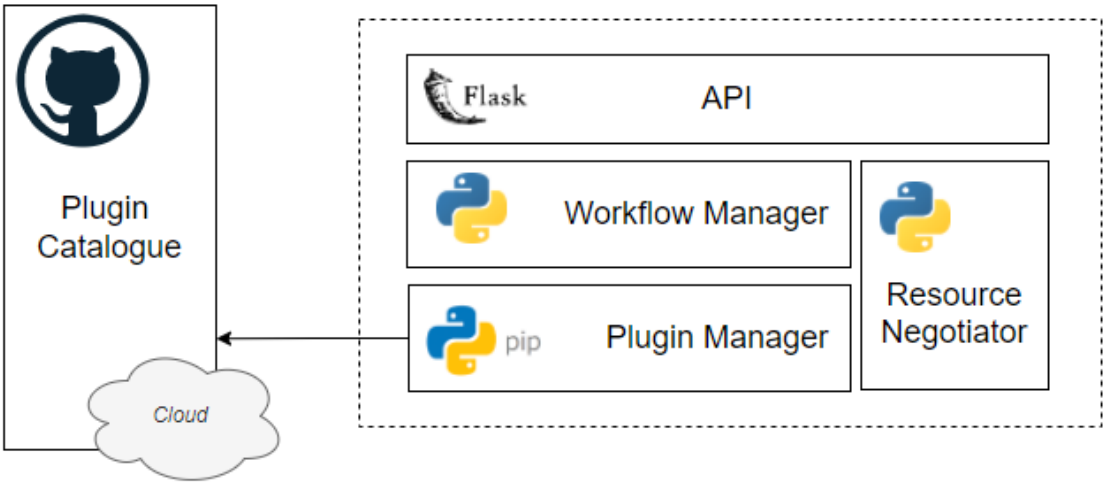

Figure 28 - Technical implementation of the edge module

The plugin catalogue is the only component of the edge module that is deployed on the cloud and it will be implemented through the **Github** MES-CoBraD community.

8.2 ANONYMISATION MODULE

Sensitive information is the most critical part of a medical platform and in general of an IT system. Several privacy and security-oriented practices will be implemented within the scope of the MES-CoBraD project.

The MES-CoBraD project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under grant agreement No 965422

One of the most important of those is the patients' data anonymisation. This will imply to keep sensitive information on the edge of the platform, by giving the possibility to recover original data (with the patients' information) only to the original submitters.

Within the project the anonymisation is intended as an iterative process. Where submitters, once data are processed by the anonymisation module, are manually verified before to being uploaded within the platform. Data that fails the manual check must be re-processed by the module by changing its configurations. Indeed, the module allows to configure several parameters such us input columns, sensitive columns, quasi-identifiers and to fine-tune the anonymisation algorithm.

The module in charge of such anonymisation process is the anonymisation module. This component, available as a plugin for the edge module, provides configurable de-personification functionalities that can be used together with the other plugins to compose data pipelines that run on the edge. As of other plugins, the anonymisation module is simply installable within the edge module at runtime.

Such module will be implemented in Python due to the versatility in handling files in different formats, the dynamic typing and the easiness to read and maintain.

Potential approaches to de-personalise patients' data could be:

- Data masking
- K Anonymity
- L Diversity
- T Closeness
- Differential Privacy models

There are several open-source python libraries that implement such algorithms, like:

- Faker [8]
- Mimesis [9]
- PyARX [10]
- PyDP [11]

8.3 DATA COLLECTION COMPONENT

Scattered data coming from different data silos needs to be ingested and processed to extract value from them and to allow analytical tools and artificial intelligence components to effectively make advantage from such data. The data collection component represents the focal point of all the ingestion and harmonisation operations, by allowing end-users to define, maintain and monitoring efficient data pipelines that can scale horizontally on the underlying hardware. Such component will be implemented by using Apache Airflow, an open-source modular workflow orchestrator that allows to schedule data pipeline defined using a low-code approach.
Airflow provides the right amount of extendibility and flexibility thanks to a vast catalogue of operators that can enrich the developer's toolkit during the pipeline design phase. Such catalogue can be easily extended with customs operators that better suit the aim of the project.
The high modularity of Airflow fits with the MES-CoBraD project requirements and make advantage from the cloud native deployment approach used for the whole platform.
Moreover, it comes with a built-in UI that allows end-users to easily monitor workflow executions, restart them partially, define new connections, configuration and check tasks logs. Such logs are only one of the tools that Airflow provides to debug workflows and tasks. Indeed, the UI can show meaningful visualisations to check the execution times at different grains and to compare executions over time.

The MES-CoBraD project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under grant agreement No 965422

The key concept in Airflow is the DAG (Directed Acyclic Graph). As mentioned in the section 5.2.3 a DAG represents a workflow composed by tasks. Each task is an instance of an operator, a function that focuses on an atomic operation that involves one or more data sources. The operators to interact with the most common external systems like relational databases, http endpoints and ftp servers. The catalogue of the operators can be enriched by installing extensions coming from official vendors (e.g., Amazon, Google, Microsoft) or by adding custom operators.
DAGs can be executed manually via API or by using the user interface. This functionality is particularly useful for workflows whose aim is to clean environments, or to restore the system to a specific snapshot. It is also useful to debug non-working DAGs by isolating the tasks and by running them in other environments always within the component. Moreover, DAGs can be automatically scheduled by specifying a CRON expression within the DAG file. Airflow supports also concurrent executions on the same DAG, where each of these can work with subset of data or with specific data sources.
The scheduler is the architectural component that spin-up the tasks using an executor. Once a task is scheduled a process is spawn to monitor task status. Once completed the scheduler checks task's dependencies and spawn next tasks.
Executor's goal is to effectively execute a task instance. Airflow supports different executors; the main ones are listed below:

- Debug Executor – This executor allows to debug DAGs within IDEs. It simply runs processes inside the host machine one-by-one, without the need of a dedicated metadata database to store executions data.
- Local Executor – Such executor can spawn concurrent processes within the host machine. It is the most used executor for single-machine installations. It required a dedicated metadata database (e.g., postgres) to enable concurrency.
- Remote Executor(s) – Airflow can run on multiple-machines by using remote executors. Such architectural components load balance process across the available machines to efficiently use the underlying hardware.

Within the MES-CoBraD project airflow will be configured to use Kubernetes executor. Such executor allows to spawn Pods (that reflect the task instances within a DAG execution) in a Kubernetes environment. The Digital Enabler, a Kubernetes-based ecosystem platform, offers Apache Airflow instances already integrated with Kubernetes by allowing end-users to just upload their DAGs and defining the connection without worrying about configurations and performance optimisation.

Airflow provides a set of tools to effectively monitor DAGs executions and to compare them each other. Out-of-the box an end-user can check a DAG execution progress by visually following the specific tasks execution. Moreover, the UI shows for each task details like technical description, input parameters and real time logs. DAG execution times can be monitored from the UI as well. Indeed, the UI has a specific section that contains tasks' execution time and display such information in a timeline. Here end-users can identify bottleneck, and focus optimisation activities to specific tasks.

Apache Airflow makes the connection with external data sources really easy. Within Airflow a connection defines the technical details to connect with such external systems. By choosing the connection type, and filling the endpoint and (eventually) the credentials the tool is able to connect with such external system without make users worry about connection handling. Apache Airflow comes with a catalogue of already installed connections. Such catalogue can be simply extended by installing additional connections, thanks to the plug-and-play nature of Python.

Moreover, Airflow allows to define custom pools that can be assign to group of tasks to limit the concurrency and better handling the execution flows. Pools can be created directly using the UI by simply define a pool name and the slots number. Then, within a task definition (in the DAG file) it will be possible to assign a pool let airflow to automatically increase/decrease the available slots within a pool during

The MES-CoBraD project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under grant agreement No 965422

task executions.

DAG can be generated dynamically based on pre-defined variables. Such variables can be modified in any moment using the UI and can change or parametrize DAG executions. Variables can be simple texts or JSONs that will be automatically parsed by the tasks within a DAG. Variables can be simply added from the UI by defining a name and the value. If the configuration name contains special words (configurable) like password or secret then the configuration becomes not visible from the UI and encrypted backend-side. Only the tasks can access secrets, by making sensitive information securely stored and not visible with tasks logs.

The following picture shows the association between each component of Airflow's architecture and the corresponding technical implementation:

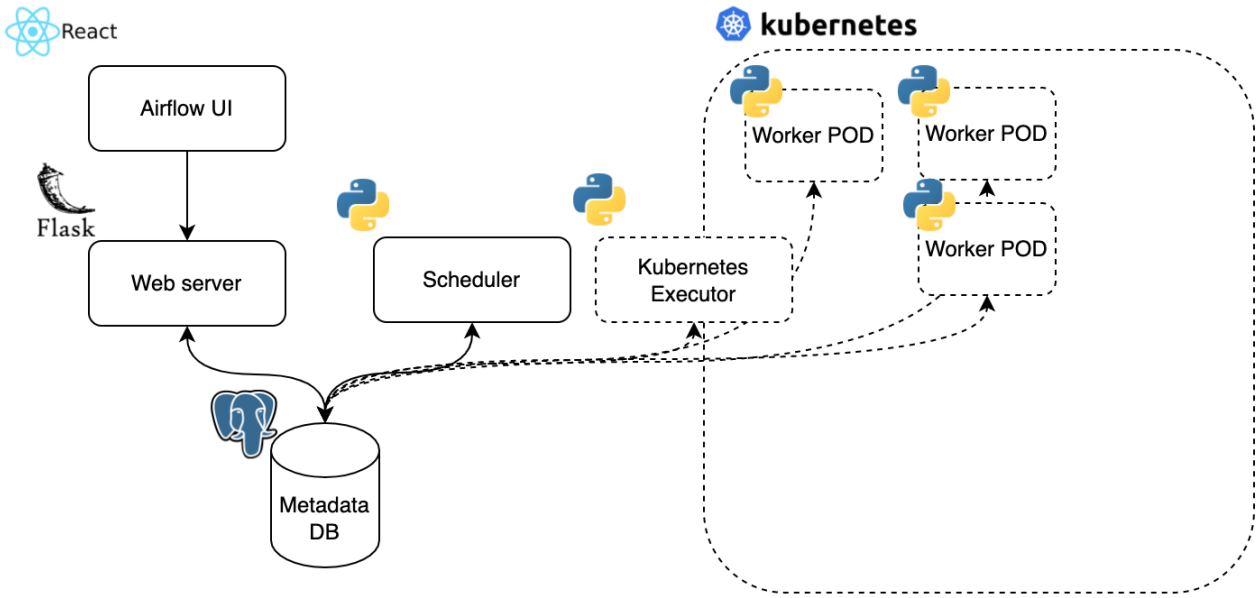

Figure 29 - Data collection component implementation details (Airflow based)

8.4 DATA LAKE

The real-world data (RWD) are the hearth of the MES-CoBraD platform. Such data must be stored efficiently and in a secure and anonymous way to be harmonised with other data source and to be analysed by end-users and AI components. The platform's data lake will be composed by several data storage solutions in order to accomplish the different requirements that comes from the kind of data the platform aim to store (real-time data, images, text files, etc...).

Anonymous data, medical images and signals will be store inside the platform's object storage. This component will be implemented by using MinIO, an open-source solution provided by the Digital Enabler ecosystem platform. MinIO is a high-performance multi-cloud object storage that runs on any Kubernetes distribution, compatible with S3 object storage. This ensures interoperability with other data lake solutions and allows other components to use the same libraries for all the data-related operations.

High availability is granted thanks to the cloud nature of the component. In contrast to other solutions, it is easy to scale the tool and to set-up data replication. Moreover, MinIO supports data encryption at-rest and versioning, providing out-of-the-box a high level of security and reliability.

As previously mentioned, MinIO is 100% compliant with Amazon S3 APIs. This means that it is possible to use AWS SDK to interact with the component, without having to deal with the complex REST APIs that

The MES-CoBraD project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under grant agreement No 965422

are available too for expert users. Moreover, MinIO comes with MinIO Console, a simple UI that allows end-users to visually consult the object storage, upload resources and share them.

Within MinIO data are collected inside "buckets", containers that are associated with a unique namespace. Buckets can be associated with departments or group of users and are accessible by granting specific permissions.

MinIO support versioning at bucket level. Once the versioning is enabled for a specific bucket, its objects are associated with a versionID. Whenever a user uploads an object with an existent name, MinIO generate a new version without removing the previous one. This ensures to being always able to retrieve old versions of the same file, avoiding overwriting existing file by mistake.

Resources within the object storage can be shared in two ways:

- Public links
- Ad-hoc permissions

Public links can be generated for specific resources, granting access to whoever receive the link. Such link contains special tokens to let the user without a MES-CoBraD account to access only that specific resource. Despite it can be dangerous to expose data lake's resources, this functionality could be helpful to share public documents, open data or to temporarily giving access to specific files. Indeed, public links can be generated giving an expiration date interval, to make a resource available publicly for a limited period of time.

Resource can be accessed only if the user's policy grants the access to it. Within MinIO each user has associated one or more access policy that specify which operations the user is authorized to perform. Such permissions can be defined at different grains (bucket, folder, file). As a consequence, a file can be shared by creating an ad-hoc permission that grant access only to the specific resource.

As mentioned, elements can be accessed through the UI or programmatically by using an SDK or the MinIO's REST API. To access the object storage programmatically a pair of Client-key and Client-secret must be generated. Service accounts are basically key/secret pairs that can be generated by end-users by using the UI. Once generated, such credentials can be used within a script or a software to interact with the object storage.

Service accounts by default are generated inheriting user's permissions. Moreover, the UI allows to to define custom policies to include subsets of user's permissions for granting access to specific resources.

The following picture shows the association between each component of MinIO's architecture and the corresponding technical implementation:

The MES-CoBraD project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under grant agreement No 965422

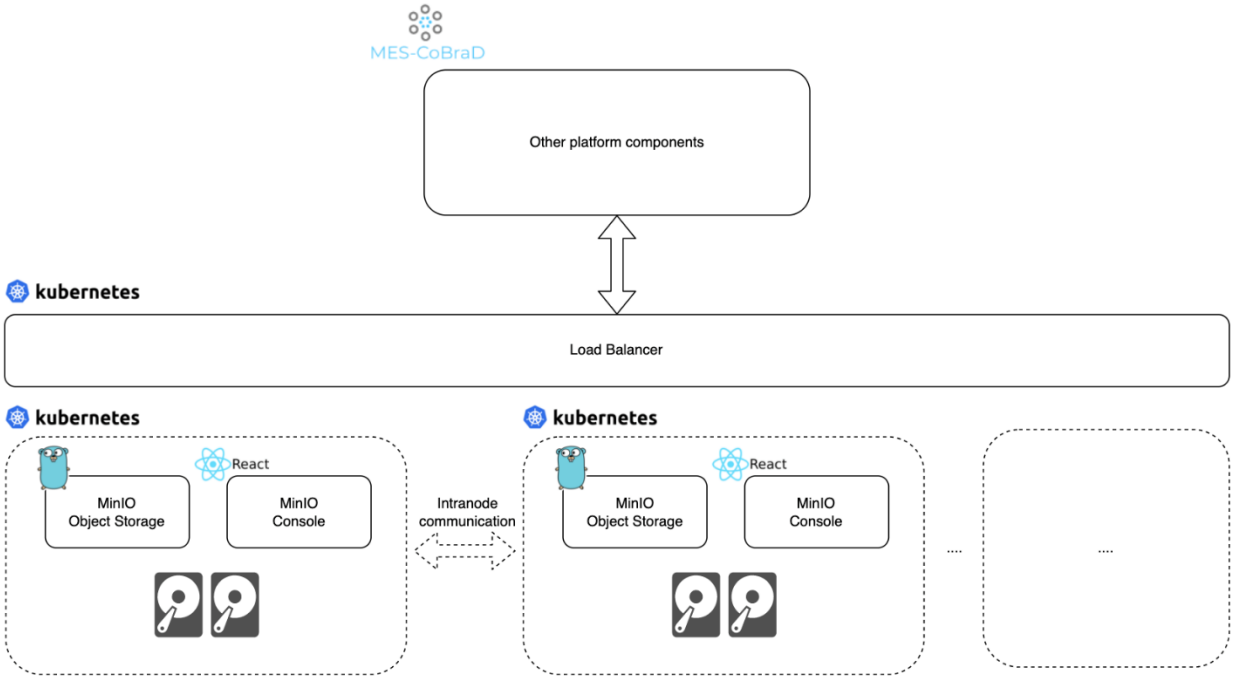

Figure 30 - Data lake object storage implementation details (MinIO based)

The MES-CoBraD data lake includes also other storage solutions to persist real-time data, unstructured data and to run analytical query:

- The platform will use a relational database (postgres) to store end-users' information, metadata and other data necessary to make the platform properly working.
- The data lake will include an OLAP database (postgres) to let analytical tools and expert to perform fast queries and generate insights and reports.
- Time related data will be stored within an InfluxDB database, a storage solution optimised for time series and analytics.
- Unstructured data will be stored in a MongoDB server, ensuring to have a solution change-friendly thanks to its flexible schemas.
- MES-CoBraD components will stream system logs to a log server (Graylog). It acts as a central point to analyse users' behaviour and to debug platform's components.

8.5 DATA HARMONISATION COMPONENT

To achieve the harmonisation process, Data Mashup Editor (DME) component of Digital Enabler will be used.

There are two ways to communicate with DME. One includes synchronous triggers via HTTP request through REST APIs, while the other is asynchronous and is done through different protocols such as MQTT, Kafka, etc.

A process which involves a HTTP request to DME and as a result returns a transformed result is called a mashup.

Mashup can be forwarded to other components like data bases, data lakes or context brokers and can be scheduled to run the same flow periodically.

The MES-CoBraD project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under grant agreement No 965422

DME has a graphical user interface which is a key point for creating mashups.

Before creating a mashup, the user should create a dataset or a data source which will be the starting point of the mashup.

After that, the user can use the graphical UI to transform the data using many different operators, including predefined data models, which can destructure data, add new properties or create new object structures.

If the operators are applied to an array of objects, the operator will be applied for each of them, which is very useful when working with the large amount of data.

It is very important to emphasize that no coding experience is needed to transform the data. All the operators can be used in a drag and drop manner.

Each mashup's logs are tracked and can be reviewed after each execution.

In cases where the original data source for the mashup is continually changed, it is possible to trigger the harmonisation process over and over to receive transformed data from the new data. This can be done from multiple sources, including HTTP requests, MQTT messages and Kafka messages.

The mashup result will be saved on the target chosen by user, which may include HTTP endpoint, MQTT or Kafka topic and Fiware Orion Context Broker.

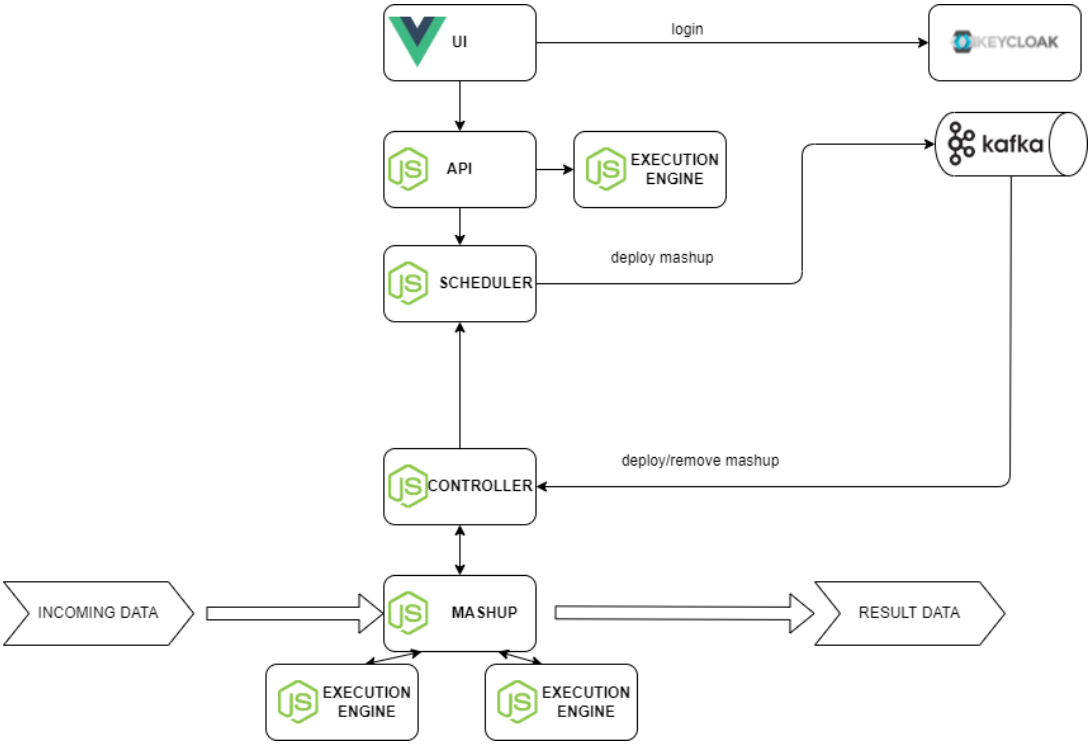

Figure 31 - Data harmonisation tool implementation details (Data Mashup Editor based)

8.6 API GATEWAY

It was decided to use Kong [7] as an API gateway because it is an open-source solution with many rich features that include load balancing, Kubernetes support, scalability and customizability, but it also has something that other tools do not – a graphical user interface.

Kong's graphical user interface is called Kong Manager. It can be used to create routes, add or remove plugins or monitor performance.

Kong can run natively on Kubernetes platforms, providing a single point of access for all the incoming

The MES-CoBraD project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under grant agreement No 965422

requests to a Kubernetes cluster.

For what concern the authentication, it is able to protect the APIs, as well as enable anonymous access to selected users.

Also, if the application is unavailable, Kong is able to reroute the requests using load balancing.

The MES-CoBraD project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under grant agreement No 965422

9 INTEGRATION

In this chapter we will encompass all mentioned components and explain in what way they are coupled together.

9.1 INTEGRATION BETWEEN COMPONENTS

In order to better understand the architecture, we must explain how the components are intertwined.

The architecture is divided into multiple layers. The first layer is comprised of data sources, in this case, medical data.

Medical data will be gathered from medical data centers or from medical devices. However, there are two steps which must be taken before sending that data to the Data Lake.

Since most medical data centers are not connected to the outside world, primarily to protect sensitive information, the first step is to upload the data to the edge module, located inside the medical center. Once the data is in the edge module, the second step is to anonymise it.

There are two ways of introducing data to the edge module:

- synchronous data provisioning (using REST APIs)
- scheduled data retrieval (through periodic querying to other medical local systems)

Synchronous data provisioning will be provided by the medical staff who will use the edge module autonomously, collect data from the patients and upload it using a specific form. They will also be responsible to check if the anonymisation has been performed successfully.

Scheduled data retrieval will be performed using workflows defined by medical staff. The workflows will gather data from medical devices periodically and send it to the edge module.

The edge module will then perform data anonymisation through the anonymisation module, which will be added as a plugin, and forward the data to the MES-CoBraD platform.

All steps from the ingestion of the data into the edge module to the transmission of the anonymised data belong to the second layer, security and anonymisation.

Once the data is secure and anonymised, they are sent to the third layer, the Data Lake, using the Data collection component. The data ingestion process can be either synchronous or asynchronous.

Synchronous ingestion should be triggered directly from the edge module using authenticated REST APIs with the data to be stored contained in the payload of the requests.

Asynchronous ingestion should be triggered using scheduled data processes that are able to retrieve data from other data sources.

Data Lake is the component where all the raw, harmonised and curated data will be safely stored and from which it will be shared to the other components. For the sharing of the data among components, both synchronous and asynchronous integration patterns are available.

For data transmission between the Data Lake and the Data Harmonisation component, it is planned to integrate a communication bus, which would be based on Apache Kafka, and a synchronous approach which would be performed through the REST APIs.

The MES-CoBraD project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under grant agreement No 965422

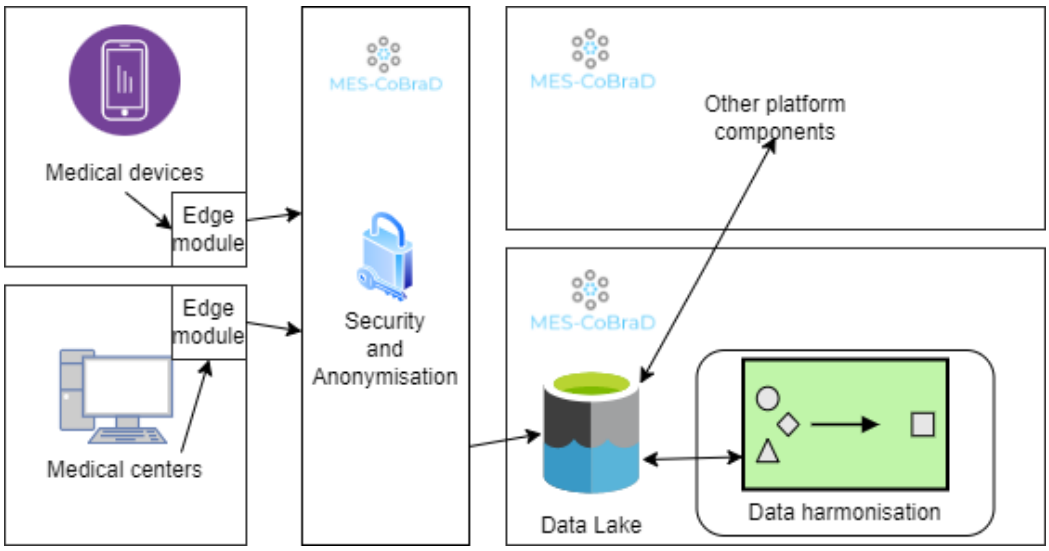

Figure 32 - Integration between the components

The integration between the Data Lake and the other MES-CoBraD platform components is subject of discussion of the next section.

9.2 INTEGRATION WITH THE OTHER MES-COBRAD PLATFORM COMPONENTS

The overall architecture is imagined as a multi-layer component system based on orchestrated Kubernetes containers in the same network.

The base of the infrastructure is presented by the MES-CoBraD platform, which consists of Hardware platform and Software platform.

The hardware platform is comprised by seven Virtual Machines hosted by VMWare, while the software platform is running on Ubuntu operating system on a virtual machine.

All the components shall use the Data Lake as a persistence layer. It is a multi-technology storage layer capable of storing different types of data, including object type storage, relational and non-relational databases, as well as time series and log server. Using this approach, Data Lake will be usable for communication between all internal components.

The data ingestion will be done through the Dataflow component of Digital Enabler based on Apache Airflow. It supports both synchronous and asynchronous ingestion through REST APIs and scheduled batch processes, respectively.

The other platform components that will be integrated with the Data Lake are mainly, but not exclusively, the Expert System's modules: i.e. the Query Builder, Data Analytics, Data Visualisation.

The MES-CoBraD project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under grant agreement No 965422

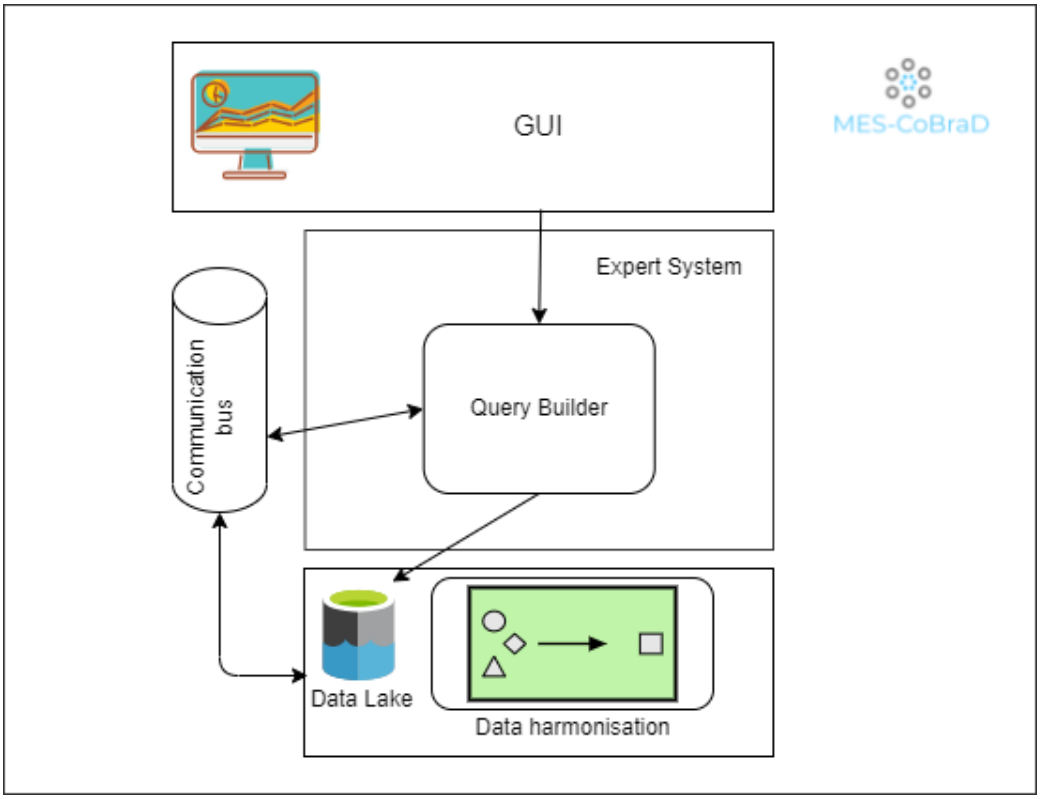

Figure 33 - Integration with other MES-CoBraD components

The MES-CoBraD project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under grant agreement No 965422

10 TESTING

In this chapter will be defined a set of user acceptance test (UAT) for each of the components discussed in this deliverable. The UAT are described according to the templated in ANNEX I of the D8.2.

10.1 EDGE MODULE

Table 6 - Edge module UAT plan

Test Case ID	Test Case Name	Summary	Expected UX	Remarks
EDG_UAT_01	Install plugin	The tester will install a plugin already available in the plugin catalogue.	1. The UI shows the list of the plugins available in the catalogue. 2. The user choose a plugin to install and click on it. 3. A configuration form (specific for the selected plugin) pops up. 4. The user submits the configuration and starts the installation. 5. When the installation ends the just selected plugin is available in the installed plugins list	-
EDG_UAT_02	Delete plugin	The tester will delete a plugin already installed.	1. The UI shows the list of the plugins installed on the instance 2. The user chooses a plugin and click on the “delete” icon 3. A confirmation prompt pops up and the user confirms 4. The plugin is removed from the system	The workflows containing the just removed plugin are labelled as “*unstable*”
EDG_UAT_03	Create a workflow	The tester will create a workflow with two steps	1. The user creates a new workflow from the dedicated form on the UI 2. The user chooses the following steps: ☐ Data manual submission ◎ Data de-personalization ☐ Data aggregation 3. When the user submits the workflow, it appears on the workflows list	-
EDG_UAT_04	Trigger a workflow	The tester will trigger a workflow through REST Api	1. The user invokes a workflow through the dedicated API 2. The user checks the response body and the status code (200)	-

The MES-CoBraD project has received funding from the European Union’s Horizon 2020 Research and Innovation Programme under grant agreement No 965422

10.2 ANONYMISATION MODULE

Table 7 - Anonymisation module UAT plan

Test Case ID	Test Case Name	Summary	Expected UX	Remarks
AM_UAT_01	Anonymisation process creation	The tester will upload a csv containing personal data to obtain a reference to the de-personalisation process	1. The UI prompts a form where the user can submit a file 2. The user submits a file with the following three columns: • Name • Surname • exam_result 3. The user selects the chosen de-personalisation protocol (pseudo-anonymisation) and submits the form 4. The UI confirm the upload of the file and gives back a link to the page where the anonymised data will be available.	At least a workflow with the anonymisation plugin must be available.
AM_UAT_02	Data pseudo-anonymisation	The tester will check that the pseudo anonymisation has been executed correctly.	1. The user periodically checks the link obtained during AM_UAT_01 2. After some time, a new csv with the following structure is produced: • patientid • exam_result	The patientid must be unique for the same name, surname couple.

10.3 DATA COLLECTION COMPONENT

Table 8 - Data collection component UAT plan

The MES-CoBraD project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under grant agreement No 965422

Test Case ID	Test Case Name	Summary	Expected UX	Remarks
DCO_UAT_01	DAG visualisation	The tester will access the tool to consult the Data pipelines deployed within the component	1. The user log-in within the component 2. The UI shows the list of available DAGs	-
DCO_UAT_02	Trigger DAG	The tester will trigger an existent DAG to check if it run correctly	1. The user log-in within the component 2. The UI shows the list of available DAGs 3. The user clicks on the 'play' icon next to the DAG of interest 4. (Optional) the user inserts configurations for the DAG execution	The tasks inside the selected DAG must start being scheduled (their contour become light green)
DCO_UAT_03	Disable DAG	The tester will disable a DAG to stop the scheduling of the tasks the DAG is composed by	1. The user log-in within the component 2. The UI shows the list of available DAGs 3. The user clicks on the switch next to the DAG of interest	Tasks must not be scheduled after disabling DAG
DCO_UAT_04	DAG logs	The tester will check the log of a specific DAG to ensure it has been executed correctly	1. The user log-in within the component 2. The UI shows the list of available DAGs 3. The user clicks on the DAG of interest 4. The user clicks on one of the tasks 5. The user selects the log option	-
DCO_UAT_05	New connection	The tester will register a new connection within the component	1. The user log-in within the component 2. The UI shows the list of available DAGs 3. The user clicks on the connection section 4. The user clicks on "new connection" 5. The user defines a new connection by specifying the type of connection, endpoint and credentials 6. The user submits the form	Connection endpoint must be reachable by the component

The MES-CoBraD project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under grant agreement No 965422

DCO_UAT_06	New configuration	The tester will add a new configuration within the component	1. The user log-in within the component 2. The UI shows the list of available DAGs 3. The user clicks on the configuration section 4. The user clicks on “new configuration” 5. The user defines a “new connection by defining the name and the value 6. The user submits the form	A new row must be available in the configuration list
DCO_UAT_07	New secret	The tester will add a new secret within the component	1. The user log-in within the component 2. The UI shows the list of available DAGs 3. The user clicks on the configuration section 4. The user clicks on “new configuration” 5. The user defines a new connection by defining the name and the value 6. The user submits the form	Secret name must contain the words “secret” or “password”; Secret value must not be visible in the UI

The MES-CoBraD project has received funding from the European Union’s Horizon 2020 Research and Innovation Programme under grant agreement No 965422

10.4 DATA LAKE

Table 9 - Data Lake UAT plan

Test Case ID	Test Case Name	Summary	Expected UX	Remarks
DLK_UAT_01	Bucket creation	The tester will create a new bucket	1. The user log-in within the component 2. The UI shows the list of available buckets 3. The user clicks on “new bucket” 4. The user type-in new bucket’s name	Bucket name must be unique
DLK_UAT_02	Folder creation	The tester will create a folder within a bucket	1. The user log-in within the component 2. The UI shows the list of available buckets 3. The user clicks on the bucket of 93nterest 4. The user clicks on “New folder”	Folder name must be unique within the same folder; Folder must contain at least one file to exist
DLK_UAT_03	File upload	The tester will upload a file within a bucket	1. The user log-in within the component 2. The UI shows the list of available buckets 3. The user clicks on the bucket of interest 4. The user clicks on “New file” 5. The user selects a file from his PC	File name must be unique within the same folder
DLK_UAT_04	Resource deletion	The tester will delete a resource within a bucket	1. The user log-in within the component 2. The UI shows the list of available buckets 3. The user clicks on the bucket of interest 4. The user clicks on the “X” icon next to the file of interest	If a folder is deleted then also its comment is deleted as well

The MES-CoBraD project has received funding from the European Union’s Horizon 2020 Research and Innovation Programme under grant agreement No 965422

DLK_UAT_05	Service account creation	The tester will create a service account for the programmatic access of the object storage	1. The user log-in within the component 2. The UI shows the list of available buckets 3. The user clicks on the Service Account section 4. The user clicks on “new service account” 5. The user defines service account permissions 6. The user submits the form	Once create service accounts must not be modified, only deleted; Service accounts must ensure the programmatic access with the specified permissions; Service account permissions must be a subset of the user’s permissions

10.5 DATA HARMONISATION COMPONENT

Table 10 - Data harmonisation component UAT plan

Test Case ID	Test Case Name	Summary	Expected UX	Remarks
DHA_UAT_01	Dataset adding	The tester will add a dataset which will be used for creating a mashup.	1. The user logs in to the platform 2. The user opens the Data Mashup Editor component 3. The user chooses datasets tab 4. The UI shows the list of available datasets and the option to create a new dataset 5. The user clicks on a button to create a new dataset and a new window appears 6. The user chooses the properties and saves the dataset 7. The user can see the new dataset in the dataset list	
DHA_UAT_02	Mashup creation	The tester will create a mashup using the provided UI.	1. The user logs in to the platform 2. The user opens the Data Mashup Editor component 3. The user chooses mashups tab 4. The UI shows the list of mashups and the option to create a new mashup 5. The user clicks on a button to create a new mashup and a new window appears 6. The user chooses a starting dataset to get data from 7. The user chooses operators from the tab on the left and drags and drops them to the mashup grid 8. The user connects the data and the operators to transform the data 9. The user can see the transformed data and can save the mashup	
DHA_UAT_03	Mashup trigger	The tester will be able to set a trigger for a mashup.	1. The user logs in to the platform 2. The user opens the Data Mashup Editor component 3. The user chooses mashups tab	

The MES-CoBraD project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under grant agreement No 965422

			4. The UI shows the list of mashups 5. The user selects one of the mashups and a new window appears 6. The user finds the trigger options on mashup details 7. The user adds the target and the condition for the trigger 8. The user saves the mashup with a trigger	

10.6 API GATEWAY

Table 11 - API Gateway UAT plan

Test Case ID	Test Case Name	Summary	Expected UX	Remarks
API_UAT_01	Retrieve a specific version of a file	The tester will be able to retrieve a specific file	1. The user logs in to the platform 2. The user accesses the path /bucket_name/full_file_path?versionId={versionId} using browser or Postman tool, where bucket_name is the name of the bucket where file is stored, full_file_path is the path to the file and versionId is the specific version to access 3. The user can view the data from the specific file	
API_UAT_02	Retrieve data via API from the data lake	The tester will be able to access the data from the data lake	1. The user logs in to the platform 2. The user submits a form using the Airflow component and keeps note of the link where the data should be available 3. The user accesses the link using browser 4. The user can access the raw data	
API_UAT_03	Trigger a mashup via API	The tester will be able to trigger a mashup	1. The user logs in to the platform 2. The user creates a mashup using the Data Mashup Editor component 3. The user chooses a target for the mashup trigger 4. The user activates the trigger using REST API by sending a POST request to /mashups/{execflowId} 5. The user can observe the new data by accessing the target saved in the mashup	

The MES-CoBraD project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under grant agreement No 965422

10.7 INTEGRATION TESTING

Table 12 - Integration tests plan

Test Case ID	Test Case Name	Summary	Expected UX	Remarks
ITE_UAT_01	Data adding	The tester will insert data which will be properly anonymised and saved in data lake.	1. The user logs in to the platform 2. The user opens the Airflow component 3. The user clicks on the connection section 4. The user clicks on “new connection” 5. The user defines a new connection by specifying the type of connection, endpoint and credentials 6. The UI prompts a form where the user can submit a file 7. The user submits a file with the following three columns: a. Name b. Surname c. exam_result 8. The user selects the chosen de-personalisation protocol (pseudo-anonymisation) and submits the form 9. The UI confirm the upload of the file and gives back a link to the page where the anonymised data will be available. 10. The user can access the raw, anonymised data via REST API	
ITE_UAT_02	Data harmonisation	The tester will use the data from the data lake to create harmonized datasets which will be exposed using APIs.	1. The user logs in to the platform 2. The user opens the Data Mashup Editor 3. The user chooses the datasets tab 4. The user enters the url of the dataset created in the Airflow component 5. The user adds other properties and saves the dataset 6. The user chooses the mashups tab 7. The user selects the newly created dataset from the list	

The MES-CoBraD project has received funding from the European Union’s Horizon 2020 Research and Innovation Programme under grant agreement No 965422

			of datasets and adds it to the mashup 8. The user transforms the data using operators 9. The user saves the mashup and is able to see the transformed data 10. The user can access the transformed data via API	
ITE_UAT_03	Data analysis	The tester will use the harmonised data to perform data analysis.	1. The user logs in to the platform 2. The user opens the Airflow component 3. The user can add the dataset harmonized by Data Mashup Editor using an API 4. The user is able to perform analytics operations on the dataset 5. The user can save the workflow and schedule it 6. The user can access the curated data via REST API	

The MES-CoBraD project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under grant agreement No 965422

11 CONCLUSIONS AND NEXT STEPS

What discussed in this document is the version one of the Data integration, harmonisation, sharing and security layers of the MES-CoBraD platform.

This document will be considered as a "guideline" for the rest of the project but a new, updated version will be submitted in month 24 (march 2023) entitled: *D5.2 Data integration, harmonisation, sharing and security – final*.

The MES-CoBraD project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under grant agreement No 965422

References

[1] https://hadoop.apache.org/docs/r1.2.1/hdfs_design.html#Introduction
[2] V.C Pezoulas, T.P. Exarchos, D.I. Fotiadis "Medical Data Sharing, Harmonization and Analytics" 2020
[3] https://www.bmc.com/blogs/batch-processing-stream-processing-real-time
[4] K. Abouelmehdi, A. Beni-Hssane, H. Khaloufi, M. Saadi "Big data security and privacy in healthcare: A Review" 2017
[5] G. Duncan, L. Stokes, "Data masking for disclosure limitation" vol. 1, 2009
[6] https://swagger.io/specification/
[7] https://docs.konghq.com/
[8] https://github.com/joke2k/faker
[9] https://github.com/lk-geimfari/mimesis
[10] https://pyarxaas.readthedocs.io/en/latest/
[11] https://github.com/OpenMined/PyDP
[12] Liang Hong, Mengqi Luo, Ruixue Wang, Peixin Lu, Wei Lu*, Long Lu*; "Big Data in Healthcare: Applications and Challenges", 2018
[13] Senthilkumar SA, Bharatendara K Rai, Amruta A Meshram, Angappa Gunasekaran, Chandrakumarmangalam S. Big Data in Healthcare Management: A Review of Literature. American Journal of Theoretical and Applied Business. Vol. 4, No. 2, 2018, pp. 57-69. doi: 10.11648/j.ajtab.20180402.14
[14] Hian Chye Koh and Gerald Tan; "Data Mining Applications in healthcare"; Journal of Healthcare Information Management — Vol. 19, No. 2
[15] Anam Sajid1 & Haider Abbas2,3; Data Privacy in Cloud-assisted Healthcare Systems: State of the Art and Future Challenges; 2016
[16] Caufield, J.H., Liem, D.A., Garlid, A.O., Zhou, Y., Watson, K., Bui, A.A., Wang, W., Ping, P. A Metadata Extraction Approach for Clinical Case Reports to Enable Advanced Understanding of Biomedical Concepts. J. Vis. Exp. (139), e58392, doi:10.3791/58392 (2018)
[17] Jane S. Saczynski, Ph.D.1,2, David D. McManus, M.D.1,2, and Robert J. Goldberg, Ph.D.2 ; Commonly Utilized Data Collection Approaches in Clinical Research; 2013
[18] Henry Anayo Okemiri, Alo Uzoma Rita, Achi Ifeany Isaiah, Oketa Kelechi Christian, Nnamene Chizoba Christopher and Chima Chinazo I. ; Patient Data Integration: A Panacea for Effective Healthcare; 2020
[19] https://www.altexsoft.com/blog/data-standards-healthcare/
[20] A K Akobeng; "Principles of evidence based medicine"; 2015

The MES-CoBraD project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under grant agreement No 965422