Project title: Accelerating the lab to market transition of AI tools for cancer management

Grant Agreement: 952172

Call identifier: H2020-SC1-FA-DTS-2019-1

Topic: DT-TDS-05-2020 AI for Health Imaging

D3.4. Final Platform Design

Leader partner: Universitat Politècnica de València (UPV)

Author(s): UPV: Ignacio Blanquer, Andrei S. Alic , Sergio López Huguet and Pau Lozano y J. Damian Segrelles Quilis

Reviewers: BAHÍA: Juan Prieto-Pena, Javier Pena Rendo

UM: Henry Woodruff, Anke Wind

Work Package: WP3

Due date: Month 18

Actual delivery date: 28/02/2022

Type: R

Dissemination level: PU

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 952172.

Table of contents

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 952172.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 952172.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 952172.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 952172.

Abbreviations

Acronym	Description
ABAC	Attribute-based access control
AI	Artificial Intelligence
API	Application Programming Interfaces
AWS	Amazon Web Service
CLUES	Cloud Elasticty System
DevOps	Development and Operations
DPO	Data Protection Officer
EC3	Elastic Compute Cluster in the Cloud
GDPR	General Data Protection Regulation
IaaS	Infrastructure as a Service
IdP	Identity Providers
IM	Infrastructure Manager
K8s	Kubernetes
LDAP	Lightweight Directory Access Protocol
MIABIS	Minimum Information About BIobank data Sharing
OCI	Open Container Initiative
OIDC	OpenID Connect
OASIS	Organization for the Advancement of Structured Information Standards
PACS	Picture Archiving and Communication System
PID	Persistent Identifier
RBAC	Role-based access control
RDBMS	Database Management Systems
RDP	Remote Desktop Protocol
RIS	Radiology Information System
SAML	Security Assertion Markup Language
SSH	Secure SHell
TOSCA	Topology and Orchestration Specification for Cloud Application

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 952172.

UML	Unified Modelling Language
UUID	Universally Unique IDentifier
VMI	Virtual Machine Image
VNC	Virtual Network Computing
WADO-RS	Web Access to DICOM Objects Restful Service

Disclaimer

The opinions stated in this report reflect the opinions of the authors and not the opinion of the European Commission.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 952172.

1. Introduction

One of the objectives of the *CHAIMELEON project* is to design an EU-wide interoperable repository (***CHAIMELEON Repository***). The repository will provide storage resources to share health images, related clinical data and molecular data from pathology and liquid biopsy samples. Also, the repository will provide advanced computational cloud infrastructures to process these data as valuable resources for the AI community to develop and test practical tools (such as Quantitative Imaging biomarkers) for improved cancer management applications. These tools will be validated in the application context of four organs: lung, colorectal, breast and prostate. The repository must be supported by a distributed infrastructure which it has built on existing initiatives at the levels of European, national, regional and individual centres.

The deliverable *"D.3.4. Final Platform Design" describes* the final platform design of the *CHAIMELEON Repository*. The content is an extension of the *"D.3.3. Interim Platform Design"* where the architecture components, use cases and requirement specifications have been refined and extended.

1.1. Scope of the Document

This document constitutes a deliverable report of WP3. It covers the final design architecture to implement the *CHAIMELEON Repository*.

This document uses as input "*D3.1 Accessible Imaging Health Data Map*" and especially "*D3.2 Requirements and Standards for CHAIMELEON Platform"* and *"D.3.3. Interim Platform Design"* to identify: a) components that compose the final repository design; b) actors and functional requirements (*User Roles and Actions*); b) interactions (*Use Cases*) among components of the architecture to support the identified *User Role* actions; c) and non-functional requirements (e.g. standards or common data models).

1.2. Target Audience

The document serves both internally and externally to the consortium. The document gathers and describes the **Components of the architecture**, ***User Roles*** and ***Actions***, ***Use Cases*** and ***Requirements*** collected from the different meetings and interactions with the users. This information will serve the developers to design and implement properly the *CHAIMELEON Repository* and the consortium partners that will use it as a guideline on what they should expect.

The document could also serve external readers interested in building up a repository, reusing both the experience and the components. As most of the components are released under Open-Source licenses, external readers could find in this document an applicable solution that could address similar problems.

Finally, it serves project coordination and reviewers as evidence to support the achievement of milestones.

1.3. Structure of the Document

This document starts with this introduction. Section 2 describes the *User Roles* and the *Actions* they can perform in the *CHAMELEON Repository* (functional requirements).

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 952172.

Section 3 presents the final design of the repository, describing the components that will be integrated and the technologies that will be used for its implementation. Then, Section 4 describes in detail the interactions among the architecture components to support the actions that can be carried out for each user role. Next, Section 5 presents all the requirements defined to be considered. Section 6 describes at architecture level the threat analysis and the recommendations for the set up and configuration of the components regarding security. Finally, section 7 ends with the conclusions.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 952172.

2. User Roles

This section focuses on the *User Roles*, which generalise those users interacting with the Repository. Depending on the roles assigned to a user, they can perform specific *Actions* corresponding to functional requirements (e.g. ingest data, create *Datasets*, deploy *Standalone Application*, publish *Processing and AI Tools* etc.), which start an interaction flow (*Use Cases*) supported by the components of the final architecture (see section 3).

Figure 1 *Generic User Roles Identified.*

The *User Roles* identified (see figure 1) have been refined with respect to deliverable D3.3. For each *Role*, a detailed description is presented, including the specific list of *Actions* (functional requirements) for which it is enabled to carry out and the *Use Case* (see Section 4) where these actions take place. Also, the partners of the consortium that will assume the user roles are identified.

Regarding the list of actions, table 1 shows the common actions that can be carried out by all user roles (any *User*).

Table 1. List of actions enabled to all Roles.

Action	**Description**	**Use Case**
Request Identity	S(he) creates an identity through an existing Identity Provider supported by CHAIMELEON Repository.	User Identity Assignment from IdP
Perform Proof of Identity	The applicant is verified as a real user who belongs to a given institution or organisation accepted by the CHAIMELEON Committee.	Proof of Identity
Authenticate	S(he) authenticates in any of the CHAIMELEON Repository components.	User Authentication
Authorise	S(he) is authorised by a component to carry out a specific *Action* in the CHAIMELEON Repository.	User Authorisation

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 952172.

Request Sign Up	The applicant initiates the process to be validated (proof of identity) by the CHAIMELEON Committee and registered in the repository with a set of specific roles.	User Registration

2.1. Cloud Services and Security Manager

Cloud Services and Security Manager role is in charge of the management of the ***CHAIMELEON Repository***. The actions carried out by this role are the following:

- **Infrastructure Management.** To manage the *Infrastructure as a Service (IaaS)*, the *Development and Operations (DevOps*) and the building of the containers that host the *Processing Applications*. These tasks serve to automate the deployment of virtual infrastructures comprising the components of *CHAIMELEON Repository* running and also the components set up on top of them in a secure and agile way.
- **Assign Roles to Accepted Users.** To manage the users accepted by the *CHAIMELEON Access Committee*, which analyses and approves all the user access requests. The *Cloud Services and Security Manager User* is in charge of the assignment of *User Roles* (see figure 1) in the base of the *actions* to perform in the repository requested by the users.
- **Security Management.** To manage security issues of the *CHAIMELEON Repository.* This user will make the security assessment, risk evaluation, apply software patches, subscribe to security vulnerability bulletins, validate backups and act as a contact point for security incidents.

UPV assumes this role in the CHAIMELEON project.

The *actions* that are enabled to this role are shown in table 2.

Table 2. List of actions enabled to the *Cloud Services and Security Manager Role.*

Action	**Description**	**Use Case**
Assign Roles to a User	S(he) assigns user roles to a given user who has previously been accepted by the *CHAIMELEON Committee*.	Role Assignment to Users
Verify Proof of Identity	S(he) consults to CHAIMELEON Committee if a given user has previously been verified as a real person and that s(he) belongs to an accepted organisation.	User Registration
Register a Processing Application	S(he) registers a *Processing Application (Standalone Applications* and *Processing or AI Applications*) in the *Application Registry*.	Processing Application Creation
Search for a Processing Application	S(he) browses and chooses a *Processing Application (Standalone Applications* and *Processing or AI Applications*) registered in the *Application Registry*.	Processing Application Query
Unregister a Processing Application	S(he) unregisters a *Processing Application (Standalone Applications* and *Processing or AI Applications*) *from the Application Registry*.	Processing Application Removal
Release a Standalone Application	S(he) undeploys a Standalone Application that is running on CHAIMELEON Repository cloud, releasing the cloud resources used.	Standalone Application Release
Browse Marketplace of	S(he) browses the *Marketplace* to choose Processing *and AI*	Processing and AI Tool

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 952172.

Tools	*Tools* published in this catalogue.	Query
Unpublish a Tool	S(he) removes a *Processing or AI Tools* from the *Marketplace,* maintaining the associated *Processing and AI Application* in the *Application Registry* for its traceability.	Unpublish Processing and AI Tools

2.2. Authorised Technical Data Manager

Authorised Technical Data Manager role deals with the preparation of medical data (clinical data and associated medical images) to be ingested into the *Data Lake* of the *CHAIMELEON* Repository. This data preparation includes tasks of collaboration with the *Clinical staff* for the selection of cases and tasks, such as de-identification, anonymisation and curation, to name a few, which are performed at the data providers centres using specific tools (e.g. Medexprim Suite(™)). These tasks are carried out entirely on the medical centre side and do not require access rights to the repository. This will be supported by the resources of the hospitals, as this information is not yet anonymised.

After performing these tasks, the *Authorised Technical Data Manager* has access rights to ingest the fully anonymised and curated prepared medical data to the *Data Lake* through services and applications provided by the *CHAIMELEON Repository.*

This role is present in every medical centre contributing medical data to the repository. Partners supporting this role are HULAFE, CERF, PSD, ULS, UNIPI, CHA, UM, CHUP.

The *actions* that are enabled to this role are shown in table 3.

Table 3. List of actions enabled to the *Authorised Technical Data Manager*.

Action	**Description**	**Use Case**
Ingest Patient Cases In-Bulk	S(he) ingests a set of patient cases (images and clinical data associated collected in an e-Form) into Data Lake of the *CHAIMELEON Repository* through an In-bulk ingestion application located in a medical centre (e.g. hospitals).	In-Bulk Data Lake Ingestion
Ingest Patient Case Manually	S(he) ingests a patient case (Images and clinical data associated collected in an e-Form) manually into Data Lake of the *CHAIMELEON Repository*.	Manual Data Lake Ingestion
Browse Patient Cases	S(he) browses patient cases (images and associated clinical data) stored in the *Data Lake* of the repository to select specific cases and update them due to incorrect detected data.	Data Lake Query
Update Patient Case	S(he) updates data (images and/or associated clinical data) stored in the *Data Lake* of the repository due to being incorrect.	Update Data Lake Data

2.3. Clinical Staff

Clinical Staff role deals with both the provision (data provider) and consumption (exploration and processing) of medical data (data consumer) and the use of high-level data *Processing and AI tools* provided by the *Marketplace* of the *CHAIMELEON Repository* (tool consumer).

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 952172.

As a data provider, the main tasks of this *Role* correspond to the selection of medical data from the different sources located in the medical centres (e.g. PACS, RIS, Clinical Data Bases) to be uploaded into the *Data Lake* by the *Authorised Technical Data Manager***.** These tasks are carried out entirely on the medical centre side, following their own protocols and procedures, and do not require access rights to the repository.

As data and tool consumers, clinical staff use the *Processing and AI Tools* provided by the *Marketplace*, as well as inspect the data available in the *Case Explorer Application*. The *Processing and AI Tools* are used to annotate, segment, enhance or extract knowledge from the medical data to improve diagnosis, prognosis, treatment diagnosis and so on.

Partners supporting this role are HULAFE, CERF, PSD, ULS, UNIPI, CHA, UM, CHUP.

The *actions* that are enabled to this role are listed in table 4.

Table 4. List of actions enabled to the *Clinical Staff*.

Action	Description	Use Case
Browse Marketplace of Tools	S(he) browses the *Marketplace* to choose Processing *and AI Tools* published in this catalogue.	Processing and AI Tool Query
Apply Tool	S(he) employs a *Processing or AI tool* published in the *Marketplace* to improve clinical processes such as diagnosis, prognosis and treatment follow-up.	Apply Processing or AI Tools
Browse Patient Cases	S(he) browses patient cases (images and associated clinical data) stored in the *Data Lake* of the repository to select specific cases.	Data Lake Query

2.4. Dataset Administrator

Dataset Administrator role deals with the registration of *Datasets* from the *Data Lake* that are used by the researchers. A *Dataset* can be considered as a publication of a research object that comprises a coherent subset of medical data and is identified through a unique Persistent Identifier. A research object (datasets or trained ai models) has a PID assigned and resolvable linked to an info page in which aggregated information is presented (e.g. author, provider). Along with this information, it has a fingerprint as a unique checksum that could be computed by the user, so the veracity of the data available (*Model* or *Dataset*) can be attested.

A dataset could not be modified once registered, except for specific metadata fields (contact person or description). Datasets will have a Persistent Identifier (PID) assigned, so to ensure reproducibility, the CHAIMELEON platform will not allow changing the data, as changing a dataset may lead to different results and incoherences in the reproducibility. If a dataset needs to be changed, a new dataset should be created. Therefore, all the references to the previous dataset will remain valid. A faulty dataset can be invalidated.

Partners supporting this role are CERF, ULS, PSD and HULAFE.

The *actions* that are enabled to this role are shown in table 5.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 952172.

Table 5. List of actions enabled to the *Dataset Administrator*.

Action	Description	Use Case
Create a Dataset	S(he) chooses a set of data (medical images and clinical data) from *Data Lake* and creates a *Dataset* in the *Repository Database*. Data has been previously uploaded by an *Authorised and Technical Data Manager*.	Dataset Creation - Dataset Creation From Data Lake. - Enriched Dataset Creation. - External Dataset Importation
Create an Enriched Dataset	S(he) creates a *Dataset* in the *Repository Database* based on an existing *Dataset* and enriched data computed through *Standalone Applications*.	Enriched Dataset Creation
Import a Dataset	S(he) creates a *Dataset* in the *Repository Database* using data sources from external institutions.	External Dataset Importation
Disable a Dataset	S(he) disables a given *Dataset*. A Dataset may be obsolete or have entries for which consent has been revoked.	Dataset Disablement
Browse for a Dataset	S(he) browses and chooses existing *Datasets* in the *Repository Database*.	Dataset Query

2.5. Data Scientist

Data Scientist role deals with accessing a *Dataset* available in the *CHAIMELEON Repository* and analysing it in a specific processing environment (*Standalone Processing Applications*) for building new AI *models* that will be published as a publication of a research object. This user will have access rights to console applications with processing tools (*Standalone Applications*) and is able to register the trained AI *models* and tools developed in the *Marketplace*. Also, the user is also able to trace the usage of AI *models* and to use other models for comparison.

If a faulty image is found, a data scientist should inform the "creator" of the dataset (the dataset manager, included in the metadata) about this fault. Data scientists are not authorized to ingest new official data in the repository, so they cannot feed them back (again, for reproducibility reasons). The dataset manager can invalidate a dataset with a faulty image and create a new one. Then, the ingestion and update of images are always performed on the case explorer application.

Partners supporting this role are HULAFE, CHA, BGU, IMPERIAL, QUIBIM, BAHIA, GEHC.

The actions that are enabled to this role are shown in table 6.

Table 6. List of actions enabled to the *Data Scientist*.

Action	Description	Use Case
Browse for a Dataset	S(he) browses and chooses existing *Datasets* in the *Repository Database*.	Dataset Query
Search for a Processing Application	S(he) browses the *Application Registry* for a *Processing Application* (both *Standalone* and *Processing/AI Applications)*	Processing Application Query
Deploy a Standalone Application	S(he) deploys *Processing Applications (Standalone Applications* or *Processing/AI Applications)* on the cloud resources of the CHAIMELEON repository.	Standalone Application Deployment

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 952172.

Release a Standalone Application	S(he) undeploys a given *Standalone Application* and releases the cloud resources used.	Standalone Application Release
Browse Standalone Access Points	S(he) searches for the access point of a deployed Standalone Application.	Standalone Application Access Point Query
Access to a Standalone Application	S(he) access to a given running Standalone Application through remote desktop or SSH protocols.	Standalone Application Access
Browse Marketplace of Tools	S(he) browses the *Marketplace* to choose Processing *and AI Tools* published in this catalogue.	Processing and AI Tool Query
Apply Tool	S(he) employs a *Processing or AI tool* published in the *Marketplace* to compare with his/her own processing or AI models.	Apply Processing or AI Tools
Publish Model	S(he) publishes his/her *Model* in the *Source code Repository*.	Publish Model
Unpublish Model	S(he) unpublished his/her *Model* in the *Source code Repository*.	Unpublish Model
Browse The Model Repository	S(he) searches for Models published in the *Source Code Repository*.	Model Query

2.6. Application Developer

Application Developer role deals with the building of software tools (such as tools exploiting the AI trained *models*, segmentation tools, analytical engine tools) that could be used by the *Clinical Staff* to carry out diagnosis, prognosis, follow-up or by *Data Scientists* for data analytics. All these software tools are provided as containerised applications by the *CHAIMELEON Repository* and are embedded in the repository by the *Cloud Services Manager.*

Partners supporting this role are CHA, BGU, IMPERIAL, QUIBIM, BAHIA, GEHC, MEDEX.

The *actions* that are enabled to this role are shown in table 7.

Table 7. List of actions enabled to the *Application Developer*.

Action	**Description**	**Use Case**
Browse for a Dataset	S(he) browses and chooses existing *Datasets* in the *Repository Database*.	Dataset Query
Search for a Processing Application	S(he) browses the *Application Registry* for a *Processing Application* (both *Standalone* and *Processing/AI Applications)*	Processing Application Query
Deploy a Standalone Application	S(he) deploys *Processing Applications (Standalone Applications* or *Processing/AI Applications)* on the cloud resources of the CHAIMELEON repository.	Standalone Application Deployment
Release a Standalone Application	S(he) undeploys a given *Standalone Application* and releases the cloud resources used.	Standalone Application Release
Browse Standalone Access Points	S(he) searches for the access point of a deployed Standalone Application.	Standalone Application Access Point Query
Access to a Standalone Application	S(he) access to a given running Standalone Application through remote desktop or SSH protocols.	Standalone Application

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 952172.

		Access
Publish a Tool	S(he) publishes in the marketplace a Processing or AI Tool. These tools provide the means to extract knowledge to assist in research, diagnosis, prognosis and treatment follow-on.	Publish Processing or AI Tools
Unpublish a tool	S(he) removes a *Processing or AI Tools* from the *Marketplace,* maintaining the associated *Processing and AI Application* in the *Application Registry* for its traceability.	Unpublish Processing and AI Tools
Browse Marketplace of Tools	S(he) browses the *Marketplace* to choose Processing *and AI Tools* published in this catalogue..	Processing and AI Tool Query
Apply Tool	S(he) employs a *Processing or AI tool* published in the *Marketplace* to compare with his/her own processing or AI models.	Apply Processing or AI Tools
Browse The Model Repository	S(he) searches for Models published in the *Source Code Repository*.	Model Query

2.7. Data Protection Officer

Data Protection Officer role deals with the surveillance of the fulfilment of the data protection regulations and the traceability of the data usage along with the project.

Partners supporting this role are clinical centres (HULAFE, CERF, PSD, ULS, UNIPI, CHARITE, UM, CHUP).

The *actions* that are enabled to this role are shown in table 8.

Table 8. List of actions enabled by the *Data Protection Officer*.

Action	**Description**	**Use Case**
Consult Dataset History	S(he) retrieves the full set of operations performed on a given *Dataset*.	Dataset History Query
Consult Accessed Datasets of a User	S(he) retrieves the details of the users who have used a given *Dataset*.	Query users who accessed a Dataset
Consult Datasets of a Model	S(he) retrieves the aggregated information of the Dataset that has been employed to develop a model.	Query Datasets used on a Specific Model
Consult Research Object Fingerprint	S(he) consults the fingerprint of a specific research object (dataset or model).	Fingerprint Research Object Query

2.8. External Researcher

External Researcher role deals with browsing the aggregated information of the AI *tools* and *models* published in the marketplaces and *Source Code Repository* to get an appraisal of the potential interest that she may have in the *CHAIMELEON Repository*. This user has limited access rights to the repository and eventually can access the *Datasets, Processing Applications* and *Tools* provided by the repository.

If a faulty image is found by an External Researcher, s(he) should inform the "creator" of the dataset (the *Dataset Manager*, included in the metadata) about this fault as it happens in the

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 952172.

Data Scientist Role when s(he) finds one. Then, the Dataset Manager proceeds the same way.

Data scientists are not authorized to ingest new official data in the repository, so they cannot feed them back (again, for reproducibility reasons). The dataset manager can invalidate a dataset with a faulty image and create a new one. Then, the ingestion and update of images are always performed on the case explorer application.

Partners supporting this role are external researchers to the consortium.

The *actions* that are enabled to this role are shown in table 9.

Table 9. List of actions enabled to the *External Researcher*.

Action	**Description**	**Use Case**
Browse for a Dataset	S(he) browses and chooses existing *Datasets* in the *Repository Database*.	Dataset Query
Search for a Processing Application	S(he) browses the *Application Registry* for a *Processing Application* (both *Standalone* and *Processing/AI Applications)*	Processing Application Query
Deploy a Standalone Application	S(he) deploys *Processing Applications (Standalone Applications* or *Processing/AI Applications)* on the cloud resources of the CHAIMELEON repository.	Standalone Application Deployment
Release a Standalone Application	S(he) undeploys a given *Standalone Application* and releases the cloud resources used.	Standalone Application Release
Browse Standalone Access Points	S(he) searches for the access point of a deployed Standalone Application.	Standalone Application Access Point Query
Access to a Standalone Application	S(he) access to a given running Standalone Application through remote desktop or SSH protocols.	Standalone Application Access
Browse Marketplace of Tools	S(he) browses the *Marketplace* to choose Processing *and AI Tools* published in this catalogue.	Processing and AI Tool Query
Apply Tool	S(he) employs a *Processing or AI tool* published in the *Marketplace* to compare with his/her own processing or AI models.	Apply Processing or AI Tools
Publish Model	S(he) publishes his/her *Model* in the *Source code Repository*.	Publish Model
Unpublish Model	S(he) unpublished his/her *Model* in the *Source code Repository*.	Unpublish Model
Browse The Model Repository	S(he) searches for Models published in the *Source Code Repository*.	Model Query

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 952172.

3. Architecture

This section presents a general view of the *CHAIMELEON Platform Architecture*. Figure 2 shows the *Components* defined in architecture without depicting any interaction. Also, the figure shows the technologies considered for its implementation. All *Components* inside the boundaries of the *CHAIMELEON Repository* are connected through an internal private overlay network. This network is created by the Orchestrator Service (K8s in the figure), and it is only accessible by the containers that run the applications.

The role *actions* defined in Section 2 (see from table 1 to table 9) supported by the *Components* will be outlined in the use cases (see section 4).

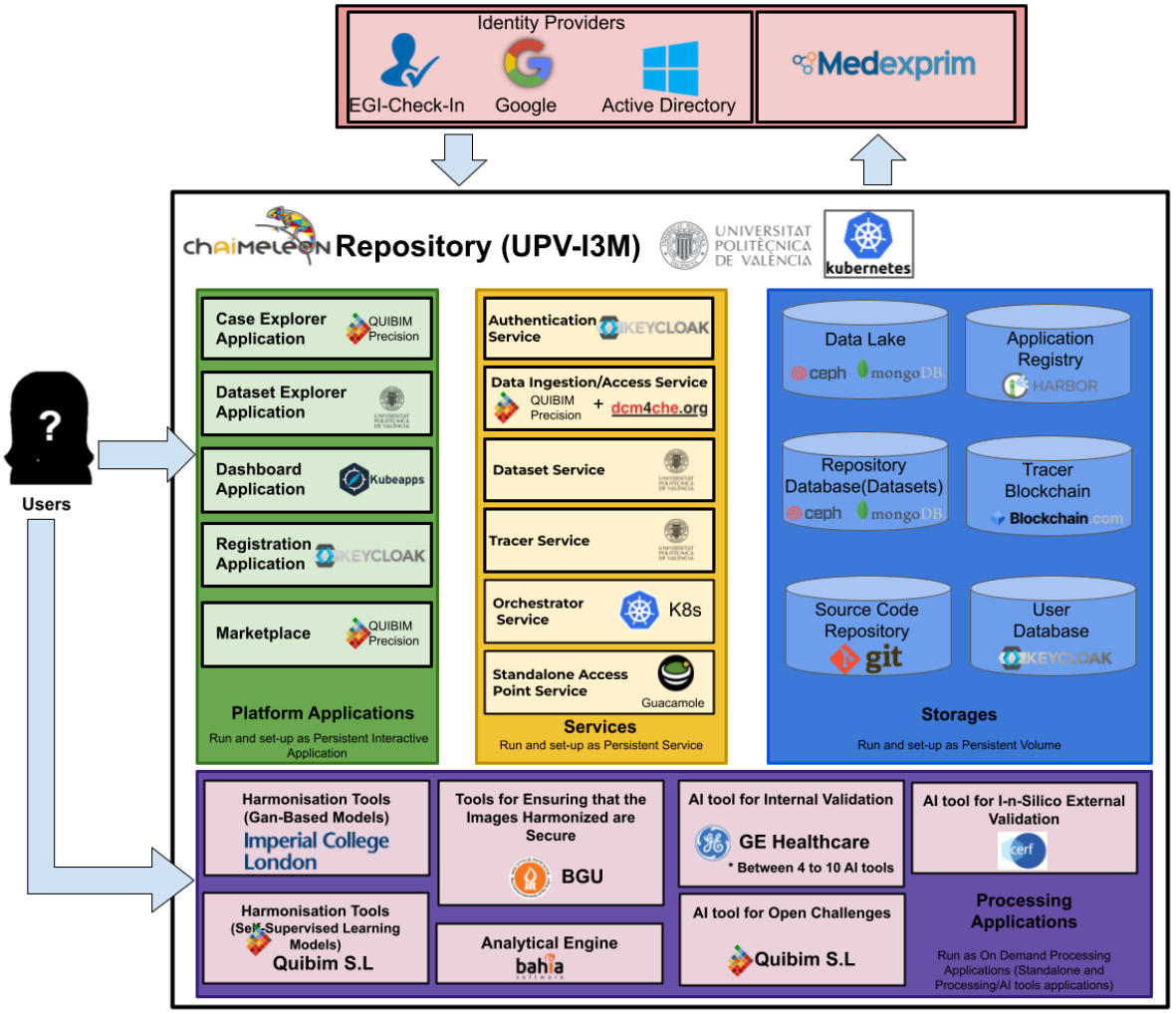


Figure 2. *Identified Components of the Final CHAIMELEON Architecture.*

3.1. Components

All *Components* of the architecture are described below. There are components that are running inside the *CHAIMELEON Repository*, employing the Cloud infrastructure provided by the UPV Cloud platform. Also, there are components that are running outside of the *CHAIMELEON Repository* boundaries, such as *MEDEXPRIM Suite* running in the medical centres for collecting and anonymizing data to ingest into the repository (managed internally

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 952172.

at the data providers with their own resources), or *IdP* supported by external entities to manage the authentication of the users, also managed externally by the Identity Providers.

Inside the boundaries of the *CHAIMELEON Repository*, there are five types of components which are the following:

- **Storages.** They provide data persistence (including medical data, application binaries and specifications, traceability logs, users' profiles) in the *CHAIMELEON Repository.* Storages are uniquely managed by *Services* or *Applications* and cannot be accessed directly by the end-users.

- **Platform Applications** are Applications providing the end-users with the functionalities (user *actions*) offered by the *CHAIMELEON Repository* through user-friendly interfaces. The users will mainly interact with the repository through these applications that implement the functionalities through the *Services*.

- **Services.** This component provides the functionality required by the end user's applications (*Platform Applications*), interaction with resources such as the *Storage,* and the processing back-ends. Some of them will be internal services (they will not be accessible even to application developers), and some others (e.g., Access/Ingestion service) will allow direct interaction with other applications of the Repository offering specific functionalities such as the data Ingestion in bulk mode.

- ***Processing Applications*.** They are hosted by the *CHAIMELEON Repository* and are used for data processing and analysis. These applications run and are set up on the *CHAIMELEON Cloud Resources*. There are two types of processing applications: *Standalone Applications* that are deployed on-demand and *Processing and AI Tools* that are managed in the *Marketplace* integrated with the *Case Explorer Application*.

The identified *Storages* are the following:

- ***User Database.*** This storage provides the persistence of all the information related to the identity of the users and their *roles* and capabilities in the repository.

- ***Data Lake.*** This storage provides the persistence of observational and research medical data coming from the medical centres. Data will be fully anonymised and comprises clinical data and its associated medical images. They are tentatively organised into four thematic sections, which are breast, lung, prostate and colorectal cancer.

- ***Repository Database (Datasets).*** This storage provides the persistence of *Datasets.* Datasets are research objects uniquely identified that comprise a coherent set of clinical data (copied from *Data Lake*) and references to its associated medical images (hosted at *Data Lake*). The *Repository Database* stores data in an appropriate way to minimise data replication but ensure data consistency, reusability and traceability.

- ***Application Registry.*** This storage provides the persistence of *Processing Applications* and *Platform Applications* as containers. Applications able to run on the *CHAIMELEON Repository* will be stored on an internal and secure registry of container images and deployment recipes, describing the application topology specifications for those applications which require different storage and computing resources to run and set up.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 952172.

- ***Source Code Repository.*** This repository provides the persistence of the source code of the analytic algorithms and AI-trained model data to be shared and used in the project.
- ***Tracer Blockchain.*** This storage provides the persistence of all data generated regarding the use and traceability of the *Datasets* and the development of *Processing and AI tools*.

The identified *Services* are the following:

- ***Authentication and Authorisation Service.*** This service provides to the *User Database* all functionalities to generate a user identity with aggregate information (e.g. institution, groups, role, capabilities). It also provides all the required information to *Platform Applications*, *Services* and *Processing Applications* to implement their user authentication and authorisation processes.
- ***Data Ingestion/Access Service***. This service provides all the necessary functionalities to the *Platform Applications* to manage the *Data Lake* data such as insertion, updating or selection of data. Also, this service allows the medical centres to connect their applications for in bulk data ingestion into the *Data Lake*.
- ***Dataset Service.*** This service provides all necessary functionalities to the *Platform Applications, Services* and *Processing Applications* to manage the registration of a *Dataset* as a set of image references hosted at *Data Lake* and clinical data copied from *Data Lake*. Furthermore, it includes functionalities to the creation of the *Dataset* access as Volumes.
- ***Tracer Service.*** This service provides all necessary functionalities to the *Services* to register the operations related to the creation and access to *Datasets* and *AI tools*.
- **Orchestrator Service:** This service provides all necessary functionalities to the *Platform Applications* to manage *Processing Applications* (containerised applications) and their cloud resources. It will deploy, inspect, update and release *Processing Applications* in the *CHAIMELEON Cloud Resources*.
- **Standalone Access Point Service.** This Service provides all necessary functionalities to connect to *Standalone Applications* which are running in the cloud resources of the Repository.

The *Platform Applications* are the following:

- **User Registration Application.** This application provides *Users* with a user-friendly web interface to sign up in the *CHAIMELEON Repository* with a specific *Role*, assigning their membership to groups and capabilities. In addition, it provides administrators with a web interface to validate the documentation required for registration and proof of entity and thus accept or deny a user registration.
- ***Case Explorer Application:*** This application provides the *Users* with a user-friendly web interface to manage the information in the *Data Lake*. Also, functionalities to select a set of data for creating a *Dataset* are provided.
- ***Marketplace.*** This application provides a market of *Processing and AI tools*. It allows *Data Scientists* and *Developers* to publish their *Processing and AI tools* (*Processing*

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 952172.

Applications), and allows *Clinical Staff* to employ them in the medical centres for fine diagnosis, prognosis, follow-up treatment and so on.

- ***Dataset Explorer Application.*** This application provides a user-friendly web interface to explore the existing *Datasets* and to retrieve information about their usage. It also is used to gather the *Dataset* identifiers to be acceded by a *Processing Application*.
- ***Application Dashboard.*** This dashboard provides a user-friendly web interface that enables *Users* to deploy applications such as data science AI frameworks, data exploration environments or system consoles to access and process the data. This application is able to query the *Application Registry* and is able to deploy the *Processing Applications* through the Orchestrator.

The *Processing Applications* are defined by the *Users* and can be deployed on-demand by *Users* on *CHAIMELEON Repository* Cloud Resources. There are two types of *Processing Applications*. They are the following:

- ***Processing or AI Applications.*** These are processing applications or trained AI models embedded as applications to extract knowledge from medical images and clinical data to fine diagnosis, prognosis, follow-up of treatments and so on. These *Processing Applications* are managed by the *Marketplace* of the *CHAIMELEON Repository*.

- ***Standalone Applications.*** These are interactive applications for exploring and processing existent *Datasets* of the CHAIMELEON Repository. These *Processing Applications* are stored in the *Application Registry*. Some examples of this type of application are analytical engines (provided by BAHIA), harmonisation tools (provided by Imperial College and QUIBIM), tools for ensuring that the Images harmonised are Secure (provided by BGU) and tools and frameworks for AI developers such as Jupyter server with Tensorflow or Pytorch.

3.2. Technologies

Most of the components described above is released under open-source technologies.

The next subsections describe the technologies involved in the deployment of the infrastructure that supports the *Components* and their implementation. This document is the final architecture design, although changes may occur during the deployment phase (D4.2 *Interim Repository deployment* and D4.3 *Final Repository deployment*). Moreover, a long-term plan on technology update will be drawn in D10.2 *Action plan for Repository sustainability*.

3.2.1. OpenStack

OpenStack[1] is a cloud management system that controls large pools of computing, storage, and networking resources throughout a data centre, all managed and provisioned through APIs with common authentication mechanisms. Beyond standard infrastructure-as-a-service functionality, additional components provide orchestration, fault management and service management amongst other services to ensure high availability of user applications.

[1] https://www.openstack.org

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 952172.

In the context of the *CHAIMELEON project*, this technology will be used to manage the cloud resources provided by the repository.

3.2.2. Infrastructure Manager (IM)

The Infrastructure Manager (IM)[2] enables the deployment of computing infrastructures on cloud providers. IM is a tool that deploys complex and customised virtual infrastructures on multiple back-ends. The IM automates the Virtual Machine Image (VMI) selection, deployment, configuration, software installation, monitoring and update of virtual infrastructures. It supports a wide variety of back-ends, making user applications Cloud agnostic. In addition, it features Development and Operation (DevOps) capabilities. DevOps is an approach in which the deployment and configuration of infrastructures mainly based on virtual resources are automated to be triggered forth and back to test and deploy updates on production infrastructures. Moreover, it also provides platform independence, facilitating the migration to a different backend. IM uses Ansible[3], a declarative language to describe resources and software configuration, to enable the installation and configuration of all the user required applications providing the user with a fully functional infrastructure. IM has been developed by UPV-I3M.

In the context of the *CHAIMELEON project*, this technology will be used to implement the deployment of the *Components* of the Architecture.

IM supports Topology and Orchestration Specification for Cloud Applications (TOSCA)[4] that it is a standard created by the Organisation for the Advancement of Structured Information Standards (OASIS).

The idea behind the TOSCA standard is to define service templates consisting of different components and the relationships among these different parts. TOSCA helps configure applications and their underlying infrastructure as well as enable moving applications in the cloud. Also, all required orchestration policies and resources can be defined using templates dedicated for that purpose.

The TOSCA standard has the main goal of answering the need for automation, portability, and interoperability along with the management challenges of complex cloud applications.

In the context of *CHAIMELEON*, TOSCA will be used to describe the topology of all *Processing Applications* to deploy on the cloud resources of the *CHAIMELEON Repository.*

3.2.3. Elastic Compute Cluster in the Cloud (EC3)

The Elastic Compute Cluster in the Cloud (EC3)[5] is a technology developed by the UPV-I3M that enables the deployment of virtual elastic hybrid clusters across Cloud infrastructures. It consists of a set of recipes and a command-line interface used as a client for the IM to deploy a custom front-end node of a virtual cluster that features: i) an instance of an IM to provision for additional computing resources (working nodes); ii) Cloud Elasticity System (CLUES), implementing the elasticity rules considering the state of the Local Resource Management System (LRMS) and iii) the specific configuration for the virtual cluster required for the execution of the applications that will be run on the cluster.

EC3 is also offered as a free online service to deploy on-demand elastic virtual clusters on Amazon Web Services, OpenNebula and OpenStack.

---

[2] https://www.grycap.upv.es/im/ GNU General Public License - version 3.0
[3] https://www.redhat.com/en/topics/automation/learning-ansible-tutorial
[4] https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=tosca
[5] www.grycap.upv.es/ec3, Apache License, Version 2.0

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 952172.

In the context of the CHAIMELEON project, EC3 will be used to implement the Flexible and Elastic Management of Cloud Resources.

3.2.4. Kubernetes (K8s)

Kubernetes[6] (K8s) is an open-source orchestration system for Docker containers, although can be configured for running other OCI[7] compliant containers. Kubernetes schedules and runs containerised applications on clusters of physical or virtual machines. Kubernetes can run on various platforms in local machines (such as minikube), cloud-managed solutions (such as Google Container Engine[8] or Azure Container Service[9]) and cloud platforms. Kubernetes clusters are composed of two types of nodes (master and workers) and a set of core components. The master node runs the services that support the connectivity among containers in the nodes (kube-dns), the internal database (etcd) and the main service that manages all the functionalities (kube-apiserver). The worker nodes are managed through the master node and run the connection services and the containers that host the applications.

Kubernetes provides different methods for authenticating users: X509 Client certificates, static tokens per user, bootstrap tokens, service account tokens, and OpenID Connect Tokens. Thus, Kubernetes can authenticate users using Keycloak. Regarding multi-tenancy, there are different approaches to isolating tenants (users). The most efficient approach considering computing resource sharing is to isolate users in their own namespace. Namespaces are the Kubernetes logical partitions of the cluster. Regarding authorisation, Kubernetes supports several modes: node grants, Attribute-based access control (ABAC), Role-based access control (RBAC), and webhook callbacks.

Pods are the minimum unit of scheduling in Kubernetes and they are groups of containers that are deployed and scheduled together. Kubernetes permits mounting external storage into the containers by using Volumes. Kubernetes supports different types of Volumes: nfs, hostPath, azureDisk, cephfs, gcePersistentDisk, awsElastickBlockStore, etc. Container images can be obtained from several repository images, such as DockerHub[10], Google Container Registry[11], Harbor[12], AWS EC2 Container Registry[13] and Azure Container Registry [14] (ACR).

Regarding the interaction with Kubernetes, there are several ways to do it. On the one hand, it is possible to connect using CLI tools such as kubectl, through REST API or by using client libraries for several programming languages. On the other hand, it is possible to interact with Kubernetes using browser graphical user interfaces (such us Kubernetes Dashboard[15] or Kubeapps[16]) or using desktop graphical user interfaces (such us Octant[17] or Lens[18]).

[6] https://kubernetes.io/, https://github.com/kubernetes/
[7] https://opencontainers.org/
[8] https://cloud.google.com/kubernetes-engine
[9] https://azure.microsoft.com/en-us/product-categories/containers/
[10] https://hub.docker.com/
[11] https://cloud.google.com/container-registry
[12] https://goharbor.io/
[13] https://aws.amazon.com/ecr/?nc1=h_ls
[14] https://azure.microsoft.com/en-us/services/container-registry/
[15] https://github.com/kubernetes/dashboard
[16] https://kubeapps.com/
[17] https://octant.dev/
[18] https://k8slens.dev/

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 952172.

3.2.5. QUIBIM Precision

Quibim Precision[19] will be the front-end (*Case Explorer Application*) of the *Data Lake*. QUIBIM Precision provides a friendly Web UI with the basic functionalities (insert, update and query) and more advanced tools like filtering, data mining and a complete DICOM viewer that provides visualisation and interaction, such as manual segmentation of images or import/export masks. It manages both the clinical data, as *e-forms* in OMOP-CDM or OSIRIS-CDM, and medical images, in DICOM or NIFTI formats.

It provides manual ingestion and automatic anonymisation in both metadata (*tags*) and image regions. It also has a REST API interface in the back-end (*Data Ingestion/Access Service*) which enables External Applications to ingest data in bulk.

Clinical data is stored in MongoDB, and medical images are stored in a File System, provided by Ceph.

A service of QUIBIM precision in the back-end has the capabilities of downloading medical images from a DICOM node (also for bulk ingestion). For this purpose, the project will use DCM4CHEE (see Section 3.2.9), which acts as temporary storage for connecting from the Picture Archiving and Communication System (PACS).

Besides, Quibim Precision has a collection of biomarkers and AI Tools (*Marketplace*) which can be extended by users and applied to any case in the *Data Lake*. QUIBIM Precision uses Nomad and Kubernetes as container orchestrators to launch and manage those analysis modules through containers.

3.2.6. Apache Guacamole

Apache Guacamole[20] is a clientless remote desktop gateway. It supports standard protocols like VNC, RDP, and SSH.

The *CHAIMELEON project* uses Apache Guacamole to implement the *Standalone Access Point Service* for accessing Standalone *Applications* running on cloud resources of the repository.

3.2.7. Kubeapps

Kubeapps is an open-source browser UI dashboard for deploying and managing applications in the Kubernetes cluster using Helm charts files which help define, install, and upgrade even the most complex Kubernetes applications. Kubeapps provides a catalogue of Helm charts to the users of multiple public chart repositories. It also allows using private chart repositories such as Harbor[21] or ChartMuseum[22]. Regarding authentication, it supports the use of OIDC providers to access both the Kubeapps dashboard and Kubernetes API.

Kubeapps will provide a graphical interface to the users to deploy and visually select and set values for those parameters defined in the different Helm charts (for example volumes to mount or daemons to run, like VNC server to connect latter through Guacamole).

[19] https://quibim.com
[20] https://guacamole.apache.org/
[21] https://goharbor.io/
[22] https://chartmuseum.com/

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 952172.

3.2.8. Keycloak

We will use Keycloak[23] to implement the Authentication and Authorisation service. Keycloak is an open-source authentication service compatible with the standard protocols OpenID Connect (an extension to OAuth 2.0) and SAML 2.0, which are the most used nowadays. It ensures the best compatibility with the actual and future applications of the CHAIMELEON platform, but also with external Identity Providers (IDPs) like EGI Check-in, eduGain, Google among many others can be configured easily with Keycloak. This service will centralise the user management in the CHAIMELEON platform offering a single sign-on to the users and avoiding the applications needing to store users' credentials or implement login forms. Once logged in Keycloak or any of the external IDPs configured, the user will be able to access the applications without providing the credentials again, even without creating a new password for that service if (s)he already has an account in the external IDPs. Keycloak provides other CHAIMELEON applications with all the required details from the users including the groups and roles (s)he belongs to. Then, applications can allow or deny the user's access to the functionality requested, according to the role or group (RBAC).

For storing the users' data, Keycloak uses a relational database through Java™ Database Connectivity (JDBC) supporting the most widely used Relational Database Management Systems (RDBMS) such as Oracle, MySQL, Microsoft SQL Server or PostgreSQL, selecting the latter for the project.

Keycloak has an Admin Console, which is a complete web UI for managing the users, groups, roles, external IDPs and all the other configurations of the service, easing that way of the admin tasks. It also has a registration (sign up) form and an Account Management Console (Web UI) for the users. The forms, the action flows and automatic mail templates are customizable, with multi-language support. Any custom attribute can be added to the forms, storing it in the database and including it as a claim within the token sent to the applications. Keycloak assigns to every user a UUID (Universally Unique IDentifier) which is sent in the 'subfield and can be used by other applications and services like the tracer to uniquely identify the users in the *CHAIMELEON platform*.

Password policies can be adjusted, and some security defences like brute force detection can be enabled and configured too. Finally, the security can be enhanced with Two-Factor Authentication (2FA) with OTP that is available by default for any user.

This service can be configured to run in cluster mode with a load balancer and multiple instances which can be scaled up if demand increases.

3.2.9. DCM4CHEE

DCM4CHEE[24] is a collection of open-source applications and utilities for the healthcare enterprise. At the core of the DCM4CHEE project is a robust implementation of the DICOM standard and it is developed in the Java programming language for performance and portability, supporting deployment on JDK 1.6 and up.

This technology will be employed to implement the *Data Ingestion/Access Service* through Web Access to DICOM Object (WADO) service provided by DCM4CHEE.

[23] Apache License, Version 2.0. https://www.keycloak.org/. https://github.com/keycloak/keycloak
[24] Open Source Clinical Image and Object Management https://www.dcm4che.org/

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 952172.

3.2.10. Harbor

Harbor is a multi-tenant and open-source registry and Chart repository that can be configured to ensure that the images are scanned and free from vulnerabilities, and signed. The container images are organised in different projects which allows applying resource quotas and controlling the access to certain images only to authenticated users and based on predefined roles for accessing projects (limited guests, guests, developers, maintainer and project admin) and for managing Harbor server. Besides, Harbor supports several ways to authenticate users: database users, LDAP or OpenID Connect.

3.2.11. Ceph

Ceph[25] is a free-software storage platform that implements object storage on a single distributed computer cluster and provides interfaces for object-, block- and file-level storage. Ceph aims primarily for completely distributed operation without a single point of failure, scalable to the exabyte level, and freely available.

Ceph replicates data and makes it fault-tolerant, using commodity hardware and requiring no specific hardware support. As a result of its design, the system is both self-healing and self-managing, aiming to minimise administration time and other costs.

Ceph stores all data as objects within pools, irrespective of the type of storage (Ceph Filesystem, Ceph Object Storage, or Ceph Block device). Pools also can divide it into namespaces. Ceph users must have access to pools to read and write data. Besides, Ceph users must have to execute permissions to use administrative commands. Ceph uses the term “capabilities” to describe authorising an authenticated user to restrict access to data within a pool, a namespace within a pool, or a set of pools based on their application tags.

3.2.12. BlockChain

Fundamentally, a blockchain is a list (chain) of records (called blocks) linked by cryptographic hashes (Figure 3). Each record should contain, at a minimum, the cryptographic hash of the previous record (unless it’s the first block), a timestamp, and a data structure. The way the chaining is done offers protection against the modification of the lists’ records since any change in a block (except the last one) would invalidate the chain. This protection by hash chaining is improved by distributing the whole chain to various parties, creating a distributed blockchain network. If a new record is considered to be added, these parties communicate with each other to validate the proposed entry (for instance, using a voting mechanism). As a result, the blockchain network becomes a decentralised database with inherent fault tolerance and security.

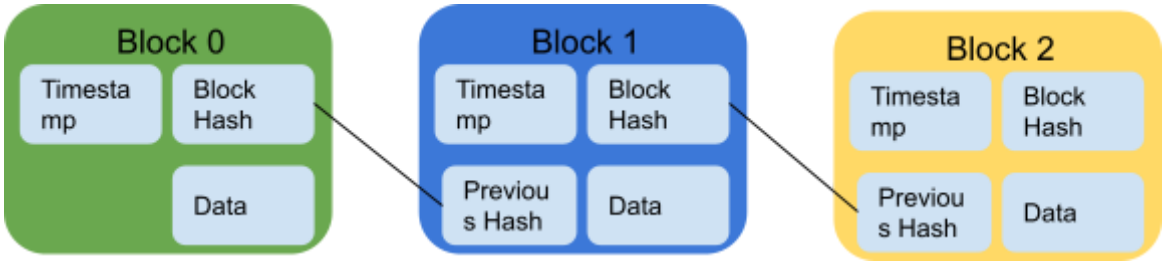


Figure 3. *The blockchain’s basic structure.*

[25] http://ceph.com/

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No. 952172.

We use blockchain technology with our Tracer service. It acts as a CR (Create-Read) instead of CRUD (create, Read, Update and Delete) database, used to trace various user actions, such as the creation of a new dataset, the use of one or more datasets in a Kubernetes pod, or the publishing of a new machine learning model. These actions include the fingerprints of the data targeted by the actions (if any, accessing an existing dataset would not need fingerprints of data since nothing new is created). For example, if a user creates a new dataset, the corresponding entry inserted into the blockchain contains the checksum of the actual imaging data. As a result, the Tracer Component isn't only a way to record the platform's usage history but a way to ensure the actual clinical data (images, patient information, dataset information) stored on our platform has not been tampered with. Blockchain technology preserves the authenticity of the data stored within, therefore the fingerprints of the actual data cannot be modified once added. Due to the requirements of the GDPR governing the rights of the user on his/her virtual data and information, we intend to store only data that cannot be used to infer protected information (such as personal user details: age, gender, occupation, home address etc.), or personal preferences (favourite colour, favourite pet etc.).

Other services running on our platform that want to register/read a trace into/from the blockchain don't access it directly. The Tracer service acts as a proxy responsible for the communication with the rest of the services (Figure 4). This way, we can ensure seamless integration of a different blockchain implementation at a later stage, if we deem it necessary (with or without the migration of the existing blocks), without changing all other services running on our platform. The use of the blockchain can be considered a data store of traces, and nothing more.

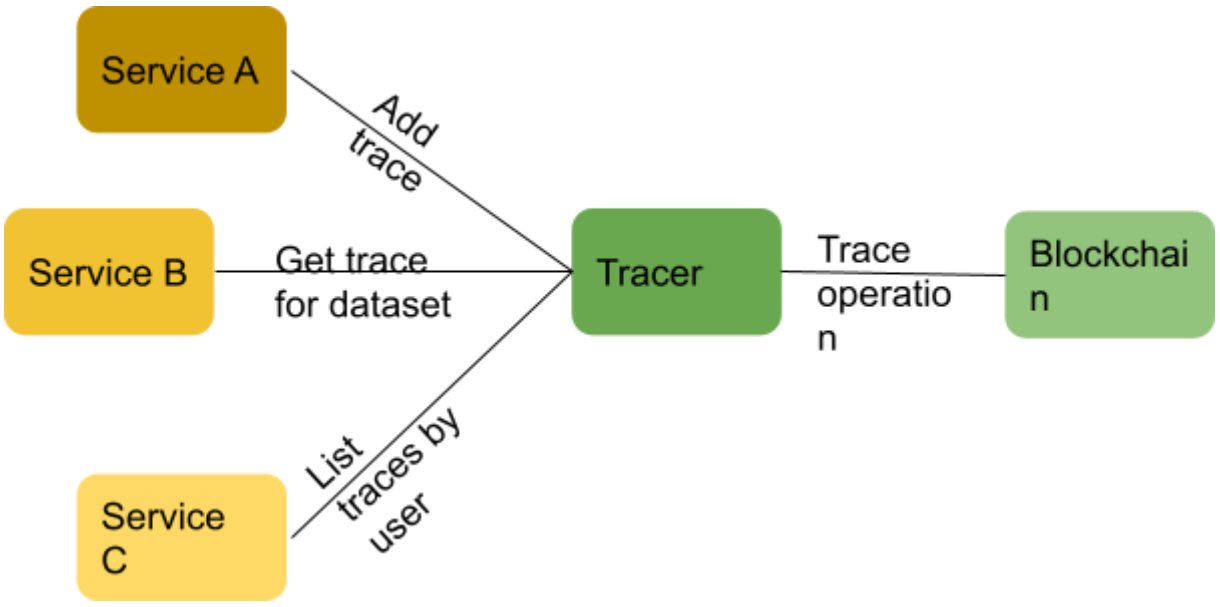


Figure 4. *Tracing operations are not querying the Blockchain directly.*

3.2.13. MongoDB

MongoDB[26] is an open-source NoSQL database management system. MongoDB is a tool that can manage document-oriented information, store or retrieve information. MongoDB supports various forms of data. As a NoSQL database, MongoDB shuns the relational

[26] GNU AGPL v3.0
https://www.mongodb.com/

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 952172.

database's table-based structure to adapt JSON-like documents that have dynamic schemas, which it calls BSON. MongoDB is built for scalability, high availability and performance from single server deployment to large and complex multi-site infrastructures.

In the context of the CHAIMELEON Project, the database will be employed for *Data Lake* data and datasets management.

3.2.14. Git

Git[27] is a free and open-source distributed version control system designed to handle everything from small to very large projects with speed and efficiency. It is software for tracking changes in any set of files.

In the context of the CHAIMELEON Project, the database will be employed for *Source Code Repository*.

[27] Git https://git-scm.com/

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 952172.

4. Use Cases

This section refines the description of *Use Cases* presented at deliverable "*D3.3. Interim Platform Design*" and new ones are identified and described.

Use Cases define the life cycle of all interactions flow among the *Users (*see Section 2*)* and *Components (*see Section 3*)* of the repository. Most *actions* enabled for each *User Role* trigger these interaction flows, although other set *actions* also happen in the middle of the *Use Case* execution as interactive processes (user/component) which are required to conclude the whole life cycle.

The next subsections describe the most important Use Cases from the perspective of the functionlities (user actions) offered by the repository. For each Use Case, a Unified Modeling Language (UML) sequence diagram is presented where all interactions among Users and Components are specified and the User Actions are highlighted in red arrows.

4.1. Authentication & Authorization

In *CHAIMELEON Repository*, Any *User* must have a verified *User Identity (Proof of Identity)* and a set of *User Roles* assigned which enables them to execute *actions* to the repository. Depending on the assigned *Roles*, the *User* will be able to be authenticated and authorised by the specific *Component* which is in charge to trigger the whole *Use Case* associated with the *action*.

Regarding the *Authentication and Authorisation (AA)* processes in the CHAIMELEON Repository*, 6 m*ain use cases (see Figure 2) have been identified and defined.

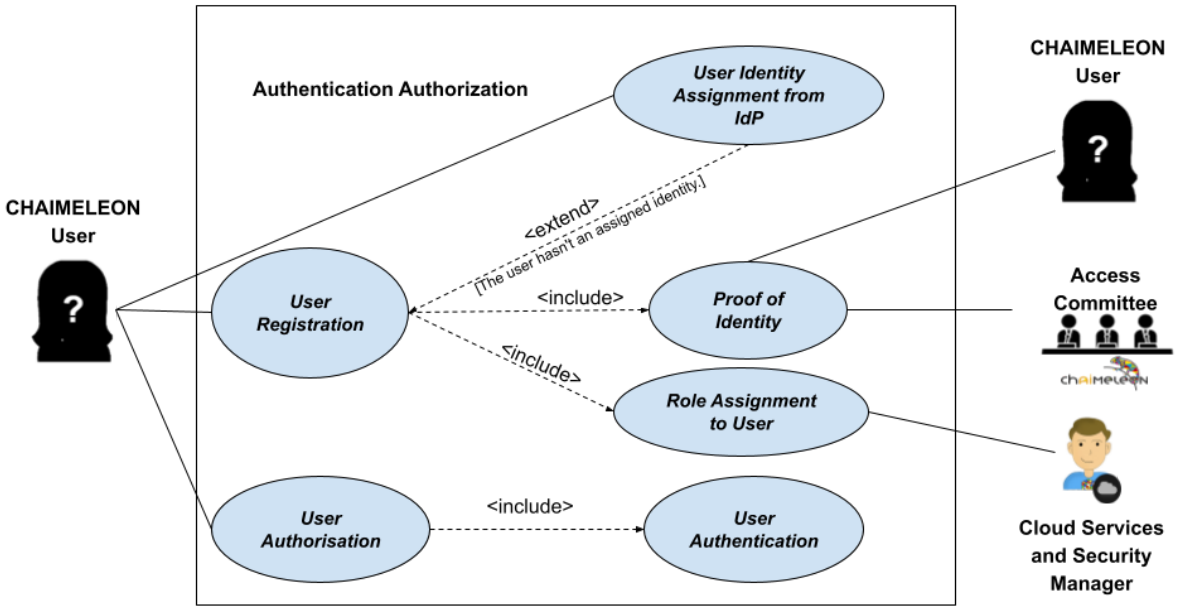


Figure 5. *UML Use Cases identified for Authentication and Authorisation Processes.*

4.1.1. User Identity Assignment from IdP

All users in CHAIMELEON Repository require a User Identity (User ID). A User ID is a logical entity used to authenticate a User on the repository. Today, most people acquire User Identities through external services named Identity Providers (IdPs). IdP is a service that creates, maintains, and manages identity information and provides authentication services.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 952172.

Thus, in the final design of the repository, *User IDs* provided by institutional IdPs associated with EGI Check-In (https://www.egi.eu/services/check-in/) and popular IdPs such as Google ID and Microsoft Active Directory are accepted. Also, the repository provides its own IdP (*CHAIMELEON IdP*), but it is only for *Users* who are not able to acquire identity from the aforementioned *IdPs*.

Figure 6 shows the UML sequence diagram corresponding to the interaction among an IdP service and *Users* to implement the acquisition of a *User ID*.

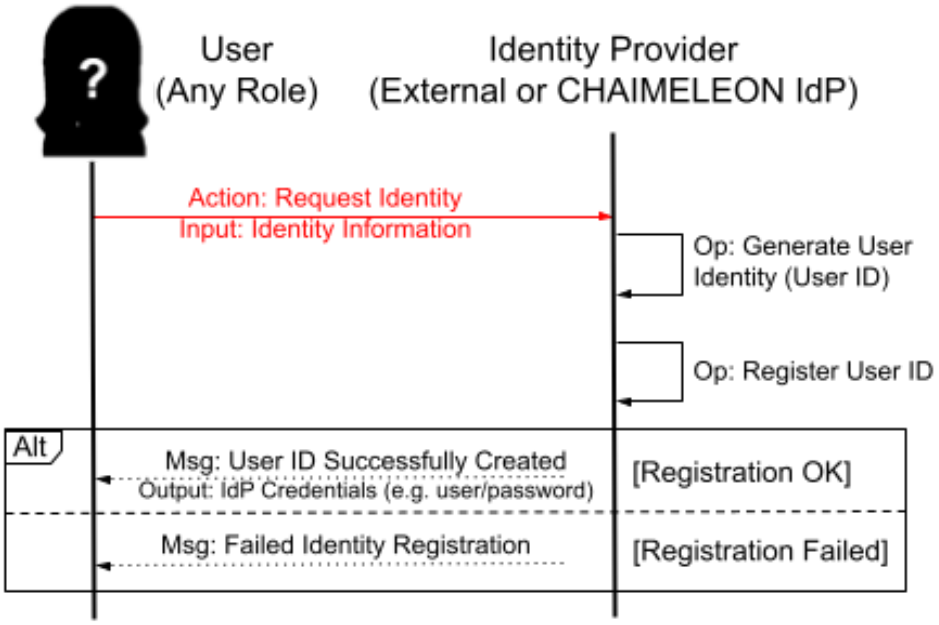


Figure 6. *UML Sequence Diagram for Identity Acquisition of Users.*

4.1.2. Proof of Identity

For the registration in the repository, the *Users* and the relation to their organisation must be proved through proof of identity. For the sake of liability, *CHAIMELEON Repository* will not provide access to generic accounts, only will provide access to users who have a *User ID* provided by the accepted IdP (see *User Identity Assignment from IdP Use Case*) and verified by an *Access Committee* composed of members from CHAIMELEON Consortium.

By binding the *User ID* to proof of identity, we guarantee that an account is linked to a specific real person and organisation, who must be liable to the actions performed in the repository. The proof of identity can be:

- Hard and preferred: A digital certificate issued by a trusted CA (e.g. TERENA).
- Soft: A copy of a passport or identity card.

Figure 7 shows the UML sequence diagram corresponding to the interaction flow among *Users* and *Access Committee* to implement the proof of identity process.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 952172.

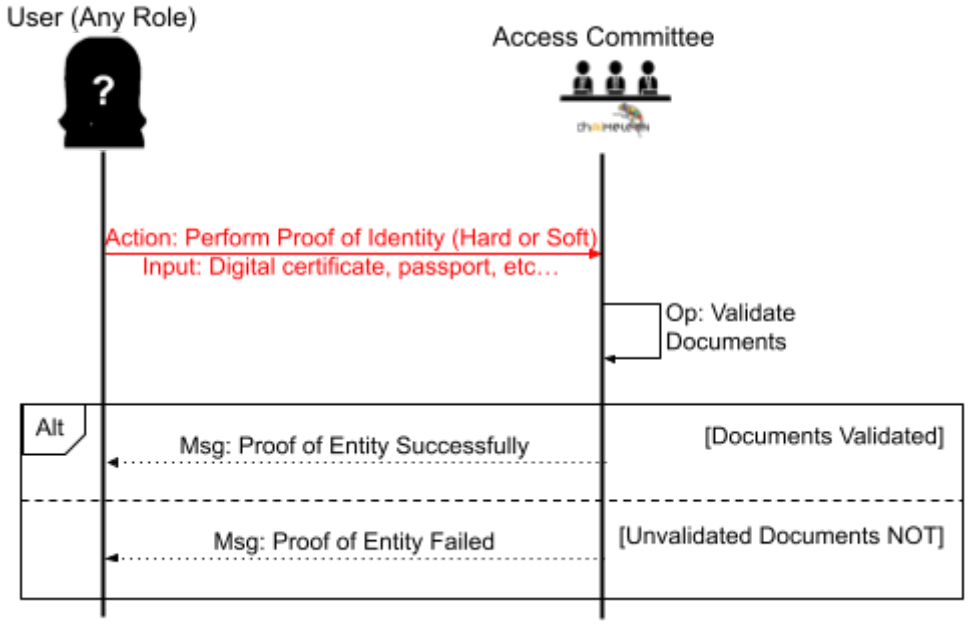


Figure 7. *UML Sequence Diagram for User Proof of Identity.*

4.1.3. User Authentication

Prior to the execution of any action by a *User* in the repository, the *User* must first be authenticated and then authorised to perform the action. The authentication process must be carried out by the *Component,* which triggers interaction flow among all repository *Components* involved in the action. Basically, in this process a verified *User ID* (see *Proof of Identity Use Case*) provided by an IdP(see *User Identity Assignment from IdP Use Case*) is checked to see if s(he) has previously been registered in the repository.

Figure 8 shows the UML sequence diagram corresponding to the interaction flow among *Users* and *Components* involved to implement any *User* authentication process.

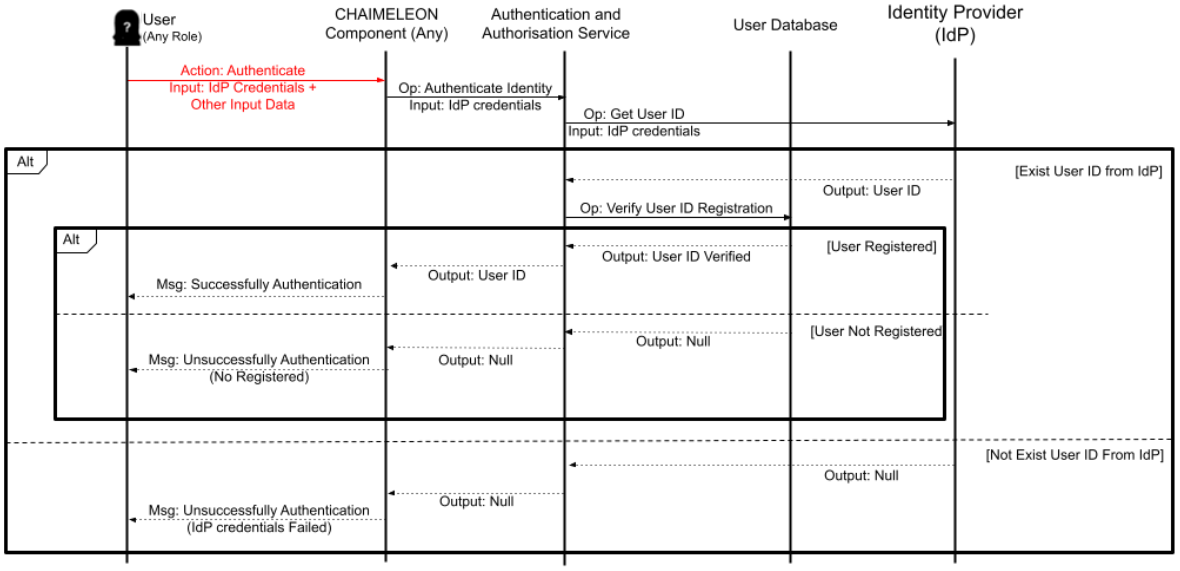


Figure 8. *UML Sequence Diagram for the User Authentication Use Case.*

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 952172.

4.1.4. User Authorisation

All *Users* must first be authenticated and then authorised to perform the action. The authorization process must be performed by the *Component* which triggers interaction flow among all repository components involved in the action and must be carried out depending on the presented authenticated identity (*User ID*) and its associated *User Roles*.

Figure 9 shows the UML sequence diagram corresponding to the interaction flow among *Users* and *Components* involved to implement any *User* authorisation process.

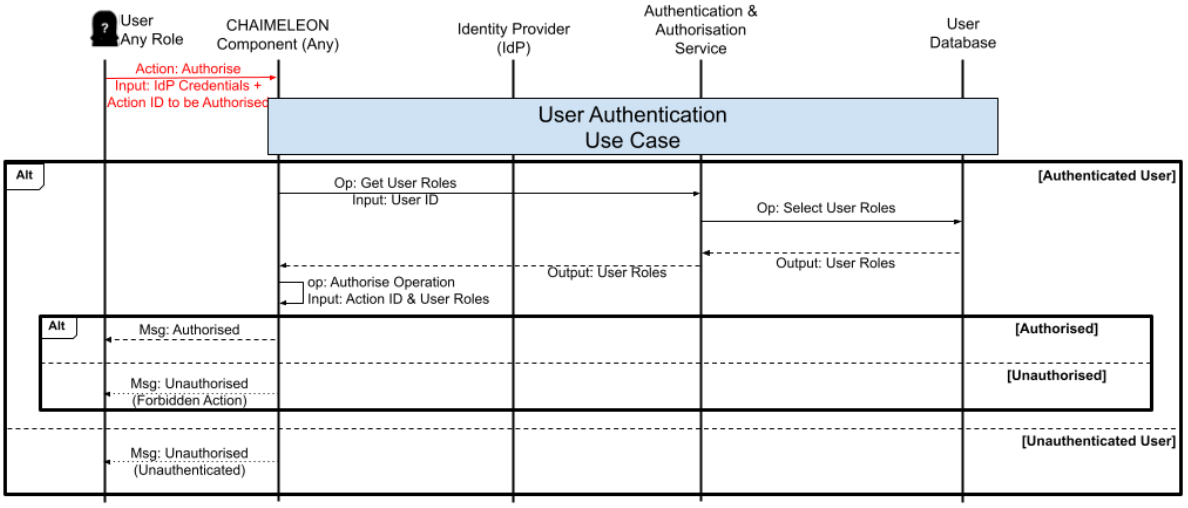


Figure 9. *UML Sequence Diagram for the User Authorisation Use Case.*

4.1.5. Role Assignment to Users

This use case describes how *Cloud Service and Security Manager* assigns roles to an existing (see *User Identity Assignment from IdP Use Case*) and verified user (see *Proof of Identity Use Case*). *Cloud Service and Security Manager* based on a list of actions provided by the user determines the roles to be assigned.

Figure 10 shows the UML sequence diagram corresponding to the interaction flow among *Users* and *Components* involved to implement the role assignment.

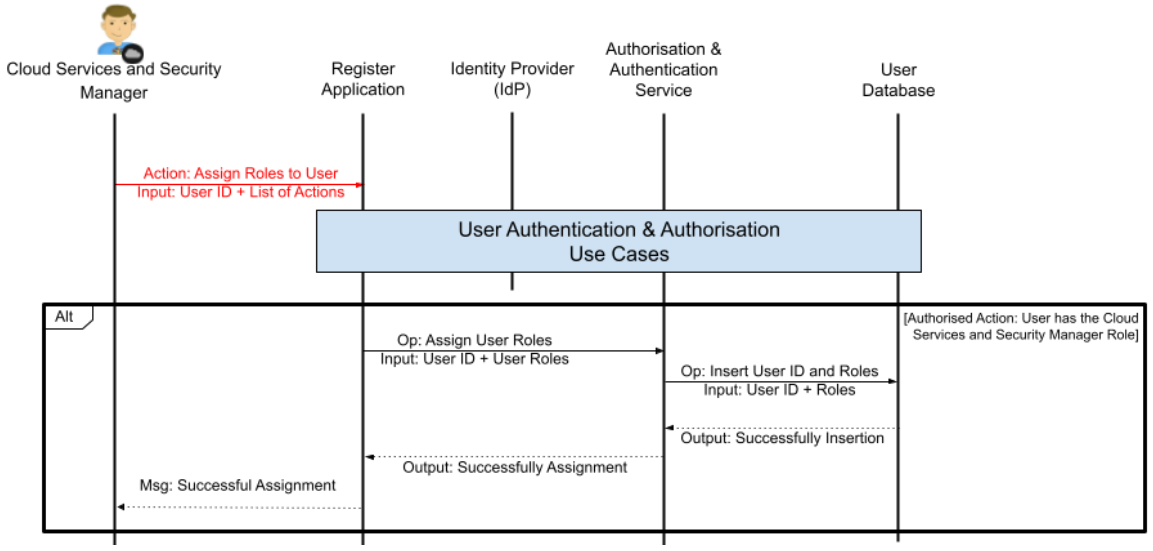


Figure 10. *UML Sequence Diagram for the Role Assignment to Users.*

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 952172.

4.1.6. User Registration

In the user registration process, users must present their *Identity* Credentials (see *User Identity Assignment from IdP Use Case*) previously verified through a proof of identity (see *Proof of Identity Use Case*) by the *CHAIMELEON Committee*. Then, they accept the *Terms of Usage* of the Repository (data usage) and provide the list of actions for which they want to be enabled.

The user registration must securely keep the successfully verified User IDs and their associated *User Roles* in the repository. The *User Roles* will be employed for enabling the execution of actions in the repository in the authorisation processes (see *User Authorisation Use Case*)

Figure 11 shows the UML sequence diagram corresponding to the interaction flow among *Users* and *Components* involved to implement the registration of a *User* including the proof of identity and the assignment of *User Roles.*

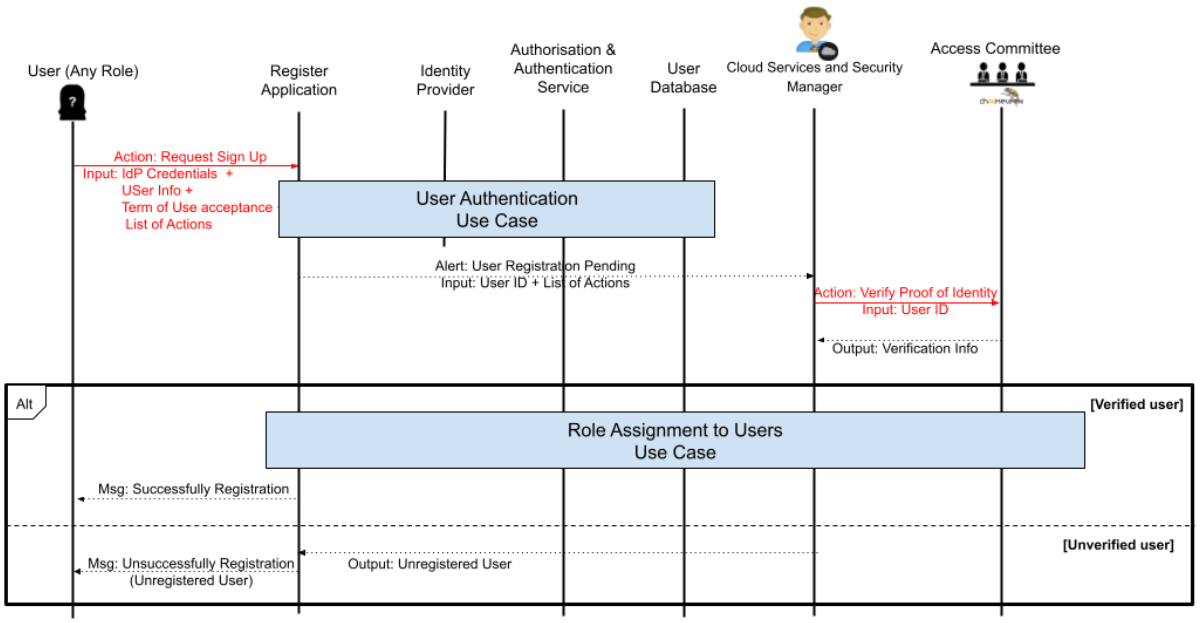


Figure 11. UML *Sequence Diagram for the Registration Process to CHAIMELEON Repository.*

4.2. Data Lake Management

In the context of the *CHAIMELEON Repository*, all Patient Cases (medical images and associated clinical data) required to create *Datasets* research objects have to be managed inside the boundaries of the repository. This management is through the *Data Lake* component where all these data are ingested, stored and consulted.

Regarding *Data Lake* management, 4 main use cases (see Figure 12) have been identified. The *Use Cases* are described in the next subsections.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 952172.

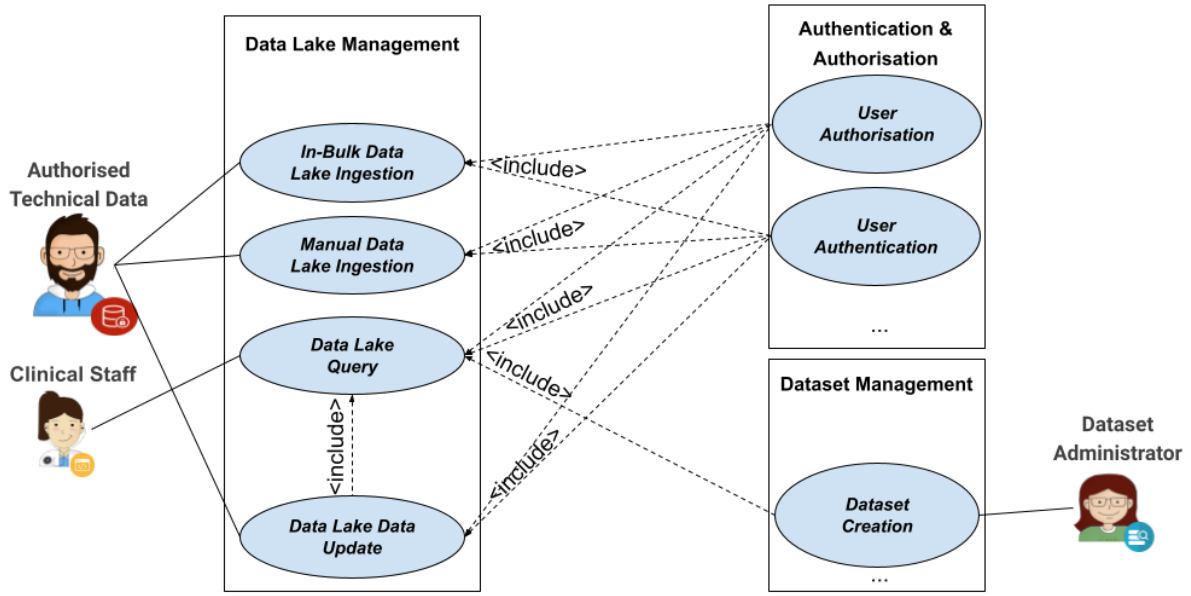


Figure 12. *UML Use Cases identified for Data Lake Management.*

4.2.1. In-Bulk Data Lake Ingestion

This *Use Case* defines the required interactions to ingest patient data (medical images and associated clinical data) into the *Data Lake* in bulk**.** All medical data to be ingested into a *Data Lake* must have been anonymised and curated in the clinical centre previously to the ingestion. Therefore, this *Use Case* takes care of ingesting completed cases.

The first one, User delegates his/her Credentials to an *in-bulk ingestion application* (Medexprim Suite™ *Component*) installed at the medical centre where the real observational and research data sources are located. In-bulk ingestion applications gather all medical data from the different involved sources (such as PACS, RIS or other Clinical Data databases). Only users with the *Authorised Technical Data Manager Role* can delegate their credentials to in-bulk ingestion applications. Finally, the *Data Ingestion/Access Service* inserts all medical data into the *Data Lake.*

Figure 13 shows the UML sequence diagram corresponding to the interaction flow among *Users* (through in-bulk application with delegated credentials by an *Authorised Technical Data Manager)* and *Components* involved to ingest data in bulk to the *Data Lake*.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 952172.

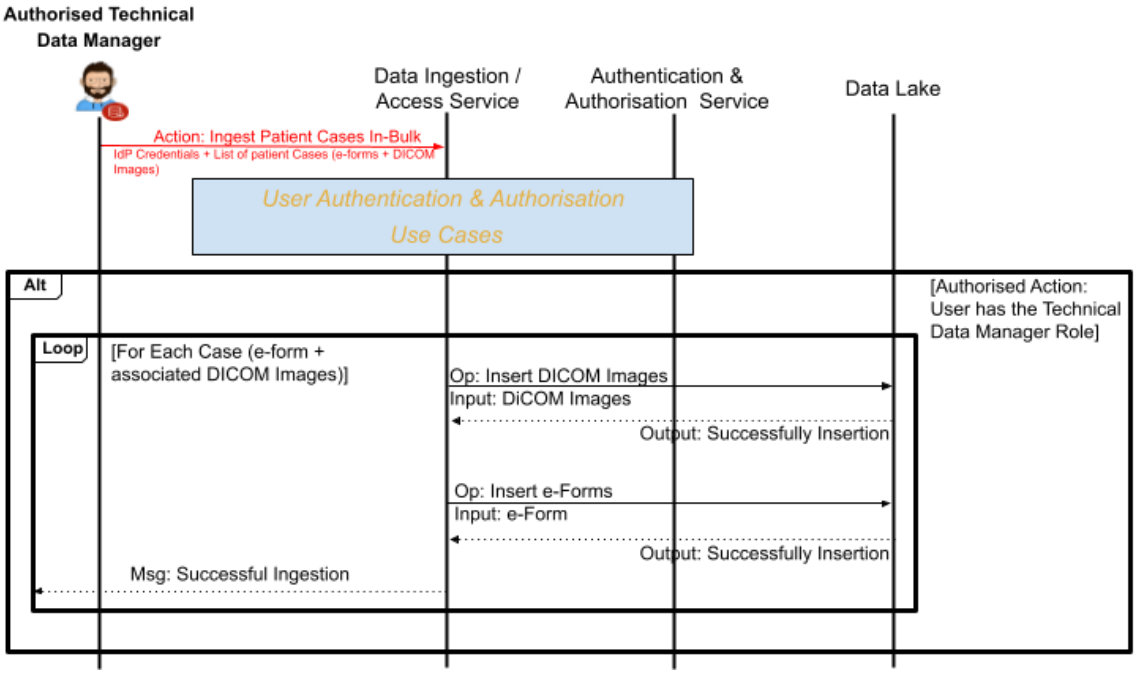


Figure 13. *UML Sequence Diagram for In-Bulk Data ingestion.*

4.2.2. Manual Data Lake Ingestion

This *Use Case* defines the required interactions to ingest Patient Cases into *Data Lake* manually. This scenario happens when an *Authorised Technical Data Manager* user needs to ingest new data corresponding to an enrichment process of a given dataset (see *Enriched Dataset Creation Use Case*) or a *Dataset* importation process from an external institution (*External Dataset Importation Use Case*).

The first step is to authenticate and authorise *Users* for carrying out the ingestion of the patient cases. Only users with an authenticated *Authorised Technical Data Manager Role* will be authorised by the *Case Explorer Application* to ingest data to *Data Lake* manually in the name of their organisations. In the second step, the *Case Explorer Application* will get the medical data from the *User* and will invoke the *Data Ingestion/Access Service, which will* insert this data into the *Data Lake*.

Figure 14 shows the UML sequence diagram corresponding to the interaction flow among *Users* and *Components* involved to ingest the data into the *Data Lake*.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 952172.

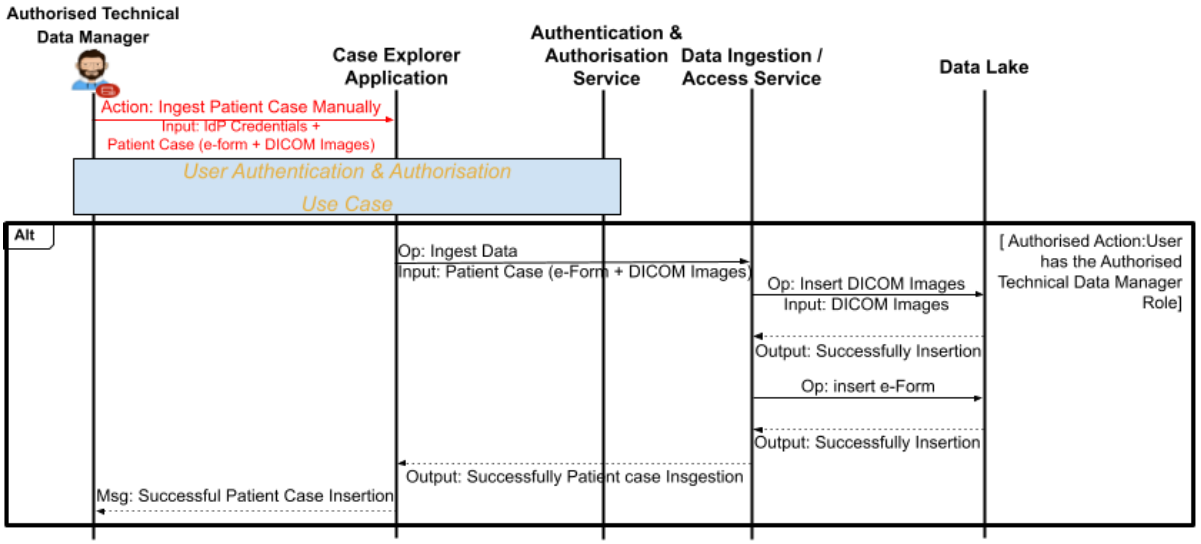


Figure 14. *UML Sequence Diagram for Manual Patient Case Data Ingestion.*

4.2.3. Data Lake Query

This *Use Case* defines the required interactions to query patient data from the *Data Lake*. This use case happens prior to updating *Data lake* data (see *Update Data Lake Data Use Case*) or the creation of *Data Sets* (see *Dataset Creation Use Case*) when a preliminary selection of cases kept in the *Data Lake* is required.

The first step is to authenticate and authorise the *User.* Only users with *ClinicalStaff Role*, *Dataset Administrator Role* or *Authorised Technical Data Manager Role* will be authorised by the *Case Explorer Application* to query patient data to the *Data Lake*. In the second step, *Case Explorer Application* queries and filters data from the *Data Lake* through the *Data Ingestion/Access Service*. Finally, in the third step the service query to *Data Lake* and retrieve the results that are returned to the application.

Figure 15 shows the UML sequence diagram corresponding to the interaction flow among *Users* and *Components* involved to query data from the *Data Lake*.

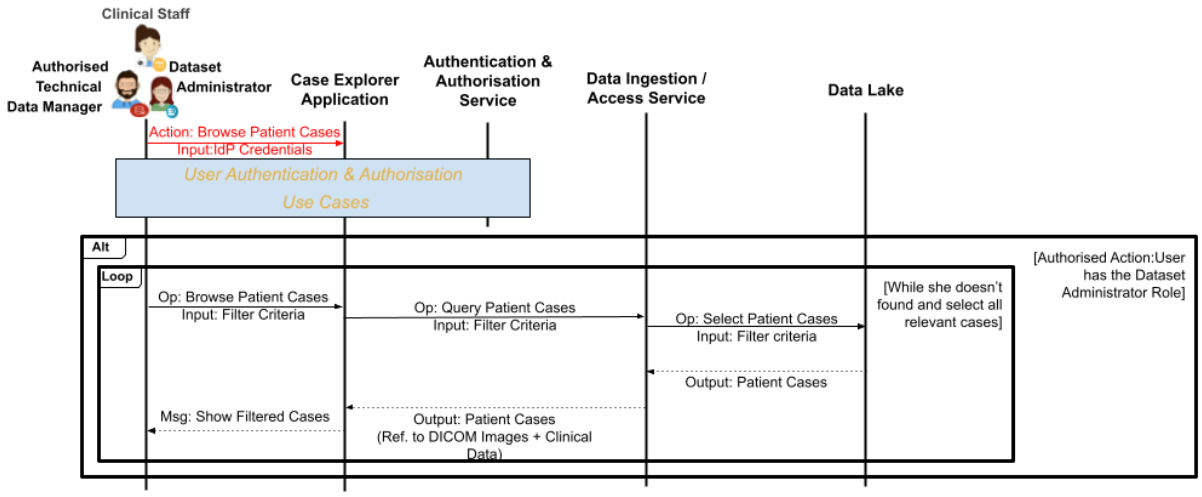


Figure 15. *UML Sequence Diagram for Data Lake Query.*

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 952172.

4.2.4. Update Data Lake Data

This *Use Case* defines the required interactions to update patient data that previously were ingested into *Data Lake* (see *In-Bulk Data Lake Ingestion Use Case* and *Manual Data Lake Ingestion Use Case*). Patient data to be updated are DICOM images or e-forms (clinical data). In the case of updating images, new images are uploaded to *Data Lake* while maintaining the old ones. This is to maintain consistency in the available *Datasets* (see *Dataset Creation Use Case*) as they use references to old images. However, in the case of updating fields of the e-forms, this is directly done at the *Data Lake* level. This is done because when *Datasets* are created, a copy of the involved e-forms is recorded, guaranteeing consistency. Finally, if data belonging to a *Dataset* has been involved in an update process, the *Dataset* must be marked as "source data updated".

Only users with *Authorised Technical Data Manager Role* authenticated will be authorised by the *Case Explorer Application* to update patient data from *Data Lake*. Then, the user looks for the patient cases to be updated/removed (See *Data Lake Query Use Case*). Next, *Case Explorer Application* Updates *Data Lake* data through the *Data Ingestion/Access Service*, which updates accordingly such data in the *Data Lake*. Finally, the *Dataset Service* gets the notification of a change, so it can mark the affected datasets.

Figure 16 shows the UML sequence diagram corresponding to the interaction flow among *Users* and *Components* involved to update *Data Lake* patient data.

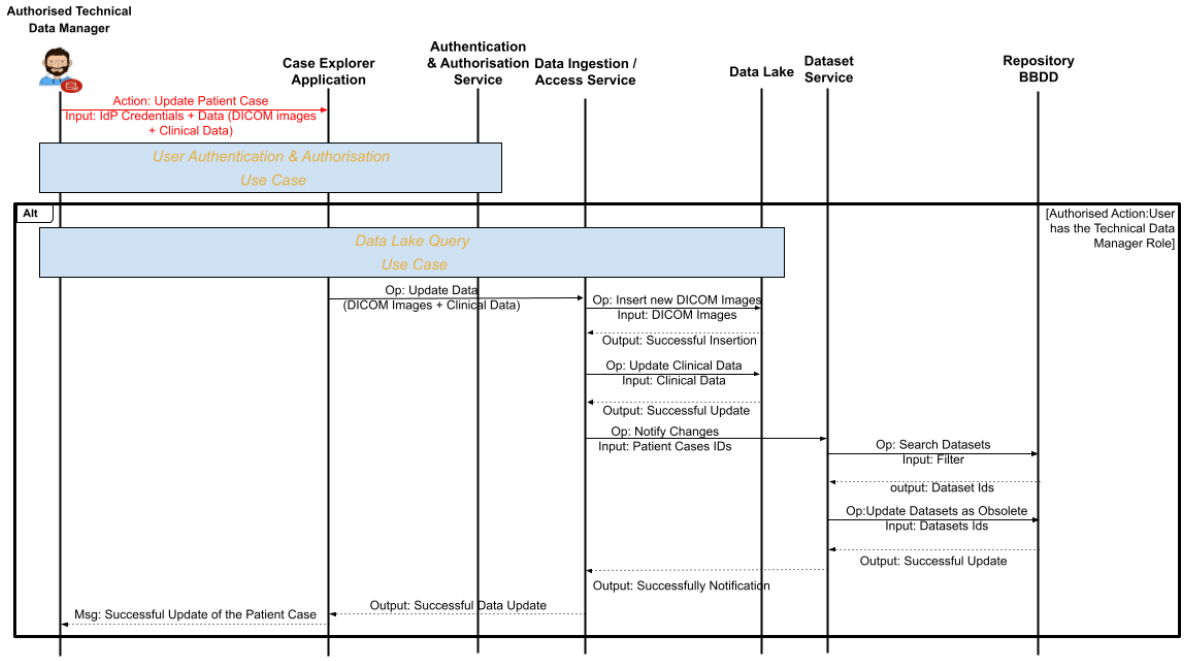


Figure 16. *UML Sequence Diagram for Updating Data Lake Data.*

4.3. Dataset Management

In the context of the *CHAIMELEON Repository*, *Datasets* are research objects managed in the repository and are composed of a set of data stored in the Data Lake. *Datasets* are employed by *Data Scientists* and *External Researchers* to develop AI *Models. Datasets* aim to give better traceability about what medical data is employed to build a given *Model* and enable reproducibility in the development of processing methods and AI Tools.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 952172.

Regarding *Datasets* management, 6 main *Use Cases* (see Figure 17) have been identified. The *Use Cases* are described in the next subsections.

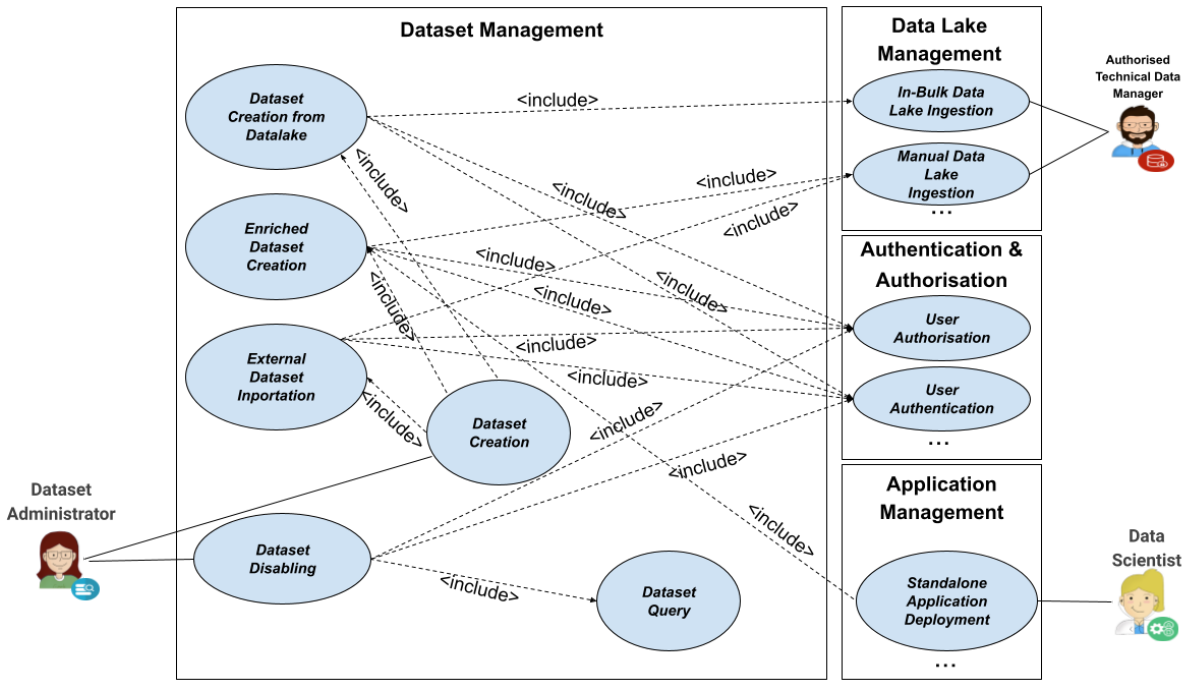


Figure 17. *UML Use Cases identified for Dataset Management.*

4.3.1. Dataset Creation

This *Use Case* defines the required interactions to create *Datasets*. Prior to creating a *Dataset*, an *Authorised Technical Data Manager* uploads the required data to the *Data Lake* (see *Dataset Creation From Data Lake Use Case*, *Enriched Dataset Creation Use Case* or *External Dataset Importation Use Case*).

Dataset Administrator Role is authenticated and authorised by the *Case Explorer Application* in this scenario**.** The *Case Explorer Application* creates a *Dataset* using the selected data through the *Dataset Service***.** The *Dataset Service* inserts the new *Dataset* in the *Repository Database* and registers the creation process (Metadata of the *Dataset*) through the *Tracer Service* for its traceability. In this process, the *Tracer Service* will collect some metadata about the *Dataset*, including anonymity checks and other data quality metrics that could be implemented.

Figure 18 shows the UML sequence diagram corresponding to the interaction flow among *Users* and *Components* involved to create *Datasets*.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 952172.

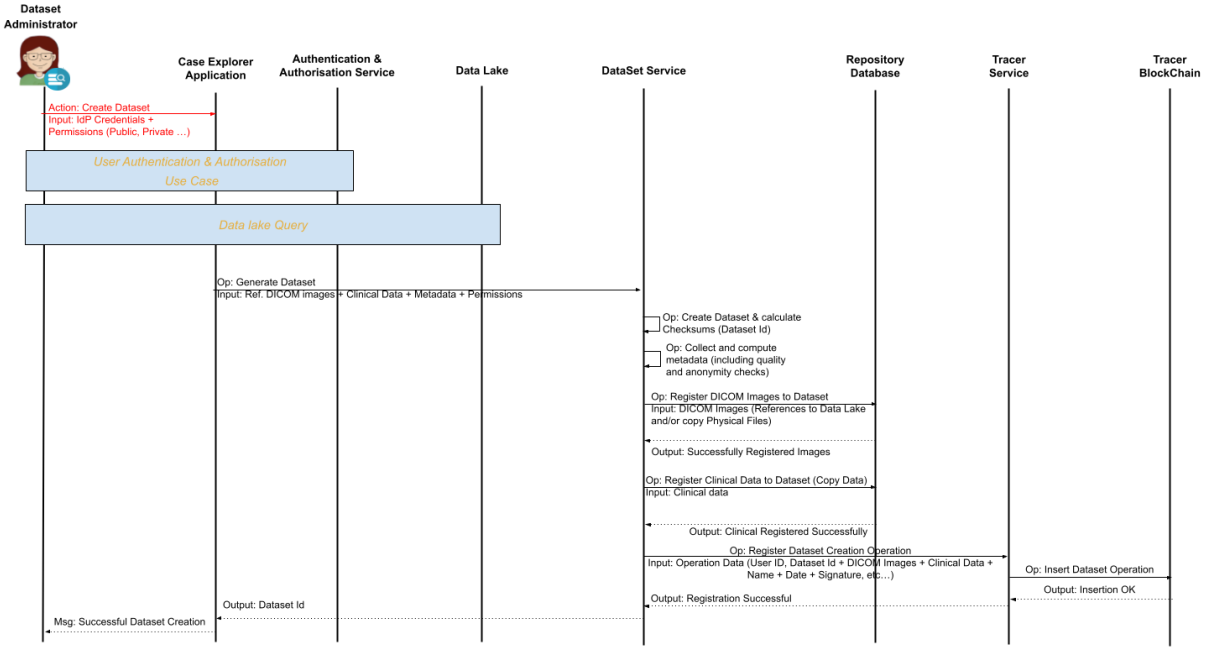


Figure 18. *UML Sequence Diagram for Creating Datasets.*

4.3.2. Dataset Creation From Data Lake

This *Use Case* defines the required interactions to create *Datasets* from patient data stored in the *Data Lake* Storage of the repository. In this scenario, data are previously anonymised and curated in the clinical centre previously to the ingestion in the *Data Lake* (*In-Bulk Data Lake Ingestion Use Case*).

First, a *Dataset Administrator Role is* authenticated and authorised by the *Case Explorer Application* and selects data from *Data Lake (*see *Data Lake Query Use Case) through Case Explorer Application*. Then, s(he) creates the Dataset *(Dataset Creation Use Case)*.

Figure 19 shows the UML sequence diagram corresponding to the interaction flow among *Users* and *Components* involved to create *Datasets*.

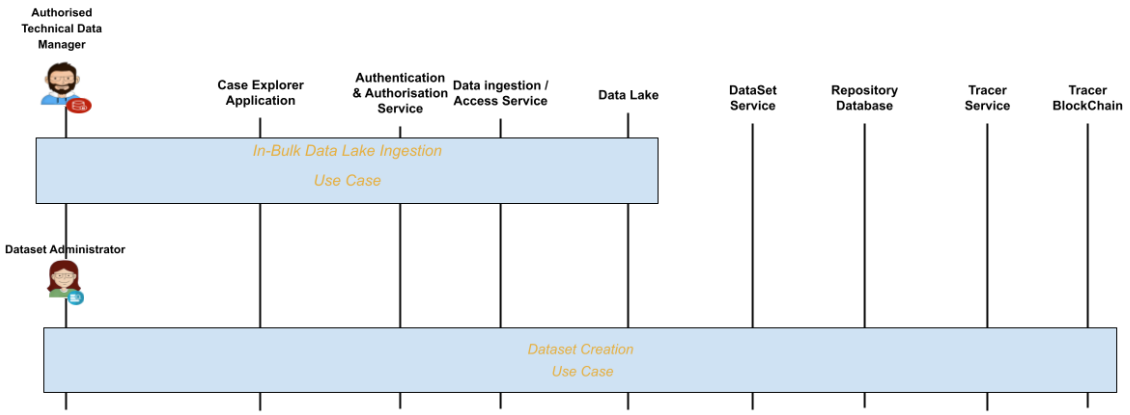


Figure 19. *UML Sequence Diagram for Creating Datasets* from *Data Lake* Data.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 952172.

4.3.3. Enriched Dataset Creation

This *Use Case* defines the required interactions to create a new enriched *Dataset* based on an existing *Dataset* of the repository. Data is provided from existing *Datasets* that have been enriched with new data (DICOM images or Clinical Data) through a *Standalone Application* (e.g. Analytical Engine or Harmonisation Processes).

First, a *Data Scientist* deploys a *Standalone Application* (*Standalone Application Deployment USe Case*) connected to an existing *Dataset*. The *Standalone Application* will be equipped with tools for enriching data (e.g. harmonisation tools). Then, the *Data Scientist* enriches the *Dataset* and prepares all data for creating a new Dataset. Next, an *Authorised and Technical Data* user inserts new data into the *Data Lake* (see *Manual Data Lake Ingestion Use Case),* then *a Dataset Administrator* creates the *Dataset* including the new data (see *Dataset Creation Use Case*).

Figure 20 shows the UML sequence diagram corresponding to the interaction flow among *Users* and *Components* involved to create enriched *Datasets*.

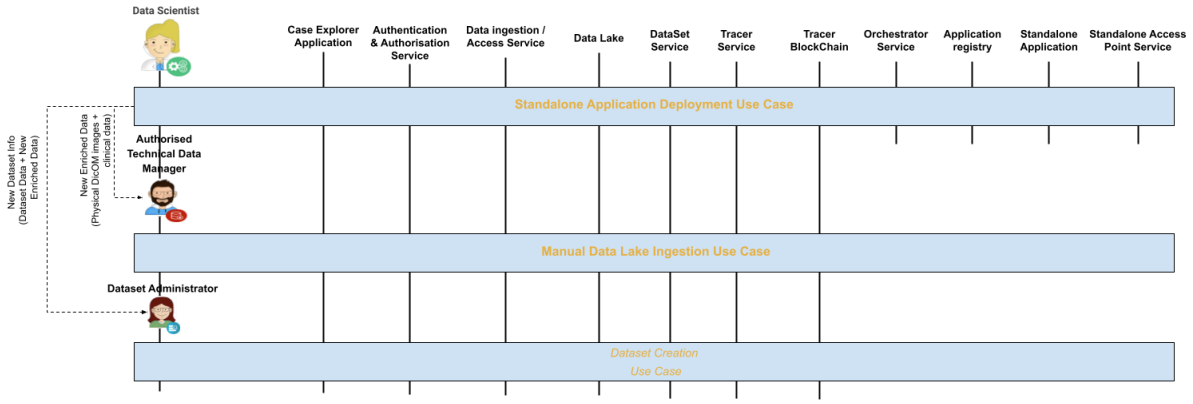


Figure 20. *UML Sequence Diagram for Creating Enriched Datasets.*

4.3.4. External Dataset Importation

This *Use Case* defines the required interactions to create a *Dataset* based on existing *Datasets* provided by an external institution.

Previous to this Use Case, the *User* acquires and prepares all data from an external institution for creating a new *Dataset* to import the repository. First, the *Authorised and Technical Data Manager* manually ingests all data into the *Data Lake* (see *Manual Data Lake Ingestion Use Case).* Next *Dataset Administrator* creates the *Dataset* (see *Dataset Creation Use Case*)

Figure 21 shows the UML sequence diagram corresponding to the interaction flow among *Users* and *Components* involved to import *Datasets*.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 952172.

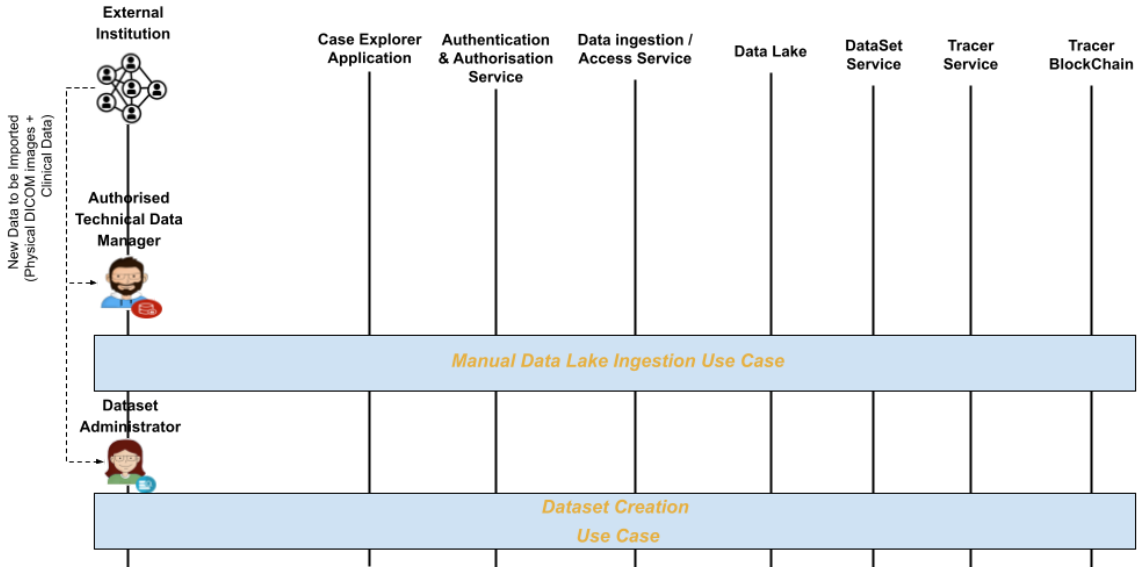


Figure 21. *UML Sequence Diagram for Importing Datasets.*

4.3.5. Dataset Query

This *Use Case* defines the required interactions to query *Datasets* from the *Repository Database***.** This *Use Case* happens prior to disabling a *Dataset* (see *Dataset Disablement Use Case*) or the deployment of *Processing Applications* (see *Standalone Application Deployment Use Case*) when a preliminary selection of a given *Dataset* kept in the *Repository Database* is required.

The first one is to authorise the *User.* Only users with an authenticated *Dataset Administrator Role, Data Scientist Role, Application Developer Role or External Researcher Role* will be authorised by the *Dataset Explorer Application* to query *Datasets* to the *Repository Database***.** In the second step**,** *Dataset Explorer Application* queries and filters *Datasets* from the *Repository Database* through the *Dataset Service***.** Finally, in the third step, the service queries the Repository *Database* and retrieves the results that are returned to the *Dataset Explorer Application*.

Figure 22 shows the UML sequence diagram corresponding to the interaction flow among *Users* and *Components* involved to query *Datasets*.

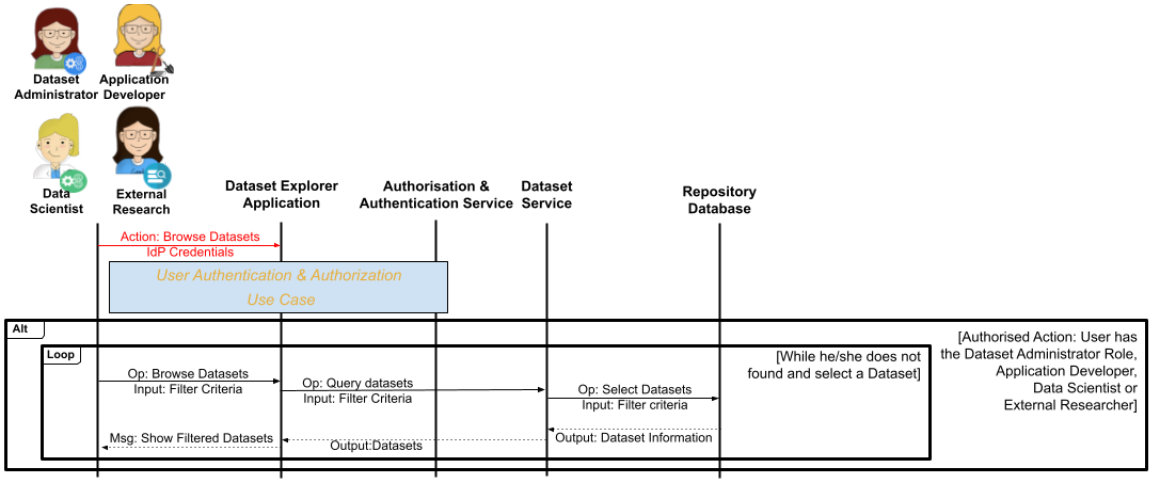


Figure 22. *UML Sequence Diagram for Dataset Query.*

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 952172.

4.3.6. Dataset Disablement

This *Use Case* defines the interactions required to disable a given *Dataset* from the *Repository Database*. Some of the reasons why a *Dataset* can be disabled are because its data has been revoked or wrongly created, to name a few.

First, User selects a given *Dataset (Dataset Query Use Case*). Only users with *Dataset Administrator Role* authenticated will be authorised by the *Dataset Explorer Application* in this scenario. Then, the *Dataset Explorer Application* will disable the selected *Dataset* through the *Dataset Service* that will update the *Dataset* as Disabled**.** Finally, the *Dataset Service* registers the action for its traceability through the *Tracer service*.

Figure 23 shows the UML sequence diagram corresponding to the interaction flow among *Users* and *Components* involved to disable a given *Dataset*.

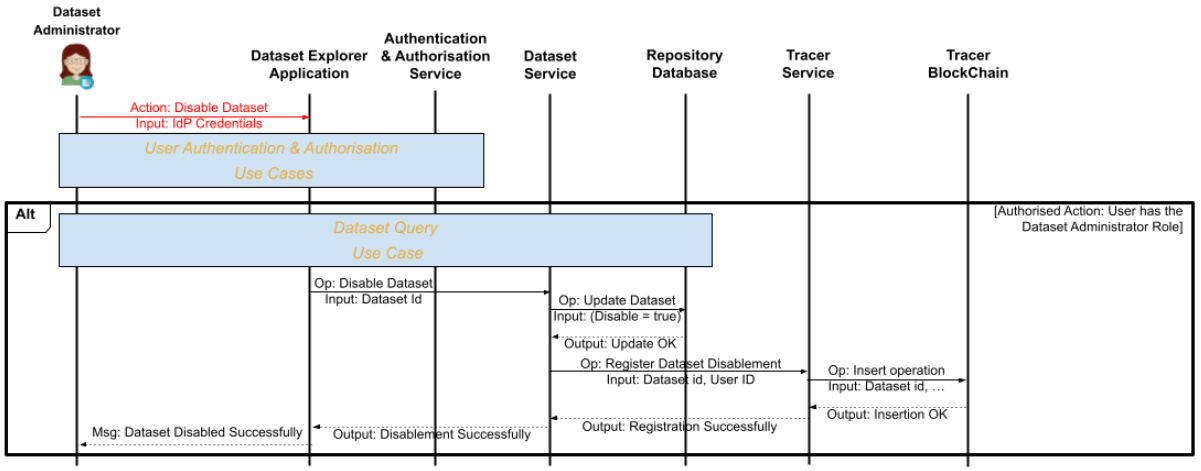


Figure 23. *UML Sequence Diagram for Disabling a Dataset.*

4.4. Processing Application Management

In the context of the *CHAIMELEON Repository*, there are two types of *Processing Applications*, these are *Standalone Applications* and *Processing or AI Applications.*

A *Standalone Application* is an application from the *Application Dashboard* catalogue that runs independently and connects to a *Dataset* volume, providing *Users* the means to carry out exploring and processing existent *Datasets* of the *CHAIMELEON Repository* for creating new AI *Models* or enriched *Datasets*. Some examples of this type of application are analytical engines (provided by BAHIA), harmonisation tools (provided by Imperial College and QUIBIM), tools for ensuring that the Images harmonised are secure (provided by BGU) and tools and frameworks for AI developers such as Jupyter server with Tensorflow or Pytorch.

Processing or AI Applications are processing applications or trained AI models embedded as applications to extract knowledge from medical images and clinical data to fine diagnosis, prognosis, follow-up of treatments and so on. These types of *Processing Applications* are associated with *Processing or AI Tools* managed by the *Marketplace* of the *CHAIMELEON Repository*.

Regarding *Processing Application Management*, 3 main *Use Cases* (Figure 24) have been defined. The *Use Cases* are described in the next subsections.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 952172.

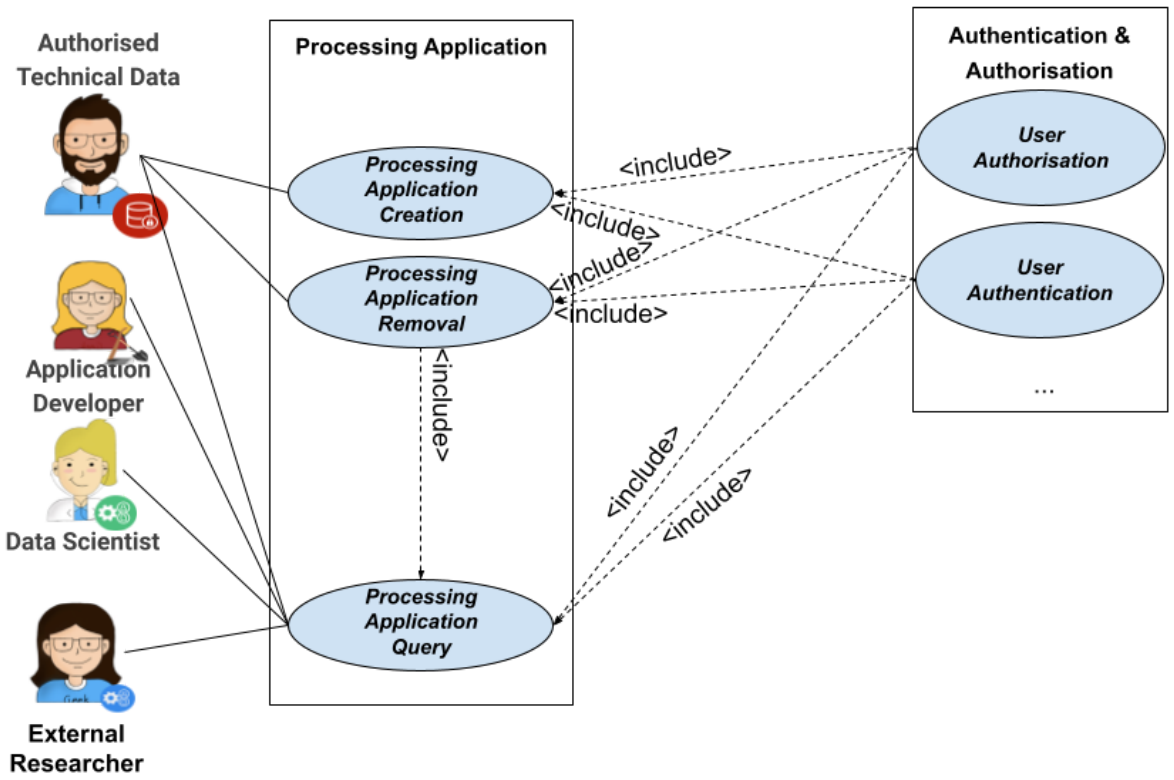


Figure 24. *UML Use Cases Identified for Processing Application Management.*

4.4.1. Processing Application Creation

This *Use Case* defines the required interactions to create and register any type of *Processing Application* (*Standalone Applications* or *Processing or AI Applications)* in the *CHAIMELEON Repository*.

Before creating and registering the *Processing Application*, the *Cloud Service and Security Management Role* is in charge of collecting all the requirements (software, hardware and configuration) to create the environment for the *Processing Application*.

Only the *Cloud Service and Security Manager* role will be authorised to add *Processing Applications to* the *Application Dashboard*, once checking that the components used by the *Processing Application* are secure. Then s(he) will register the *Processing Application* and store the application container image, required configuration and deployment parameters in the *Application Registry*.

Figure 25 shows the UML sequence diagram corresponding to the interaction flow among Users and Components involved to create and register a *Processing Application*.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 952172.

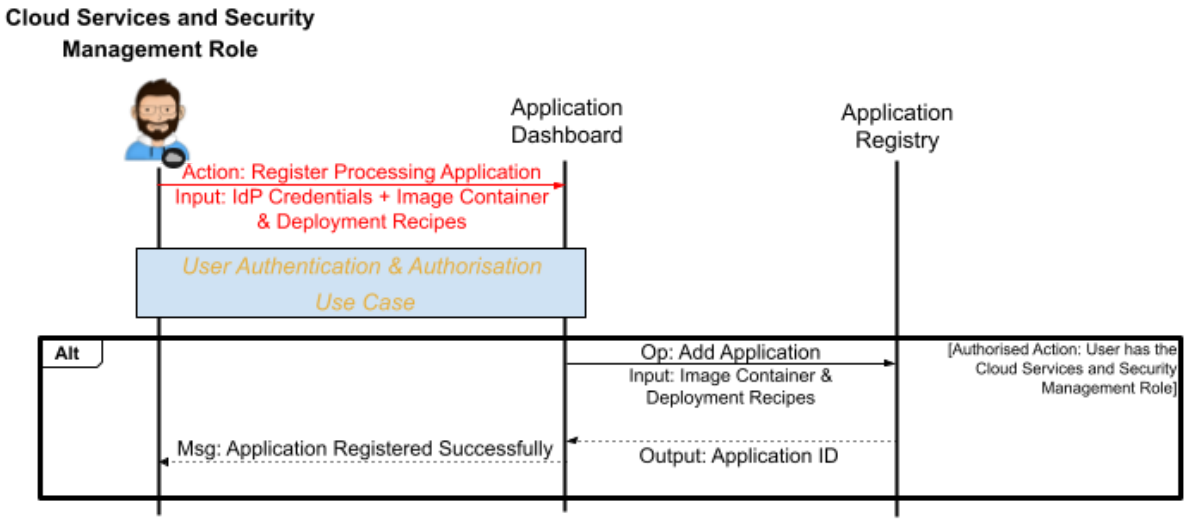


Figure 25. *UML Sequence Diagram for Processing Application Creation.*

4.4.2. Processing Application Removal

This *Use Case* defines the required interactions to remove and unregister any type of *Processing Applications* (*Standalone Applications* or *Processing or AI Applications)* in the *CHAIMELEON Repository*. Only the *Cloud Service and Security Manager* role will be authorised to remove *Processing Applications* to the Application Dashboard.

Figure 26 shows the UML sequence diagram corresponding to the interaction flow among *Users* and *Components* involved to remove and unregister a *Processing Application*.

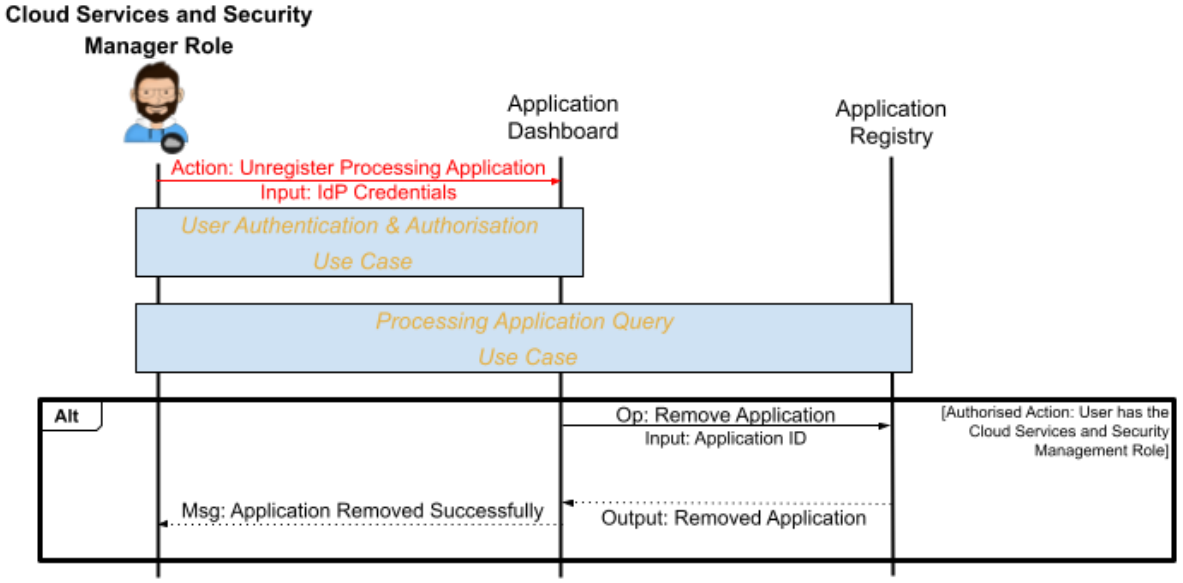


Figure 26. *UML Sequence Diagram for Processing Application Removal.*

4.4.3. Processing Application Query

This *Use Case* defines the required interactions to query any *Processing Applications* (*Standalone Applications* or *Processing or AI Applications)* from the *Application Registry*. This use case happens prior to creating/removing a *Processing Application (Processing Application Creation Use Case* or *Processing Application Removal Use Case*) or deploying/releasing *Standalone Applications* (*Standalone Application Deployment Use Case* and *Standalone Application Release Use Case)*. The first one is to authenticate and

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 952172.

authorise the *User.* Only users with *Data Scientist Role, Application Developers Role, External Researcher Role or Cloud Services and Security Manager* will be authorised by the *Application Dashboard* to query *Processing Applications* to the *Application Registry*. Next, *Application Dashboard* queries and filters applications from the *Application Registry*. Finally, it retrieves the results that are returned to the *Application Dashboard*.

Figure 27 shows the UML sequence diagram corresponding to the interaction flow among *Users* and *Components* involved to query *Processing Applications*.

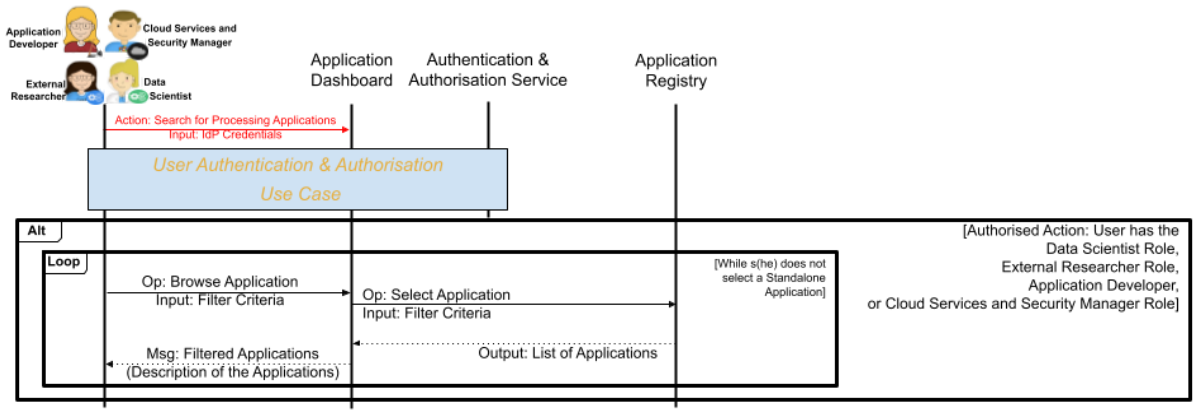
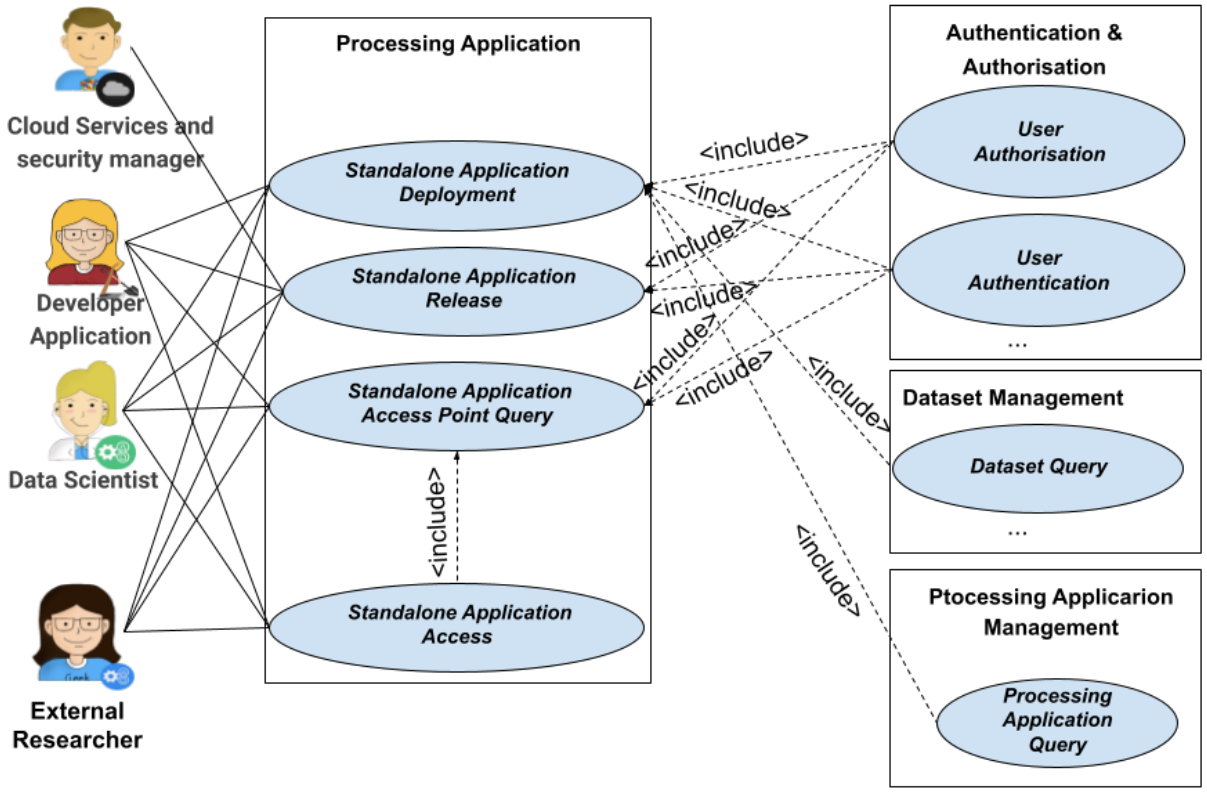


Figure 27. *UML Sequence Diagram for Standalone Application Query.*

4.5. Standalone Application Management

This section provides the specific uses cases related to the *Standalone Application*. 3 main *Use Cases* (Figure 28) have been defined. The *Use Cases* are described in the next subsections.



This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 952172.

Figure 28. *UML Use Cases Identified for Standalone Application Management.*

4.5.1. Standalone Application Deployment

This *Use Case* defines the interactions required to deploy a *Standalone Application* on the cloud resources provided by the *CHAIMELEON Repository*.

Before deploying a *Standalone Application* (first step), the execution of the *Dataset Query Use Case* for selecting a *Dataset id* must be produced. After that, authenticated users with the *Data Scientist Role, Application Developer Role* or *External Researcher Role* will be authorised by the *Application Dashboard* in this scenario. The *User* selects the application (see *Standalone Application Query Use Case*) and deploys it through the *Application Dashboard*. The *Application Dashboard* carries out the deployment through the *Orchestrator Service* that downloads the container image from the *Application Registry* and deploys the application and attaches a volume to access all data from the *Dataset* selected by the *User*. The information of the Data volume to mount in the *Standalone Application* is collected from the *Dataset Service* and the operation is registered in the *Tracer Service* for its traceability annotating the use of a given *Dataset* by a specific *Standalone Application* and a given *User*.

Figure 29 shows the UML sequence diagram corresponding to the interaction flow among *Users* and *Components* involved in the deployment of a *Standalone Application.*

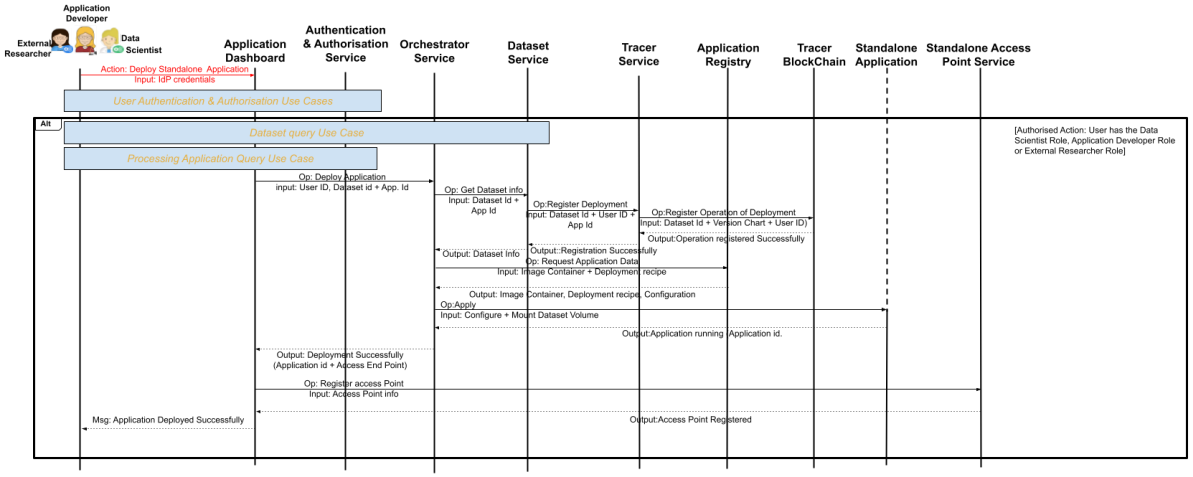


Figure 29. *UML Sequence Diagram for Standalone Application Deployment.*

4.5.2. Standalone Application Release

This *Use Case* defines the required interactions to destroy a running *Standalone Application* (see *Standalone Application Deployment Use Case*) and release the cloud resources used by the application.

Authenticated *Users* with the *Data Scientist Role*, *Application Developer Role* and *External Research Role* will be authenticated and authorised by the *Application Dashboard* in this scenario to destroy a *Standalone Application.* Applications deployed by a user can only be destroyed by the same user, except the *Cloud Services and Manager Role* who are able to destroy any application. The *User* selects the application to delete (see *Processing Application Use Case*) and destroys it through the *Application Dashboard*. The *Application Dashboard* carries out the release of the resources through the *Orchestrator Service*. The

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 952172.

back-end resources will be automatically provisioned and released on demand reacting to the workload managed by the orchestrator.

Figure 30 shows the UML sequence diagram corresponding to the interaction flow among *Users* and *Components* involved to release cloud resources and destroy Standalone Applications.

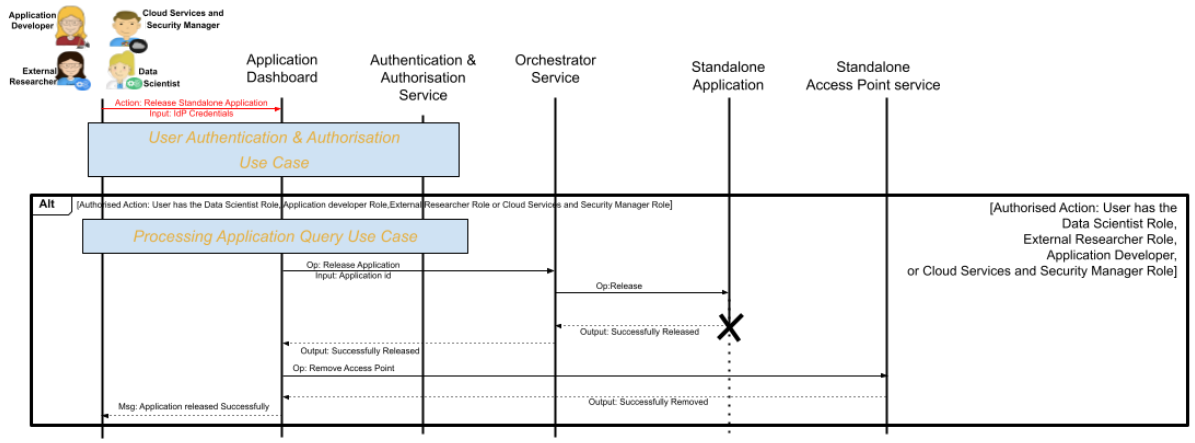


Figure 30. *UML Sequence Diagram for Standalone Application Release.*

4.5.3. Standalone Application Access Point Query

This *Use Case* defines the required interactions to get access to a *Standalone Application* running on the processing cloud resources provided by the *CHAIMELEON Repository*. All these applications must be deployed previously (see *Standalone Application Deployment Use Case*).

First, one is to authorise the *User.* Only authenticated users with the *Data Scientist Role*, *Application Developer Role* and *External Researcher Role* will be authorised by the *Application Dashboard.* The *Application Dashboard* will list the *Standalone Applications* deployed by the user, and then information about each one of them, including the access point, can be obtained. Internally, the *Application Dashboard* could query the access point or additional information to the *Orchestrator*.

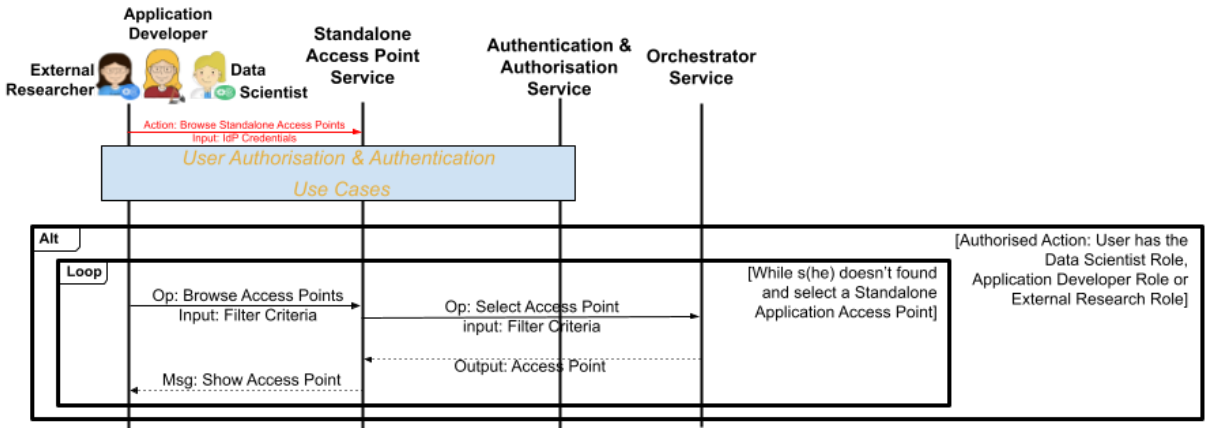


Figure 31. *UML Sequence Diagram for listing Standalone Application Access Points.*

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 952172.

Figure 31 shows the UML sequence diagram corresponding to the interaction flow among *Users* and *Components* involved in the listing of *Standalone Application* Access Points.

4.5.4. Standalone Application Access

This *Use Case* defines the interactions required to Access a *Standalone Application* that is running on cloud resources provided by the *CHAIMELEON Repository*.

Before accessing a *Standalone Application,* the access point and access credentials must be known (see the *Standalone Application Access Point Query Use Case*). After that, *Users* access their *Standalone Applications* using the access point and required credentials.

Figure 32 shows the UML sequence diagram corresponding to the interaction flow among *Users* and *Components* involved to access a *Standalone Application.*

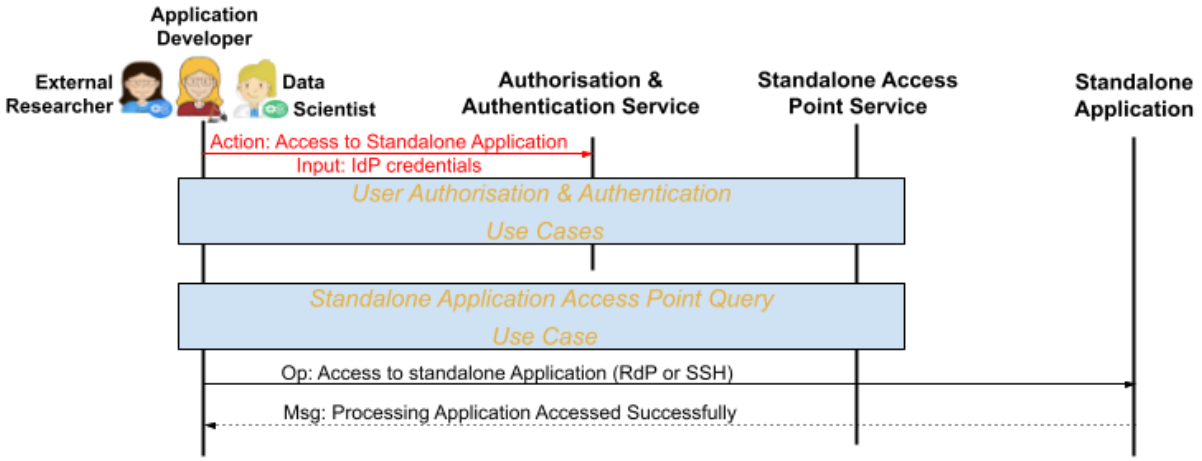


Figure 32. *UML Sequence Diagram for accessing Standalone Application.*

4.6. Model Management

In the context of the *CHAIMELEON Repository*, a *Model* is a research object created by *Data Scientists* or *External Researchers*. A model is an Artificial Intelligence program that has been trained employing *Datasets* provided by the repository. For developing the *Models*, a *Standalone Application* running on the repository cloud resources is used. *Models* can be published and employed by *Applications Developers* to build *Processing or AI Tools* for fine diagnostics, follow-on disease etc.

Regarding *Model Management*, 3 main *Use Cases* (see Figure 33) have been defined. The *Use Cases* are described in the next subsections.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 952172.

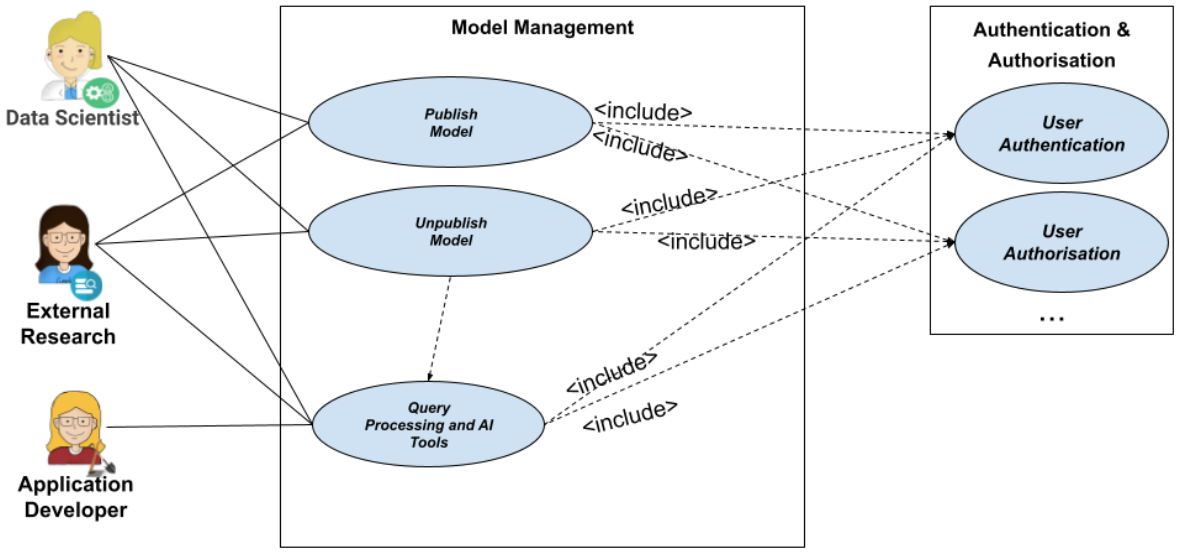


Figure 33. *UML Use Cases identified for Model Management.*

4.6.1. Publish Model

This *Use Case* defines the required interactions to publish a *Model* in the *CHAIMELEON Repository*. Such *Models* will provide *Application Developers* with trained AI *Models* which can be embedded as applications (*Processing or AI Applications*) in the marketplace (see *Publish Processing or AI Tools Use Case*).

Prior to the publication of a *Model,* a *Data Scientist or External Researcher* deploys a *Standalone Application* (*Standalone Application Deployment USe Case*) which has access to a set of *Datasets* of the repository and provides all required AI frameworks (e.g. TensorFlow) to train *the AI Models*. Then, the user creates the model and prepares all data for its publication. The *Model* is published in the *Source Code Repository* by authenticated and authorised users (Only users with *Data Scientist Role* or *External Researchers)*, providing the information about the *Model (*e.g. employed Datasets, authors etc...*)*. The operation of publishing a model by a given user is registered by the *Tracer Service* for its traceability.

Figure 34 shows the UML sequence diagram corresponding to the interaction flow among *Users* and *Components* involved in the publishing of a *Model*.

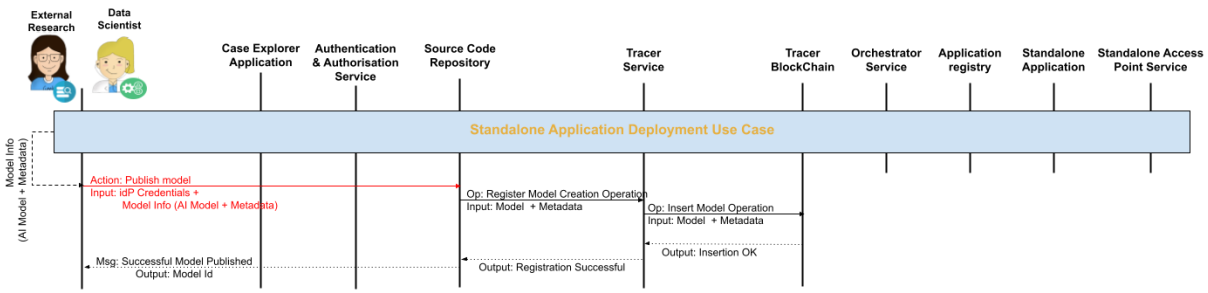


Figure 34. *UML Sequence Diagram for Publishing a Model.*

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 952172.

4.6.2. Unpublish Model

This *Use Case* defines the required interactions to unpublish a *Model* in the *CHAIMELEON Repository*.

Models will be unpublished to the *Source Code Repository* by authenticated and authorised users (Only users with *Data Scientist Roles* or *External Researchers)*. First, the user chooses a *Model* (see Model Query Use Case). Next, the *Model* is unpublished from the *Source Code Repository*.

Figure 35 shows the UML sequence diagram corresponding to the interaction flow among *Users* and *Components* involved in the publishing of a *Model*.

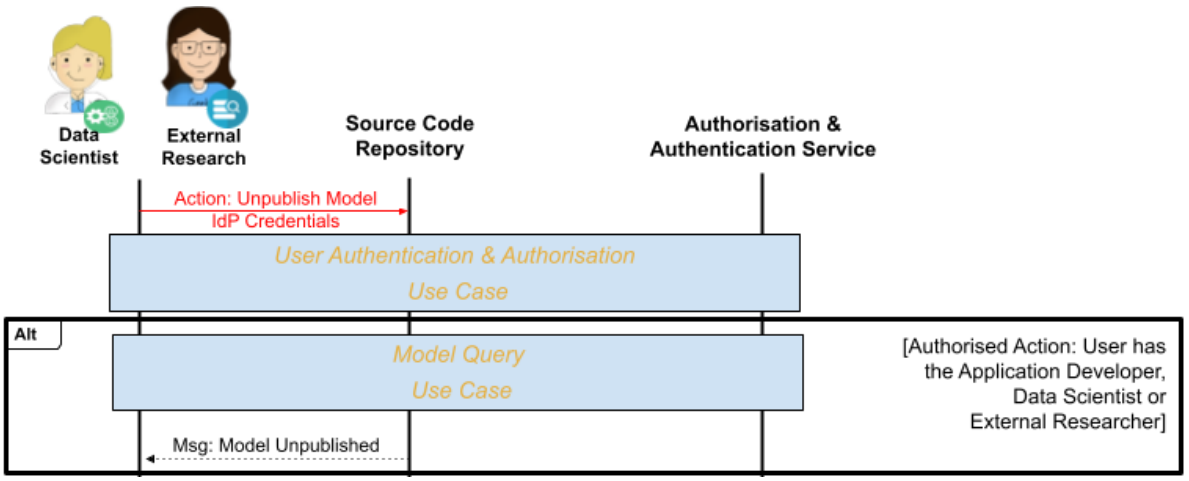


Figure 35. *UML Sequence Diagram for Unpublishing a Model.*

4.6.3. Model Query

This *Use Case* defines the required interactions to query *Models* stored and published at the *Source Code Repository*.

The first one is to authorise the *User.* Only users with an authenticated *Data Scientist Role, Application Developer Role or External Researcher Role* will be authorised by the *Source Code Repository* to query *Models*.

Figure 36 shows the UML sequence diagram corresponding to the interaction flow among *Users* and *Components* involved to query *Models*.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 952172.

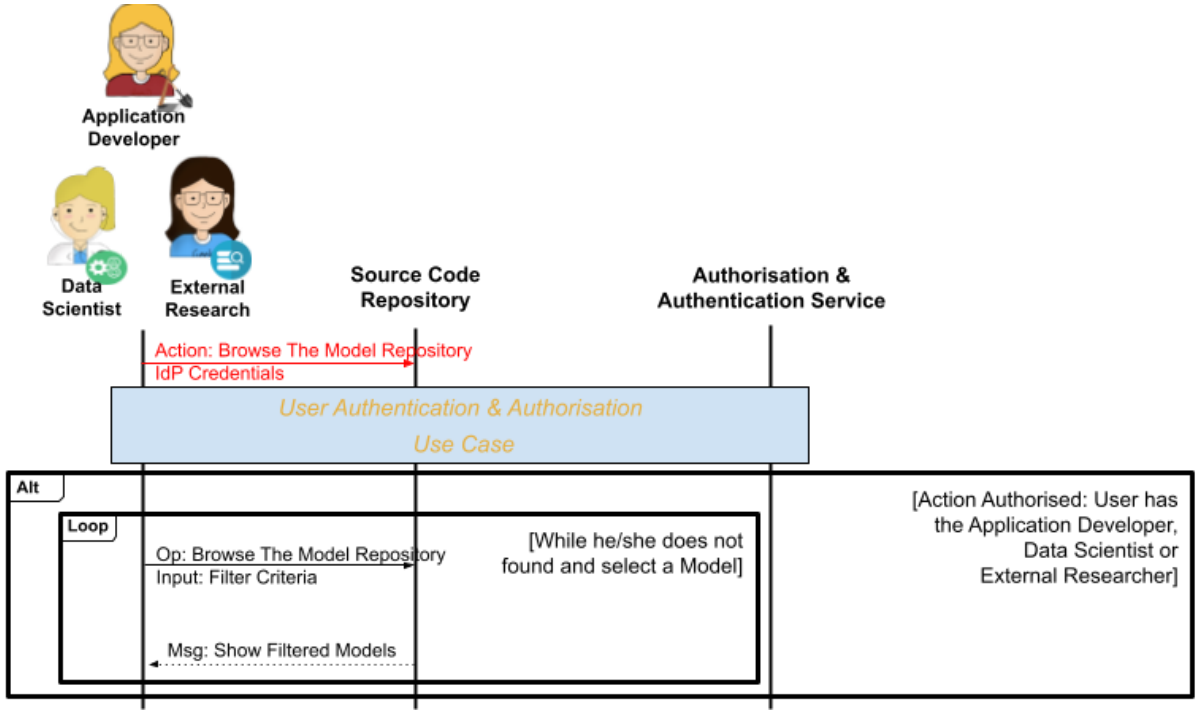


Figure 36. *UML Sequence Diagram for Consulting Models.*

4.7. Marketplace Management

Marketplace provides to users a set of *Processing or AI Tools* based on existing *Models* developed by *Data Scientists*. In the context of *CHAIMELEON Repository*, the *Processing or AI Tools* will be created by *Application Developers* using existing *models* Datasets created by the Data Scientists.

Regarding Marketplace *Management*, 4 main *Use Cases* (Figure 37) have been defined. The *Use Cases* are described in the next subsections.

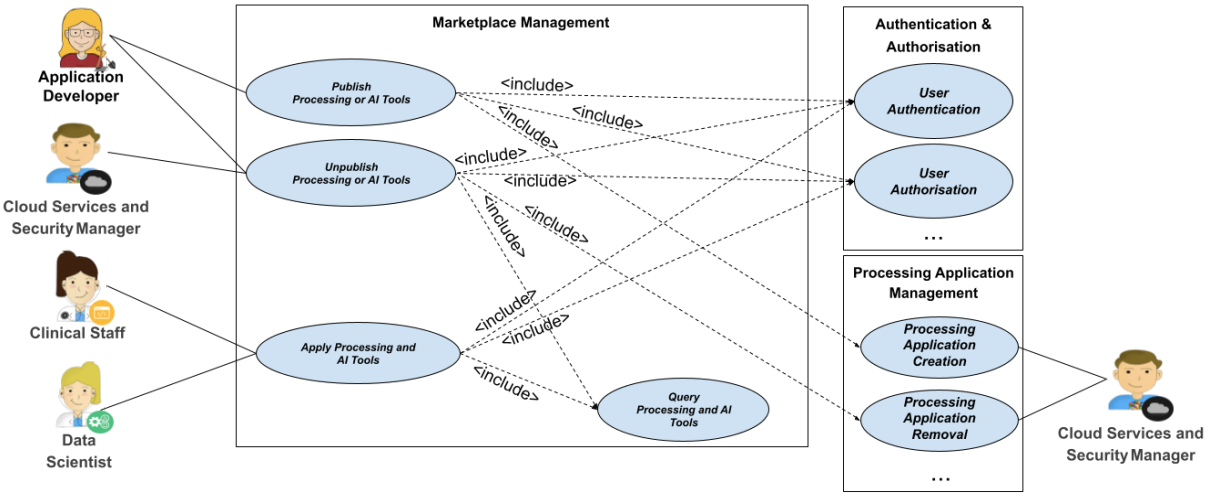


Figure 37. *UML Use Cases identified for Marketplace Management.*

4.7.1. Publish Processing or AI Tools

This *Use Case* defines the required interactions to develop and publish *Processing or AI Tools* in the *CHAIMELEON Repository*. Such *Tools* will provide *Users* with the means to extract knowledge to assist in research, diagnosis, prognosis and treatment follow-on. In the

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 952172.

context of the project, these *Tools* could be processing tools or trained AI models embedded as applications (*Processing or AI Applications*).

Tools will be integrated through a *Marketplace* by authenticated and authorised users (Only users with *Application Developer Role)*, providing the information about the execution arguments and providing a user-friendly interface to end-users for its execution. The computational processes of these *Tools* will be executed through new *Processing or AI Applications which* will require the *Cloud Services and Security Manager* intervention to the creation of their specifications (image containers and recipes) and register them in the *Application Registry* of the repository (*Processing Application Creation Use Case*).

Figure 38 shows the UML sequence diagram corresponding to the interaction flow among *Users* and *Components* involved in the publishing of *Processing or AI tools*.

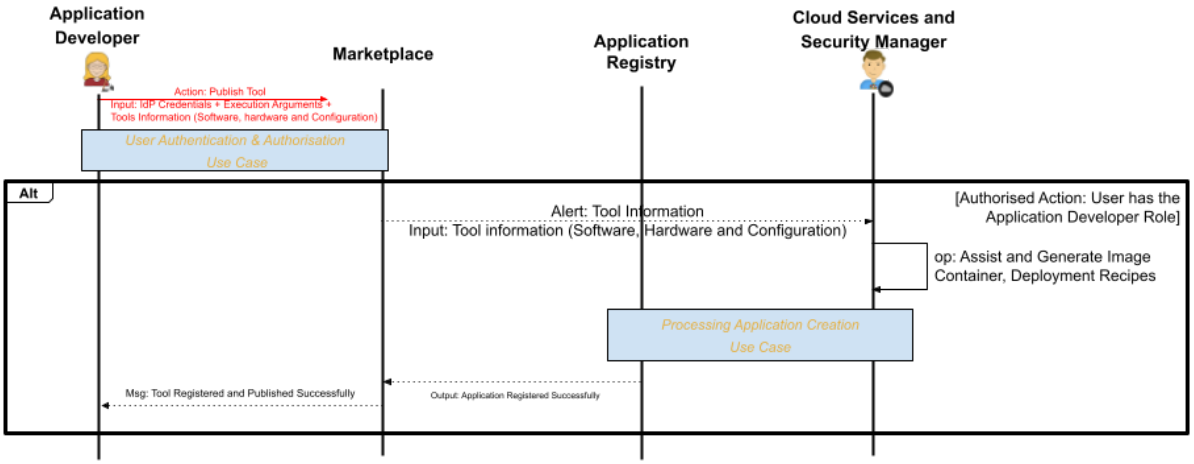


Figure 38. *UML Sequence Diagram for publishing Processing or AI tools.*

4.7.2. Unpublish Processing or AI Tools

This *Use Case* defines the required interactions to unpublish a *Processing or AI Tools* from Marketplace.

Authenticated *Users* with the *Application Developer Role* will be authenticated and authorised by the *Marketplace* in this scenario to unpublish a Tool. Tools deployed by a user can only be unpublished by the same user, except the *Cloud Services and Manager Role,* who are able to unpublish any tool. The *User* selects the tool to unpublish (see *Processing or AI Tool Query Use Case*) and unpublish it through the *Marketplace*. Then, the associated *Processing or AI Application* is maintained in the Application registry for its traceability.

Figure 39 shows the UML sequence diagram corresponding to the interaction flow among *Users* and *Components* involved in the unpublishing of *Processing or AI tools*.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 952172.

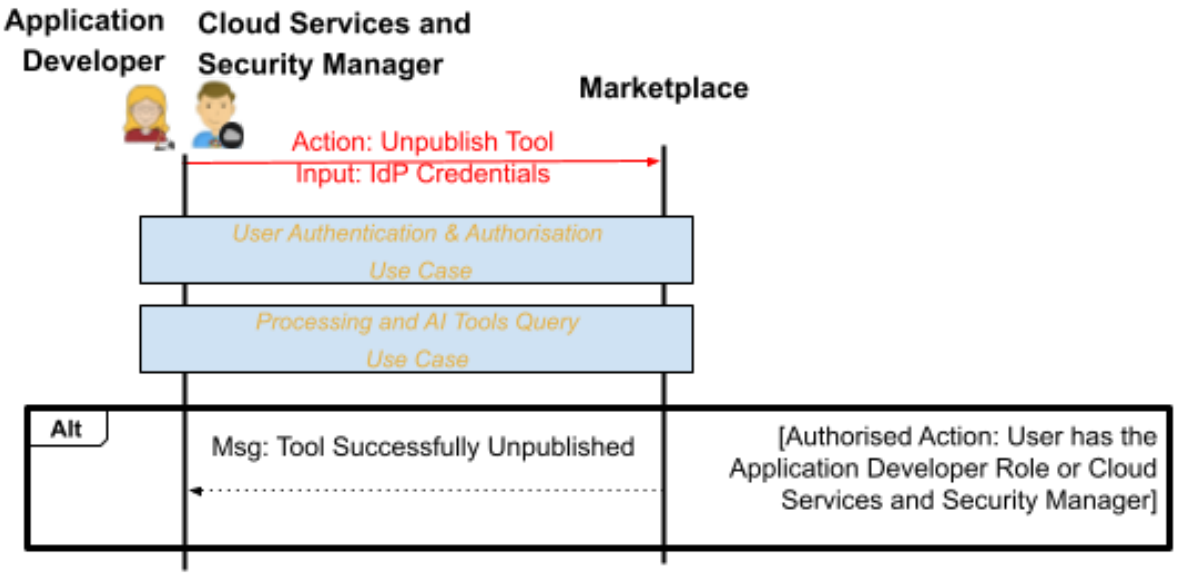


Figure 39. *UML Sequence Diagram for publishing Processing and AI Tools.*

4.7.3. Processing or AI Tool Query

This *Use Case* defines the required interactions to query *Processing and AI Tools* published in the *Marketplace*. This use case happens prior to using a tool (see *Apply Processing and AI Tools Use Case*). The first one is to authenticate and authorise the *User.* Only users with the *Data Scientist Role, Clinical Staff Role, External Researcher Role or Cloud Services and Security Manager Roles* will be authorised by the *Marketplace* to query the published tools. In the second step, the *Marketplace* queries and filters the tools.

Figure 40shows the UML sequence diagram corresponding to the interaction flow among *Users* and *Components* involved to list *Processing and AI Tools*.

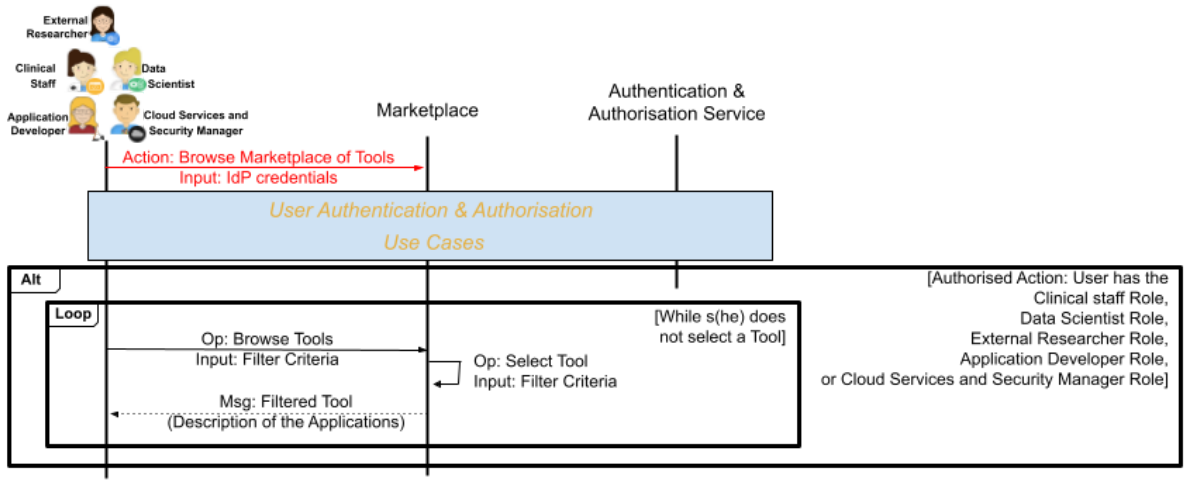


Figure 40. *UML Sequence Diagram for the processing or AI Tool Query.*

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 952172.

4.7.4. Apply Processing or AI Tool

This *Use Case* defines the required interactions to use the *Processing and AI tools* provided by the *Marketplace* of the *CHAIMELEON Repository* to assist on research, diagnosis, prognosis and treatment follow-on.

The first one is to authenticate and authorise the *User.* Only users with *External researcher Role, Application Developer Role*, *Data Scientist Role or Clinical Staff Roles* will be authorised by the *Marketplace* to use the tools previously published (*Publish Processing and AI Tools use case*). Then, the *User* through *Marketplace* queries and filters *Tools* (see *Processing and AI Tool Query use Case*) and selects one (e.g. the computation of the angiogenesis for evaluating the aggressiveness of a tumour). Next, the *User* selects its own or an existing case to analyse and after that, the *Marketplace* runs the associated process (*Processing or AI Application*) through the *Orchestrator Service.* This may require fetching the application image and the additional files required from the *Application Registry* and then running the associated *Processing or AI Application*. Finally, the Marketplace collects the results from the *Processing or AI Application* and shows them to the *User* through the *Marketplace*. When the process finishes and results are displayed by the *Marketplace*, the *Orchestrator Service* releases cloud resources of the associated *Processing or AI Application.*

Figure 41 shows the UML sequence diagram corresponding to the interaction flow among *Users* and *Components* involved to run *Processing / AI Tools*.

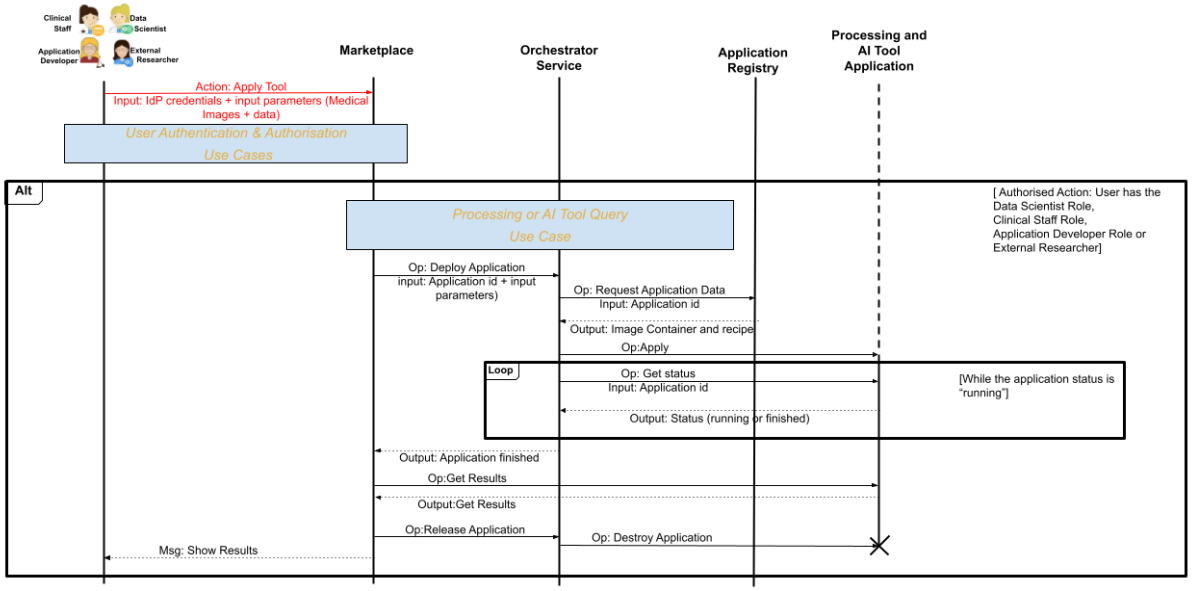


Figure 41. *UML Sequence Diagram for Applying processing or AI Tool.*

4.8. Tracing Data

This *Use Case* defines the required interactions to provide a traceability service to register and inspect the operations performed on a specific *Dataset* and *Models*. The objective of traceability is twofold. On the one hand, data providers can query the service to retrieve the users who have accessed the *Dataset* and the models trained on it. On the other hand, users

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 952172.

can use the tracking information on a public blockchain to verify that the model or data they are using is the one used in the registration of an object.

4 main *Use Cases* (Figure 42) have been defined. The *Use Cases* are described in the next subsections.

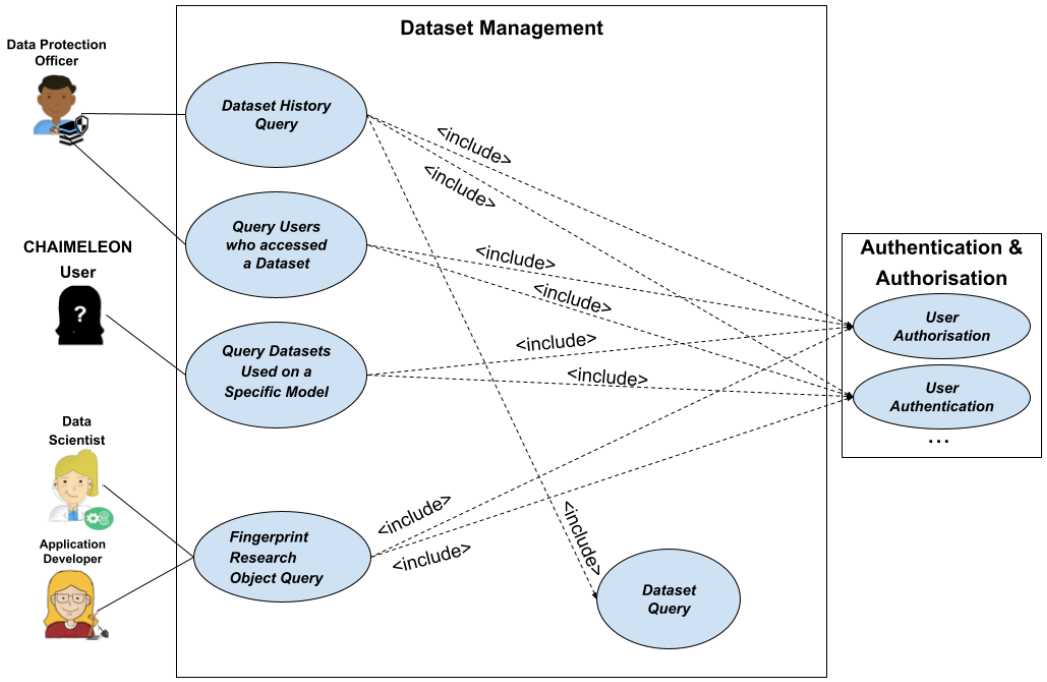


Figure 42. *UML Use Cases identified for Tracing Data Management.*

4.8.1. Dataset History Query

The *Dataset* history is the full set of operations performed on a given *Dataset*, such as the creation, the access and the models created using such a *Dataset*. This *Use Case* will be available only to the *Data Protection Officer* (DPO) who would like to monitor the access history of a given *Dataset*.

The first one is to authenticate and authorise the *User.* Only users with *Data Protection Officer Roles* will be authorised by the *Tracer*. Then, the *User* chooses a given *Dataset* (see *Dataset Query Use Case*) to consult the history through *Tracer Service.*

Figure 43 shows the UML sequence diagram corresponding to the interaction flow among *Users* and *Components* involved to query the *Dataset* history.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 952172.

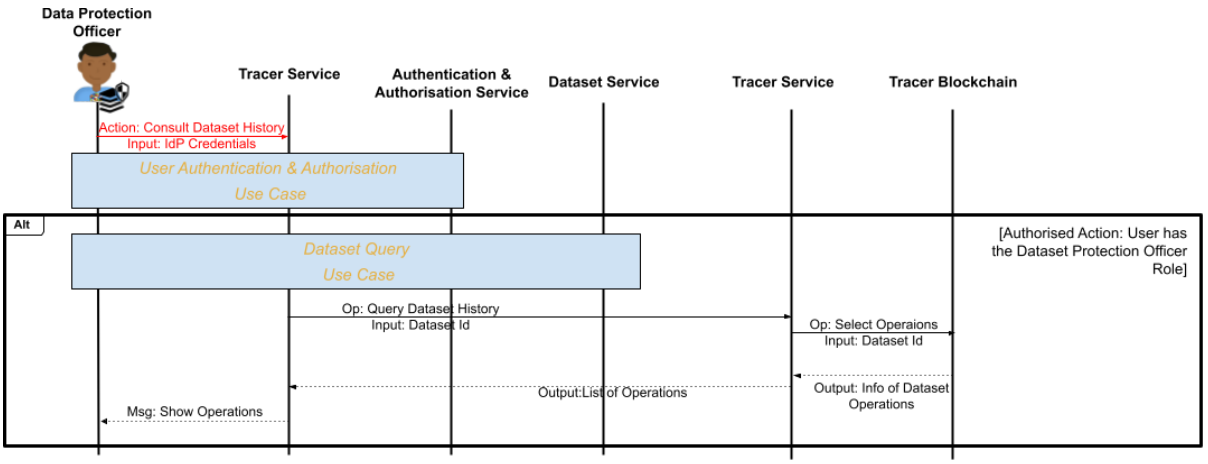


Figure 43. *UML Sequence Diagram for Dataset History Query.*

4.8.2. Query Users who accessed a Dataset

The Tracer service registers access to a *Dataset* performed in the frame of the CHAIMELEON repository. *Datasets* are made available to users as volumes mounted on virtual infrastructures (*Standalone Applications*), so access to the *Datasets* is duly registered on the tracing system. The *Data Protection Officer* (DPO) could retrieve the details of the users who have mounted and therefore had the right to access a given *Dataset*.

The first one is to authenticate and authorise the *User.* Only users with *Data Protection Officer Roles* will be authorised by the *Tracer Service*. Then, the *User* chooses a given *Dataset* (see *Dataset Query Use Case*) to consult the users who have accessed the *Dataset* through *Standalone Applications*.

Figure 44 shows the UML sequence diagram corresponding to the interaction flow among *Users* and *Components* involved to query the users who have accessed a given *Dataset*.

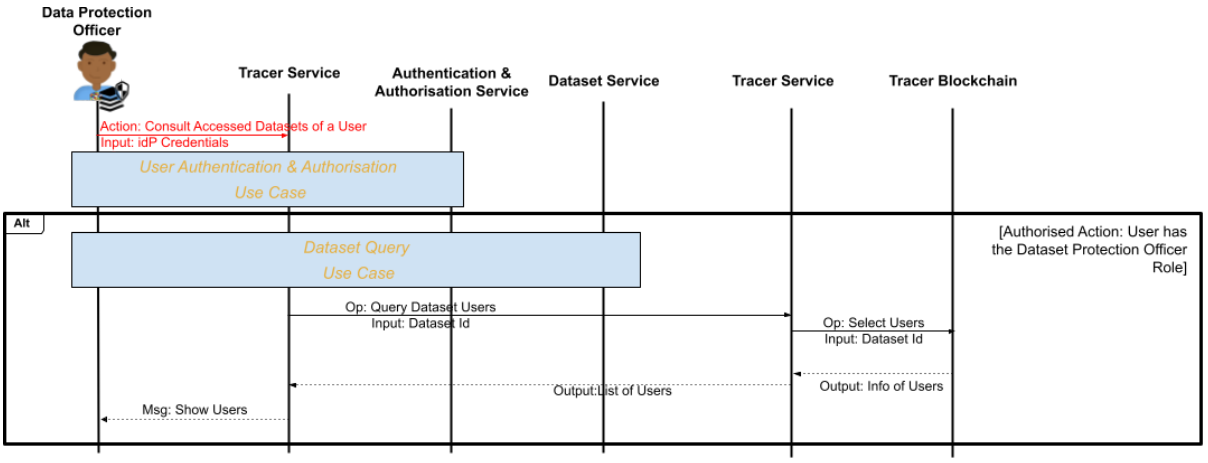


Figure 44. *UML Sequence Diagram for Consulting Dataset Users Query.*

4.8.3. Query Datasets Used on a Specific Model

Key information for selecting the rightmost model could be the accuracy or the sensitivity. However, this information may be biased depending on the *Dataset* used for the training. During the registration of a model, the developer has included information about the *Datasets*

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 952172.

used in training, so a user of the model can retrieve the aggregated information of the *Dataset* and make more informed opinions.

The first one is to authenticate and authorise the *User.* Any CHAIMELEON user will be authorised by the *Tracer Service*. Then, the *User* chooses a given *model* (see Model Query Use Case) employed in a given *Processing or AI Tool* to consult the *Datasets* employed for its development.

Figure 45 shows the UML sequence diagram corresponding to the interaction flow among *Users* and *Components* involved to query the *Datasets* used to develop a specific *Model*.

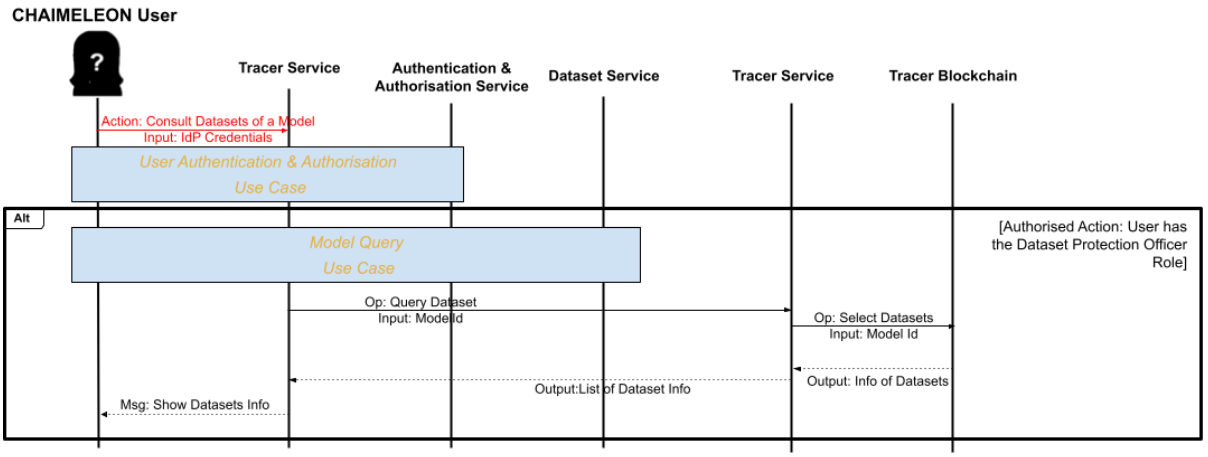


Figure 45. *UML Sequence Diagram for consulting Dataset Used on a Specific Model.*

4.8.4. Fingerprint Research Object Query

Research objects such as *Datasets* and *Models* have a PID assigned. This resolvable PID is linked to an info page in which aggregated information is presented, such as the author, provider, and a summary including some aggregated data (such as the number of cases, gender balance, modalities and other related information according to the MIABIS format in the case of datasets). Along with this information, *Datasets* and *Models* have *a fingerprint* computed from the information of images, clinical data or model parameters. This information provides a unique checksum that could be computed by the user, so the veracity of the data available (*Model* or *Dataset*) can be attested.

The first one is to authenticate and authorise the *User.* Only users with *Data Scientists* and *Application Developers* will be authorised by the *Tracer Service*. Then, the *User* chooses a given research object (*Dataset or Model*) (see *Dataset Query Use Case* or see *Model Query Use Case*) to consult its fingerprint.

Figure 46 shows the UML sequence diagram corresponding to the interaction flow among *Users* and *Components* involved to query the *Datasets* used to develop a specific model.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 952172.

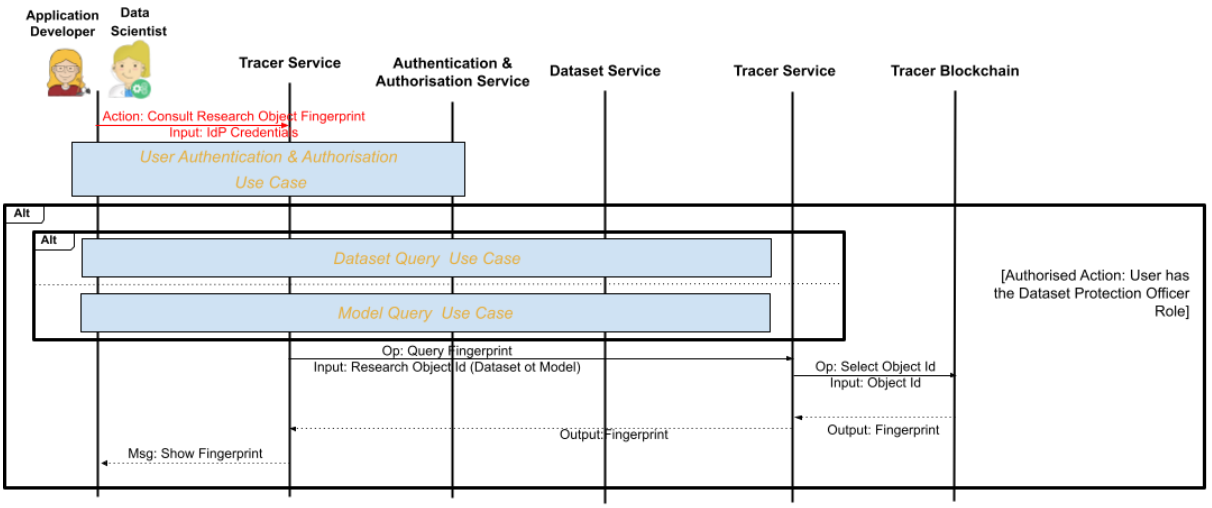


Figure 46. *UML Sequence Diagram for consulting Fingerprints.*

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 952172.

5. Requirements

After the analysis of the user roles (see section 2), components (see section 3), and use cases (see section 4) we proceed to identify, list, describe and prioritise the requirements to the final repository design. Starting from the information of Deliverable 3.2, which outlines non-functional requirements, this deliverable compiles and formally describes those requirements (non-functional requirements) and standards for the design of the *CHAIMELEON Repository*. Additionally, it includes those requirements coming from the analysis of the *User Role Actions* (functional-requirements) and *Use Cases*.

The requirement elicitation follows the IEEE-830 standard, in which requirements are prioritised according to the following categories: **Mandatory** (requirements that must be considered); **Recommendable** (requirements that will provide relevant features to the platform); **Optional** (requirements that will be nice to have but can be postponed).

The requirements are described according to the following structure:

- Code: RE#.#, where the first number relates to the category of the requirement (TRA - Transversal, UDB - User Database, DLK - Data Lake, RDS - Repository Database (Datasets), APR - Application Registry, SCD - Source Code Repository, BCH - Blockchain Repository, AUS - Authentication Service, IAS - Data Ingestion/Access Service, DSS - Dataset Service, TRS - Tracer Service, SAPS- Standalone Access Point Service, ORC - Orchestrator Service, URA - User Registration Application, CEA - Case Explorer Application, MKT - Marketplace, DSE - Dataset Explorer Application, APP - Application Dashboard and the second identifies the number of the requirement in such a group.
- Name: Short title of the requirement.
- Description: Concise but comprehensive description of the requirement.
- Type: Functional or Non-Functional.
- Relevance: Mandatory / Recommendable / Desirable.

5.1. Transversal Requirements

Next requirements are applied or involved to a set of *CHAIMELEON Components; these* are Storages, *Services*, *Platform Applications* and *Processing Applications*.

Code: RETRA.1	**Name:** Cloud Resources Management	
Type: Non-Functional	**Relevance:** Mandatory	**Technology:** Openstack
Description: *CHAIMELEON Components* run on top of cloud resources. The management of the back-end resources **must** be performed through a cloud manager.		

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 952172.

Code: RETRA.2	**Name:** Isolated Cloud Resources Management	
Type: Functional	**Relevance:** Mandatory	**Technology:** Openstack
Description: The *CHAIMELEON Components* **must** run on an isolated environment using dedicated cloud resources, which could run on a pool of shared computing resources.		

Code: RETRA.3	**Name:** Flexible and Elastic Management of Cloud Resources	
Type: Functional	**Relevance:** Recommendable	**Technology:** EC3
Description: *CHAIMELEON Components* **should** serve all requests, scaling up storage and processing cloud resources based on the demand and scaling down the processing resources if the demand decreases.		

Code: RETRA.4	**Name:** Deploy *Storages, Services, Platform Applications* and *Processing Applications* on the Cloud	
Type: Functional	**Relevance:** Mandatory	**Technology:** TOSCA, IM
Description: *CHAIMELEON Components* **must** be described following the Infrastructure as Code (IaC) paradigm to be automatically deployed on an on-premise or a public cloud.		

Code: RETRA.5	**Name:** *Storages, Services, Platform Applications* and *Processing Applications* connected through a Private Network	
Type: Non-Functional	**Relevance:** Mandatory	**Technology:** K8s
Description: All *CHAIMELEON Components* **must** be linked through an isolated private network. Components that require external access will also have a public end.		

Code: RETRA.6	**Name:** Interactions between *Users/Storages*	
Type: Functional	**Relevance:** Mandatory	**Technology:** --
Description: *Users* **must not** be able to access the *Data Lake* directly, *Repository Database*, *Application Registry* and *Tracer Blockchain* but will use the *applications* and services instead.		

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 952172.

Code: RETRA.7	**Name:** User access to the repository	
Type: Functional	**Relevance:** Mandatory	**Technology:** SSL, HTTPs, SFTP, DICOMWeb (WADO-RS)
Description: The functionalities of the *CHAIMELEON Repository* are mainly provided to Users through *Platform Applications.* Thus, *Platform Applications* **must** expose an interface to *Users*.		

Code: RETRA.8	**Name:** User access to ingestion service	
Type: Functional	**Relevance:** Mandatory	**Technology:** SSL, HTTPs, SFTP, DICOMWeb (WADO-RS)
Description: Direct interaction between *Users* with any *Services* will not be provided **except** between *authorised Technical Data Managers* and the *Data ingestion/access Service* to carry out the data ingestion in bulk using external tools with delegated credentials.		

Code: RETRA.9	**Name:** Interactions between *Users/Processing Applications*	
Type: Functional	**Relevance:** Mandatory	**Technology:** SSH, Guacamole
Description: All *Processing Applications* **must** provide a user interface to the *Users* through consoles or a remote desktop.		

Code: RETRA.10	**Name:** User authorisation Processes Implemented at *Platform Application Level* and *Data ingestion/Access Service*	
Type: Non-Functional	**Relevance:** Mandatory	**Technology:** Keycloak, OpenID
Description: As the *Platform Applications* and *Ingestion/Access Service* are the unique access points for the *Users* to perform the functionalities offered by the *CHAIEMELON Repository*, they **must** provide authentication and authorisation mechanisms for *Users*.		

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 952172.

Code: RETRA.11	**Name:** Encrypted communications for direct interactions between *Users* and *CHAIMELON Components*	
Type: Non-Functional	**Relevance:** Mandatory	**Technology:** SSL, HTTPs, SFTP, DICOMWeb (WADO-RS) and Apache Guacamole.
Description: *CHAIMELON Components* **must** guarantee confidentiality and integrity of communications through secure channels using encrypted communications protocols when dealing with Applications accessible by users (web interfaces, command line interfaces, remote desktop, REST API).		

Code: RETRA.12	**Name:** Encrypted communications for internal interaction among *CHAIMELON Components*	
Type: Non-Functional	**Relevance:** Desirable	**Technology:** SSL, HTTPs and SFTP.
Description: All *CHAIMELEON Components* **may** support secure channels through encrypted communications on the *CHAIMELEON Private Network* for direct communications among them, in case that the applications or services request it.		

Code: RETRA.13	**Name:** Capability to Resize Storage Backend Capacity	
Type: Functional	**Relevance:** Mandatory	**Technology:** Ceph, K8s
Description: Backend resources supporting the *Storages* **must** be able to scale up providing additional capacity.		

5.2. User Database

Code: REUDB.1	**Name:** Storage of User Identities and Associated Metadata	
Type: Non-Functional	**Relevance:** Mandatory	**Technology:** Keycloak
Description: This *Storage* **must** store all *User Identities* and his/her associated metadata (groups, roles and capabilities).		

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 952172.

Code: REUDB.2	**Name:** API/REST interface to offer the required functionalities to manage *User* Data	
Type: Functional	**Relevance:** Mandatory	**Technology:** Keycloak
Description: *This storage* **must** provide an API/REST interface to the *User* data management application (including adding user identity, assigning or removing groups/roles/capabilities, decommissioning a user, etc.).		

Code: REUDB.3	**Name:** Access to Storage only from the *User Registration Application*.	
Type: Functional	**Relevance:** Mandatory	
Description: The *User Registration Application* **must** be the only *CHAIMELEON Component* able to access the *User Database Storage* (via API/REST interface). Direct access from any other *CHAIMELEON Component* will not be allowed.		

5.3. Data Lake Storage

Code: REDLK.1	**Name:** It Stores Medical Images and Associated Clinical Data	
Type: Non-Functional	**Relevance:** Mandatory	**Technology:** Keycloak
Description: This *Storage* **must** store Clinical Data and associated Medical Images from the medical centres in the four areas involved in the project (breast, prostate, lung and colorectal).		

Code: REDLK.2	**Name:** Anonymised and Curated Data	
Type: Non-functional	**Relevance:** Mandatory	**Technology:** --
Description: This *Storage* **must** store fully anonymised and curated data (clinical data and images). The repository **must not** include any personal information, and Users will commit to preventing from running and reidentification process on the data.		

Code: REDLK.3	**Name:** DICOM Standard Format for Storing Medical Images	
Type: Non-Functional	**Relevance:** Mandatory	**Technology:** DICOM Standard
Description: It **must** store medical imaging data as files in DICOM format and compliant to the Common Data Model (CDM) defined in the project.		

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 952172.

Code: REDLK.4	**Name:** NIFTI Standard Format for Storing Medical Images	
Type: Non-Functional	**Relevance:** Recommendable	**Technology:** NIFTI
Description: It **should** store medical imaging data in files following other formats than DICOM such as NIFTI.		

Code: REDLK.5	**Name:** OMOP-CDM for Storing Observational Real-World Studies (Clinical Data)	
Type: Non-Functional	**Relevance:** Recommendable	**Technology:** OMOP-CDM
Description: This *Storage* **must** keep the clinical data following the Observational Medical Outcomes Partnership Common Data Model (OMOP-CDM) for observational, real-world studies.		

Code: REDLK.6	**Name:** OSIRIS-CDM for Storing Research Data (Omic and Biomarker Data)	
Type: Non-Functional	**Relevance:** Desirable	**Technology:** OSIRIS-CDM
Description: This *Storage* **may** consider other types of research data (in particular -omics and biomarker data) to complement the OMOP-CDM following the OSIRIS-CDM.		

Code: REDLK.7	**Name:** MIABIS-CDM for fusion with External Biobanks	
Type: Non-Functional	**Relevance:** Desirable	**Technology:** MIABIS-CDM
Description: This *Storage* **may** consider the MIABIS data model to allow the fusion with traditional biobanks and guarantee the sustainability of recollected data.		

Code: REDLK.8	**Name:** API/REST interface to offer basic functionalities for managing *Clinical Data* and *Medical Images*	
Type: Functional	**Relevance:** Mandatory	**Technology:** Keycloak
Description: This *Storage* **must** provide an API/REST interface for the management (at least insert, update, select and delete) of Clinical Data and Medical images.		

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 952172.

Code: REDLK.9	**Name:** Access Restricted to the *Data Ingestion/Access Service*	
Type: Non-Functional	**Relevance:** Mandatory	**Technology:** QUIBIM Precision
Description: *The Data Ingestion/Access Service* **must** be the only *CHAIMELEON Component* able to access the *Data Lake Storage* (via an API/REST interface). Direct access from any other *CHAIMELEON Component* is not allowed.		

Code: REDLK.10	**Name:** POSIX File System for Management of Medical Images	
Type: Functional	**Relevance:** Mandatory	**Technology:** QUIBIM Precision (File System), POSIX, Ceph.
Description: This *Storage* **must** make Medical images files accessible using standard POSIX calls.		

5.4. Repository Database

Code: RERDS.1	**Name:** Store Datasets as Isolated Volumes	
Type: Non-Functional	**Relevance:** Mandatory	**Technology:** Keycloak
Description: This *Storage* **must** create *Datasets* as data volumes that can be mounted in a *CHAIMELEON Processing Application.* A data volume will be a folder with symbolic links to medical images files (DICOM or other supported formats such as NIFTI) hosted in the *Data Lake Storage*. The research data (OSIRIS-CDM) and clinical data (OMOP-CDM) will be a copy of the e-forms files related.		

Code: RERDS.2	**Name:** Data Volume POSIX Access by *Processing Applications*	
Type: Functional	**Relevance:** Mandatory	**Technology:** CEPH, POSIX, JSON (e-forms)
Description: The data volumes (Datasets) created in this storage **must** be mountable as a folder in a POSIX file system, with symbolic links to medical images files (DICOM or other supported formats such as NIFTI) hosted at the *Data Lake Storage*.		

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 952172.

Code: RERDS.3	**Name:** Access to Medical Images of the Volumes through DICOM Web protocol	
Type: Non-Functional	**Relevance:** Mandatory	**Technology:** DCM4CHEE
Description: The images of the Volumes stored in this s*torage* **must** be accessible by *CHAIMELEON Processing Application* through a PACS System compatible with the DICOM web protocol.		

Code: RERDS.4	**Name:** Access Restricted to the *Dataset Service*	
Type: Non-Functional	**Relevance:** Mandatory	**Technology:** --
Description: The *Dataset Service* **must** be the only *CHAIMELEON Component* able to access the *Repository Database Storage (*via API/REST interface). Direct access from any other *CHAIMELEON Components* is not allowed.		

5.5. Application Registry

Code: REAPR.1	**Name:** Storage of *Processing Applications* as Containers	
Type: Non-Functional	**Relevance:** Mandatory	**Technology:** Harbor Registry, Docker, Helm Charts, YAML
Description: The *Application Registry* **must** store *Processing Applications* (*Standalone Applications* and *Processing/AI tools*) as containers, including all the files required for their deployment and set-up, such as Helm Charts, YAML files.		

Code: REAPR.2	**Name:** API/REST interface to manage *Processing Applications* as Containers	
Type: Functional	**Relevance:** Mandatory	**Technology:** Keycloak
Description: This *Storage* **must** provide an API/REST interface for managing the uploading, downloading, updating and querying information about the containers of the applications and all the associated files.		

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 952172.

Code: REAPR.3	**Name:** Access Restricted to the *Dashboard Application*, *Case Explorer (Marketplace)* and the *Orchestrator Service*	
Type: Non-Functional	**Relevance:** Mandatory	**Technology:** --
Description: The *Application Registry* ***must*** only be accessible by the *Dashboard Application*, the *Case Explorer* and the *Orchestrator Service* **(**via API/REST).		

5.6. Source Code Repository

Code: RESCD.1	**Name:** Storage of Source Code	
Type: Non-Functional	**Relevance:** Mandatory	**Technology:** Git Repository
Description: This *Storage* **must** store the Source Code created by the developers (e.g. *Processing and AI tools*, *Standalone Applications*, *Services*) in the context of the CHAIMELEON project.		

Code: RESCD.2	**Name:** CLIs and Web interface to offers basic functionalities to manage Source Codes	
Type: Functional	**Relevance:** Mandatory	**Technology:** Git Repository
Description: This S*torage* **must** provide Command-Line and web interfaces for managing the source code repository (at least pull, push, checkout and commit)		

Code: RESCD.3	**Name:** Access Restricted to Application Developers	
Type: Non-Functional	**Relevance:** Mandatory	**Technology:** Git Repository
Description: *Developers of the CHAIMELEON project* **must** be able to access the Source code Repository.		

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 952172.

5.7. Tracer Blockchain

Code: REBCH.1	**Name:** Storage of Data for Traceability of Datasets and Tools.	
Type: Non-Functional	**Relevance:** Mandatory	**Technology:** MongoDB
Description: This *Storage* **must** store all data required to trace the access to data (through the mounting and release of a data volume from a Dataset) and the usage of Processing Applications (which processing application has used which dataset and under which user), as well as in the development of processing tools (which dataset have been used for creating which processing tool).		

Code: REBCH.2	**Name:** API/REST interface offering a basic functionality to manage traceability data.	
Type: Functional	**Relevance:** Mandatory	**Technology:** MongoDB
Description: This *Storage* **must** provide an API/REST interface for the secure management of traceability data traceability. The operations supported should preserve the integrity and have transactional behaviour.		

Code: REBCH.3	**Name:** Restricted access to the *Tracer Service* only	
Type: Non-Functional	**Relevance:** Mandatory	**Technology:** --
Description: *Tracer Service* **must** be the only *CHAIMELEON Component* allowed to access via API/REST interface the *Tracer Blockchain Storage*.		

Code: REBCH.4	**Name:** Restricted access to Blockchain	
Type: Non-Functional	**Relevance:** Mandatory	**Technology:** --
Description: The *Tracer Service* **must not** be of public access.		

Code: REBCH.5	**Name:** Blockchain does not store sensitive information	
Type: Non-Functional	**Relevance:** Mandatory	**Technology:** --
Description: The *Tracer Service* **must** store only pseudonymised or anonymised data identifiers, for access tracking purposes.		

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 952172.

5.8. Authentication & Authorisation Service

Code: REAUS.1	**Name:** Restrict access to the Authentication &Authorisation Service to the *Platform Applications* and *Data Ingestion/Access Services* only	
Type: Non-Functional	**Relevance:** Mandatory	**Technology:** Keycloak
Description: *Platform Applications* and *Data Ingestion/Access Service* must be the only *CHAIMELEON Components* able to access this *Service*.		

Code: REAUS.2	**Name:** *User* Authentication and authorisation compatible with OpenID Connect (OAuth 2.0)	
Type: Non-Functional	**Relevance:** Mandatory	**Technology:** Keycloak
Description: This *Service* **must** be compatible with OpenID Connect (OAuth 2.0) standard, enabling all *Platform Applications* and *Data Ingestion/Access Service* to identify and authenticate the users. This standard also provides basic profile information where roles and groups as "claims" can be added to authorise or deny the *User* access.		

Code: REAUS.3	**Name:** *User* Authentication and authorization compatible with SAML 2.0	
Type: Non-Functional	**Relevance:** Recommendable	**Technology:** Keycloak
Description: This *Service* **should** be compatible with SAML 2.0 standard to provide the possibility for communicating with external Identity Providers such as EduGain.		

Code: REAUS.4	**Name:** *User* Authentication Management	
Type: Functional	**Relevance:** Mandatory	**Technology:** Keycloak
Description: This *Service* **must** provide the basic functionalities for managing Users' identities such as sign up, list, get information and disable. If a *User* provides an identity from an external IDP (such as Google or EduGain), proof of identity will be requested to process the request.		

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 952172.

Code: REAUS.5	**Name:** *User* authorization Management	
Type: Functional	**Relevance:** Mandatory	**Technology:** Keycloak
Description: This *Service* **must** provide the basic functionalities for managing groups and Users' roles or capabilities (such as creation, disabling, query, role assignment and release, group membership).		

Code: REAUS.6	**Name:** API/REST Interface for Offering the Basic Authentication and authorization Functionalities	
Type: Non-Functional	**Relevance:** Mandatory	**Technology:** Keycloak
Description: This *Service* **must** offer an API/REST interface to provide *Platform Applications* and *Data Ingestion/Access Service* with the basic functionalities for user authentication and authorization management.		

5.9. Data Ingestion/Access Service

Code: REIAS.1	**Name:** Restrict access to Data Ingestion/Access Service to *Case Explorer Application* and External applications with delegated credentials only	
Type: Non-Functional	**Relevance:** Mandatory	**Technology:** QUIBIM Precision Web Interface, MEDEX Suite
Description: *CHAIMELEON Case Explorer Application* and *External Applications (*to ingest data in bulk) **must** be able to access this service for managing *Data Lake Storage* and execute the offered functionalities (data ingestion, selection etc…). The external applications for in-bulk data ingestion must have delegated credentials of an *Authorised Technical Data Manager*. Direct access from any other *CHAIMELEON Component* is not allowed.		

Code: REIAS.2	**Name:** Basic Functionalities to Manage the Data present in the *Data Lake*	
Type: Functional	**Relevance:** Mandatory	**Technology:** --
Description: This *Service* **must** provide basic functionalities, such as insert, select, retrieve, update, disable to manage the Data Lake data (both clinical data as JSON e-forms in OMOP-CDM or OSIRIS-CDM and medical images as DICOM or NIFTI files) stored in the *Data Lake Storage*.		

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 952172.

Code: REIAS.3	**Name:** API/REST Interface for Managing the Data Lake Data	
Type: Non-Functional	**Relevance:** Mandatory	**Technology:** API/Rest, DICOM Web
Description: This *Service* **must** offer a REST/API interface (and DICOM Web protocol for medical images) to the *Case Explorer Application* and *External Applications* used to ingest data in bulk.		

5.10. Dataset Service

Code: REDSS.1	**Name:** Restricted access to the Dataset Service from *Data Ingestion/Access Service, Orchestrator Service, Case Explorer Application and Dataset Explorer Application only.*	
Type: Non-Functional	**Relevance:** Mandatory	**Technology:** --
Description: Only *Data Ingestion/Access Service, Orchestrator Service, Case Explorer Application and Dataset Explorer Application* **must** be able to access this service. Direct access from any other *CHAIMELEON Component* is forbidden.		

Code: REDSS.2	**Name:** Basic Functionalities to Manage Dataset Data	
Type: Functional	**Relevance:** Mandatory	**Technology:** --
Description: This *Service* **must** provide basic functionalities for managing Datasets (create, consult, etc.) stored in the *Repository Database*.		

Code: REDSS.3	**Name:** API / REST Interface for Offering Dataset data Management	
Type: Non-Functional	**Relevance:** Mandatory	**Technology:** API/REST
Description: This *Service* **must** expose a REST / API interface.		

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 952172.

5.11. Tracer Service

Code: RETRS.1	**Name:** Restricted access to the Tracer Service from Dataset Service only	
Type: Non-Functional	**Relevance:** Mandatory	**Technology:** --
Description: The *Tracer Service* **must** be (in principle) only accessible through the *Dataset Service*.		

Code: RETRS.2	**Name:** Provide a Basic Functionality to Manage the Traceability	
Type: Functional	**Relevance:** Mandatory	**Technology:** --
Description: This *Service* **must** provide basic functionalities for the management of data traceability, registering at least the events of dataset creation, dataset access through a volume, dataset release, application registration, application release, application access, IA models access. This data is stored in the *Tracer Blockchain*.		

Code: RETRS.3	**Name:** API/REST Interface for Traceability management	
Type: Non-Functional	**Relevance:** Mandatory	**Technology:** API/REST
Description: This *Service* **must** offer a REST/API interface to *Dataset Service*.		

5.12. Orchestrator Service

Code: REORC.1	**Name:** Restricted access to Orchestrator Service from *Application Dashboard* and *Case Explorer (Marketplace) only*	
Type: Non-Functional	**Relevance:** Mandatory	**Technology:** Kubeapp, QUIBIM Precision
Description: This *Service* **must** be accessible only by the *Application Dashboard and Case Explorer*.		

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 952172.

Code: REORC.2	**Name:** Basic Functionalities to Manage the Applications	
Type: Functional	**Relevance:** Mandatory	**Technology:** Kubernetes
Description: This service **must** provide the basic functionality for the management of *Processing Applications*, including deployment, listing, inspecting, reconfiguring and deleting. Applications will be stored in the *Application Registry*.		

Code: REORC.3	**Name:** API / REST Interface for Offering Application Management	
Type: Non-Functional	**Relevance:** Mandatory	**Technology:** Kubernetes
Description: This S*ervice* **must** offer a REST/API interface to the *Application Dashboard*.		

5.13. Standalone Access Point Service

Code: SAPS.1	**Name:** User-Friendly Web Interface	
Type: Functional	**Relevance:** Mandatory	**Technology:** Apache Guacamole
Description: This application **must** offer a User-Friendly web interface to manage access points to standalone Applications through SSH or RDP connections.		

Code: SAPS.2	**Name:** Functionalities for Standalone Applications access points	
Type: Functional	**Relevance:** Mandatory	**Technology:** Apache Guacamole
Description: This application **must** offer a set of functionalities to manage access points to Standalone Applications. These functionalities **must** only be enabled to the Data Scientist Role, Application Developer Role, External *Researcher Role* and *Clinical Staff Role*.		

5.14. User Registration Application

Code: REURA.1	**Name:** User-Friendly Web Interface	
Type: Functional	**Relevance:** Mandatory	**Technology:** Keycloak
Description: This application **must** offer a set of functionalities to manage *User* Identities and associated Metadata through a user-friendly and usable interface.		

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 952172.

Code: REURA.2	**Name:** Users' Sign-up	
Type: Functional	**Relevance:** Mandatory	**Technology:** Keycloak
Description: This application **must** allow users to sign up requesting a specific role. The application will implement identity assignment, user registration and the uploading of the proof of identity and the registration of groups and capabilities.		

Code: REURA.3	**Name:** Access from *Cloud Services and Security Manager Role* to the registration management	
Type: Functional	**Relevance:** Mandatory	**Technology:** Keycloak
Description: This application **must** provide the *Cloud Services and Security Manager Role* an interface to manage the registration (including the validation of the identity and the assignment of groups and capabilities) of any *User Role* in the CHAIMELEON Repository.		

5.15. Case Explorer Application

Code: RECEA.1	**Name:** User-Friendly Web Interface	
Type: Functional	**Relevance:** Mandatory	**Technology:** QUBIM Precision
Description: This application **must** offer a User-Friendly web interface to manage *Data Lake Data*, *Datasets* and *Marketplace* depending on the *User Role*.		

Code: RECEA.2	**Name:** Functionalities for manual ingestion into the *Data Lake*	
Type: Functional	**Relevance:** Mandatory	**Technology:** QUIBIM Precision (DCM4CHEE Node),
Description: This application **must** offer a set of functionalities to manage *Data* ingestion into the *Data Lake*, offering a web interface to introduce for each case the clinical data (e-form compliant to OMOP-CDM / OSIRIS-CDM) and upload medical images associated (e.g. DICOM images). These functionalities **must** only be enabled to the *Authorised Technical Data Manager Role.*		

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 952172.

Code: RECEA.3	Functionalities for Filtering *Data Lake* Data	
Type: Functional	**Relevance:** Mandatory	**Technology:** QUIBIM Precision
Description: This application **must** offer a set of functionalities to select *Data* from the *Data Lake*, offering a web interface to select the cases that comprise a Dataset (including clinical data and associated medical images). These functionalities **must** only be enabled to the *Authorised Technical Data Manager Role* and *Dataset Manager Role.*		

Code: RECEA.4	**Name:** Functionalities for Updating *Data Lake Data*	
Type: Functional	**Relevance:** Mandatory	**Technology:** QUIBIM Precision (DCM4CHEE Node)
Description: This application **must** offer a set of functionalities to manage the update of *Data Lake Data*, through a web interface. These functionalities **must** only be available to the *Authorised Technical Data Manager Role.*		

Code: RECEA.5	**Name:** Functionalities for Creating Datasets	
Type: Functional	**Relevance:** Mandatory	**Technology:** QUIBIM Precision
Description: This application **must** offer a set of functionalities to create *Datasets using the Data Lake* Data. These functionalities **must** only be enabled to the *Dataset Manager Role* and will include the request of the dataset creation to the *Dataset Management Service* providing the list of cases and the associated metadata.		

5.16. Marketplace

Code: REMKT.1	**Name:** User Friendly and Usable Web Interface	
Type: Functional	**Relevance:** Mandatory	**Technology:** QUBIM Precision
Description: This application **must** expose the management of *Processing and AI Tools* through a User-Friendly web interface.		

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 952172.

Code: REMKT.2	Functionalities for Publishing *Processing and AI Tools*	
Type: Functional	**Relevance:** Mandatory	**Technology:** QUIBIM Precision
Description: This application **must** offer a set of functionalities to publish *Processing and AI Tools*, offering a web interface to upload the developed tool and validate it. These functionalities **must** only be accessible to the *Application Developed Role* who could upload tools to the marketplace and *Cloud Services and Security Manager Role* to validate the tools for publishing.		

Code: REMKT.3	Functionalities for Filtering *Processing and AI Tools*	
Type: Functional	**Relevance:** Mandatory	**Technology:** QUIBIM Precision
Description: This application **must** offer a set of functionalities to select *Processing and AI Tools* published in the *marketplace*, offering a web interface to filter tools. These functionalities **must** only be enabled to the *Clinical staff Role* and *Data Scientist Role.*		

Code: REMKT.4	Functionalities for Executing *Processing and AI Tools*	
Type: Functional	**Relevance:** Mandatory	**Technology:** QUIBIM Precision
Description: This application **must** offer a set of functionalities to execute *Processing and AI Tools* published in the *marketplace*. These functionalities **must** only be enabled to the *Clinical staff Role* and *Data Scientist Role.*		

5.17. Dataset Explorer Application

Code: REDSE.1	**Name:** User-Friendly Web Interface	
Type: Functional	**Relevance:** Mandatory	**Technology:** QUBIM Precision
Description: This application **must** provide a User-Friendly web interface to manage *Datasets*.		

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 952172.

Code: REDSE.2	Select and query Datasets	
Type: Functional	**Relevance:** Mandatory	**Technology:** QUIBIM Precision
Description: This application **must** offer a set of functionalities to select *Datasets* from the *Repository Database*, offering a web interface to filter them and to obtain information from a specific one. These functionalities **must** be available only to *Dataset Administrator Role, Data Scientist Role, Application Developer Role and External Researcher Role.*		

Code: REDSE.3	Disable Datasets	
Type: Functional	**Relevance:** Mandatory	**Technology:** QUIBIM Precision
Description: This application **must** provide the means to disable a given *Dataset* of the *Repository Database*, so this dataset will not appear on further searching operations. This functionality is restricted to the *Dataset Administrator Role.*		

5.18. Application Dashboard

Code: REAPP.1	**Name:** User-Friendly Web Interface	
Type: Functional	**Relevance:** Mandatory	**Technology:** Kubeapps
Description: This application **must** provide a User-Friendly web interface to manage *Standalone Applications*.		

Code: REAPP.2	**Name:** Create *Standalone Applications*	
Type: Functional	**Relevance:** Mandatory	**Technology:** Helm Charts, Kubeapps
Description: This application **must** enable the creation of *Standalone Applications*, including the uploading of image containers, Helm charts and additional specification and configuration files. These functionalities **must** only be enabled to the *Cloud Services and Security Management Role.*		

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 952172.

Code: REAPP.3	Filter and Select *Standalone Applications*	
Type: Functional	**Relevance:** Mandatory	**Technology:** Kubeapps
Description: This application **must** offer a set of functionalities to filter and select *Standalone Applications* from the *Application Registry* through a web interface. This **must** be provided only to the *External Researchers, Data Scientist Role* and *Application Developer Roles.*		

Code: REAPP.4	Deployment of *Standalone Applications*	
Type: Functional	**Relevance:** Mandatory	**Technology:** Kubeapps
Description: This application **must** expose a web interface to deploy *Standalone Applications* in the ChAIMELEON repository cloud resources. The application will enable defining configuration parameters (such as the dataset id, the resource claim and other access credentials or configuration options). These functionalities **must** only be enabled to the *External Researchers, Data Scientist Role* and *Application Developer Roles.*		

Code: REAPP.5	Retrieve Access Points and other information from running *Standalone Applications*	
Type: Functional	**Relevance:** Mandatory	**Technology:** Kubeapps
Description: This application **must** offer a web interface to filter applications and list *Standalone Applications* access points *of running applications*. This **must** be available only to the *External Researchers, Data Scientist Role* and *Application Developer Roles.*		

Code: REAPP.6	Release Standalone Applications	
Type: Functional	**Relevance:** Mandatory	**Technology:** Kubeapps
Description: This application **must** offer a set of functionalities to remove a running *Standalone Application* and release the associated resources. This **must** be available only to the *External Researchers, Data Scientist Role* and *Application Developer Roles.*		

Code: REAPP.7	Access Control to Running Standalone Application to their Owners	
Type: Functional	**Relevance:** Mandatory	**Technology:** Kubeapps
Description: The Standalone Applications which are running must only be managed (list access point, release) by the User which s(he) has deployed it.		

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 952172.

6. Security Considerations

This section describes at architecture level the threat analysis and the recommendations for the setup and configuration of the components of the architecture regarding security.

6.1. Threat Analysis

In order to identify the risk of intrusions and privacy leakages, we performed an analysis of the reasonable threats that the platform may face. This threat model is described in this section.

6.1.1. Assumptions

The model considers the following assumptions:

- Users' credentials may be weak or get exposed.
- Resources may not be properly patched and may suffer from vulnerabilities.
- Access to Virtual Machines cannot be granted from inside a Docker container.
- Granting access to the overlay network that connects the containers will not increase exposure risk.
- The Platform will only run official CHAIMELEON Docker Containers.
- Users have accepted some “Terms of Usage” before accessing the platform. And these “Terms of Use” will include the obligation to follow a security breach procedure.

These assumptions will be the basis for identifying the threats and the corrective measures to take into account.

6.1.2. Threats

The threats are those undesired situations that could appear as a consequence of an issue that could lead to a security breach. The threats are grouped by 5 situations:

- A user may expose his/her data access token to a user adversary, who …
  - … could access any data stored on the cloud (s)he is authorised to.
  - … could remove any data stored in the cloud by his/her own, although will not be able to remove general shared data (unless the token is from the administrator, which should never be used).
  - … could update clinical of imaging data fom the Case Explorer.
  - ... could not access the VMs where the K8s containers run, nor the ceph resources.
  - … could not access the K8s services for creating, deleting or inspecting any resource.

- A user may expose his/her IdP credentials to a user adversary, who …
  - … could access any K8s job and service deployed and any data stored on their own, to access or delete the data.
  - … could create new resources and even force a DoS on his/her partition of resources
  - ... could not access the VMs where the K8s containers run, nor the ceph resources.
  - … could not exhaust resources from other users or force a huge consumption of resources.

- VM vulnerability can be exploited or VM credentials from a system administration obtained by an adversary who ...

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 952172.

- … could access the services running in the VM, stopping them or even creating new ones under any user partition.
- … could access data from any resource.
- ... could not create resources in the cloud unless access to the front end is granted.
- … could not destroy resources in the cloud unless access to the front end is granted.

- An application developer (or System Administrator) adversary can gain credentials for building Docker Images...
  - … could inject malicious code in the Docker Containers used by the Users and Data Managers to retrieve access credentials.
  - … could inject malicious code in the applications.
  - ... could not impersonate the user in the processing cloud.
  - … could not access resources from other users.

- Cloud infrastructure credentials from a system administration can be obtained by an adversary who ...
  - … could destroy the virtual infrastructure or create new resources on behalf of the system administrator.
  - … could not easily access data from any resource, enter in the VMs or access the K8s services. It will require stopping VMs, dumping the disks, mounting on other resources and inspecting the data to search for the credentials, which requires inspecting the filesystems of the containers running on the system.

6.1.3. Recommendations

In order to minimise those risks, the threat model identifies the following recommendations that will be the basis for the technical and operational measures implemented.

- Recommendation for All roles:
  - Do not store credentials (including access tokens) in GitLab repositories (even in private ones).

- Recommendations for System AdminIstrators:
  - VMs should be periodically patched (Ubuntu, Docker, Kubernetes).
  - SSH access to the Front-end VM is limited to a specific IP range.
  - Access to resources is limited to the necessary ports, no SSH access from outside the UPV is allowed.

- Recommendations for Application Developers:
  - Docker images are periodically re-built (automatically) based on official images.

6.2. Technical and Organisational Security Measures

6.2.1. Organisational Security Measures

The organisational security measures are those regulations and procedures that support focus on increasing the security of the resources and reducing the risk or the liability of the resource provider. These measures take into account the obligations stated in the regulations covering the processing and storage of sensitive data within the CHAIMELEON project and other additional measures addressing the users of the services and data.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 952172.

6.2.2. Organisational Security Measures with respect to the Cloud Infrastructure

The cloud infrastructure is located on the premises of the Grid and High Performance Computing (GRyCAP) research group from the Institute of Instrumentation for Molecular Imaging (I3M) at the Universitat Politècnica de València (UPV). These resources have the following restrictions:

- The physical access to the resources is restricted to duly authorised administrative operators by means of an electronic key. All accesses are logged in including date, time and electronic key used.
- The administrative access to the computing machines that support the cloud infrastructure is limited to the GRyCAP system administrators.
- The access to the virtual machines where the services of the CHAIMELEON cloud infrastructure run is limited to the Cloud Services and Security Manager (although system administrators may be able to grant access to the resources).
- Any other user could only access the infrastructure through the CHAIMELEON services (dashboard and datahub) with their specific access credentials described in section 2.
- The passwords will follow the procedures recommended by the UPV (https://www.upv.es/entidades/ASIC/catalogo/467508normali.html).

The procedures that should be taken into account by the system administrators and the application managers are the following:

- Keeping the software up to date
  - The resources supporting the cloud infrastructure must be proactively patched. In particular, the system administrators will:
    - Patch the operating system of the physical machines at least twice a year and every time a security patch is released.
    - Patch the Virtual Machine Images used in the cloud platform with the same regularity.
    - Patch the underlying services such as OpenStack every time a security update is available.
    - Proactively update the recipes for the software configurations used (especially Docker, Kubernetes and CEPH).
  - The application managers will update the deployment of the Kubernetes, CEPH and Docker every time a new security patch is available.
  - For those purposes, the system administrators will inform the cloud resource application managers (included in a mailing list) about any important update that should be taken into account.
- Reducing the surface exposure
  - CHAIMELEON cloud infrastructure integrates many services. However, only a fraction of them need to be accessible from outside of the UPV. In order to provide access from the Internet, the application manager has to explicitly configure this in both the Kubernetes cluster and OpenStack. However, this will not be sufficient for regular (not VPN) access from outside of the UPV and will require opening such a port in the UPV's institutional firewall. For this purpose, the Application Manager has to request it (on writing) to the System Administrator. The system administrator will submit this request to the UPV's network administrator team who may approve or not, depending on the risks.
  - Moreover, the UPV will proactively monitor the connections and could switch off resources if suspicious traffic is taking place.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 952172.

6.2.3. Organisational Security Measures with respect to the users of the Cloud Infrastructure

The users of the cloud infrastructure must sign the *Terms of Usage* (ToU) when subscribing to the KeyCloak Authentication and Authorisation service. This ToU includes the following terms:

CHAIMELEON brings together cloud computing and human resources to decision making in the clinical management of malignant tumors, offering predictive tools to assist diagnosis, prognosis, therapies choice and treatment follow-up, based on the use of novel imaging biomarkers, advanced visualization of predictions with weighted confidence scores and machine-learning based translation of this knowledge into predictors for the most relevant, disease-specific, Clinical End Points

For this purpose, CHAIMELEON aims at:

- Providing a platform for storing securely anonymised data from oncology.
- Provide a platform for running computational medical imaging biomarkers on the cloud.

The users of the CHAIMELEON Platform will be application developers and applications managers that want to support scientists from this community. By registering as a user you declare that you have read, understood and will abide by the following conditions of use:

1. You shall only use the resources/services to perform work, or transmit or store data consistent with the stated goals, policies and conditions of use as defined by the body or bodies granting you access.

2. You shall provide appropriate acknowledgement of support or citation for your use of the resources/services provided as required by the body or bodies granting you access.

3. You shall not use the resources/services for any purpose that is unlawful and not (attempt to) breach or circumvent any administrative or security controls.

4. You must not carry out any data processing that leads to the reidentification of the subjects to which the data belongs. This include crossing of data with other sources of information like Open Data Platforms.

5. In the event that you need to copy data outside the CHAIMELEON Platform resources for a processing activity that cannot be done with those resources, you will be responsible for their custody and agree to delete them completely at the end of the activity.

6. You shall respect intellectual property and confidentiality agreements.

7. You shall protect your access credentials (e.g. private keys or passwords).

8. You shall keep all your registered information correct and up to date.

9. You shall immediately report any known or suspected security breach or misuse of the resources/services or access credentials to the specified incident reporting locations and to the relevant credential issuing authorities.

10. You use the resources/services at your own risk. There is no guarantee that the resources/services will be available at any time or that their integrity or confidentiality will be preserved or that they will suit any purpose.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 952172.

11. You agree that logged information, including personal data provided by you for registration purposes, may be used for administrative, operational, accounting, monitoring and security purposes. You agree that this logged information may be disclosed to other authorised participants via secured mechanisms, only for the same purposes and only as far as necessary to provide the services.

12. You agree that the body or bodies granting you access and resource/service providers are entitled to regulate, suspend or terminate your access without prior notice and without compensation, within their domain of authority, and you shall immediately comply with their instructions.

13. You are liable for the consequences of your violation of any of these conditions of use, which may include but are not limited to the reporting of your violation to your home institute and, if the activities are thought to be illegal, to appropriate law enforcement agencies.

14. You must inform CHAIMELEON platform administrators about any possible security breach that you could be aware of or you have detected, like possible loss of access credentials, any misconfiguration or in general any possible security gap.

15. You are aware that information included in CHAIMELEON platform is related personal data, but anonymised and if you detect any possible remaining personal info or any possibility of re-identification, you have the duty to inform the system administrators.

16. You must be aware that tracking and tracing procedures of your activity in the CHAIMELEON platform could be carried out and some of your logging information could be stored.

6.2.4. Technical Security Measures

The technical security measures aim at reducing the exposure of the platform to the threats and risks identified in the previous sections.

- Reducing the surface exposure
    - CHAIMELEON cloud infrastructure integrates many services. However, only a fraction of them need to be accessible from outside of the UPV. For this purpose, we have limited the access by three means:
        - Access limitations using the UPV's institutional firewall. Only the ports 80 (K8s Ingress) and 443 (K8s Ingress) are accessible from outside of the UPV's network. This prevents:
            - Gaining access through SSH.
            - Gaining direct access to the project databases.
        - Access limitations through the OpenStack Security Groups firewall. Only ports 22 (SSH), 80 (K8s Ingress), 443 (K8s Ingress), 6901 (Medexprim), 8443 (Ceph Dashboard), 8800 (IM), 8899 (IM), 9080 (PACS), 9090 (Ceph Object Gateway), 9443 (PACS), 10000 (Medexprim), 10001 (Medexprim) 10443 (Harbor), 14443 (Notary), and 30443 (K8s Dashboard) are accessible. These ports are used by the applications deployed in the cluster. This prevents malicious applications from creating applications that communicate through other ports.
        - Access limitations through the Kubernetes overlay network. The database services communicate through the overlay network in the specific ports required and are not available from outside of the Kubernetes cluster.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 952172.

    - I3M cloud infrastructure resources. The cloud infrastructure dashboard and the service endpoints are not accessible from outside the UPV.
- Actively monitoring suspicious traffic. The UPV performs an active monitoring of the network traffic aimed at detecting inbound and outbound abnormal traffic. In case of suspicious events, corrective measures are automatically applied. Therefore, the UPV uses heuristics such as the access to ports during short periods of time, abnormal communication patterns, communication cuts and even MAC spoofing. The corrective measures include from sending warning messages to the system administrators to service cuts of a network segment.
- Use of encrypted communication and digital certificates for all the communications performed among the services, including the traffic in the private networks.
- Anonymous access is only allowed to dataset metadata summaries, which include public, aggregated metadata according to the FAIR principles.
- Use of OpenID authentication to validate users and KeyCloak roles to authorise them, so only users included in the KeyCloak can access the CHAIMELEON services. User's permissions can be revoked by removing or disabling the membership of the users, so following accesses will be rejected.
- Session credentials when accessing the Dashboard expire after 10 minutes, and they are automatically renewed, checking periodically the validity of the credentials.
- Physical access to the resources is restricted. Resources are hosted in a secure room with electronic keys.
- Kubernetes cluster protection:
    - Kubernetes cluster uses OpenID authentication for users.
    - Users only can interact with K8s using Kubeapps.
    - Role-based access policies (RBAC). User's actions are delimited by using a restricted RBAC policies defined by the cloud and security administrators[28].
    - Isolation of users. Users only can perform the actions defined by the RBAC policies in their or his namespaces or in a shared namespace between the all project (*chaimeleon-shared*).
    - The Pod Security Standard confiured is "Baseline"[29] and it is implemented using the K8s policy manager Kyverno[30]. Besides, we added more general security policies[31] using Kyverno to decrease the amount of attack posibilities. Furthermore, we use Kyverno to ensure 1) users only can use as root path of the ingress their username[32] and, 2) users only can mount the storage using their unique GID[33].
    - A personalised K8s Operator[34] was implemented for ensuring that users that request mounting a dataset using K8s can do it.
- Logs of the operations on the different services are recorded for:
    - OpenStack services on the OpenStack Platform.
    - Kubernetes and CEPH services on the Virtual Cluster.

[28] https://github.com/chaimeleon-eu/k8s-deployments/blob/master/kubernetes-users/authorization.yml
[29] https://kubernetes.io/docs/concepts/security/pod-security-standards/#baseline
[30] https://kyverno.io/
[31] https://github.com/chaimeleon-eu/k8s-deployments/tree/master/kyverno/policies
[32] https://github.com/chaimeleon-eu/k8s-deployments/blob/master/kube-authorizer/user_creation/rootfs/templates/kyverno-policies-ingress.yml.tpl
[33] https://github.com/chaimeleon-eu/k8s-deployments/blob/master/kube-authorizer/user_creation/rootfs/templates/kyverno-policies-security-context-gid.yml.tpl
[34] https://github.com/chaimeleon-eu/k8s-chaimeleon-operator

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 952172.

7. Conclusions

A final design of the *CHAIMELEON Repository* has been performed through a systematic and extensive process (see "*D3.3. Interim Platform Design*" for more details) that took into account the information on standards and technologies of previous deliverables. This deliverable performs a thorough process of identification of the expected functionality (*User Actions*) of the repository. The CHAIMELEON repository design is based on three main concepts: a) The *CHAIMELEON Repository* as a platform to store and process the data; b) *Datasets* as research objects comprising a subset of anonymised and annotated data that has a Persistent Identifier; c) Data Analytics *Models* trained on the data with a Persistent Identifier assigned, which can be embedded into tools in a marketplace.

This document is a guideline for the implementation of the repository and all its components. The document is especially relevant for WP4 members, but also for the other Work Packages to find the way they could interact with the repository. As the repository will be mainly developed using tools released under Open-Source licences, the document also serves as a guide for external developers who would like to contribute to the *CHAIMELEON Repository*.

The users of the repository are classified into 8 *User Roles* or user profiles. Each role has a different way of interacting with the repository (*User Actions*) and, therefore, will have different permissions. A user may have multiple roles, but operations are defined at the level of individual roles to limit permissions and accessibility. These actions trigger interactions among components defined in the architecture, specified through 34 Use Cases organised in 8 areas: Authentication and authorization, Data Lake Management, Dataset Management, Processing Application Management, Standalone Application Management, Model Management, Marketplace Management and Tracing Data.

In the final architecture, the *CHAIMELEON Repository* identified 5 *Platform Applications* as Persistent applications that mainly expose the functionality (User Registration Application, Case Explorer Application, Marketplace, Dataset Explorer Application and Application Dashboard), 6 *Services* that interact directly with the resources (Authentication Service, Data Ingestion/Access Service, Dataset Service, Tracer Service and Orchestrator Service) and 6 *Storages* (for user data, clinical data, datasets, application binaries, source code and traceability). This process ended up with 85 requirements covering 15 transversal requirements, which are applied to all *Storages*, *Services* and *Applications*, 28 requirements applied to the *Storage*, 18 for *Services* and 24 for *Platform Applications*.

One of the key aspects of the repository is traceability. Despite the use of anonymised data, the repository restricts the processing of the data to its local environment, so data is not downloaded outside of the platform. Even in this case, the repository annotates the main research objects and actors (datasets, models, users and applications) with persistent identifiers and keeps track of the interactions. This increases the trustworthiness of the repository and provides higher confidence to data providers.

This final architecture of the *CHAIMELEON Repository* will be implemented in the frame of WP4 and will be exposed to the user community as planned in the project.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 952172.