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

D3.3. Interim Platform Design

Leader partner:	Universitat Politècnica de València (UPV)
Author(s):	UPV: J. Damian Segrelles Quilis, Ignacio Blanquer, Andrei S. Alic, Sergio López Huguet and Pau Lozano.
	MEDEX: Karine Seymour, Thomas Trouillard and Samuel Boucher
	UV: Pr. Ricard Martinez, Francisco R. Soriano
	QUIBIM: Ana Jimenez-Pastor
	BAHÍA: Manuel Suárez Taboada, Miguel Boubeta Martínez, Juan Prieto-Pena
	HULAFE: Ana Miguel, Gloria Ribas
	MATICAL: Amelia Suarez
	IMPERIAL: Guang Yang, Nan Yang
	BGU: Yisroel Mirsky
Reviewers:	UM: Henry Woodruff, Shruti Atul Mali
	BAHÍA: Juan Prieto-Pena
	MEDEX: Karine Seymour
Work Package:	WP3
Due date:	Month 12
Actual delivery date:	31/08/2021
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

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant

This project has received funding from the European Union's
Horizon 2020 research and innovation programme under grant

Abbreviations

AI	Artificial Intelligence
FAIR	Findable, Accessible, Interoperable and Reusable
PID	Persistent Identifier
POSIX	Portable Operating System Interface
REST	Representational State Transfer
DICOM	Digital Imaging and Communication On Medicine
GUI	Graphical User Interface
API	Application Programming Interface
UAP	User Access Policy
DevOps	Development Operations
DPO	Data Protection Officer
IaaS	Infrastructure as a Service
PACS	Picture Archiving and Communication System
RIS	Radiology Information System
UML	Unified Modelling Language
IdP	Identity Provider
EC3	Elastic Compute Cluster in the Cloud
IM	Infrastructure Manager
TOSCA	Topology and Orchestration Specification for Cloud Applications
K8s	Kubernetes
WADO	Web Access to DICOM Objects

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant

CDM	Common Data Model
NIFTI	Neuroimaging Informatics Technology Initiative
OMOP	Observational Medical Outcomes Partnership
OSIRIS	Interoperability and data sharing of clinical and biological data in oncology
MIABIS	Minimum Information About BIobank Data Sharing
YAML	Yet Another Markup Language
CLI	Command Line Interface
DoA	Description of Action
CRUD	Create, Read, Update and Delete

Disclaimer

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

All intellectual property rights are owned by the consortium of CHAIMELEON under terms stated in their Consortium Agreement and are protected by the applicable laws. Reproduction is not authorized without prior written agreement. The commercial use of any information contained in this document may require a license from the owner of the information.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant

1. Introduction

One of the objectives of *CHAIMELEON project* is to design an EU-wide interoperable repository (***CHAIMELEON Repository***). The repository will provide storage resources to share health images, related clinical and molecular data from pathologic and liquid biopsy samples, and also will provide advanced computational resources (such as high-memory, GPU-enabled and multicore nodes) to process these data as useful resources for the AI community to develop and test effective tools (such as Quantitative Imaging biomarkers) for improved cancer management applications. These tools will be validated in the application context of the four most prevalent cancers: lung, colorectal, breast and prostate. The repository must be supported by a distributed infrastructure, building on existing initiatives at European, national, regional and individual centres levels.

1.1 Scope of the Document

This document constitutes a deliverable report of WP3. It covers the requirement analysis and the interim design architecture to implement the *CHAIMELEON Repository*, including a description of the employed methodology.

This document uses as input D3.1 Accessible Imaging Health Data Map and especially *D3.2 Requirements and Standards for CHAMELEON Platform* to identify actors, interactions 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, describes and analyses the ***User Roles***, ***User Stories***, ***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. This analysis will also guide the selection of the key technological components that could fit the different functional boxes (***Components***) of the repository architecture.

The document also serves external readers who are interested in building up a repository to validate the approach and reuse for their own purposes. 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 a piece of evidence to support the achievement of milestones.

1.3 Structure of the Document

This document starts with this introduction. Section 2 describes the role of the repository and the types of components that will comprise the interim architecture design. Section 3 describes the methodology that has been used to define the interim repository architecture. Basically, the procedure goes through the phase of elicitation (section 4) where *User Roles* and *User Stories* are collected and analysed. Then, section 5 describes the *Use Cases* extracted from those *User Stories*, where the interaction among components of the repository are outlined, and section 6 describes the *Requirements* identified. Next, section 7 presents a set of technologies that can address such requirements and a set of big pictures of the interim architecture design. Finally, section 8 ends with the conclusions.

This project has received funding from the European Union's
Horizon 2020 research and innovation programme under grant

2. Role of the Repository

It is important to stress that the CHAIMELEON Repository consists of the infrastructure and services where anonymized research data is stored and hosts the computing services for inspecting, processing, analysing, visualizing and referencing such data and output produced.

The repository focuses on addressing three important aims:

- To gather the data available at the local repositories in the hospitals in a secure, organized and efficient way.
- To serve on the annotation and data improvement to create a set of high-quality ***Datasets*** and processing tools.
- To provide a secure and efficient environment for the processing of those *Datasets*.

The repository will foster the FAIR (*Findable, Accessible, Interoperable and Reusable*) principles in the following directions:

- Findable through the creation of annotated *Datasets* and processing tools, including quality metadata and a recognized *Persistent Identifier* (PID).
- Accessible through their availability to authorized users in a processing environment.
- Interoperable through the use of standard formats and protocols.
- Reusable, by providing a secure environment to process such data.

For this purpose, we define the concept of *Dataset as an anonymized,* coherent and annotated set of images and associated data with a unique PID that can be shared among authorized researchers within the boundaries of the *CHAIMELEON Repository*.

It is very important to consider that we are dealing with clinical data. Despite data being stored anonymized, access to the data should be protected. Data access can only be granted to authorized users who abide by specific *Terms of Usage* for this data. Therefore, the repository should provide the means to limit the extraction of such data outside of the repository, thus recommending the processing of the data inside the repository resources, unless the specific conditions of the processing require a different approach.

Despite this coming out from a preliminary analysis, it is important to outline a set of definitions that will help to understand the following sections.

The *CHAIMELEON Repository* comprises three types of components (see Figure 1):

- **Storages**. Hosting anonymized medical images, associated clinical data, application binaries, source code, processing and AI tools. Storages provide several access models, such as POSIX, REST or DICOMWeb depending on the entity that accesses the data. The repository will host at least: A *Data Lake* for the data coming from the hospitals and the annotations performed, comprising DICOM images and a database with the clinical data; a *Repository Database* that contains references and metadata of the *Datasets*; an *Application Registry* to host application descriptions and container images; a *Source Code Repository*, and a Registry of Accesses (*Tracer Blockchain*).

- **Services**. They provide access to the resources such as the processing components or the storage. They are transparent to users, although application developers may also access them through APIs. The *CHAIMELEON Repository* will include the following services: an *Authentication Service* providing coherent authentication and authorization to the whole framework; a *Data Ingestion/Access Service* capable of managing the data (both images and clinical data) coming from the hospitals; a *Tracer Service* to register the access of *Datasets* and processing tools or IA models; a *Dataset*

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant

Service to create and list the *Datasets*; and an *Orchestrator Service* to manage containerised applications and their resources.

- **Platform Applications**. Providing *Users* with the functionality of the *CHAIMELEON* Repository. In general, *Users* are supposed to access the repository through GUI Applications only. The repository will include at least: A *Case Explorer Application* to explore the information in the *Data Lake*; a *Dataset Explorer Application* to manage *Datasets* information; an *Application Dashboard* to create and list *Processing Applications* created on demand such as analytical engines; and a *Marketplace* with *Processing and AI tools*.

- **Processing Applications.** *Providing users tools f*or data processing and analysis. The repository will include at least: tools for the harmonization of data, analytical engines and AI tools.

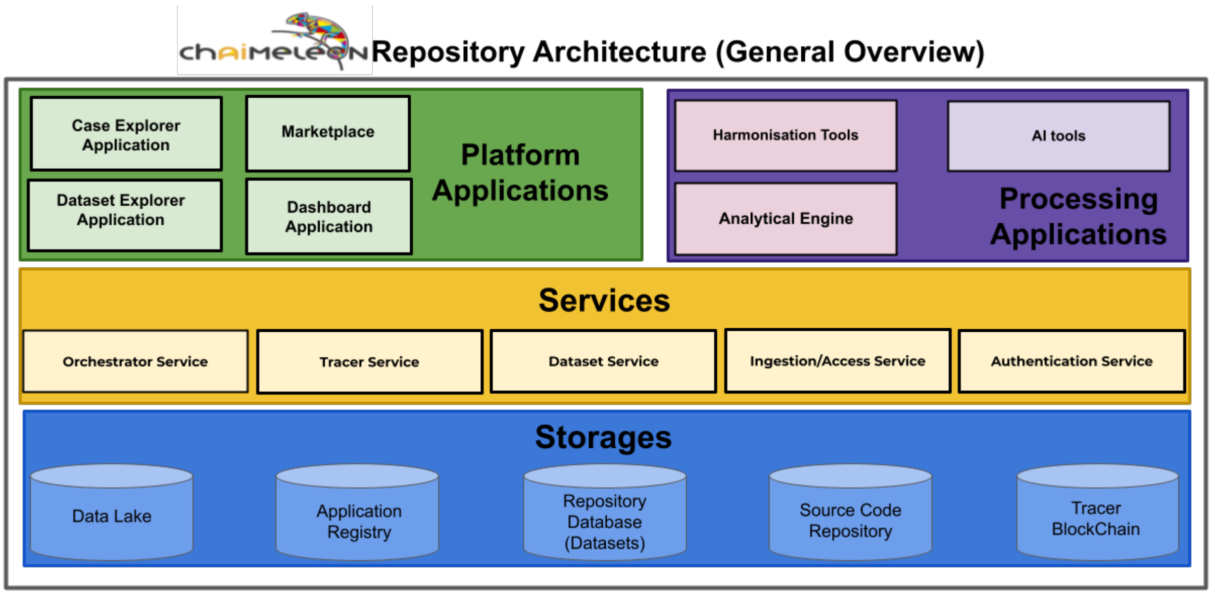


Figure 1. *General Overview of CHAIMELEON Repository Architecture.*

The information of *D3.2 Requirements and standards for CHAIMELEON Platform* states several considerations in relation of the type of information to store in the repository, formats, protocols and legal considerations. The repository will store anonymized data and will provide computational capabilities to minimise data downloading. The platform should also implement security measures and access constraints to ensure that access to data is only performed by users who abide by a specific User Access Policy (UAP). A second major consideration is interoperability. Compliance to FAIR principles are non-functional requirements that the repository should fulfil.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant

3. Methodology

This section describes the phases carried out to define the interim architecture of the *CHAIMELEON Repository*, identifying the *User Roles*, *Use Cases*, *Requirements* and *Components* (including technologies).

The method involves iterating over five phases which are the following:

1. **Elicitation.** Definition of *User Roles* and their *Stories* in *CHAIMELEON project* context.
2. **Use cases.** Definition of the *Use cases* extracted from *User Stories*.
3. **Requirements.** Extraction of *Requirements* and their prioritisation from *Use Cases.*
4. **Specification.** Definition of Big Pictures of the *CHAIMELEON Repository* architecture, and existing technologies to apply in its implementation.
5. **Validation.** Users validate the requirements.

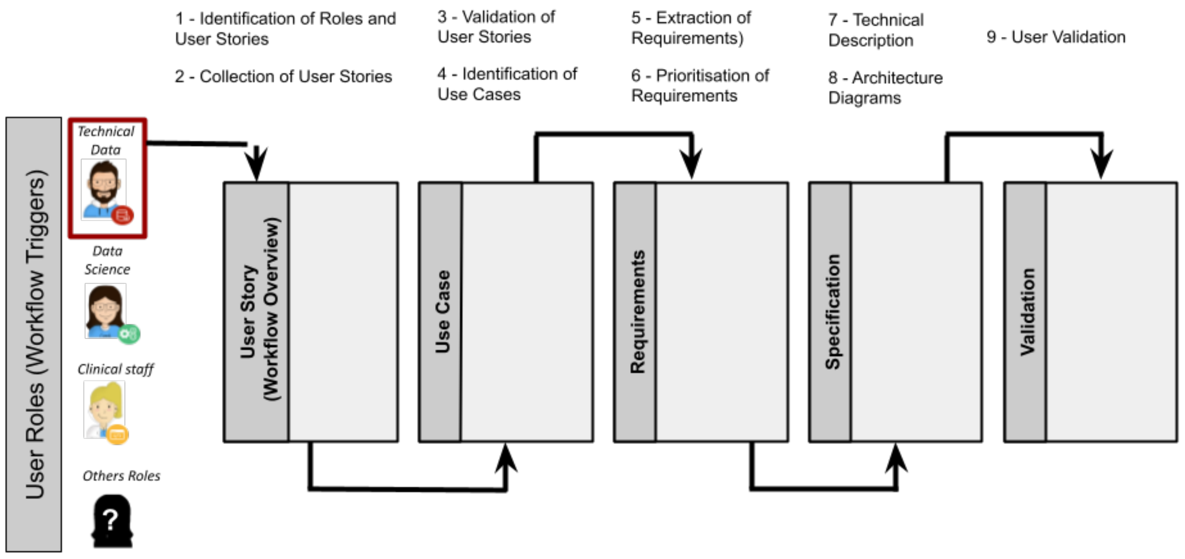

1 - Identification of Roles and User Stories
2 - Collection of User Stories
3 - Validation of User Stories
4 - Identification of Use Cases
5 - Extraction of Requirements)
6 - Prioritisation of Requirements
7 - Technical Description
8 - Architecture Diagrams
9 - User Validation
User Roles (Workflow Triggers)
Technical Data
Data Science
Clinical staff
Others Roles
User Story (Workflow Overview)
Use Case
Requirements
Specification
Validation

Figure 2. *Phases of the method to define the design interim architecture of the CHAIMELEON Repository.*

Figure 2 shows the order of the phases carried out up to the architecture definition for the first 12 months of the project. In the next subsections, the aforementioned phases will be described in detail along with the results obtained in each one.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant

4. Elicitation

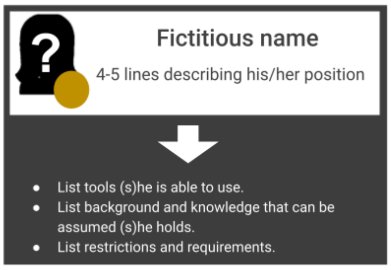
The purpose of this first phase was the identification and definition of the *User Roles* and an overview of their *User Stories* regarding the use of data/processing resources within the *CHAIMELEON Repository*.

The Elicitation phase was carried out through two different phases. First, UPV-I3M, as partner responsible for the *T3.3. Design of the repository architecture and main functionalities* of WP3, identified and described a set of generic *User Roles* and *User Stories*. For this purpose, UPV-I3M did a deep analysis of the project's Description of Action (DoA) and *Deliverables D3.1* and *D3.2* to identify generic *User Roles* and *User Stories* based on its experience in other similar European or regional projects. In the second phase, all these *User Roles and Stories* were presented to the entire *CHAIMELEON consortium* so that each partner could identify with one of them or otherwise define a new *User Roles* or new *User Stories*.

It is important to outline that the result of this phase is not the definitive list of *User Roles* and *User Stories*. Other ones may arise at different parts of the project as this progresses.

4.1 User Roles

User Roles are a generalization of those users interacting with the *CHAIMELEON Repository*. Each *User Role* was identified through an avatar (see Figure 3) exposing briefly his/her position, and a list describing the tools that s(he) was able to use, background and knowledge that could be assumed s(he) holds, and restrictions and requirements of the role.



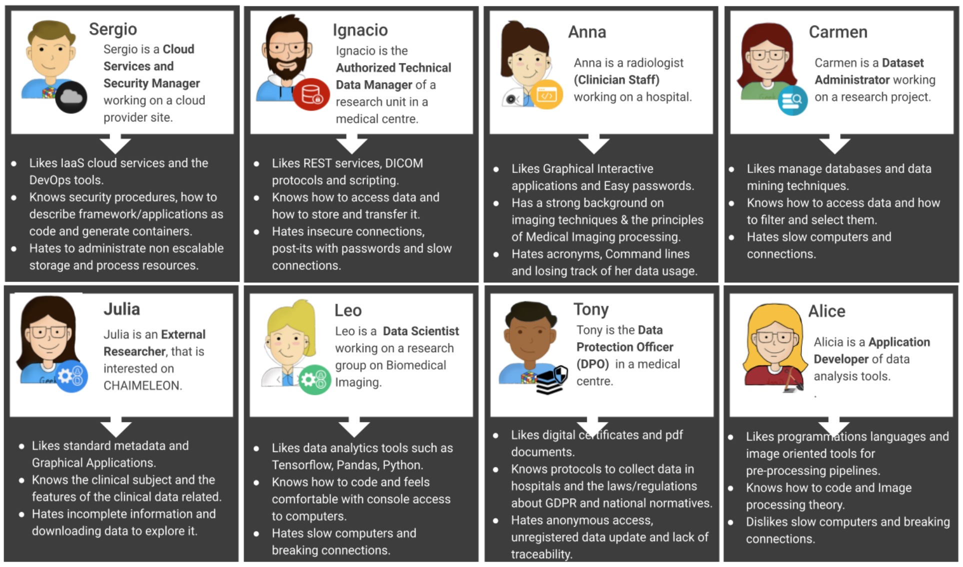
Fictitious name

4-5 lines describing his/her position

- List tools (s)he is able to use.
- List background and knowledge that can be assumed (s)he holds.
- List restrictions and requirements.

Figure 3. *Avatar Template of User Roles.*

As result, 8 *User Roles were identified*, shown in Figure 4.



Sergio

Sergio is a **Cloud Services and Security Manager** working on a cloud provider site.

- Likes IaaS cloud services and the DevOps tools.
- Knows security procedures, how to describe framework/applications as code and generate containers.
- Hates to administrate non escalable storage and process resources.

Ignacio

Ignacio is the **Authorized Technical Data Manager** of a research unit in a medical centre.

- Likes REST services, DICOM protocols and scripting.
- Knows how to access data and how to store and transfer it.
- Hates insecure connections, post-its with passwords and slow connections.

Anna

Anna is a radiologist **(Clinician Staff)** working on a hospital.

- Likes Graphical Interactive applications and Easy passwords.
- Has a strong background on imaging techniques & the principles of Medical Imaging processing.
- Hates acronyms, Command lines and losing track of her data usage.

Carmen

Carmen is a **Dataset Administrator** working on a research project.

- Likes manage databases and data mining techniques.
- Knows how to access data and how to filter and select them.
- Hates slow computers and connections.

Julia

Julia is an **External Researcher**, that is interested on CHAIMELEON.

- Likes standard metadata and Graphical Applications.
- Knows the clinical subject and the features of the clinical data related.
- Hates incomplete information and downloading data to explore it.

Leo

Leo is a **Data Scientist** working on a research group on Biomedical Imaging.

- Likes data analytics tools such as Tensorflow, Pandas, Python.
- Knows how to code and feels comfortable with console access to computers.
- Hates slow computers and breaking connections.

Tony

Tony is the **Data Protection Officer (DPO)** in a medical centre.

- Likes digital certificates and pdf documents.
- Knows protocols to collect data in hospitals and the laws/regulations about GDPR and national normatives.
- Hates anonymous access, unregistered data update and lack of traceability.

Alice

Alicia is a **Application Developer** of data analysis tools.

- Likes programmations languages and image oriented tools for pre-processing pipelines.
- Knows how to code and Image processing theory.
- Dislikes slow computers and breaking connections.

Figure 4. *Generic User Roles Identified.*

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant

4.2 User Stories

User Stories are descriptions of full interactions of a *User Role* with the *CHAIMELEON Repository*, described in natural language. *User Stories* define in general terms the needs, restrictions, performance limitations, desired features, innovation capabilities and business models for the repository.

As a result, 15 *User Stories* were recollected and are shown from table 1 to table 15.

Table 1. *Data Ingestion Story.*

Story name: Data Ingestion	**Provided by:** UPV-I3M, MEDEX
Triggered by User Role (Fictitious Name): authorized Technical Data Manager (Ignacio)	**Other User Roles (Fictitious Name):** Clinical Staff (Anna)
Potential User Role can be founded in: HULAFE, CERF, PSD, ULS, UNIPI, CHA, UM, CHUP.	**Related WP:** WP5
Overview Description: Member of a clinical partner's staff, Ignacio (*authorized Technical Data Manager* role) wants to populate a repository section (sections may focus on areas such as breast, lung, prostate or colorectal cancer) in the *Data Lake* storage of the *CHAIMELEON Repository.* The *Data Lake* stores fully anonymized and curated data (both medical imaging and associated clinical data) coming from the hospitals or medical centres. To do so, data must be curated and anonymized before being uploaded (manually or in-bulk) into the repository. Before Ignacio starts uploading the data into the *Data Lake*, he should have received permission from Anna (*Clinical Staff* role) to extract all information from the different clinical databases involved in the hospital, who should have performed the actions of curation, de-identification and anonymization. Then, Ignacio starts uploading studies (both images and clinical data). Every study uploaded is signed by Ignacio. For uploading cases, Ignacio has two options. The first one is an in-bulk ingestion, he can use a specific tool (*In-Bulk Ingestion tool*) installed at his hospital or medical centre, that connects directly the local databases with the *Data Lake* through the *Data Access/Ingestion Service*. The second one, he can directly access the *Case Explorer Application* provided by the repository for manual ingestion (case by case). Both cases require presenting valid credentials to check and get authorized (if applicable) to perform the operation in the repository through the *Authentication Service*.	

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant

Table 2. *Dataset Creation Story.*

Story name: Dataset Creation	**Provided by:** UPV-I3M
Triggered by User Role (Fictitious Name): Dataset Administrator (Carmen)	**Other User Roles (Fictitious Name):** Data Scientists (Leo) and Application Developers (Alice)
Potential User Role can be found in: CERF (Lung), ULS (Colorectal), PSD (Breast), Prostate (HULAFE).	**Related WP:** WP3, WP4, WP5
Overview Description: From a research group, Carmen (*Dataset Administrator* role) wants to release a subset of data stored in the *Data Lake* of the *CHAIMELEON Repository*. This subset of data will be called a *Dataset***.** A *Dataset* is a curated, annotated and coherent subset of the repository information that can be used for its processing, and can be furtherly referenced. These *Datasets* will be used by Leo (*Data Scientists* role*) and* Alice (*Application Developers* role*)* to build, improve or evaluate processing and AI tools such as AI models, harmonization of Data, Annotation, to name a few. For this purpose, Carmen has to crawl the *Data Lake* to find the data relevant to a specific criterion (for example, breast cancer MRI studies and associated clinical data). The creation of a *Dataset* will have a reference to the institutions providing the data and will include the appropriate information for research discovery. Thus, Carmen gets access to the *Case Explorer Application* provided by *CHAIMELEON Repository* where she has got permissions to access specific existing sections of the repository and can explore *Data Lake* by means of Data Access/Ingestion Service. Then, she searches for the corresponding cases that are relevant and extracts a list of studies, along with the images and the clinical data. Next, Carmen requests the *Case Explorer Application* by means of the *Dataset Service* to create a new *Dataset* with the list of the cases selected. This request creates the *Dataset* as a separate entity in the *Repository Database* and populates it with shortcuts (symbolic links) to the image files in the *Data Lake*. Also, this action causes the registration of the *Dataset* in the *Tracer Service* provided by the repository with the list of studies, the date, and the user and creates a signature, producing a *Dataset Id* as a unique Persistent Identifier (PID). Eventually, Carmen may split the *Dataset* into two subsets (training and validation) ensuring that the splitting does not introduce any bias. Additionally, Carmen can create a *Dataset* as a request of Leo (*Data Scientist* role) including the data that is relevant to her study. The *Dataset* will become available for its access from the *Dataset Explorer* to the authorized users.	

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant

Table 3. *AI model creation Story.*

Story name: AI model creation	**Provided by:** UPV-I3M, MEDEX, BGU, BAHIA, IMPERIAL
Triggered by User Role (Fictitious Name): Data Scientist (Leo)	**Other User Roles (Fictitious Name):** NONE
Potential User Role can be found in: CHA, BGU, IMPERIAL, QUIBIM, BAHIA, GEHC	**Related WP:** WP3, WP4, WP6, WP7, WP8, WP9
Overview Description: From a research group, Leo (*Data Scientist* role) wants to use an existing *Dataset* to research on a specific field or to develop an Artificial Intelligence (AI) model. For this purpose, Leo explores all available *Datasets* and selects the most appropriate to develop the AI model through *Dataset Explorer Application* provided by the *CHAIMELEON Repository***.** Then, Leo accesses the *Application Dashboard* of the repository and chooses a *Processing Application* to perform an analysis of the *Dataset* to gain more understanding about the data either using the Analytical Engine incorporated into the *CHAIMELEON Repository* (information about data structure, contents, relationships) or by creating a processing console to run Data Analytics tools (e.g. TensorFlow, Python Torch) linked on the previous selected *Dataset* (see table 2). Leo trains and evaluates a new AI model using the linked *Dataset*, and she uses a personal storage space provided by the repository for her own data. As a result of the training and evaluation, Leo generates a piece of software (AI model), which is versioned. Once the AI model is ready, she registers it in the *Tracer Service* of the *CHAIMELEON Repository*, along with the *Dataset ID* and the software version. This service registers that the model has been created with this specific *Dataset* (eventually even with the IDs of the individual studies excluded or included) at a specific point in time. She could always track back (and even prove) whether a specific study was or was not available when training a specific AI model. Eventually, the platform could add specific metadata of the model to the registration, so a future user could have a guarantee of the data used in the creation of the model. Finally, Leo publishes the model in the *Source Code Repository* of *CHAIMELEON Repository* for usage, following the format requested for interoperability and reusability. The model could be embedded in an AI tool in a further stage and to be published in the *marketplace* of the *CHAIMELEON Repository* for its exploitation (see table 7).	

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant

Table 4. *AI model Improvement.*

Story name: AI model Improvement	**Provided by:** UPV-I3M
Triggered by User Role (Fictitious Name): Data Scientist (Leo)	**Other User Roles (Fictitious Name):** NONE
Potential User Role can be found in: CHA, BGU, IMPERIAL, QUIBIM, BAHIA, GEHC	**Related WP:** WP3, WP4, WP6, WP7, WP8, WP9
Overview Description:	

From a research group, Leo (*Data Scientist* role) wants to improve an existing AI model available on the ***Source code Repository*** of the *CHAIMELEON Repository* or use it as a pre-trained model for extending it to deal with a different *Dataset*.

For this purpose, Leo decides to retrain the same AI model but using different parameters or using new or updated data provided by Ignacio (*authorized Technical Data Managers* role).

Leo accesses the *Application Dashboard* of the *CHAIMELEON Repository* and choose a *Processing Application* that provide a friendly user interface to perform Data Analysis on the new *Dataset* either by using the already in place Analytical Engine tool of the *CHAIMELEON Repository* (gaining information about data structure, contents or relationships), or by creating a processing console to run Data Analytics tools (e.g. TensorFlow, Pytorch) linked on the new *Dataset* (if applicable). Leo trains and evaluates a new AI model using the linked *Dataset* (if applicable) or using different parameters, and a personal storage space provided by the repository for her own data. Then she can compare the performance of both AI models. As a result of the comparison, Leo generates a piece of software (AI model), which is a new version of the existing model.

Once the specialised AI model is ready, she registers the AI model in the *Tracer Service* of the repository, along with the pre-trained *AI Model id* and the software version. This service registers that the model has been created with this specific pre-train AI Model at a specific point in time. She could always track back (and even prove) whether a specific model was or was not available when re-training a specific AI model.

Finally, Leo publishes the new version in the *Source Code Repository***.** The traceability system includes a reference to the original model for traceability.

Finally, Leo publishes the model in the *Source Code Repository* of *CHAIMELEON Repository* for usage, following the format requested for interoperability and reusability. The improved model could be embedded in an AI tool in a further stage and to be published in the *marketplace* of the *CHAIMELEON Repository* for its exploitation (see table 7).

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant

Table 5. *Image Processing Tool Development Story.*

Story name: Image Processing Tool Development	**Provided by:** MEDEXPRIM
Triggered by User Role (Fictitious Name): Application Developer (Alice)	**Other User Roles (Fictitious Name):** Data Scientists (Leo), Cloud Services and Security Manager (Sergio)
Potential User Role can be found in: IMPERIAL, CHA, BGU, QUIBIM, BAHIA, GE	**Related WP:** WP5, WP6, WP7
Overview description: From a research group, Alice (*Application Developer* role) wants to use an existing *Dataset* registered in the repository to develop an *Image Curation Tool* to allow a more efficient or fully automated image curation on the images (e.g. classify data and ensure the presence of a specific organ or the modality of the image) or an image processing tool that could be used on a pre-processing pipeline (e.g. convolutional filter useful to enlighten specific features of an image, that would help achieve better precision for the classification or detection model). Those tools could be furtherly used by *Data Scientists* to pre-process a *Dataset* before training an AI model. In some cases, an *Application Developer* can also be a *Data Scientist*. For this purpose, Alice accesses the *Application Dashboard* of the *CHAIMELEON Repository* and runs a processing environment linked on one or several *Datasets* (Dataset Creation Story - see table 2) previously created for this purpose. The processing environment will include the necessary processing tools and libraries (such as a Linux console with Python, Numpy, TensorFlow, etc.). She could use several programming third party components and specialized hardware depending on the task (e.g. CUDA library for allowing GPGPU for the aforementioned convolutional filter). Once the application is developed, Alice requests the registration of the application in the *Application Registry* to Sergio (*Cloud Services and Security Manager* role) who will validate it and insert it in the registry. However, in the case of AI tools, the registration is performed in the *marketplace* of the Case Explorer, as AI tools can be exploited (see table 7) and used for further annotations and additional information of the existing images and include them in new *Datasets*.	

This project has received funding from the European Union's
Horizon 2020 research and innovation programme under grant

Table 6. *Data Update Story.*

Story name: Data Update	**Provided by:** UPV-I3M
Triggered by User Role (Fictitious Name): Authorized Technical Data Manager (Ignacio)	**Other User Roles (Fictitious Name):** Data Scientists (Leo) and Application Developers (Julia)
Potential User Role can be found in: HULAFE, CERF, PSD, ULS, UNIPI, CHA, UM, CHUP.	**Related WP:** WP3, WP4, WP5
Overview Description: From a Hospital, Ignacio (*authorized Technical Data Manager* role) wants to update or add new data to the *Data Lake* of the *CHAIMELEON Repository* in an existing section (such as breast, lung, prostate or colorectal cancer trial). Data updates could be needed to correct erroneous values detected during the exploration of the data, and new data could be new cases, synthetic data and additional information. New data such as annotations, segmentation objects, radiomics features, etc could be obtained from analytic processes and will enrich the information of a *Dataset*. A feasibility search could be performed in advance to search for the existing data to create new *Datasets*. Data integrity should be guaranteed so existing datasets are not altered when data is updated. During the creation (see table 3) or upgrade (see table 4) of AI models by Leo (*Data Scientists* role), or during the creation of processing tool by Alice (*Application Developers* role) (see table 5), Ignacio added new data to the *Data Lake*. This does not affect Leo and Alice as the changes in the data are incremental and therefore they will access the same data, so her study is consistent. New data will not become available until a new *Dataset* is created.	

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant

Table 7. *Exploitation Story.*

Story name: Exploitation	**Provided by:** UPV-I3M
Triggered by User Role (Fictitious Name): Clinical Staff (Ana)	**Other User Roles (Fictitious Name):** NONE
Potential User Role can be found in: HULAFE, CERF, PSD, ULS, UNIPI, CHA, UM, CHUP.	**Related WP:** WP3, WP4, WP5, WP9
Overview description: Ana (*Clinical Staff*) is a Radiologist from a Medical Imaging Department who wants to extract relevant information from a set of new studies either by using an existing AI tool that was published on the *Marketplace* of the *CHAIMELEON Repository* or by using the basic exploratory data section of the Analytical Engine of the *CHAIMELEON Repository* over a set of new studies to extract relevant information according to the aim of the tool**.** In the case of AI tools, Ana gets access to the *Marketplace* and gets permission to use a published AI tool. Then, she checks the AI tools available and their information details. She is sure that the information is validated and can decide which model is more trustworthy. Finally, she executes the AI model using the new studies as input data through a user-friendly interface provided by the *Marketplace*. On the other hand, in the case of the Analytical Engine, Ana can use the basic functionalities of the Analytical Engine meant for no experts in coding or Data Analytics to obtain information (such as data structure, contents, relationships) from the set of new studies in an easy way (e.g., dashboards, basic statistics).	

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant

Table 8. *Traceability (Models created with a specific Dataset) Story.*

Story name: Traceability (Models created with a specific *Dataset*)	**Provided by:** UPV-I3M
Triggered by User Role (Fictitious Name): Data Scientist (Leo)	**Other User Roles (Fictitious Name):** NONE
Potential User Role can be found in: CHA, BGU, IMPERIAL, QUIBIM, BAHIA, GEHC	**Related WP:** WP3, WP4, WP6, WP7, WP8
Overview description: From a research centre or group, Leo (*Data Scientist* role) wants to track back which AI models were trained using a specific *Dataset,* since a new *Dataset* with updated data is available, and wants to retrain the models. For this purpose, Leo gets access to the *Tracer Service* of the *CHAIMELEON repository* and queries the models using those *Dataset Ids* and identifies the new ones to be trained again. She is confident that she could track back all of them by consulting the *Tracer Service* of the repository.	

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant

Table 9. *Traceability of Data Usage Story.*

Story name: Traceability (Data usage)	**Provided by:** UPV-I3M
Triggered by User Role (Fictitious Name): Clinical Staff (Anna), authorized Technical Data Manager (Ignacio), Institutional DPO (Tony)	**Other User Roles (Fictitious Name):** NONE
Potential User Role can be found in: HULAFE**,** UM, UNIPI, CHA, CERF, CHUP, CHA, UM**.**	**Related WP:** WP3, WP4
Overview description: Ana (*Clinical Staff* role) has contributed to the *CHAIMELEON Repository* with some studies from her institution. Months later, she wants to track who has used and which models have been created using the data from her institution. This request can also come from Tony (*Data Protection Officer* role) or Ignacio (authorized *Technical Data Manager* role). S(he) enters the *Dataset Explorer Application* of the repository and gets the *Datasets* where her institution has contributed. With this information, she can query which users accessed the *Dataset* and which models have been created using the *Dataset* through the *Tracer Service***.**	

Table 10. *Repository Applications Management.*

Story name: Repository Applications Management	**Provided by:** UPV-I3M
Triggered by User Role (Fictitious Name): Cloud Services and Security Manager (Sergio)	**Other User Roles (Fictitious Name):** Application Developers and Data Scientists
Potential User Role can be founded in: UPV	**Related WP:** WP4, WP3, WP6, WP7, WP8
Overview Description: Sergio (*Cloud Services and Security Manager* role) has a request from Alice (*Application Developer* role*)* or Leo (*Data Scientist* role) to provide an execution environment to run a set of specific tools required for their development or research. Sergio has administrative credentials to the platforms and especially to the *Application Registry* of the repository. Similarly, Sergio detects that some of the applications available need to be updated for security reasons or by demand of Alice or Leo. For this purpose, Sergio uses DevOps tools to automate the building and uploading of the applications according to the requirements provided by the researchers. Sergio codes the building of the application environment and after verifying it, he uploads it in the *Application*	

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant

Registry. From this point on, the application becomes available to the users of CHAIMELEON.

Table 11: *Image Annotation and Segmentation Story.*

Story name: Image Annotation and Segmentation	**Provided by:** MEDEXPRIM
Triggered by User Role (Fictitious Name): Clinical staff (Anna)	**Other User Roles (Fictitious Name):** Technical Data manager (Ignacio) and Data scientist (Leo)
Potential User Role can be found in: HULAFE, CERF, PSD, ULS, UNIPI, CHA, UM, CHUP.	**Related WP:** WP5
Overview description: Anna (*Clinical Staff* role) is a Radiologist from a Medical Imaging Department of a Hospital who wants to annotate and segment images using tools from the *CHAIMELEON Repository*, and also to make a qualitative validation of output from AI models through an image Viewer. Ana wants to enrich data that will be stored in the *Data Lake* storage of *CHAIMELEON Repository* (e.g. add annotations or segmentations to images that lack those information for the training and validation of an AI model that a *Data Scientist* wants to develop before a *Dataset Administrator* creates a Dataset (table 2)**.** Anna accesses to the *Application Dashboard* of the *CHAIMELEON Repository* and runs a processing console (e.g. tools to quickly draw binary mask, select targeted features such as a simply click and drag to create a bounding box on a specific organ, segmentation tools, Image viewers, labialize images). The results become available for the creation of a new *Dataset* and they have to upload means of Ignacio (*Technical Data Manager* role).	

Table 12. *Evaluation of Model by Peer Review Story.*

Story name: Evaluation of Model by Peer Review	**Provided by:** MEDEXPRIM
Triggered by User Role (Fictitious Name): Data Scientist (Leo)	**Other User Roles (Fictitious Name):** Clinical Staff (Anna*)*
Potential User Role can be found in: CHA, BGU, IMPERIAL, QUIBIM, BAHIA, GEHC	**Related WP:** WP8
Overview description:	

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant

Leo (*Data Scientist*) wants to do a peer review for validating an AI model published by her or other researcher in the *Source Code Repository* of the *CHAIMELEON Repository*, and reinforce the confidence in a methodology for solving specific problems. For this purpose, Leo gets access to the *Application Dashboard* of *CHAIMELEON Repository* to deploy an instance of such a model, retrieved from the *Source Code Repository,* in a processing environment. She uses her own *Dataset* to make a validation of the model. Leo publishes her peer review result as an evaluation of the AI model in the *Source Code Repository*. This information could be supported by supplementary information about a specific model, or by giving a set of quantitative data evaluating the efficiency and precision of the classification, detection or other output given by the model. Review information on the models is useful to *Ana (Clinical Staff* role*)* to help her selecting the most suitable AI tool based on these models while searching a tool through the *marketplace*, as well as for the developer of the AI model, who could even ask other roles to do this task, in the same way that peer reviews for scientific articles work.

Table 13. *Feasibility Search Story.*

Story name: Feasibility Search	**Provided by:** MEDEXPRIM
Triggered by User Role (Fictitious Name): External Researcher (Julia)	**Other User Roles (Fictitious Name):** NONE
Potential User Role can be found in: External organizations	**Related WP:** WP8
Overview description: From a research centre or group, Julia (*External Researcher* role) wants to know what could be available (AI tools, AI models or *Datasets*) for her in the *CHAIMELEON Repository* and be able to use them for an external project. For this purpose, Julia gets access to the *Source Code Repository*, *Marketplace* or *Dataset Explorer Application* and he has got permission to search published AI models and tools and existing *Datasets*. She could see aggregated information and run some basic exploratory tools to get an appraisal of the quality and relevance of the data available in *the repository* for her research interest.	

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant

Table 14. *Data Privacy Evaluation Story.*

Story name: Data privacy Evaluation	**Provided by:** MEDEXPRIM
Triggered by User Role (Fictitious Name): Data Protection Officer (Tony)	**Other User Roles (Fictitious Name):** NONE
Potential User Role can be found in: HULAFE, UM, UNIPI, CHA, CERF, CHUP, CHA, UM	**Related WP:** WP2, WP5
Overview description: Tony (*Data Protection Officer Role*) is the Institutional DPO of a centre and needs to create reports to support the ethical reports and audits that guarantee that the appropriate privacy-preservation mechanisms have been applied. In particular, he would like to get a report on the anonymization of the data available in the *Data Lake* storage of *CHAIMELEON Repository* and quantify the risk of re-identification on them. For this purpose, Tony gets access to the existing *Datasets* through *Dataset Explorer Application* and gets the results that include an evaluation of the risk of re-identification. Additionally, he can get a report of who has used the *Dataset* and which models have been created and for which purpose.	

Table 15. *Open Challenges Story.*

Story name: Open Challenges	**Provided by:** HULAFE-GIBI230
Triggered by User Role (Fictitious Name): External Research (Juan)	**Other User Roles (Fictitious Name):** NONE
User Role can be found in: External Researchers (Open Challenge Participants)	**Related WP:** WP8
Overview description: From a research centre or group, Juan (*External Research* role) wants to participate in an Open Challenge promoted by *CHAIMELEON Project*. He wants access to an application deployed for this purpose in the *CHAIMELEON Repository* to build and validate the models of the open challenge. For this purpose, Juan gets the application which has limited access to those *Datasets* and models that are to be used in the Open Challenge only and provides a framework where he can carry out all actions required to participate in the challenge (such as develop, train and test his AI tool, and provide results).	

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant

Table 16 summarizes all these *User Stories*.

Table 16. *Generic User stories.*

US#	**User Stories**	**Description**	**Trigger User Role**
1	*Data Ingestion*	*Uploading (manually or in-bulk) of data in the Data Lake of the repository.*	authorized Technical Data Manager
2	*Dataset Creation*	*A subset of the data (Dataset) in the Data Lake is registered as a shareable research object in the repository.*	*Dataset Administrator*
3	*AI Model Creation*	*Develop an AI model using all or part of the data stored in a Dataset.*	*Data Scientist*
4	*AI model Improvement*	*Rebuild an existing AI model using new, updated data or different parameters or extend it to create a new tool or model for different data.*	*Data Scientists*
5	*Image Processing Tool Development*	*Develop an Image Processing tool*	*Application Developer*
6	*Data Update*	*Data Lake data is updated in the repository.*	*authorized Technical Data Manager*
7	*Exploitation*	*Apply an existing analytical or AI tool to new data.*	*Clinical Staff*
8	*Traceability (Models created with a specific dataset)*	*Check what AI tools have been developed using a specific Dataset which has been updated.*	*Data Scientist*
9	*Traceability of Data Usage*	*Check which users have accessed a specific Dataset containing specific cases and which models have been created.*	*Clinical Staff, authorized Technical Data Manager, Institutional DPO*
10	*Repository Applications Management*	*Cloud resources are managed to install, configure and deploy specialized frameworks/applications.*	*Cloud Services and Security Manager*
11	*Image Annotation and Segmentation Story*	*Enrich data with annotation, segmentation, biomarkers, etc.*	*Clinical Staff*
12	*Evaluation of Model by Peer Review*	*Peer review of AI Tools in the Marketplace.*	*Data Scientists*
13	*Feasibility Search*	*Exploration of the Aggregated information of the existing AI tools and Datasets.*	*External Researcher*
14	*Data privacy Evaluation*	*Data Anonymity evaluation.*	*Institutional DPO*
15	*Open Challenges*	*Participation in Open Challenges.*	*External Researcher*

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant

4.3 Results

As a result of the elicitation phase, we derived a set of *User Roles* and *Components* that composes the architecture of *CHAIMELEON Repository*. These are described in the next subsections.

4.3.1 User Roles

User Roles are described below along with the project partners who assume them in the context of the project.

- **Cloud *Services and Security Management role (Sergio):*** This role is in charge of different tasks to administer and manage the *CHAIMELEON Repository*. These are the follows:
    - To manage the Infrastructure as a Service (IaaS), the Development and Operations (DevOps) and the building of the containers that host the applications. These tasks serve to automate the deployment of virtual infrastructures that comprise the *CHAIMELEON Repository* and storages, services and applications that run and set-up on top of it in a secure and agile way.
    - To manage the users accepted by the *CHAIMELEON Access Committee*, which analyses and approves all the user access requests. These tasks correspond to the management tasks such as assignment of roles, groups and capabilities to the accepted users.
    - To manage security issues of the *CHAIMELEON Repository.* This user will make the security assessment, evaluate risks, apply software patches, subscribe to security vulnerability bulletins, validate backups and act as a contact point for security incidents.

  Partners supporting this role: UPV.

- **authorized Technical Data Manager (Ignacio):** This role deals with 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, anonymization and curation, to name a few, which are performed at the data providers centres through specific tools (e.g. Medexprim Suite(™)). These tasks are carried out entirely in the medical centre side and do not require access rights to the repository.

  After performing these tasks, this user has access rights to ingest the fully anonymized 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: HULAFE, CERF, PSD, ULS, UNIPI, CHA, UM, CHUP.

- **Clinical Staff (Anna)**: This role deals with both the provision (provider) and consumption (consumer) of medical data and the use (consumer) of high-level data processing tools provided by the *CHAIMELEON Repository.*

  As a 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 and ingested into the *Data Lake* by the *authorized Technical*

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant

*Data Manager***.** These tasks are carried out entirely on the medical centre side and do not require access right to the repository.

As consumers, they use the AI tools provided by the *Marketplace* of the CHAIMELEON *Repository*. These AI tools process medical data, annotate, segment and enrich data and extract knowledge to refine diagnosis, prognosis, treatment diagnosis and so on.

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

- **Dataset Administrator (Carmen)**: This 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.

  Partners supporting this role: CERF (Lung), ULS (Colorectal), PSD (Breast), Prostate (HULAFE).

- **Data Protection Officer (Tony)**: This role deals with the surveillance of the fulfilment of the data protection regulations and the traceability of the data usage along the project.

  Partners supporting this role: HULAFE, CERF, PSD, ULS, UNIPI, CHARITE, UM, CHUP.

- **Data Scientist (Leo)**. This role deals with accessing the *Dataset* available at the *CHAIMELEON Repository* and requesting specific processing environments to deal with them. This user has access rights to a console with processing tools and is able to register the AI trained models and tools developed in the *Marketplace*. The user is also able to trace the usage of an AI model and to use other models to compare to.

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

- **External Researcher (Julia)**. This role deals with browsing the aggregated information of the AI tools and models published in the *Source Code Repository* and *marketplace* 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 to the *Datasets* and processing tools provided by the repository.

  Partners supporting this role: External to the consortium.

- **Application Developer (Julia).** This 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 *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 Management.*

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

4.3.2 Components of the Architecture

The components of the repository architecture are described below. We have identified five types of components, which are the following:

- **Storages.** They provide persistence of data (including medical data, application binaries and specifications, traceability logs, users' profile) in the *CHAIMELEON*

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant

Repository. They are uniquely managed by *Services* or *Applications* and cannot be accessed directly by the end users.
- **Platform Applications.** Applications providing the end-users with the functionalities offered by the *CHAIMELEON Repository* through user-friendly interfaces. The users will only interact with the repository through these applications that implement the functionalities through the *Services*.
- **Services.** *Services* provide the functionality required by the end-user's applications (*Platform Applications*), interacting with the resources such as the *Storages* 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 in the *Case Explorer Application*.

The *Storages* identified 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 anonymized 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 minimize data replication but ensuring 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 a 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.
- ***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 development of *Processing and AI tools*.

The *Services* are the following:

- ***Authentication 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 to *Platform Applications*, *Services* and *Processing*

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant

Applications all required information to implement their user authentication and authorization processes.

- ***Data Ingestion/Access Service***. This service provides to the *Platform Applications* all the necessary functionalities 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 to ingest data (in bulk) into the *Data Lake*.

- ***Dataset Service.*** This service provides to the *Platform Applications, Services* and *Processing Applications* all necessary functionalities to manage the registration of *Datasets* as sets 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*.

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* is 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 Applications*), and allows *Clinical Staff* to employ it in the medical centers 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 the information about its 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 follows:

- ***Processing and AI tools.*** These tools are processing tools 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*.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant

- ***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), harmonization tools (provided by Imperial College and QUIBIM), tools for ensuring that the Images harmonized are Secure (provided by BGU) and tools and frameworks to AI developers such as Jupyter server with Tensorflow or Pytorch.

This project has received funding from the European Union's
Horizon 2020 research and innovation programme under grant

5. Use Cases

In the second phase, *Use Cases* have been extracted and defined from the collected *User Stories*. The *Use Cases* define a complete interaction flow supported by the *CHAIMELEON Repository* among the identified components of the architecture (*Users*, *Platform Applications*, *Services*, *Storages* and *Processing Applications*). These refer (but are not limited) to the authentication and authorization processes, *Data Lake* data management, creation and management of *Datasets*, deployment of Processing Applications and so on.

Next subsections describe the most important *Use Cases* identified in the project.

5.1 Authentication & authorization

Regarding the Authentication and authorization (AA) processes in the *CHAIMELEON Repository*, 8 main use cases (see Figure 5) have been identified and defined. In all *User Stories* there are different types of connections where AA processes are carried out. Any *User*, regardless of their *User Role*, must be authenticated, and subsequently must be authorized by the *Platform Application* or *Service* which has been accessed based on his/her *User Role*.

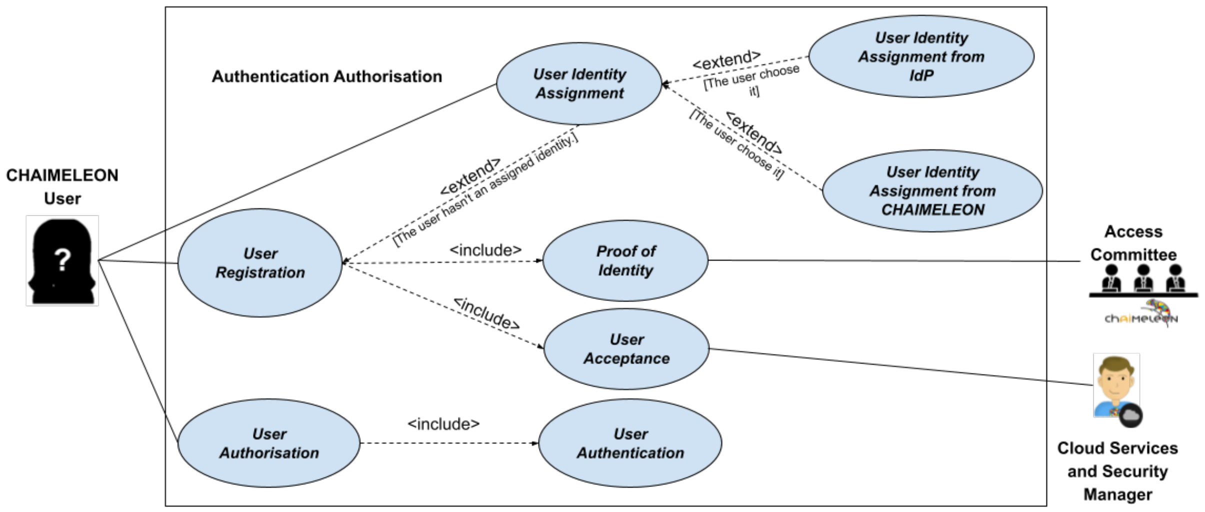


Figure 5.*UML Use Cases identified for Authentication and authorization Processes.*

5.1.1 User Identity Assignment from IdP

For any AA process where a user accesses an *Platform Application or Service*, an identity is required for the user, which is usually translated into a *User ID* (e.g. e-mail, token or UID). Today, most people have an account on a recognized Identity Provider (IdP). Institutional IdPs associated with EDUGain (https://edugain.org/) will be preferred, although other popular IdPs such as Google ID, Microsoft Live ID, Linked-In or Facebook may be accepted.

Thus, any *User* must acquire an identity from an IdP accepted by the *CHAIMELEON Repository*. Figure 6 shows the UML sequence diagram corresponding to the interaction among IdP services and *Users* to implement the acquisition of a user identity from an IdP.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant

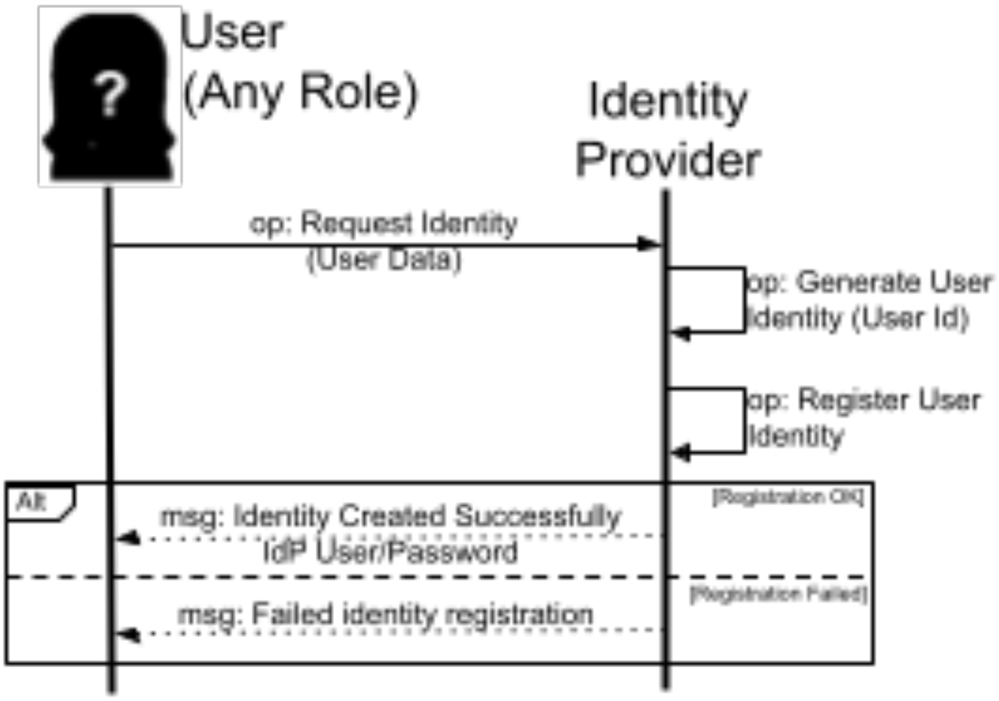


Figure 6. UML *Sequence Diagram for Identity Acquisition of Users.*

5.1.2 User Identity Assignment from CHAIMELEON

Another procedure to acquire an identity to access the *CHAIMELEON Repository* is when the local repository provides the identity through User and Password. This use case is only for *Users* who are not able to acquire an identity from the repository accepted *IdPs*. The identity from IdPs is a priority but this scenario is also considered.

Figure 7 shows the UML sequence diagram corresponding to the interaction flow among *Users* and *Components* involved to implement the acquisition of a user identity (*User ID*) from *CHAIMELEON Repository*.

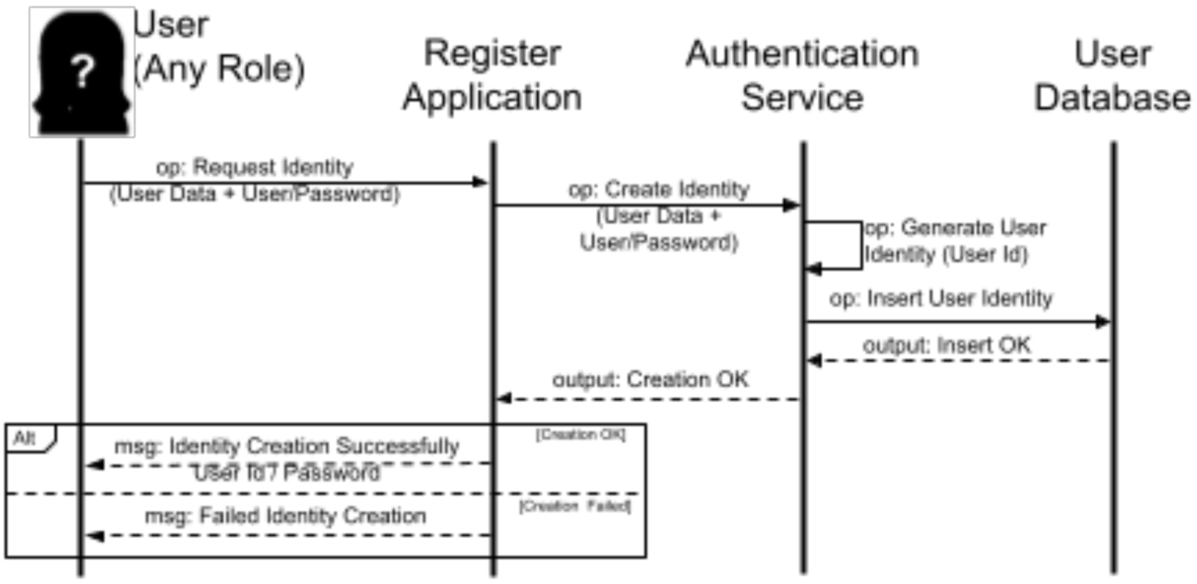


Figure 7. UML *Sequence Diagram for Identity Creation of Users from CHAIMELEON Repository.*

This project has received funding from the European Union's
Horizon 2020 research and innovation programme under grant

5.1.3 User Registration & Proof of Identity

CHAIMELEON Users must be registered in the *CHAIMELEON Repository* through his/her identity (*User ID)* provided by an IDP or directly from *CHAIMELEON Repository*.

For the registration, the identity of the user and the relation to his/her organization must be proved. For the sake of liability, *CHAIMELEON Repository* will not provide access to generic accounts or unauthenticated users. By binding the *User ID* to a proof of identity, we guarantee that an account is linked to a specific real person, who must be liable to the actions performed in any *Platform Application* with the account. The registration will include a document with the Terms of Usage of the platform and the data.

The proof of identity could be:

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

The user registration must be managed by a specific *Platform Application* named *User Registration Application***.** This application must securely keep the successfully registered *User IDs* and its associated metadata (user role, groups and capabilities) in the *User Database* through the *Authentication Service*. The metadata information will be used by the *Platform Applications* for granting access to users.

Figure 8 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.

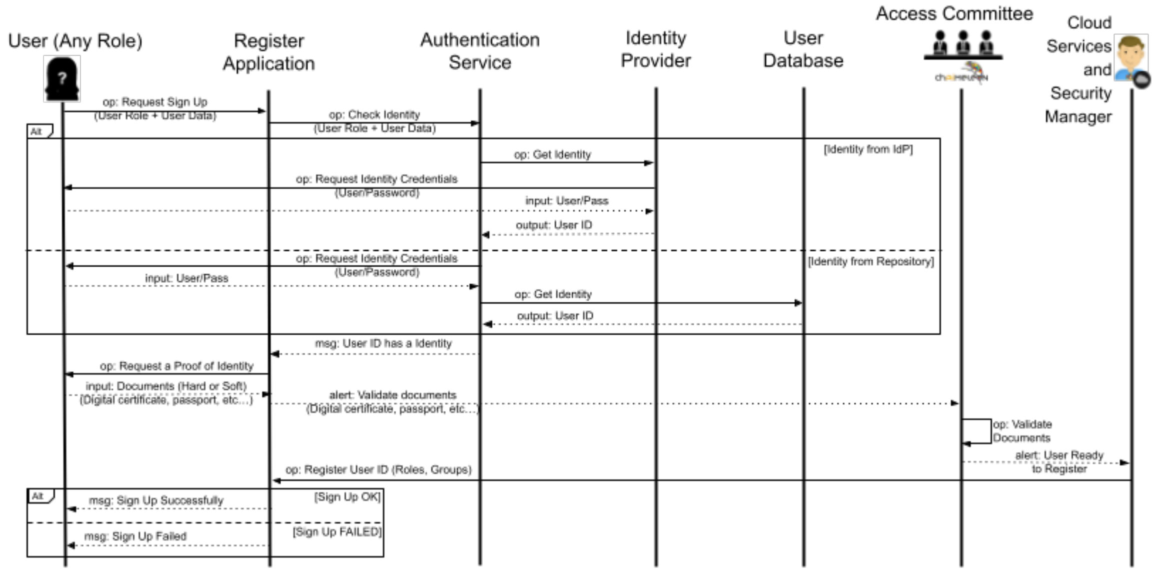

User (Any Role)
Register Application
Authentication Service
Identity Provider
User Database
Access Committee
Cloud Services and Security Manager
op: Request Sign Up (User Role + User Data)
op: Check Identity (User Role + User Data)
Alt
op: Get Identity
[Identity from IdP]
op: Request Identity Credentials (User/Password)
input: User/Pass
output: User ID
op: Request Identity Credentials (User/Password)
[Identity from Repository]
input: User/Pass
op: Get Identity
output: User ID
msg: User ID has a Identity
op: Request a Proof of Identity
input: Documents (Hard or Soft) (Digital certificate, passport, etc…)
alert: Validate documents (Digital certificate, passport, etc…)
op: Validate Documents
alert: User Ready to Register
op: Register User ID (Roles, Groups)
Alt
msg: Sign Up Successfully
[Sign Up OK]
msg: Sign Up Failed
[Sign Up FAILED]

Figure 8. UML *Sequence Diagram for the Registration Process to CHAIMELEON Repository.*

5.1.4 User Authentication

Users must be authenticated by the *Platform Application* and in the case of in-bulk data ingestion processes by the *Data Ingestion/Access Service* through his/her identity (*User ID)* provided by an IDP or directly from *CHAIMELEON Repository* for verifying that s(he) is a real user.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant

Figure 9 shows the UML sequence diagram corresponding to the interaction flow among *Users* and *Components* involved to implement the any *User* authentication process in any *Platform Application* or the *Data Ingestion/Access Service*.

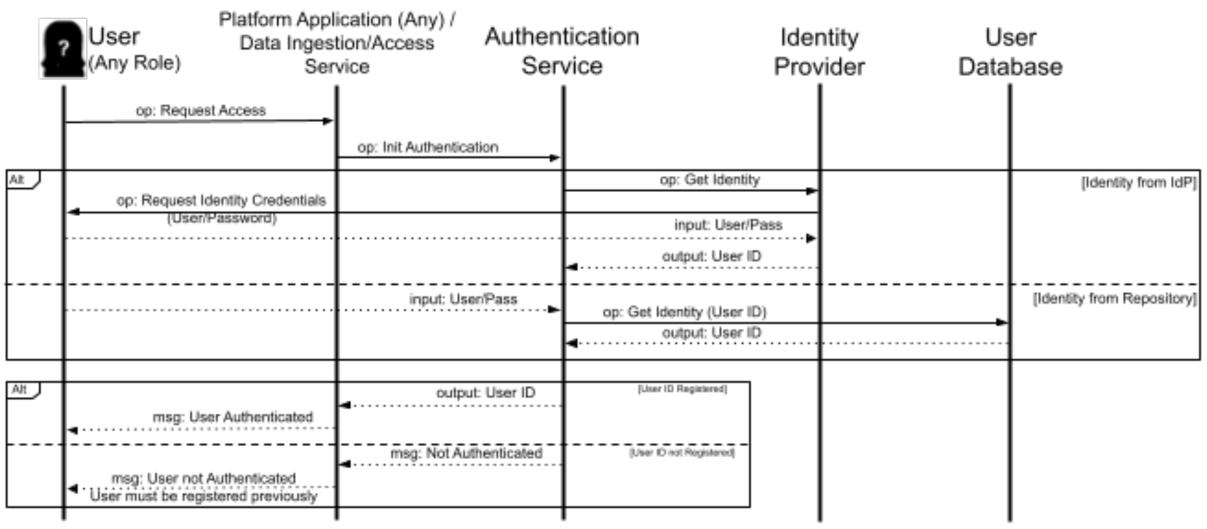


Figure 9. *UML Sequence diagram for the Authentication process.*

5.1.5 User authorization

Users must be authorized to perform any specific action in a *Platform Applications Platform Application* and in the case of in bulk data ingestion processes by the *Data Ingestion/Access Service*. This authorization process must be carried out by the own *applications* or *Data ingestion/Access services* in function of the presented authenticated identity (*User ID*) and its associated metadata (e.g. user role, group, capabilities) provided by the *Authentication Service*.

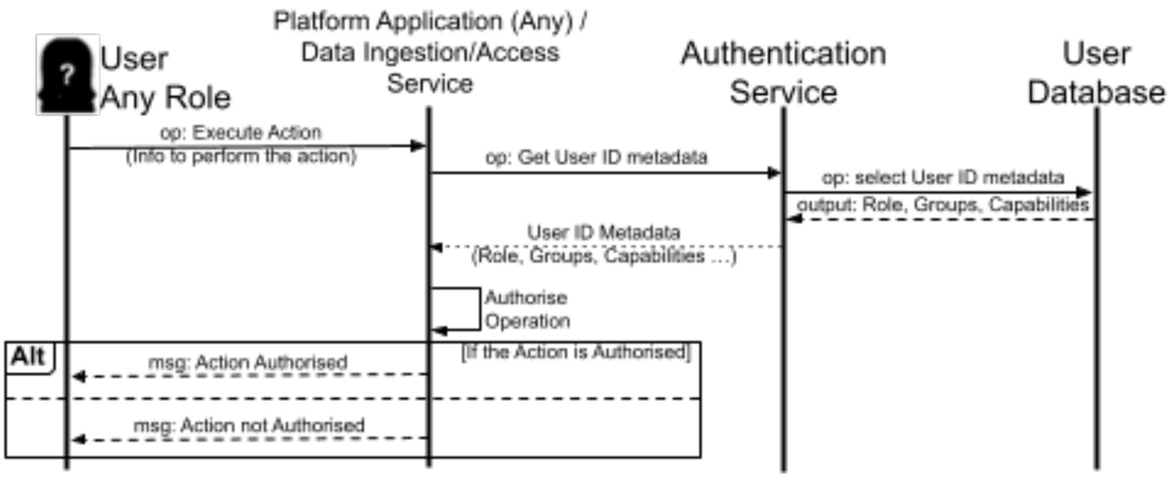

User Any Role
Platform Application (Any) / Data Ingestion/Access Service
Authentication Service
User Database
op: Execute Action (Info to perform the action)
op: Get User ID metadata
op: select User ID metadata
output: Role, Groups, Capabilities
User ID Metadata (Role, Groups, Capabilities ...)
Authorise Operation
Alt
[If the Action is Authorised]
msg: Action Authorised
msg: Action not Authorised

Figure 10. *UML Sequence diagram for User authorization.*

Figure 10 shows the UML sequence diagram corresponding to the interaction flow among *Users* and *Components* involved to implement the authorization process in any *Platform Application* and *Data Ingestion/Access Service* to allow a *User* to execute a specific action.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant

5.2 Data Lake Management

Regarding *Data Lake* data management, 4 main use cases (see Figure 11) have been identified from the stories. The *Use Cases* are described in next subsections.

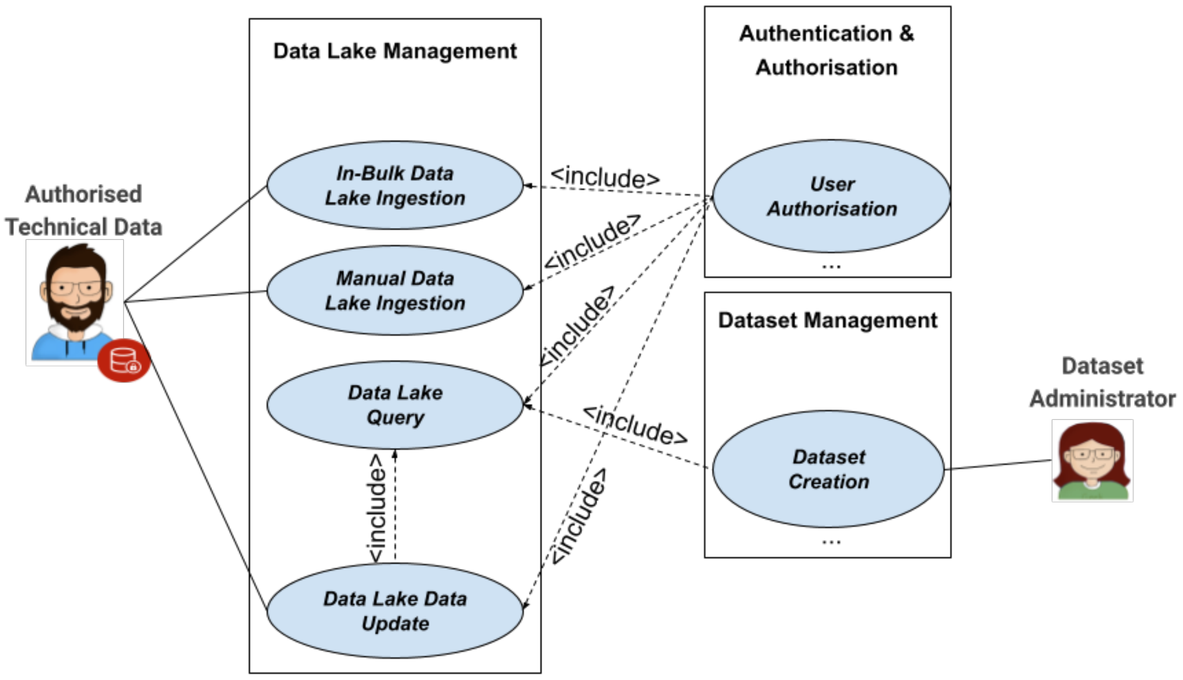


Figure 11.*UML Use Cases identified for Data Lake Management.*

5.2.1 In-Bulk Data Lake Ingestion

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

As Figure 12 shows, this high-level interaction requires two steps. The first one, a *User* delegates his/her Credentials to an in-bulk ingestion application (such as Medexprim Suite™) 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 *authorized 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.*

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant

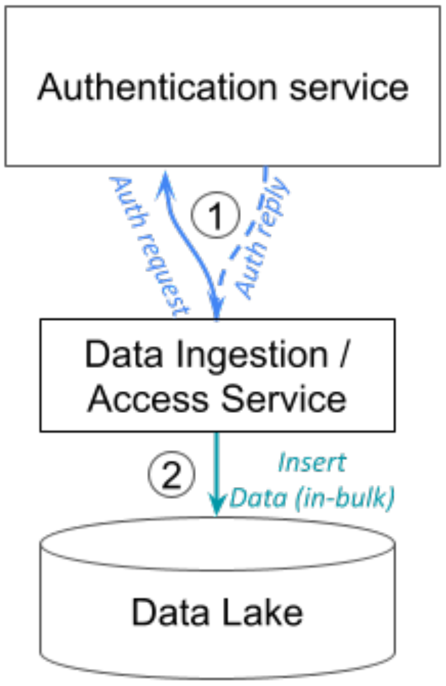


Figure 12. *In-Bulk Ingestion Data High Level Interactions.*

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

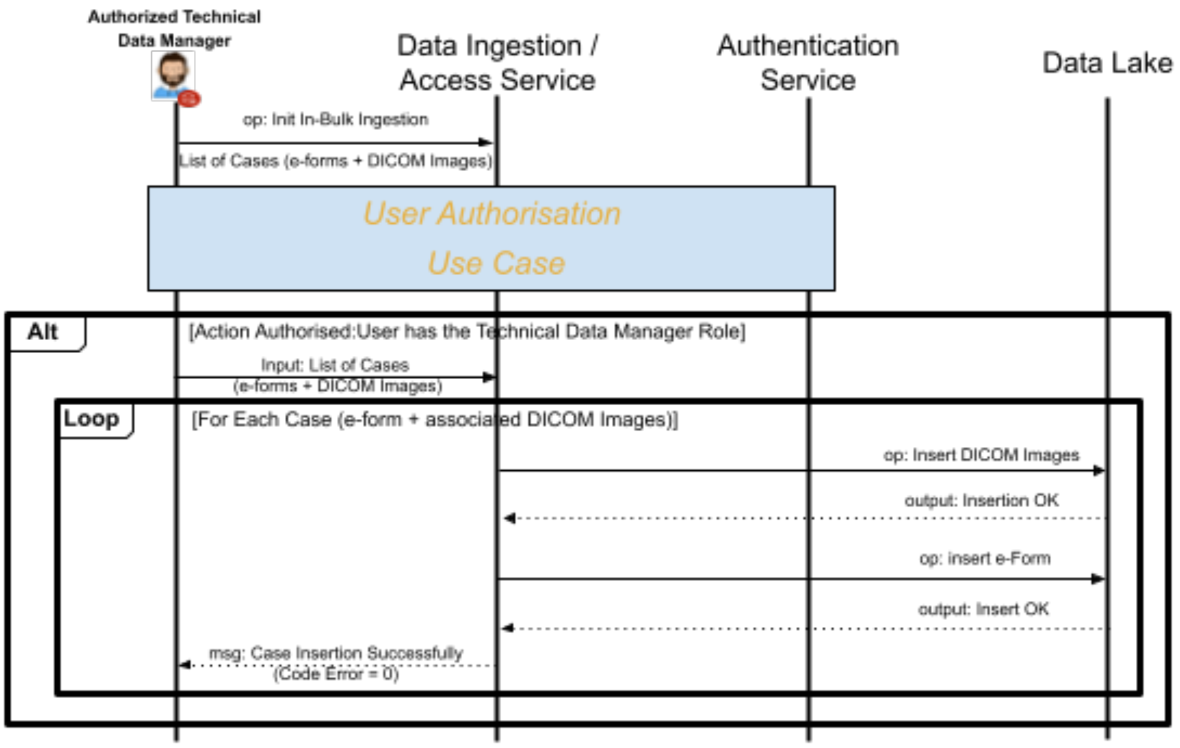

Authorized Technical Data Manager
Data Ingestion / Access Service
Authentication Service
Data Lake
op: Init In-Bulk Ingestion
List of Cases (e-forms + DICOM Images)
User Authorisation Use Case
Alt
[Action Authorised:User has the Technical Data Manager Role]
Input: List of Cases (e-forms + DICOM Images)
Loop
[For Each Case (e-form + associated DICOM Images)]
op: Insert DICOM Images
output: Insertion OK
op: insert e-Form
output: Insert OK
msg: Case Insertion Successfully (Code Error = 0)

Figure 13. *UML Sequence Diagram for In Bulk Ingestion Data.*

5.2.2 Manual Data Lake Ingestion

This *Use Case* defines the required interactions to ingest data into *Data Lake* manually**.**

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant

As Figure 14 shows, this high-level interaction requires three steps. The first one is to authorize *Users* for carrying out the ingestion of data. Only users with an authenticated *authorized Technical Data Manager Role* will be authorized by the *Case Explorer Application* to ingest data to *Data Lake* manually in the name of their organizations. 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*.

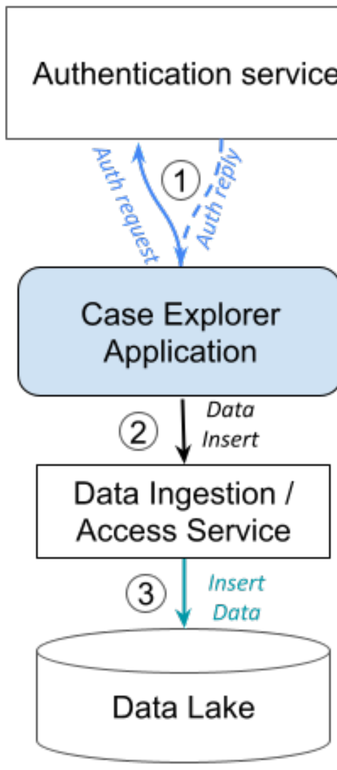
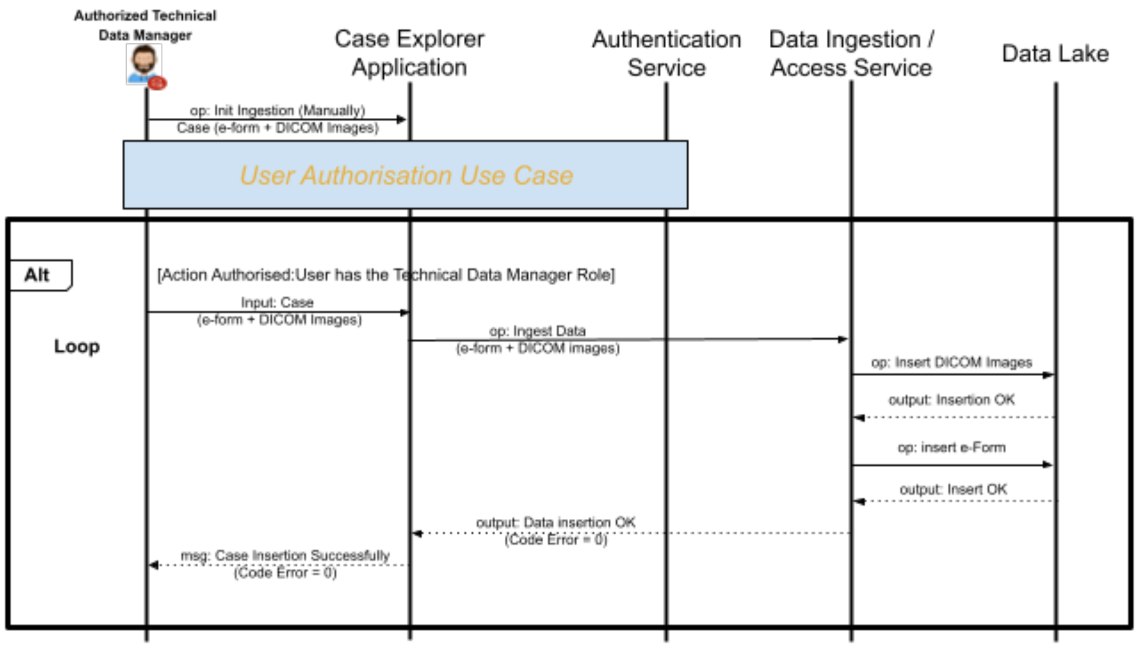


Figure 14. *In-Bulk Ingestion Data High Level Interactions.*

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



Authorized Technical Data Manager
Case Explorer Application
Authentication Service
Data Ingestion / Access Service
Data Lake
op: Init Ingestion (Manually) Case (e-form + DICOM Images)
User Authorisation Use Case
Alt
[Action Authorised:User has the Technical Data Manager Role]
Loop
Input: Case (e-form + DICOM Images)
op: Ingest Data (e-form + DICOM images)
op: Insert DICOM Images
output: Insertion OK
op: insert e-Form
output: Insert OK
output: Data insertion OK (Code Error = 0)
msg: Case Insertion Successfully (Code Error = 0)

Figure 15. *UML Sequence Diagram for Manual Ingestion Data.*

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant

5.2.3 Data Lake Query

This *Use Case* defines the required interactions to query data from the *Data Lake***.** This use case happens prior to updating *Data lake* data (see *Data Lake Data Update 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.

As Figure 16 shows, the high-level interaction requires three steps. The first one is to authorize the *User.* Only users with *Dataset Administrator Role* or *authorized Technical data manager Role* authenticated will be authorized by the *Case Explorer Application* to query data to the *Data Lake***.** In the second step**,** *Case Explorer Application* query and filter 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.

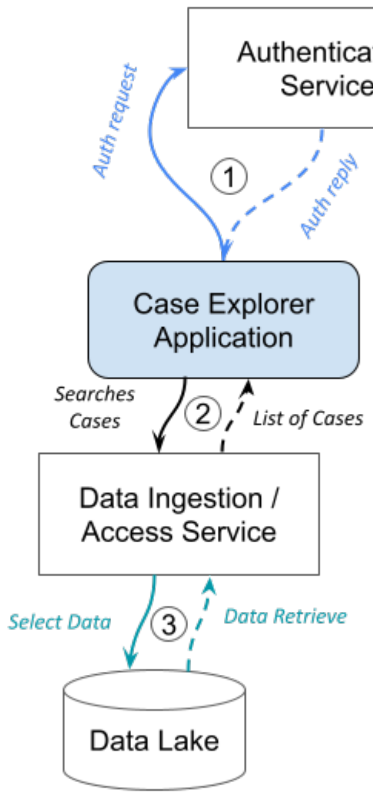


Figure 16. *Data Lake Query High Level Interactions.*

Figure 17 shows the UML sequence diagram corresponding to the interaction flow among *Users* and *Components* involved to query data from the *Data Lake*.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant

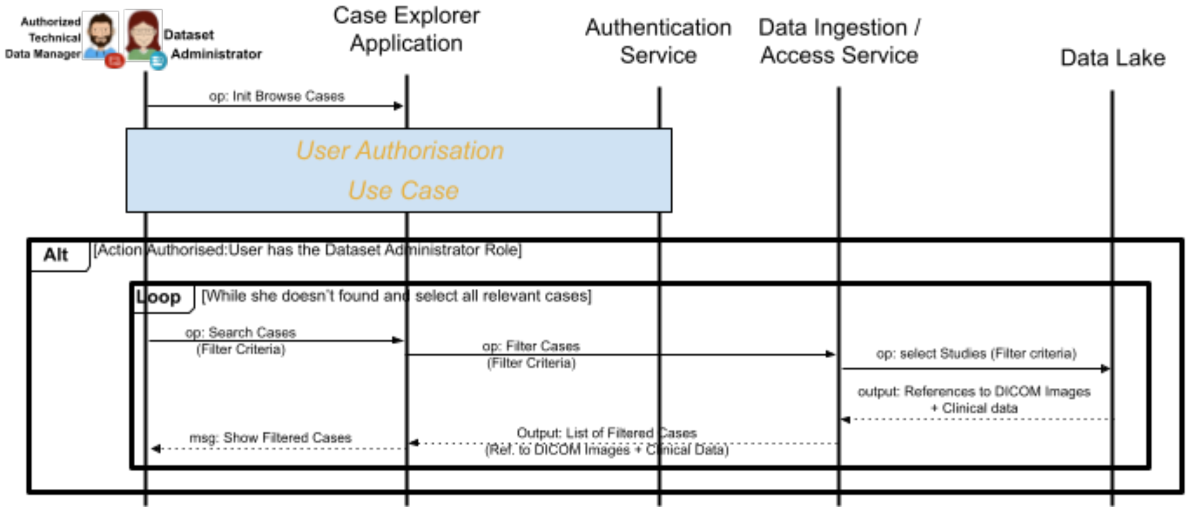

Authorized Technical Data Manager
Dataset Administrator
Case Explorer Application
Authentication Service
Data Ingestion / Access Service
Data Lake
op: Init Browse Cases
User Authorisation Use Case
Alt
[Action Authorised:User has the Dataset Administrator Role]
Loop
[While she doesn't found and select all relevant cases]
op: Search Cases (Filter Criteria)
op: Filter Cases (Filter Criteria)
op: select Studies (Filter criteria)
output: References to DICOM Images + Clinical data
Output: List of Filtered Cases (Ref. to DICOM Images + Clinical Data)
msg: Show Filtered Cases

Figure 17. *UML Sequence Diagram for Data Lake Query.*

5.2.4 Update Data Lake Data

This *Use Case* defines the required interactions to update or remove data that previously were ingested into *Data Lake* (see *Data Lake Ingestion Use Case*)**.** 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 (see *Dataset Creation Use Case*), a copy of the involved e-forms is recorded (these are not references), guaranteeing the consistency. Finally, if data belonging to a *Dataset* has been involved in an update process, the *Dataset* must be marked as "source data updated".

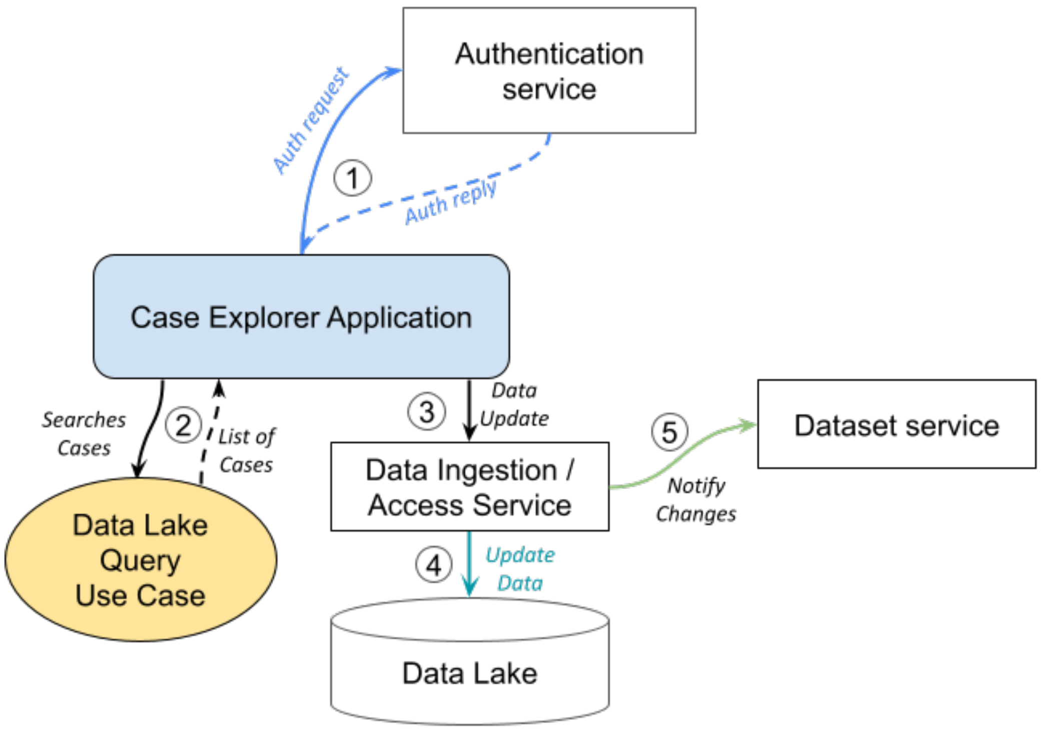


Figure 18. *Data Update High Level Interactions.*

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant

As Figure 18 shows, the high-level interaction requires five steps. The first one is to authorize *Users* for carrying out the data update. Only users with *authorized Technical Data Manager Role* authenticated will be authorized by the *Case Explorer Application* to update data from *Data Lake*. In the second step, the user looks for the cases to be updated (See *Data Lake Query Use Case*). Next, in the third step, *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 19 shows the UML sequence diagram corresponding to the interaction flow among *Users* and *Components* involved to update *Data Lake* data.

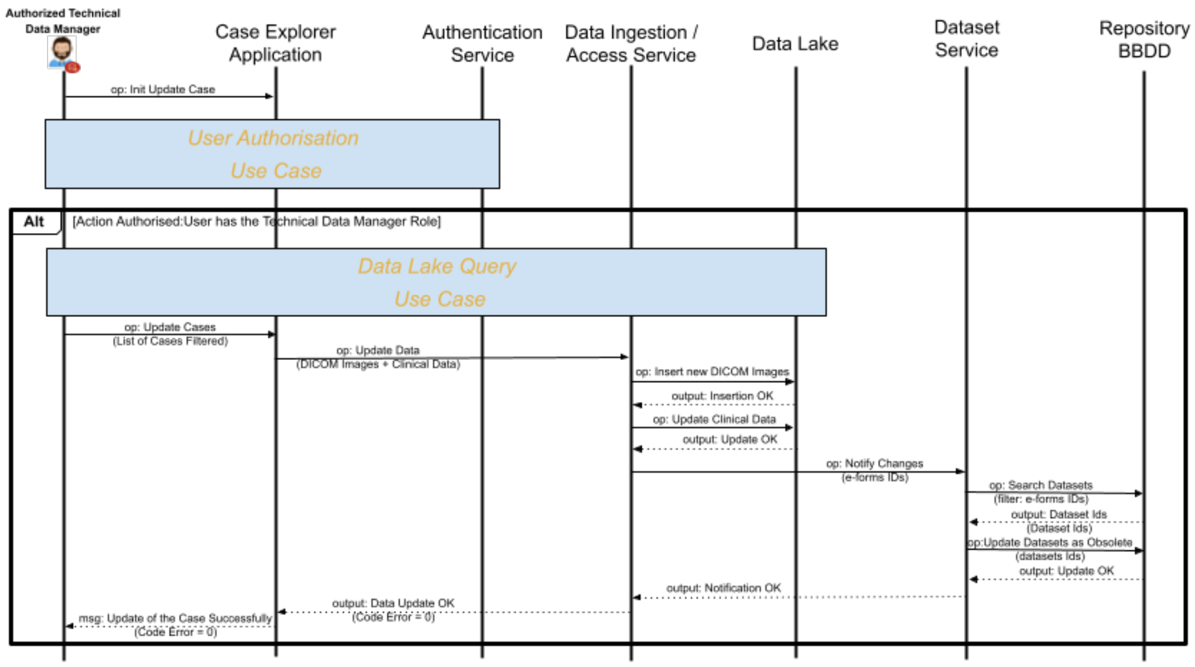

Authorized Technical Data Manager
Case Explorer Application
Authentication Service
Data Ingestion / Access Service
Data Lake
Dataset Service
Repository BBDD
op: Init Update Case
User Authorisation Use Case
Alt
[Action Authorised:User has the Technical Data Manager Role]
Data Lake Query Use Case
op: Update Cases (List of Cases Filtered)
op: Update Data (DICOM Images + Clinical Data)
op: Insert new DICOM Images
output: Insertion OK
op: Update Clinical Data
output: Update OK
op: Notify Changes (e-forms IDs)
op: Search Datasets (filter: e-forms IDs)
output: Dataset Ids (Dataset Ids)
op:Update Datasets as Obsolete (datasets Ids)
output: Update OK
output: Notification OK
output: Data Update OK (Code Error = 0)
msg: Update of the Case Successfully (Code Error = 0)

Figure 19. *UML Sequence Diagram for Updating Data Lake Data.*

5.3 Dataset Management

Regarding *Datasets* management, 3 main *Use Cases* (see Figure 20) have been identified from the stories. The *Use Cases* are described in next subsections.

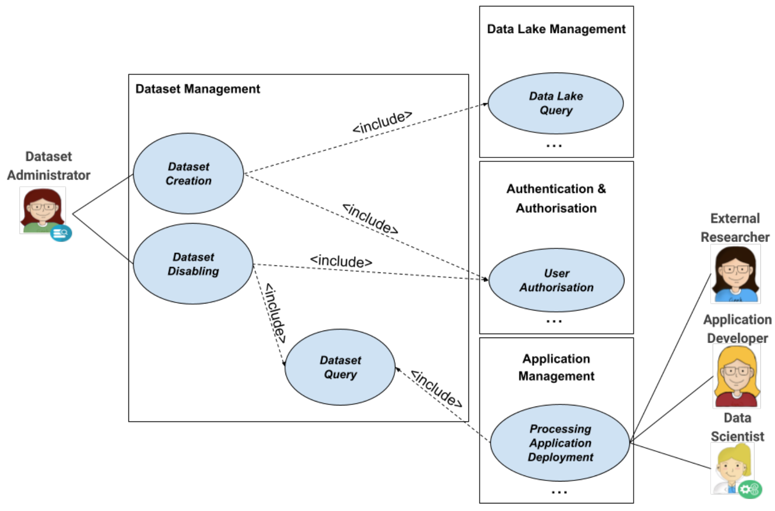


Figure 20.*UML use cases identified for Dataset Management.*

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant

5.3.1 Dataset Creation

This *Use Case* defines the required interactions to create *Datasets* employing data from the *Data Lake*. *Datasets* aim to give better traceability about what medical data is employed to build a given tool and enable reproducibility in the development of processing methods and AI Tools.

As Figure 21 shows, the first three steps correspond to the *Data Lake Query Use Case* to select a set of *Data Lake* data. Only users with *Dataset Administrator Role authenticated* will be authorized 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.

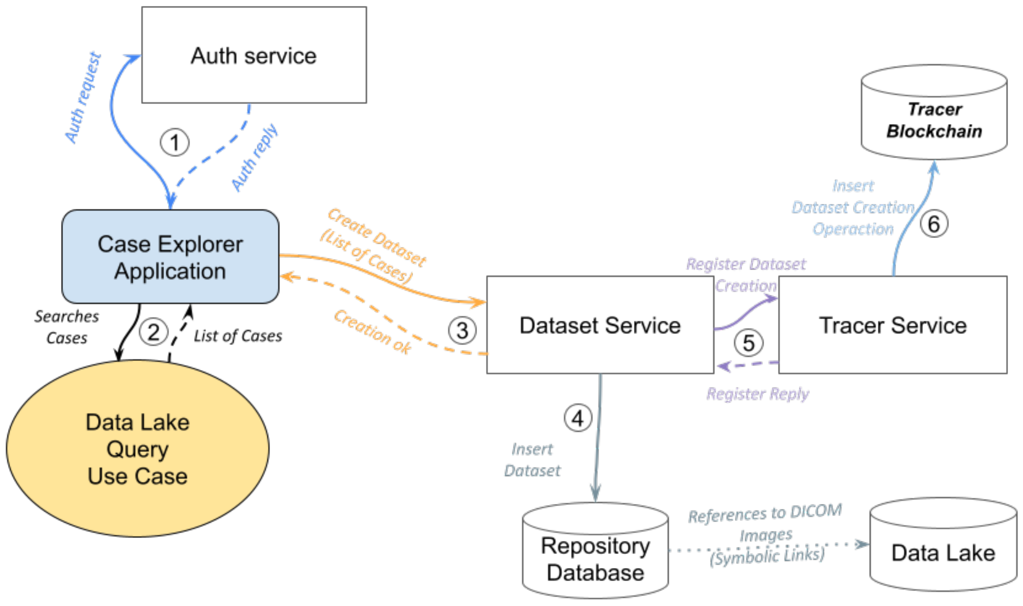


Figure 21. *Dataset Creation High Level Interactions.*

Figure 22 shows the UML sequence diagram corresponding to the interaction flow among *Users* and *Components* involved to create datasets from *Data Lake.*

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant

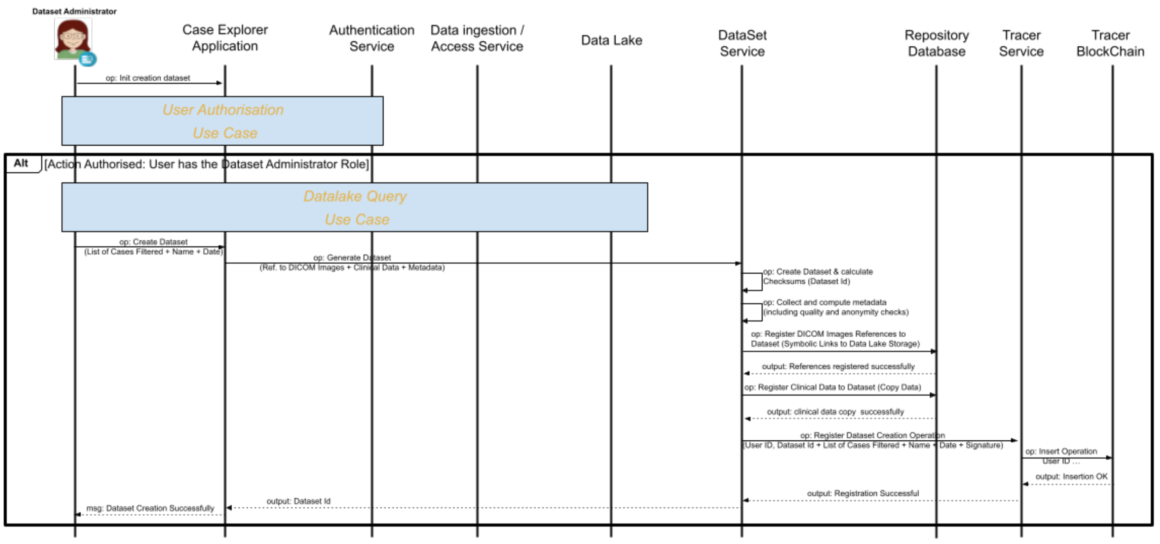

Dataset Administrator
Case Explorer Application
Authentication Service
Data ingestion / Access Service
Data Lake
DataSet Service
Repository Database
Tracer Service
Tracer BlockChain
op: Init creation dataset
User Authorisation Use Case
Alt
[Action Authorised: User has the Dataset Administrator Role]
Datalake Query Use Case
op: Create Dataset (List of Cases Filtered + Name + Date)
op: Generate Dataset (Ref. to DICOM Images + Clinical Data + Metadata)
op: Create Dataset & calculate Checksums (Dataset Id)
op: Collect and compute metadata (including quality and anonymity checks)
op: Register DICOM Images References to Dataset (Symbolic Links to Data Lake Storage)
output: References registered successfully
op: Register Clinical Data to Dataset (Copy Data)
output: clinical data copy successfully
op: Register Dataset Creation Operation (User ID, Dataset Id + List of Cases Filtered + Name + Date + Signature)
op: Insert Operation User ID …
output: Insertion OK
output: Registration Successful
output: Dataset Id
msg: Dataset Creation Successfully

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

5.3.2 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 *Disable Dataset Use Case*) or the deployment of *Processing Applications* (see *Processing Application Deployment Use Case*) when a preliminary selection of a given *Dataset* kept in the *Repository Database* is required.

As Figure 23 shows, this high-level interaction requires three steps. The first one is to authorize the *User.* Only users with an authenticated *Dataset Administrator Role, Data Scientist Role, Application Developer Role or External Researcher Role* will be authorized 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*.

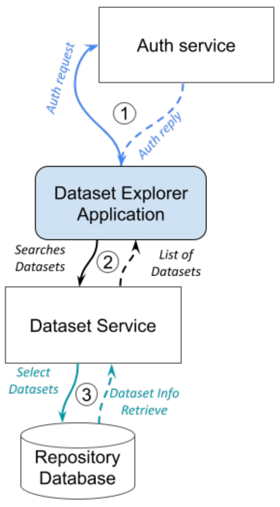


Figure 23. *Dataset Query High Level Interactions.*

Figure 24 shows the UML sequence diagram corresponding to the interaction flow among *Users* and *Components* involved to query *Datasets.*

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant

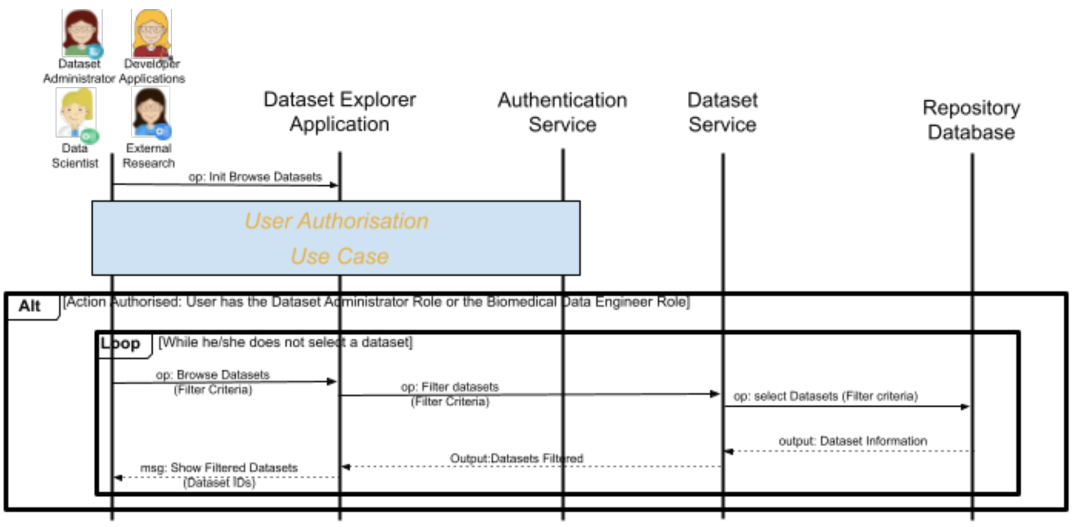

Dataset Administrator
Developer Applications
Data Scientist
External Research
Dataset Explorer Application
Authentication Service
Dataset Service
Repository Database
op: Init Browse Datasets
User Authorisation Use Case
Alt
[Action Authorised: User has the Dataset Administrator Role or the Biomedical Data Engineer Role]
Loop
[While he/she does not select a dataset]
op: Browse Datasets (Filter Criteria)
op: Filter datasets (Filter Criteria)
op: select Datasets (Filter criteria)
output: Dataset Information
Output:Datasets Filtered
msg: Show Filtered Datasets (Dataset IDs)

Figure 24. *UML Sequence Diagram for Dataset Query.*

5.3.3 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.

As Figure 25 shows, the first two steps correspond to the *Dataset Query Use Case* for selecting a given *Dataset*. Only users with *Dataset Administrator Role* authenticated will be authorized 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*.

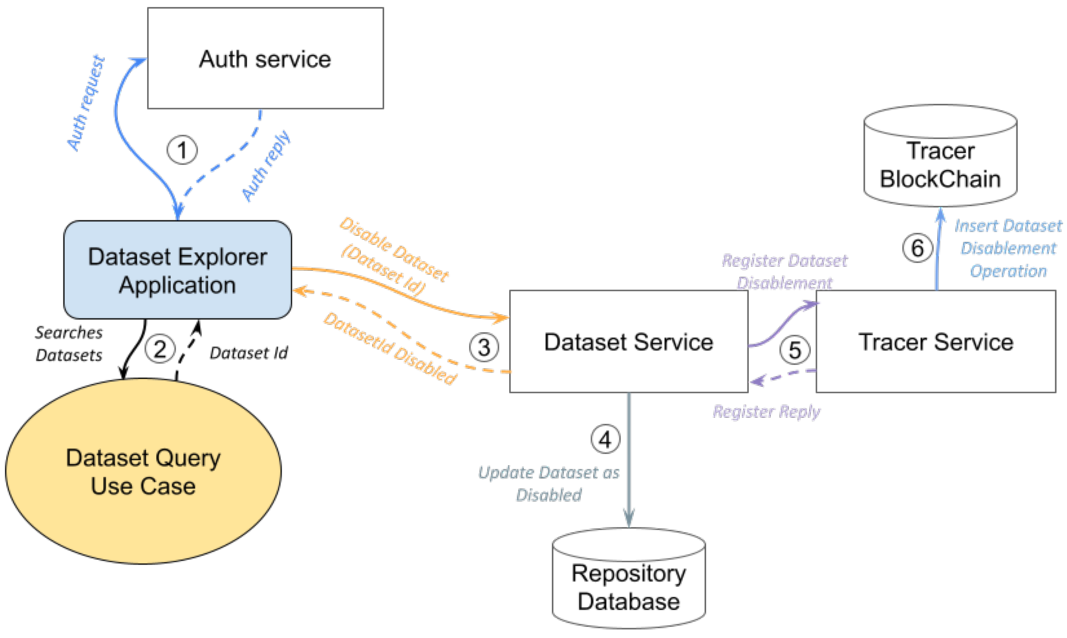


Figure 25. *Dataset Disablement High Level Interactions.*

Figure 26 shows the UML sequence diagram corresponding to the interaction flow among *Users* and *Components* involved to disable a given *Dataset.*

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant

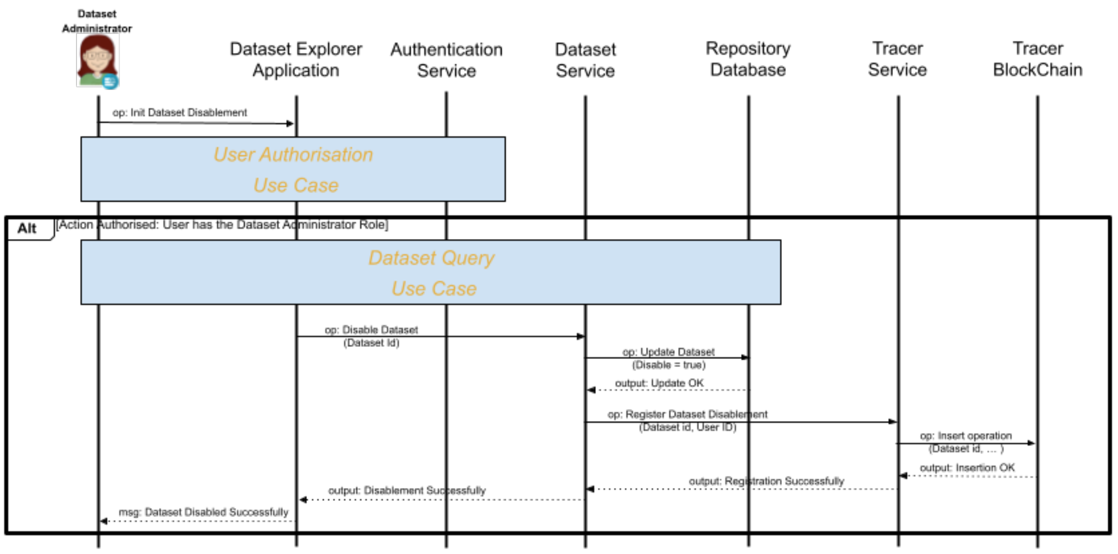

Dataset Administrator
Dataset Explorer Application
Authentication Service
Dataset Service
Repository Database
Tracer Service
Tracer BlockChain
op: Init Dataset Disablement
User Authorisation Use Case
Alt
[Action Authorised: User has the Dataset Administrator Role]
Dataset Query Use Case
op: Disable Dataset (Dataset Id)
op: Update Dataset (Disable = true)
output: Update OK
op: Register Dataset Disablement (Dataset id, User ID)
op: Insert operation (Dataset id, ...)
output: Insertion OK
output: Registration Successfully
output: Disablement Successfully
msg: Dataset Disabled Successfully

Figure 26. *UML Sequence Diagram for Dataset Disablement.*

5.4 Standalone Application Management

For *Standalone Application Management*, *6 main use cases* (Figure 27) have been identified from the *User Stories*. These *Use Cases* are described in next subsections.

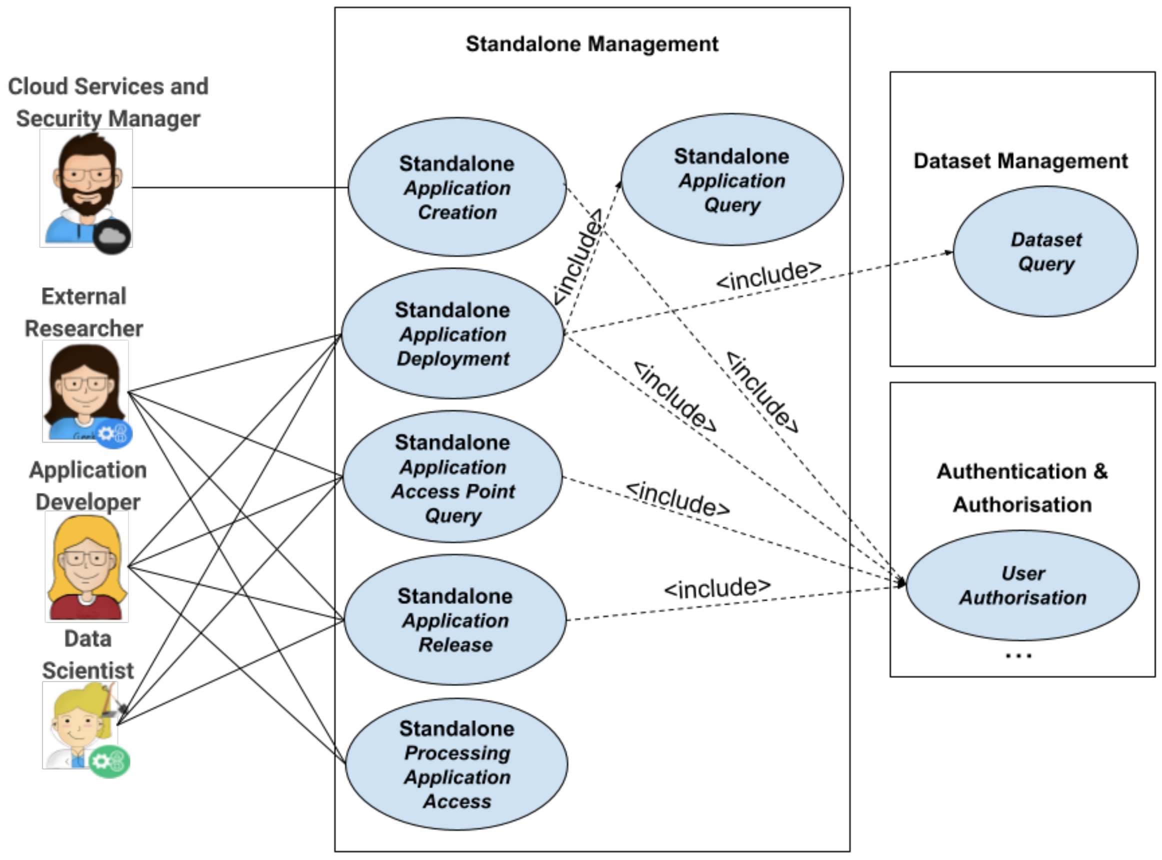

Standalone Management
Cloud Services and Security Manager
External Researcher
Application Developer
Data Scientist
Standalone Application Creation
Standalone Application Query
Standalone Application Deployment
Standalone Application Access Point Query
Standalone Application Release
Standalone Processing Application Access
<include>
<include>
<include>
<include>
<include>
<include>
Dataset Management
Dataset Query
Authentication & Authorisation
User Authorisation
...

Figure 27.*UML use cases identified for Standalone Applications Management.*

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant

5.4.1 Standalone Application Creation

This *Use Case* defines the required interactions to create and register a *Standalone Application* in the *CHAIMELEON Repository*. A *Standalone Application* provides *Users* the means to carry out statistical analysis, correlations, annotations, harmonization or the use of advanced AI frameworks to develop AI tools depending on the target of the application. Before creating and registering the *Standalone 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 *Standalone Application*.

As Figure 28 shows, only users with *Application Development* or *the Data Scientist Roles* will be authorized by the *Application Dashboard*. Only the *Cloud Service and Security Management* role will be authorized to *add Standalone Applications to* the *Application Dashboard*, once checking that the components used by the *Standalone Application* are secure. Then s(he) will register the *Standalone Application* and store the application container image, required configuration and deployment parameters in the *Application Registry*.

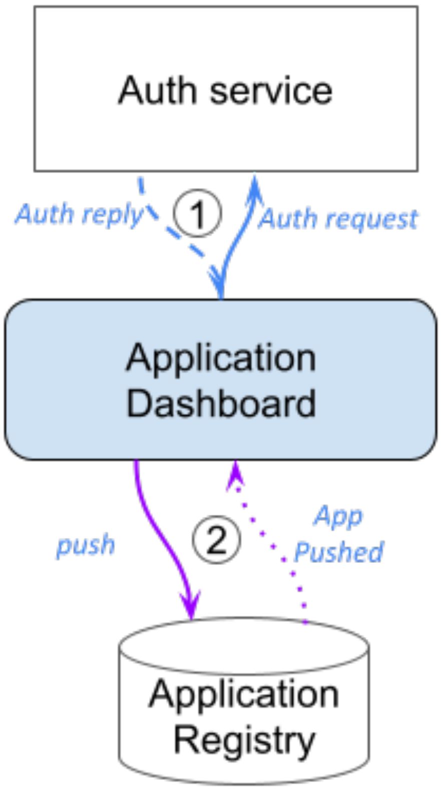


Figure 28. *Standalone Application creation High Level Interactions.*

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

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant

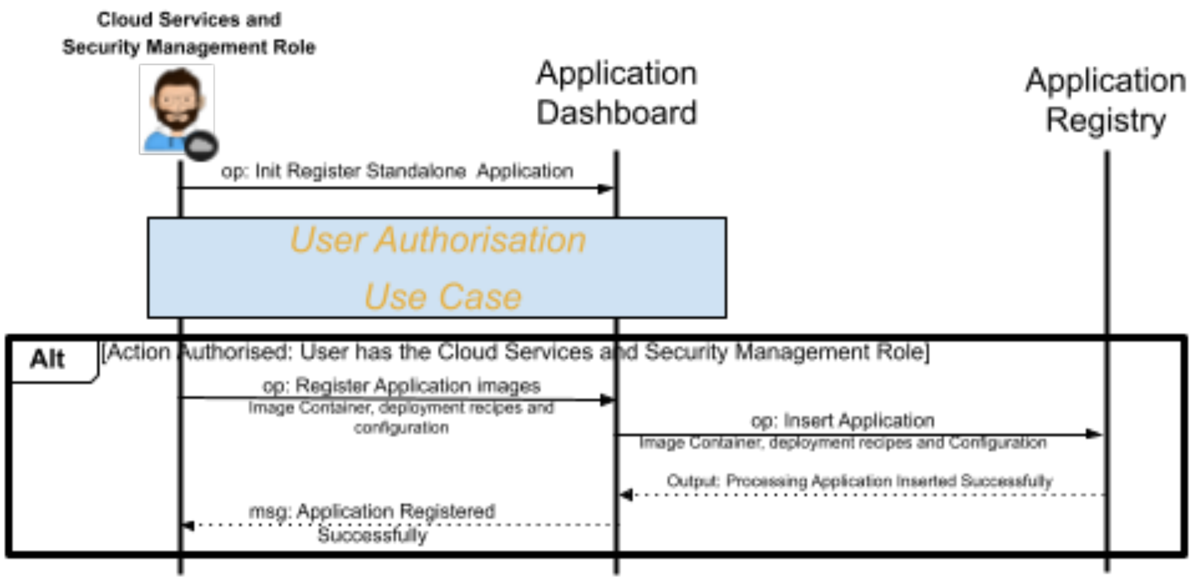

Cloud Services and Security Management Role
Application Dashboard
Application Registry
op: Init Register Standalone Application
User Authorisation Use Case
Alt
[Action Authorised: User has the Cloud Services and Security Management Role]
op: Register Application images
Image Container, deployment recipes and configuration
op: Insert Application
Image Container, deployment recipes and Configuration
Output: Processing Application Inserted Successfully
msg: Application Registered Successfully

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

5.4.2 Standalone Application Query

This *Use Case* defines the required interactions to query *Standalone Applications* from the *Application Registry*. This use case happens prior to deploying a *Standalone Application* (see *Standalone Application Deployment Use Case*).

As Figure 30 shows, this high-level interaction requires three steps. The first one is to authorize the *User*. Only users with *Data Scientist Role, Application Developers Role and External Researcher Role* will be authorized by the *Application Dashboard* to query *Standalone Applications* to the *Application Registry*. In the second step, *Application Dashboard* queries and filters applications from the *Application Registry*. Finally, it retrieves the results that are returned to the *Application Dashboard*.

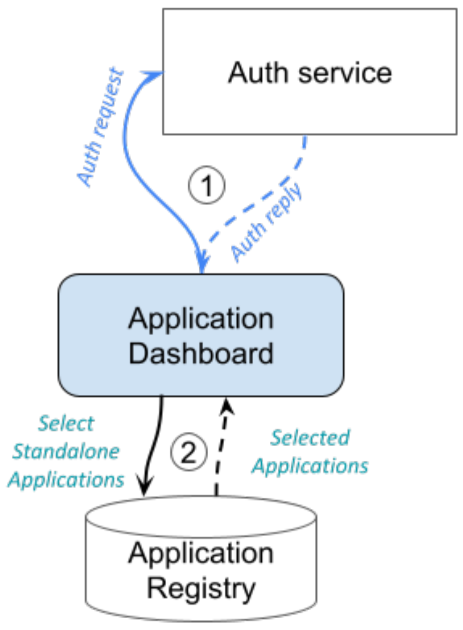


Figure 30. *Standalone Application Query High Level Interactions.*

Figure 31 shows the UML sequence diagram corresponding to the interaction flow among *Users* and *Components* involved to query *Standalone Applications.*

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant

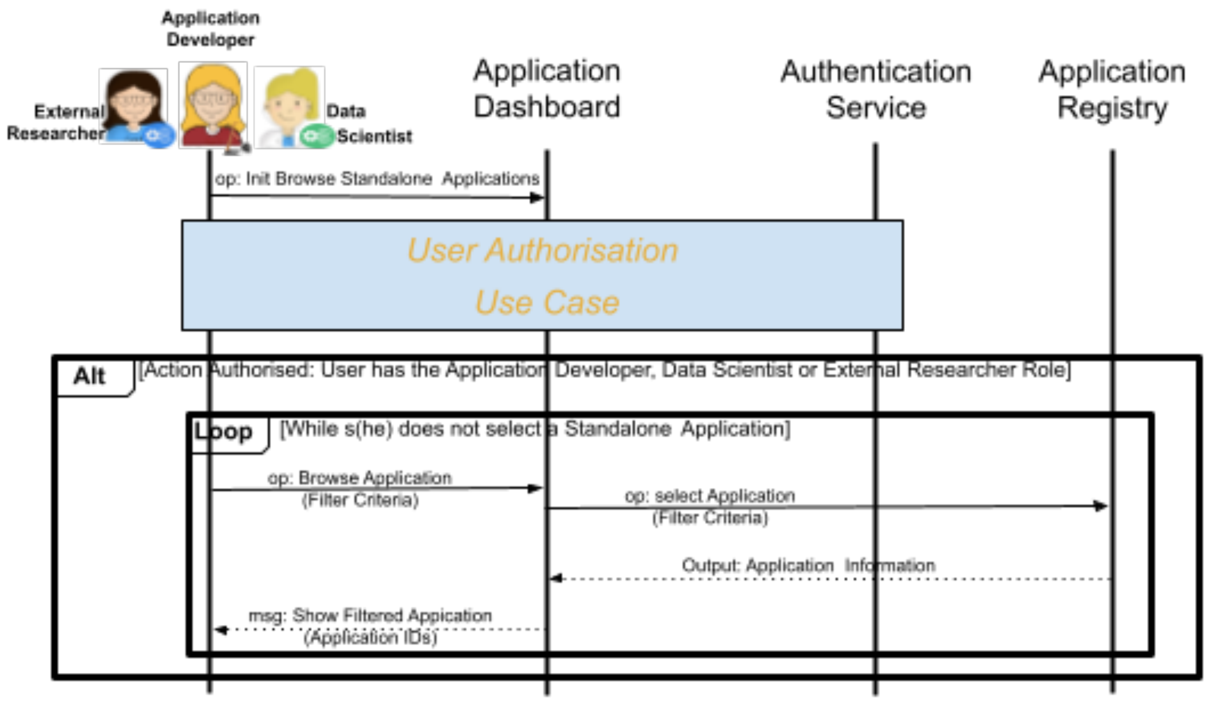

Application Developer
External Researcher
Data Scientist
Application Dashboard
Authentication Service
Application Registry
op: Init Browse Standalone Applications
User Authorisation Use Case
Alt
[Action Authorised: User has the Application Developer, Data Scientist or External Researcher Role]
Loop
[While s(he) does not select a Standalone Application]
op: Browse Application (Filter Criteria)
op: select Application (Filter Criteria)
Output: Application Information
msg: Show Filtered Appication (Application IDs)

Figure 31. *UML Sequence Diagram for Standalone Application Query.*

5.4.3 Standalone Application Deployment

This *Use Case* defines the interactions required to deploy a *Standalone Application* on the cloud resources provided by the *CHAIMELEON Repository*. A *Standalone Application* is an application from the Application Dashboard catalogue that runs independently and connects to a Dataset volume, providing *Users* with tools for carrying out statistical analysis, correlations, annotations, harmonization of data, frameworks to develop AI tools or other advanced processes on a previously selected *Dataset*.

As Figure 32 shows, 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 authorized 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 *Processing 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*.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant

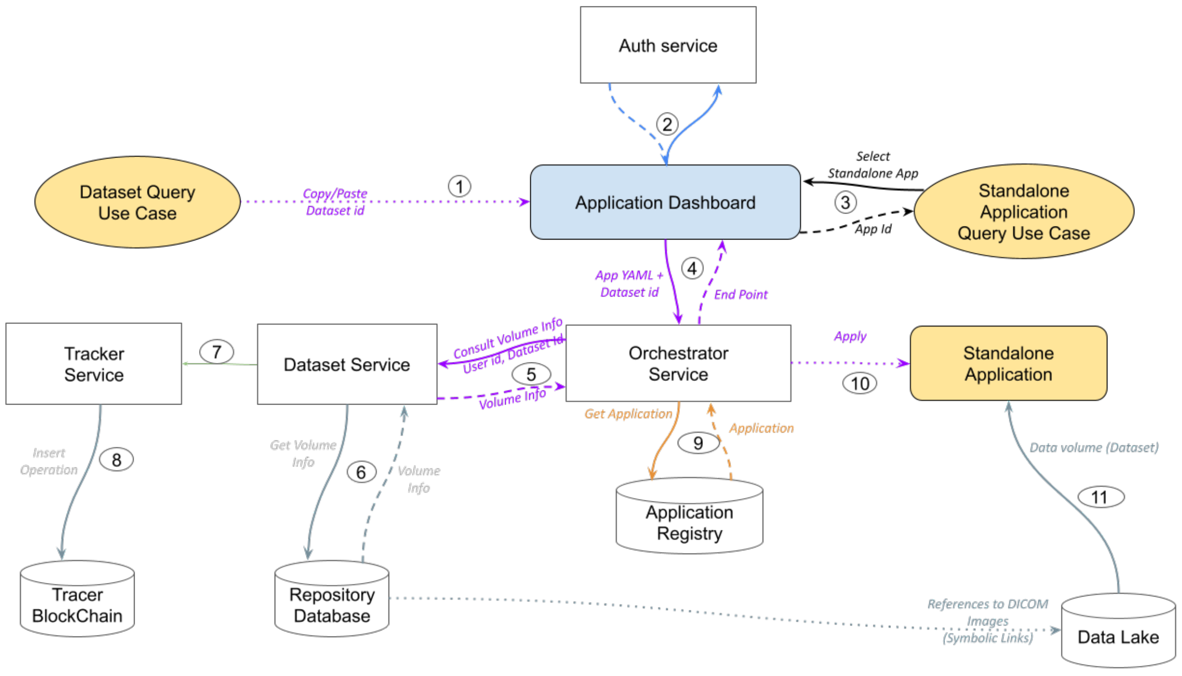


Figure 32. *Standalone Application deployment High Level Interactions.*

Figure 33 shows the UML sequence diagram corresponding to the interaction flow among *Users* and *Components* involved in the deployment of a *Standalone Application.*

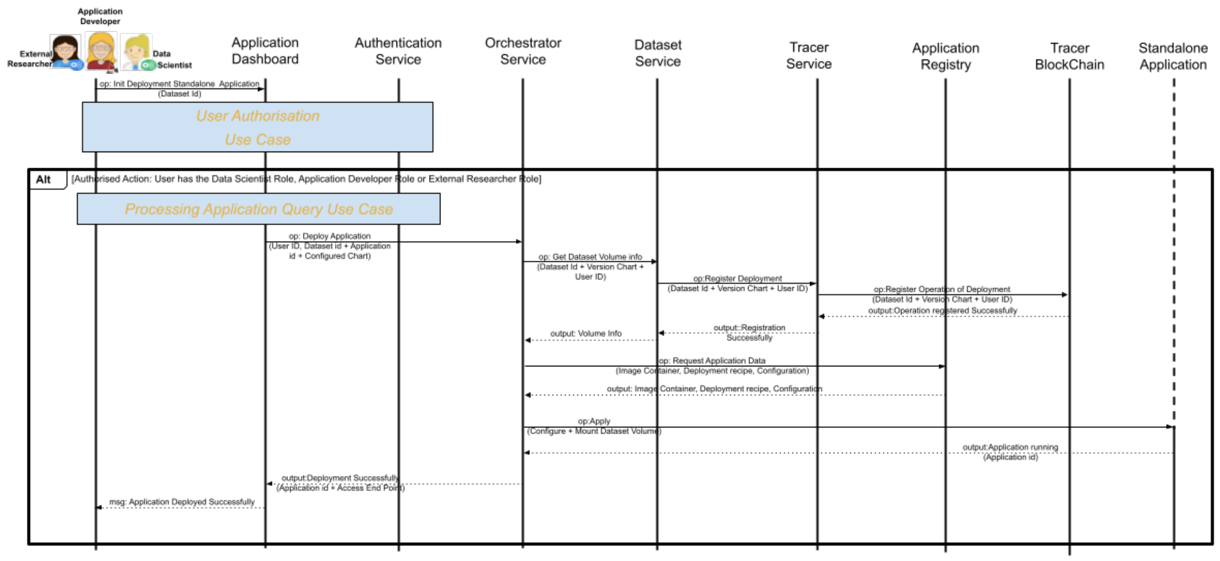

Application Developer
External Researcher
Data Scientist
Application Dashboard
Authentication Service
Orchestrator Service
Dataset Service
Tracer Service
Application Registry
Tracer BlockChain
Standalone Application
op: Init Deployment Standalone Application (Dataset Id)
User Authorisation Use Case
Alt
[Authorised Action: User has the Data Scientist Role, Application Developer Role or External Researcher Role]
Processing Application Query Use Case
op: Deploy Application (User ID, Dataset id + Application id + Configured Chart)
op: Get Dataset Volume info (Dataset Id + Version Chart + User ID)
op:Register Deployment (Dataset Id + Version Chart + User ID)
op:Register Operation of Deployment (Dataset Id + Version Chart + User ID)
output:Operation registered Successfully
output::Registration Successfully
output: Volume Info
op: Request Application Data (Image Container, Deployment recipe, Configuration)
output: Image Container, Deployment recipe, Configuration
op:Apply (Configure + Mount Dataset Volume)
output:Application running (Application id)
output:Deployment Successfully (Application id + Access End Point)
msg: Application Deployed Successfully

Figure 33. *UML Sequence Diagram for deploying a Standalone Application.*

5.4.4 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*).

As Figure 34 shows, this high-level interaction requires two steps. The first one is to authorize the *User.* Only authenticated users with the *Data Scientist Role*, *Application Developer Role*

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant

and *External Researcher Role* will be authorized 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*.

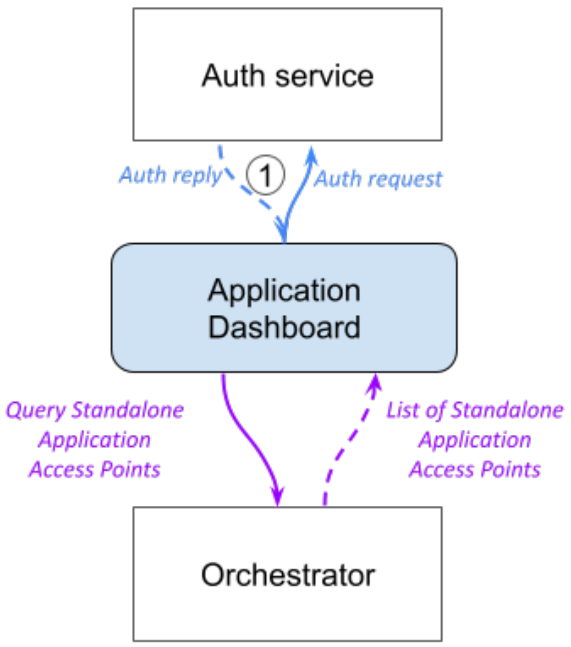

Auth service
Auth reply
1
Auth request
Application Dashboard
Query Standalone Application Access Points
List of Standalone Application Access Points
Orchestrator

Figure 34. *High Level Interactions for querying the Access Point of a Standalone Application.*

Figure 35 shows the UML sequence diagram corresponding to the interaction flow among *Users* and *Components* involved in the listing of *Standalone Application* Access Points.

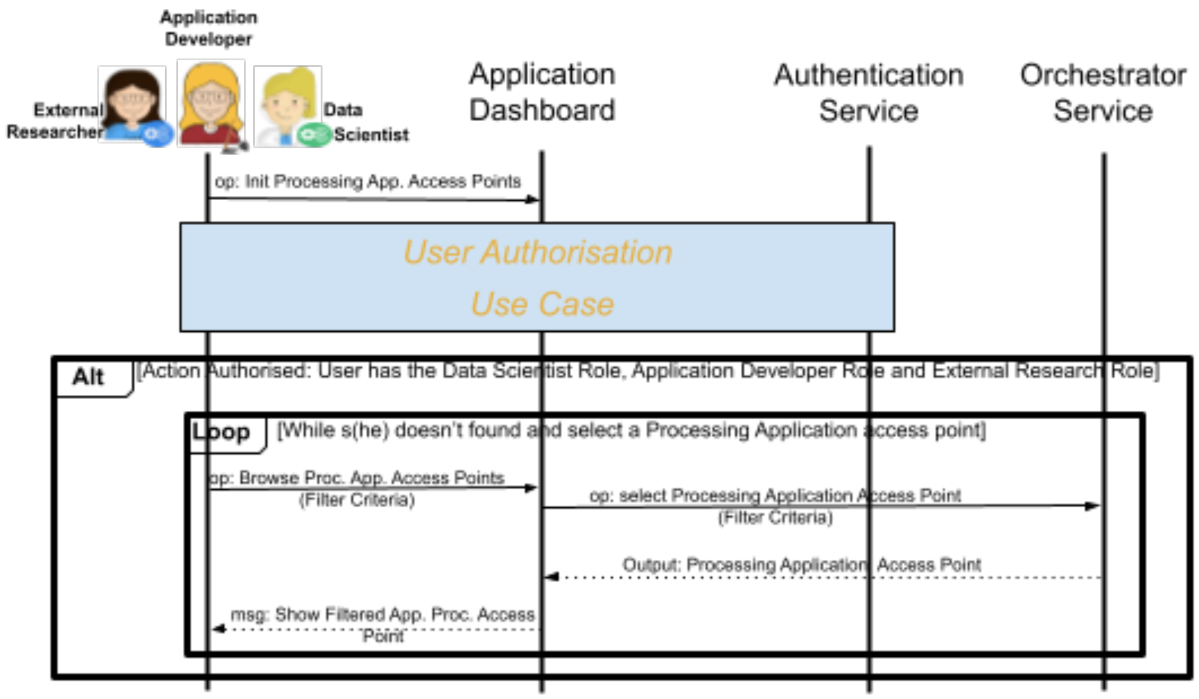

Application Developer
External Researcher
Data Scientist
Application Dashboard
Authentication Service
Orchestrator Service
op: Init Processing App. Access Points
User Authorisation Use Case
Alt
[Action Authorised: User has the Data Scientist Role, Application Developer Role and External Research Role]
Loop
[While s(he) doesn't found and select a Processing Application access point]
op: Browse Proc. App. Access Points (Filter Criteria)
op: select Processing Application Access Point (Filter Criteria)
Output: Processing Application Access Point
msg: Show Filtered App. Proc. Access Point

Figure 35. *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

5.4.5 Standalone Application Release

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

As Figure 36 shows, authenticated *Users* with the Data Scientist Role, Application Developer Role and External Research Role will be authorized by the *Application Dashboard* in this scenario to release a *Standalone Application.* Applications deployed by a user can only be removed by the same user. The *User* selects the application to delete (see *Standalone Application Query Use Case*) and removes it through the *Application Dashboard*. The *Application Dashboard* carries out the release of the resources through the *Orchestrator Service*. The back-end resources will be automatically provisioned and released on demand reacting to the workload managed by the orchestrator. Technologies such as Elastic Kubernetes will be used for that.

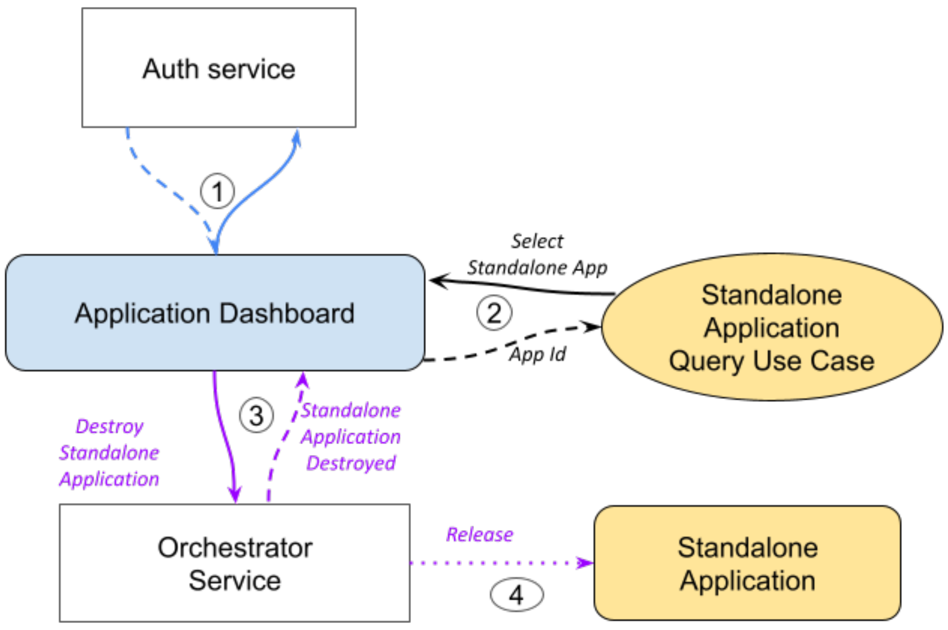


Figure 36. *Standalone Application Release High Level Interactions.*

Figure 37 shows the UML sequence diagram corresponding to the interaction flow among *Users* and *Components* involved to remove a *Standalone Application.*

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant

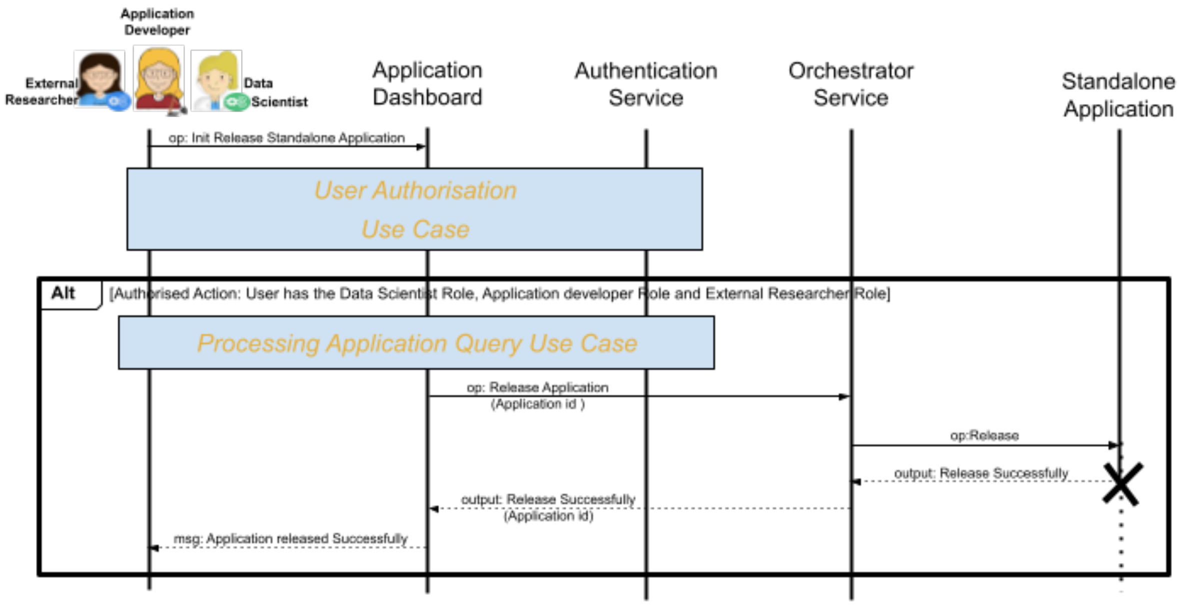

Application Developer
External Researcher
Data Scientist
Application Dashboard
Authentication Service
Orchestrator Service
Standalone Application
op: Init Release Standalone Application
User Authorisation Use Case
Alt
[Authorised Action: User has the Data Scientist Role, Application developer Role and External Researcher Role]
Processing Application Query Use Case
op: Release Application (Application id)
op:Release
output: Release Successfully
output: Release Successfully (Application id)
msg: Application released Successfully

Figure 37. *UML Sequence Diagram for releasing a Standalone Application.*

5.4.6 Standalone Application Access

This *Use Case* defines the interactions required to Access a *Standalone Application* which 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 38 shows the UML sequence diagram corresponding to the interaction flow among *Users* and *Components* involved to access a *Standalone Application.*

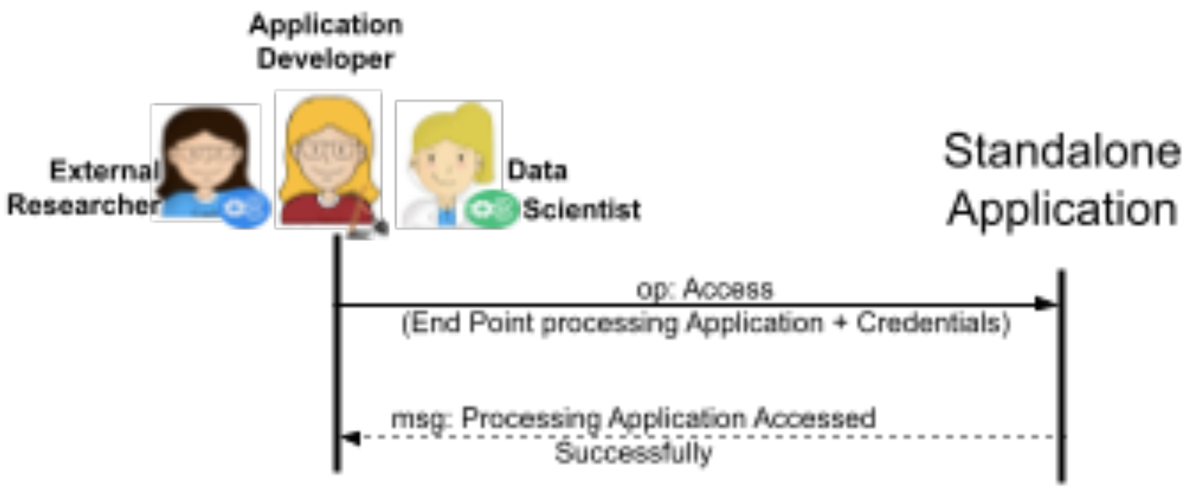

Application Developer
External Researcher
Data Scientist
Standalone Application
op: Access (End Point processing Application + Credentials)
msg: Processing Application Accessed Successfully

Figure 38. *UML Sequence Diagram for accessing a Standalone Application.*

5.5 Case Explorer (Marketplace) Management

Regarding Marketplace *Management*, *3 main use cases* (Figure 39) have been identified from the *User Stories*. The *Use Cases* are described in next subsections.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant

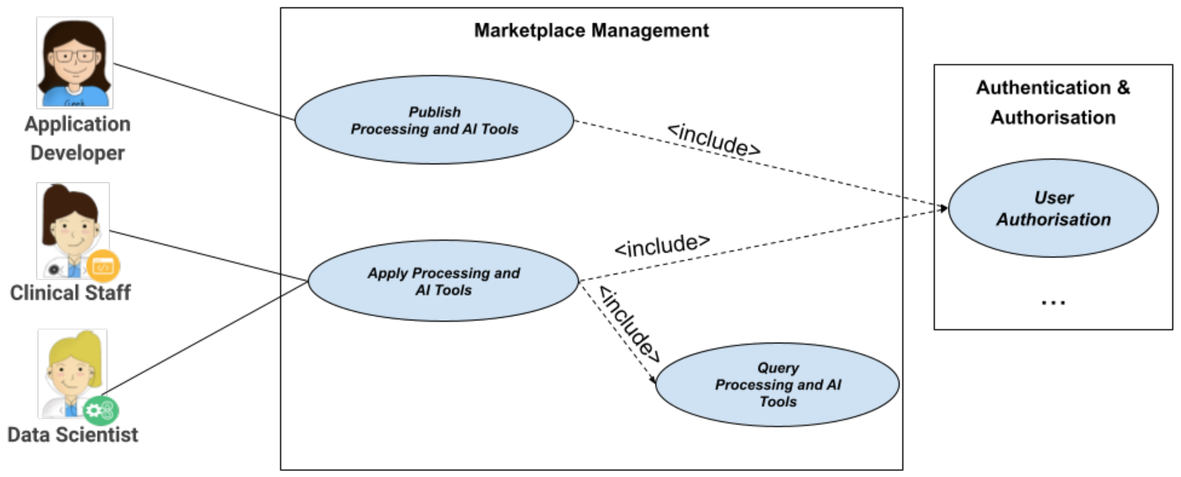

Application Developer
Clinical Staff
Data Scientist
Marketplace Management
Publish Processing and AI Tools
Apply Processing and AI Tools
Query Processing and AI Tools
<include>
<include>
<include>
Authentication & Authorisation
User Authorisation
...

Figure 39. *UML use cases identified for Marketplace Management.*

5.5.1 Publish Processing and AI Tools

This *Use Case* defines the required interactions to develop and publish *Processing and AI tools* in the *CHAIMELEON Repository*. Such tools will provide *Users* with the means to extract knowledge to assist on research, diagnosis, prognosis and treatment follow-on. In the context of the project, these tools could be processing tools or trained AI models embedded as applications by the *Application Developer Role*.

Publication of tools differs from the publication of standalone applications. Tools to be run in the *Case Explorer* will be integrated through this application by authorized users, providing some information about the execution arguments. New standalone applications will require the *Cloud Services and Security Manager* to create the specification and publish it in the *Application Dashboard* registry. This latter use case is not included in this description.

As Figure 40 shows, only users with *Application Developer Role* will be authorized by the *Case explorer* (Marketplace) to *add tools to* the *Marketplace.* Then, the *Application Developer* will register and publish the tool (container image, required configuration and deployment parameters) in the Case Explorer, which will store the configuration information in its database and the binaries in the *Application Registry*. Before a tool becomes available, the *Cloud Services and Security Manager* will validate the parameters indicated.

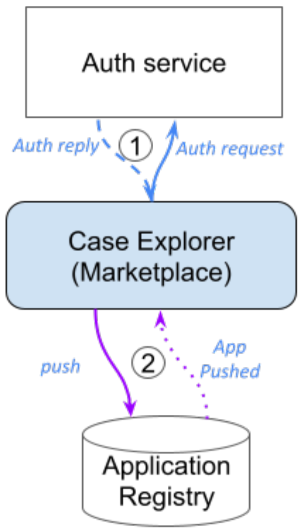


Figure 40. *High-Level Interactions for the publication of a Processing or AI tool.*

Figure 41 shows the UML sequence diagram corresponding to the interaction flow among Users and Components involved in the publishing of a Processing or AI tools.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant

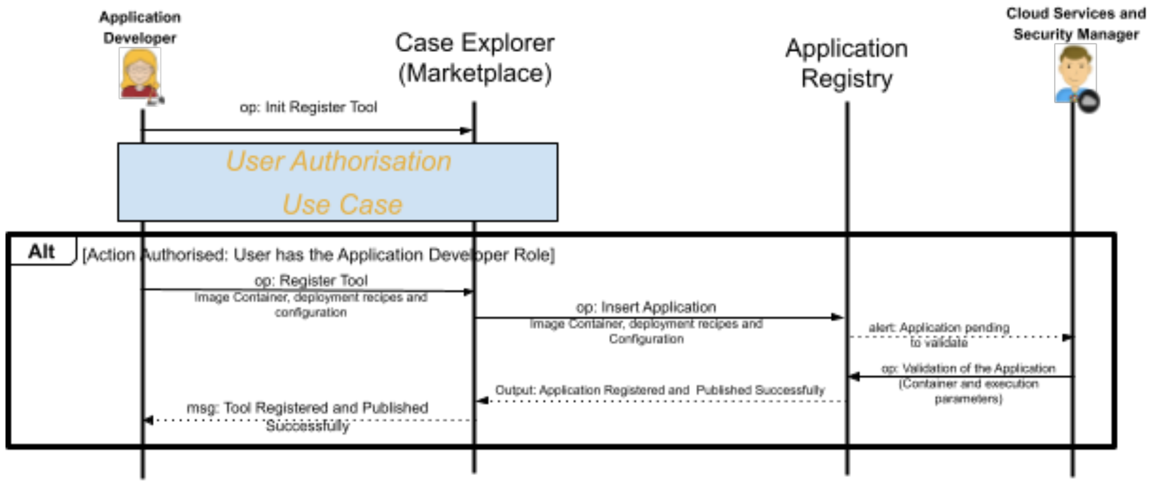

Application Developer
Case Explorer (Marketplace)
Application Registry
Cloud Services and Security Manager
op: Init Register Tool
User Authorisation Use Case
Alt
[Action Authorised: User has the Application Developer Role]
op: Register Tool
Image Container, deployment recipes and configuration
op: Insert Application
Image Container, deployment recipes and Configuration
alert: Application pending to validate
op: Validation of the Application (Container and execution parameters)
Output: Application Registered and Published Successfully
msg: Tool Registered and Published Successfully

Figure 41. *UML Sequence Diagram for Publishing Processing and AI tools.*

5.5.2 Processing and 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*). As Figure 42 shows, this high-level interaction requires two steps. The first one is to authorize the *User.* Only users with the *Data Scientist Role or Clinical Staff Role and External Researcher Role* will be authorized by the *Case Explorer (Marketplace)* to query the published tools. In the second step, the *Case Explorer (Marketplace)* queries and filters the tools.

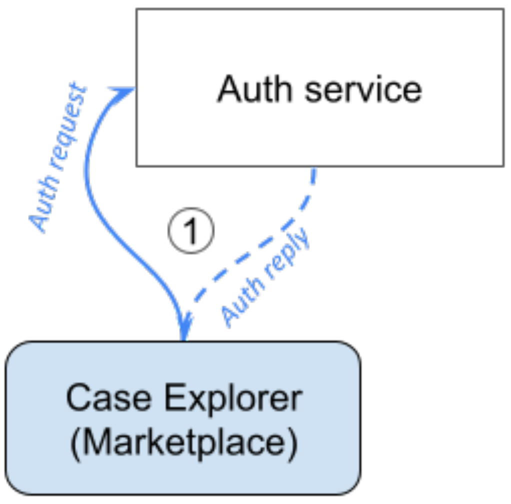


Figure 42. *Processing and AI Tool Query High Level Interactions.*

Figure 43 shows the UML sequence diagram corresponding to the interaction flow among *Users* and *Components* involved to list *Processing and AI Tools.*

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant

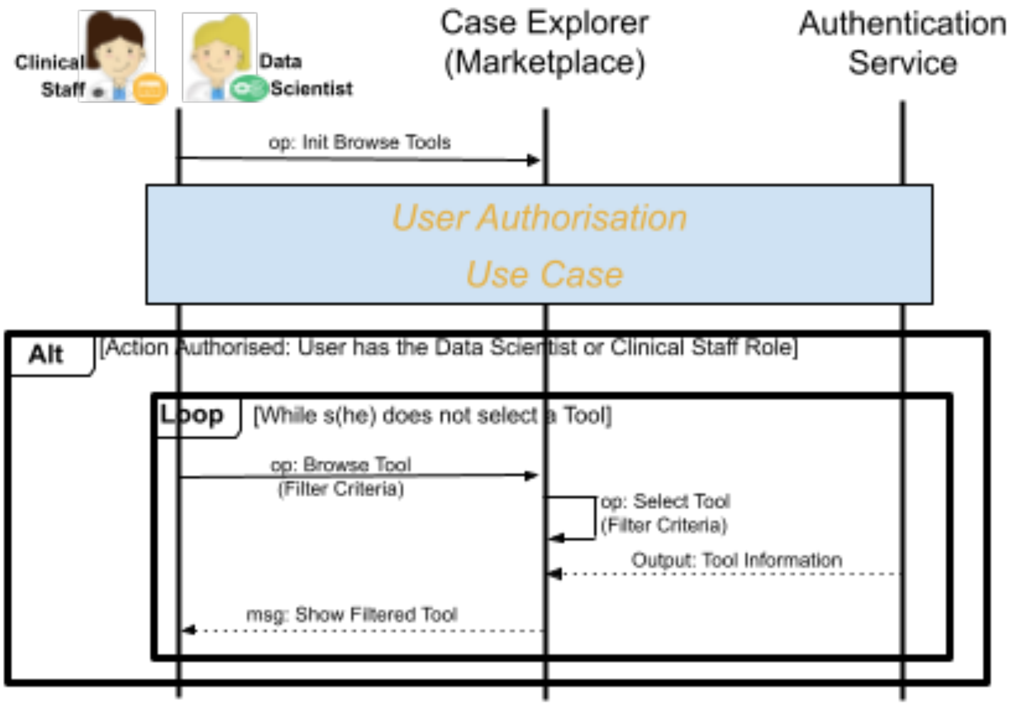

Clinical Staff
Data Scientist
Case Explorer (Marketplace)
Authentication Service
op: Init Browse Tools
User Authorisation Use Case
Alt
[Action Authorised: User has the Data Scientist or Clinical Staff Role]
Loop
[While s(he) does not select a Tool]
op: Browse Tool (Filter Criteria)
op: Select Tool (Filter Criteria)
Output: Tool Information
msg: Show Filtered Tool

Figure 43. *High-Level Interactions for the processing or AI Tool Query.*

5.5.3 Apply Processing or AI Tools

This *Use Case* defines the required interactions to use the *Processing and AI tools* provided by the marketplace of the *CHAIMELEON Repository*.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant

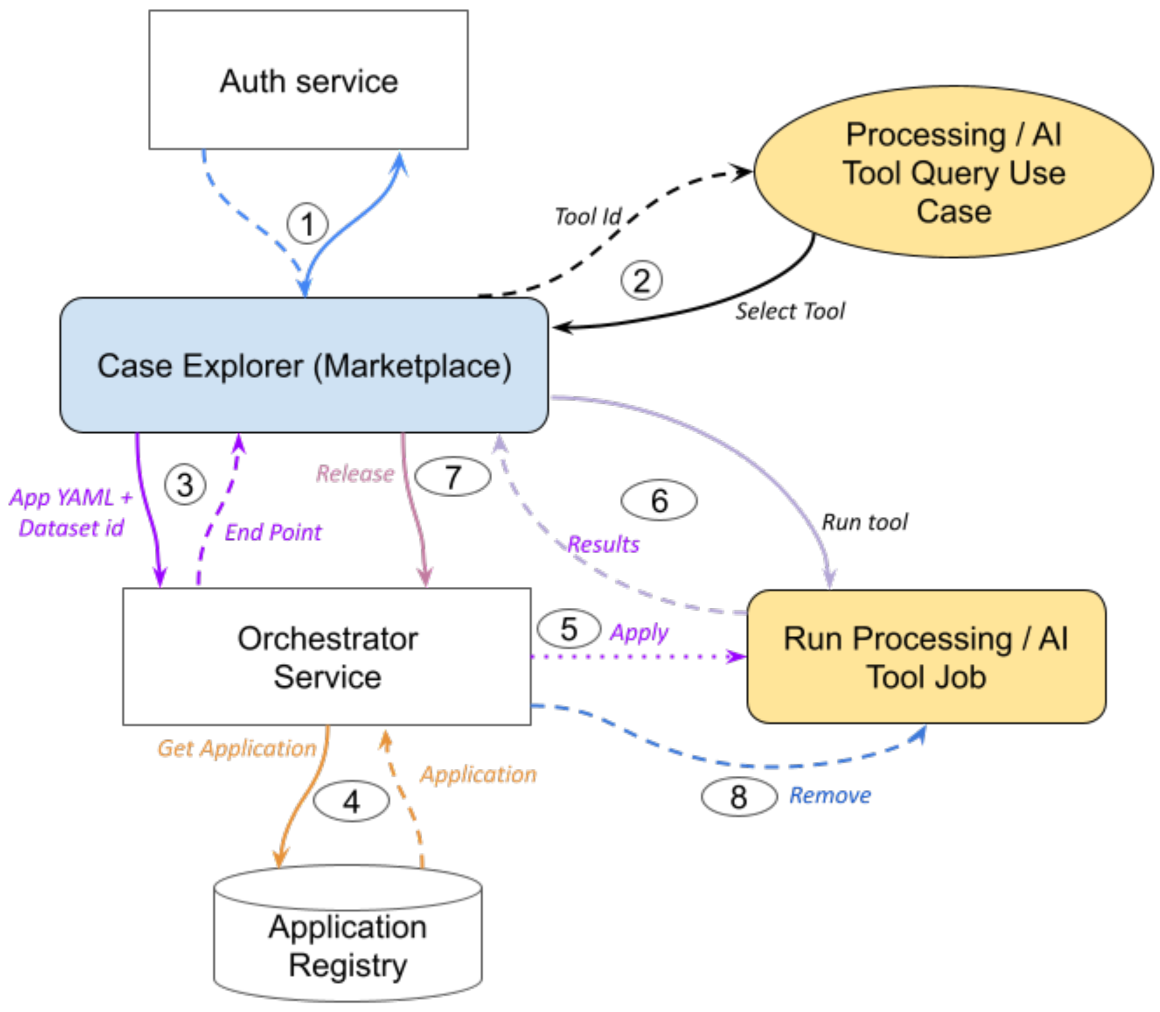


Figure 44. *Processing and AI Tool Apply High Level Interactions.*

As Figure 44 shows, this high-level interaction requires seven steps. The first one is to authorize the *User.* Only users with *Data Scientist Role or Clinical Staff Role* will be authorized by the *Case Explorer (Marketplace)* to use the tools previously published (*Publish Processing and AI Tools use case*). In the second step, the *User* through *Case Explorer (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). Then (third step), the *User* selects its own or an existing case to analyse and after that the *Case Explorer* runs the process through the *Orchestrator Service.* This may require fetching the application image and the additional files required from the *Application Registry* (fourth step) and after that running the process (fifth step). Finally, the *Case Explorer* collects the results from *Processing and AI Tool Application* (sixth step) and shows them to the *User*. When the process finishes and results are displayed by the Case Explorer, the *Orchestrator Service* removes the *Processing / AI Tool Application.*

Figure 45 shows the UML sequence diagram corresponding to the interaction flow among *Users* and *Components* involved to run *Processing / AI Tools.*

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant

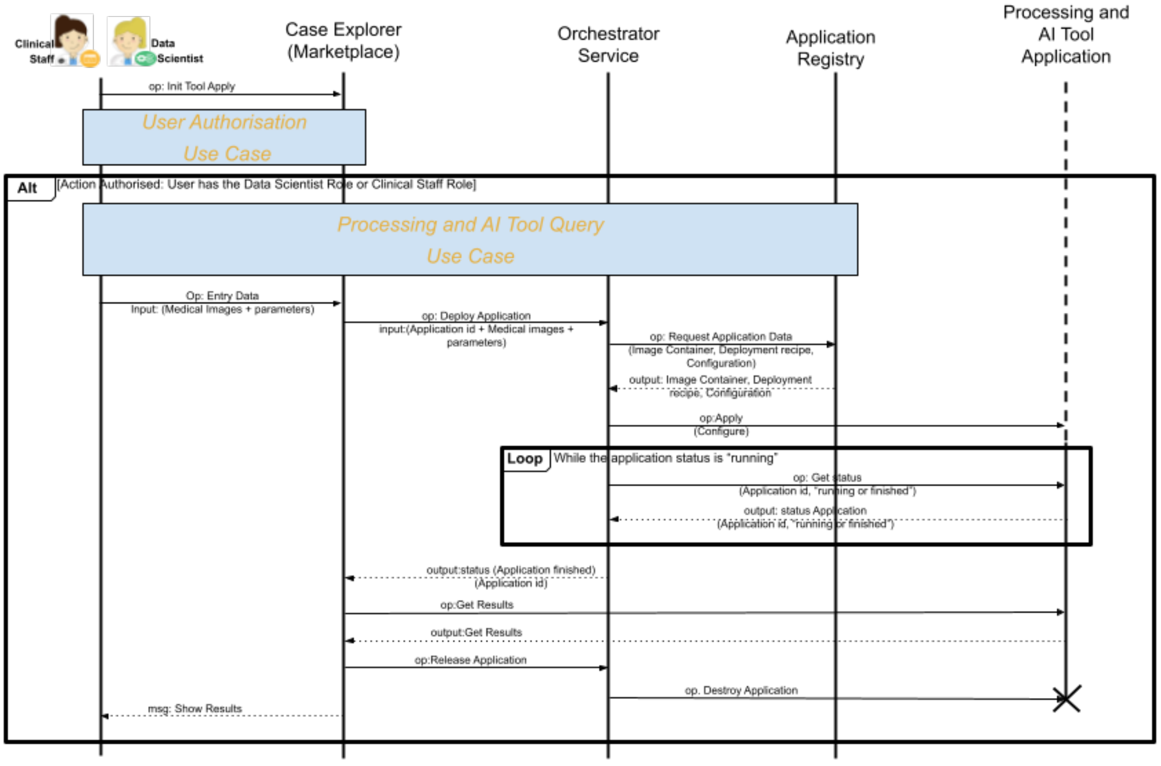

Clinical Staff
Data Scientist
Case Explorer (Marketplace)
Orchestrator Service
Application Registry
Processing and AI Tool Application
op: Init Tool Apply
User Authorisation Use Case
Alt
[Action Authorised: User has the Data Scientist Role or Clinical Staff Role]
Processing and AI Tool Query Use Case
Op: Entry Data Input: (Medical Images + parameters)
op: Deploy Application input:(Application id + Medical images + parameters)
op: Request Application Data (Image Container, Deployment recipe, Configuration)
output: Image Container, Deployment recipe, Configuration
op:Apply (Configure)
Loop
While the application status is "running"
op: Get status (Application id, "running or finished")
output: status Application (Application id, "running or finished")
output:status (Application finished) (Application id)
op:Get Results
output:Get Results
op:Release Application
op. Destroy Application
msg: Show Results

Figure 45. *High-Level Interactions for running processing / AI Tools.*

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant

6. Requirements

After the analysis of the use cases we proceed to identify, describe and prioritise the requirements. Starting from the information of Deliverable 3.2, which outlined non-functional requirements, this deliverable compiles and formally describes those requirements and standards for the design of the *CHAIMELEON Repository*. Additionally, it includes those requirements coming from the analysis of the 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, 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 requirement number 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.

6.1 Transversal Requirements

Next requirements are applied or involved to a set of *CHAIMELEON Components,* these are Storages, *Services*, *Platform Applications* and *Processing Applications* identified in the stories and appearing in the use case descriptions.

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.		

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.		

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant

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 directly access the Data Lake, 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

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 provided to Users through *Platform Applications* only, except for the data ingestion. 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 *authorized 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 authorization 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 authorization mechanisms for *Users*.		

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.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant

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:** Recommendable	**Technology:** Ceph, K8s
Description: Backend resources supporting the *Storages* **should** be able to scale up providing additional capacity.		

6.2 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).		

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.).		

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant

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.		

6.3 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 centers in the four areas involved in the project (breast, prostate, lung and colorectal).		

Code: REDLK.2	**Name:** anonymized and Curated Data	
Type: Non-functional	**Relevance:** Mandatory	**Technology:** --
Description: This *Storage* **must** store fully anonymized and curated data (clinical data and images). The repository **must not** include any personal information and Users will commit to prevent 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.		

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.		

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant

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.		

Code: REDLK.9	**Name:** Access Restricted to the *Data Ingestion/Access Service*	
Type: Non-Functional	**Relevance:** Mandatory	**Technology:** QUIBIM Precision

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant

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.		

6.4 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*.		

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 *storage* **must** be accessible by *CHAIMELEON Processing Application* through a PACS System compatible with the DICOM web protocol.		

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant

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.		

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant

6.5 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.		

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).		

6.6 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

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant

Type: Functional	**Relevance:** Mandatory	**Technology:** Git Repository
Description: This *Storage* **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.		

6.7 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:** Access Restricted to the *Tracer Service* only	
Type: Non-Functional	**Relevance:** Mandatory	**Technology:** --
Description:		

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant

Tracer Service **must** be the only *CHAIMELEON Component* allowed to access via API/REST interface the *Tracer Blockchain Storage*.

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

This project has received funding from the European Union's
Horizon 2020 research and innovation programme under grant

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 anonymized data identifiers, for access tracking purposes.		

6.8 Authentication Service

Code: REAUS.1	**Name:** Access Restricted to the *Platform Applications* and *Data Ingestion/Access Service* 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 authorization 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 authorize 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

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant

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.

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.		

6.9 Data Ingestion/Access Service

Code: REIAS.1	**Name:** Access Restricted to *Case Explorer Application* and External applications with delegated credentials	
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 *authorized Technical Data Manager*. Direct access from any other *CHAIMELEON Component* is not allowed.		

Code: REIAS.2	**Name:** Basic Functionalities to Manage the *Data Lake* data	
Type: Functional	**Relevance:** Mandatory	**Technology:** --

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant

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*.

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.		

6.10 Dataset Service

Code: REDSS.1	**Name:** Restricted access from *Data Ingestion/Access Service, Orchestrator Service, Case Explorer Application and Dataset Explorer Application*	
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

6.11 Tracer Service

Code: RETRS.1	**Name:** Restricted access from Dataset Service	
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*.		

6.12 Orchestrator Service

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

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*.		

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant

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

6.13 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.		

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 groups and capabilities requesting.		

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.		

6.14 Case Explorer Application

Code: RECEA.1	**Name:** User-Friendly Web Interface	
Type: Functional	**Relevance:** Mandatory	**Technology:** QUBIM Precision
Description:		

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant

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 *authorized Technical Data Manager Role.*		

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 *authorized 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 *authorized 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.		

This project has received funding from the European Union's
Horizon 2020 research and innovation programme under grant

6.15 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.		

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.*		

6.16 Dataset Explorer Application

Code: REDSE.1	**Name:** User Friendly Web Interface	
Type: Functional	**Relevance:** Mandatory	**Technology:** QUBIM Precision

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant

Description: This application **must** provide a User-Friendly web interface to manage *Datasets*.

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.*		

6.17 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.*		

Code: REAPP.3	Filter and Select *Standalone Applications*

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant

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

7. Specification

The specification section includes a proposal of technologies and existing tools for implementing the components of the *CHAIMELEON Repository* architecture, as well as some technical diagrams and descriptions. Interaction diagrams about components and services from all the layers in *CHAIMELEON Repository* will be developed and refined as the project evolves.

7.1 Technologies and Existing Tools

This section briefly describes the technologies proposed for the implementation of the components of the CHAIMELEON Architecture. Standards and Data models have been already presented in deliverable D3.2.

7.1.1 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 (IaaS) functionality, additional components provide orchestration, fault management and service management amongst other services to ensure high availability of user applications.

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

7.1.2 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 customized 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, thus making user applications Cloud agnostic. In addition, it features DevOps capabilities, based on Ansible 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 *Storages*, *Services*, *Platform Applications* and *Processing Applications*.

7.1.3 Elastic Compute Cluster in the Cloud (EC3)

The Elastic Compute Cluster in the Cloud (EC3)[3] 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 customized front-end node of a virtual cluster that features: i) an instance of an IM to provision for additional computing resources (working nodes); ii) 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.

[1] https://www.openstack.org
[2] https://www.grycap.upv.es/im/ GNU General Public License - version 3.0
[3] 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

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

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

7.1.4 TOSCA

Topology and Orchestration Specification for Cloud Applications (TOSCA)[4] is a standard created by OASIS. TOSCA (sometimes dubbed 'TOSCA Cloud') is known as one of the fastest-growing standards in OASIS and has numerous use cases and implementations that have already been announced.

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.*

7.1.5 EduGain

The eduGAIN[5] service provides an efficient, flexible way to interconnect participating federations and their affiliated users and services. Most European universities and research centers are affiliated to eduGAIN, and therefore, in the context of CHAIMELEON, Users can be easily and securely authenticated and authorized to manage access to services reusing their existing institutional authentication systems. Around the world, more than 55 federations joined or are in the process of joining eduGAIN, with more than 2,600 identity providers and 1,800 service providers.

7.1.6 Keycloak

We will use Keycloak[6] to implement the Authentication 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 that can be configured easily with Keycloak. This service will centralize the user management in the CHAIMELEON platform offering a single sign-on to the users and avoiding the applications needing to store user's 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

[4] https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=tosca
[5] https://edugain.org
[6] Apache License, Version 2.0.
https://www.keycloak.org/.
https://github.com/keycloak/keycloak

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant

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 'sub' field and can be used by the 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.

7.1.7 Kubernetes (K8s)

Kubernetes[7] (K8s) is an open-source orchestration system for Docker containers, although can be configured for running other OCI[8] 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[9] or Azure Container Service[10]) and cloud platforms. Kubernetes clusters are composed of two types of nodes (master and workers) and a set of core components (etcd, kube-dns, kube-apiserver, etc.)

Kubernetes provides different methods for authenticating users: X509 Client certificates, static tokens per users, bootstrap tokens, service account tokens, and OpenID Connect Tokens. Thus, Kubernetes can authenticate users using Keycloak. Regarding multi-tenancy, there are different approaches to isolate 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 authorization, 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

[7] https://kubernetes.io/, https://github.com/kubernetes/
[8] https://opencontainers.org/
[9] https://cloud.google.com/kubernetes-engine
[10] https://azure.microsoft.com/en-us/product-categories/containers/

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant

obtained from several repository images, such as DockerHub[11], Google Container Registry[12], Harbor[13], AWS EC2 Container Registry[14] and Azure Container Registry[15] (ACR).

Regarding the interaction with Kubernetes, there are several ways to do it. For 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 other hand, it is possible to interact with Kubernetes using browser graphical user interfaces (such us Kubernetes Dashboard[16] or Kubeapps[17]) or using desktop graphical user interfaces (such us Octant[18] or Lens[19]).

7.1.8 Kubeapps

Kubeapps is an open-source browser UI dashboard for deploying and managing applications in the Kubernetes cluster using Helm charts. Kubeapps provides a catalog of Helm charts to the users of multiple public chart repositories. It also allows using private chart repositories such as Harbor[20] or ChartMuseum[21]. 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).

7.1.9 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 organized 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.

7.1.10 Ceph

Ceph[22] 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 minimize 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 in

[11] https://hub.docker.com/
[12] https://cloud.google.com/container-registry
[13] https://goharbor.io/
[14] https://aws.amazon.com/ecr/?nc1=h_ls
[15] https://azure.microsoft.com/en-us/services/container-registry/
[16] https://github.com/kubernetes/dashboard
[17] https://kubeapps.com/
[18] https://octant.dev/
[19] https://k8slens.dev/
[20] https://goharbor.io/
[21] https://chartmuseum.com/
[22] http://ceph.com/

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant

namespaces. Ceph users must have access to pools to read and write data. Besides, Ceph users must have executed permissions to use administrative commands. Ceph uses the term “capabilities” to describe authorizing 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.

7.1.11 BlockChain

Fundamentally, a blockchain is a list (chain) of records (called blocks) linked by cryptographic hashes (Figure 45). 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 decentralized database, with inherent fault tolerance and security.

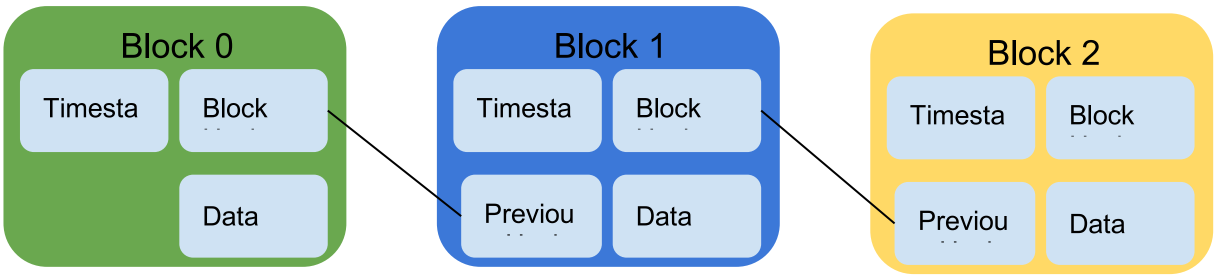


Figure 46. *The blockchain’s basic structure.*

We use blockchain technology with our Tracer service. It acts as a Create-Read (CR instead of CRUD) 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. The 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 47). 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.

This project has received funding from the European Union’s
Horizon 2020 research and innovation programme under grant

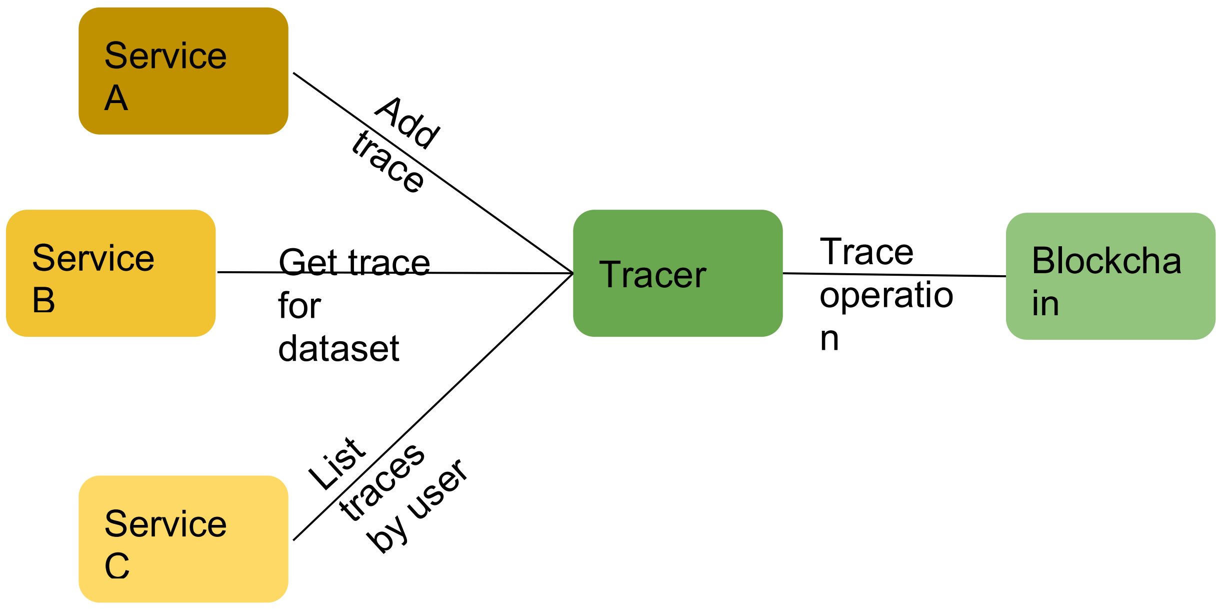


Figure 47. *Tracing operations are not querying the Blockchain directly.*

7.1.12 MongoDB

MongoDB[23] 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 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.

7.1.13 QUIBIM Precision

And to manage the *Data Lake*, the solution of Quibim Precision[24] will be used. It provides, as a front-end (*Case Explorer Application*), a friendly Web UI with the basic functionalities (insert, update and consult) but also more advanced tools like filters, data miner with charts or a complete DICOM viewer which, for example, enables the user for 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 format or other formats such as NIFTI.

An assistant is included for manual ingestion with automatic anonymization in metadata (*tags*) of the images and deletion of image regions. But it also has a REST API interface in the back-end (*Data Ingestion/Access Service*) which includes an operation to ingest data in bulk by External Applications.

Clinical data is stored in the MongoDB, and medical images are stored in the POSIX File System, provided by Ceph in this project.

A service of QUIBIM precision in the back-end has the capabilities of downloading medical images from a DICOM node (also for bulk ingestion).

[23] GNU AGPL v3.0
https://www.mongodb.com/
[24] https://quibim.com

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant

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*. In order to launch and manage those analysis modules it can currently use Nomad as the container orchestrator but also it will be able to use Kubernetes for the CHAIMELEON project.

7.2 Draft Architecture

This section presents a set of diagrams about the *CHAIMELEON Platform Architecture*. In these diagrams, components are exposed and the candidate technologies for their implementation are shown. All details about the interactions between Storages, services and applications have already been outlined in the use cases section.

7.2.1 Components of the CHAIMELEON Architecture

Figure 48 shows the main components that will be implemented in the architecture of the *CHAIMELEON Repository* without interactions. All of these components are connected among them through an internal private network and the interaction with the users through the applications.

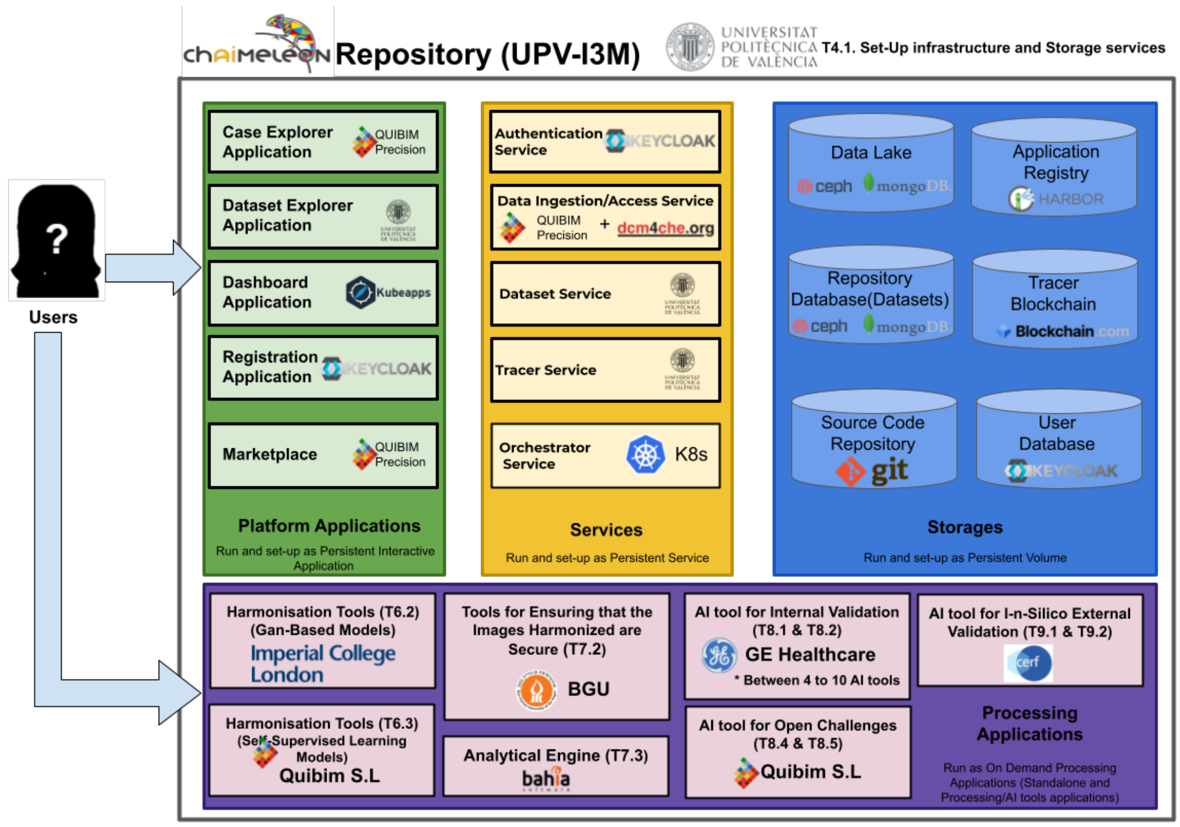

Repository (UPV-I3M)
UNIVERSITAT POLITÈCNICA DE VALÈNCIA
T4.1. Set-Up infrastructure and Storage services
Users
Case Explorer Application
QUIBIM Precision
Dataset Explorer Application
Dashboard Application
Kubeapps
Registration Application
KEYCLOAK
Marketplace
QUIBIM Precision
Platform Applications
Run and set-up as Persistent Interactive Application
Authentication Service
KEYCLOAK
Data Ingestion/Access Service
QUIBIM Precision + dcm4che.org
Dataset Service
Tracer Service
Orchestrator Service
K8s
Services
Run and set-up as Persistent Service
Data Lake
ceph mongoDB
Application Registry
HARBOR
Repository Database(Datasets)
ceph mongoDB
Tracer Blockchain
Blockchain.com
Source Code Repository
git
User Database
KEYCLOAK
Storages
Run and set-up as Persistent Volume
Harmonisation Tools (T6.2) (Gan-Based Models)
Imperial College London
Tools for Ensuring that the Images Harmonized are Secure (T7.2)
BGU
AI tool for Internal Validation (T8.1 & T8.2)
GE Healthcare
* Between 4 to 10 AI tools
AI tool for I-n-Silico External Validation (T9.1 & T9.2)
cerf
Harmonisation Tools (T6.3) (Self-Supervised Learning Models)
Quibim S.L
Analytical Engine (T7.3)
bahia
AI tool for Open Challenges (T8.4 & T8.5)
Quibim S.L
Processing Applications
Run as On Demand Processing Applications (Standalone and Processing/AI tools applications)

Figure 48. *Identified Entities of the Interim CHAIMELEON Architecture*

7.2.2 Data Lake Management

Figure 49 shows the main components that will be implemented in the architecture of the *CHAIMELEON Repository* for *Data Lake* management including interactions with *Users* and among components.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant

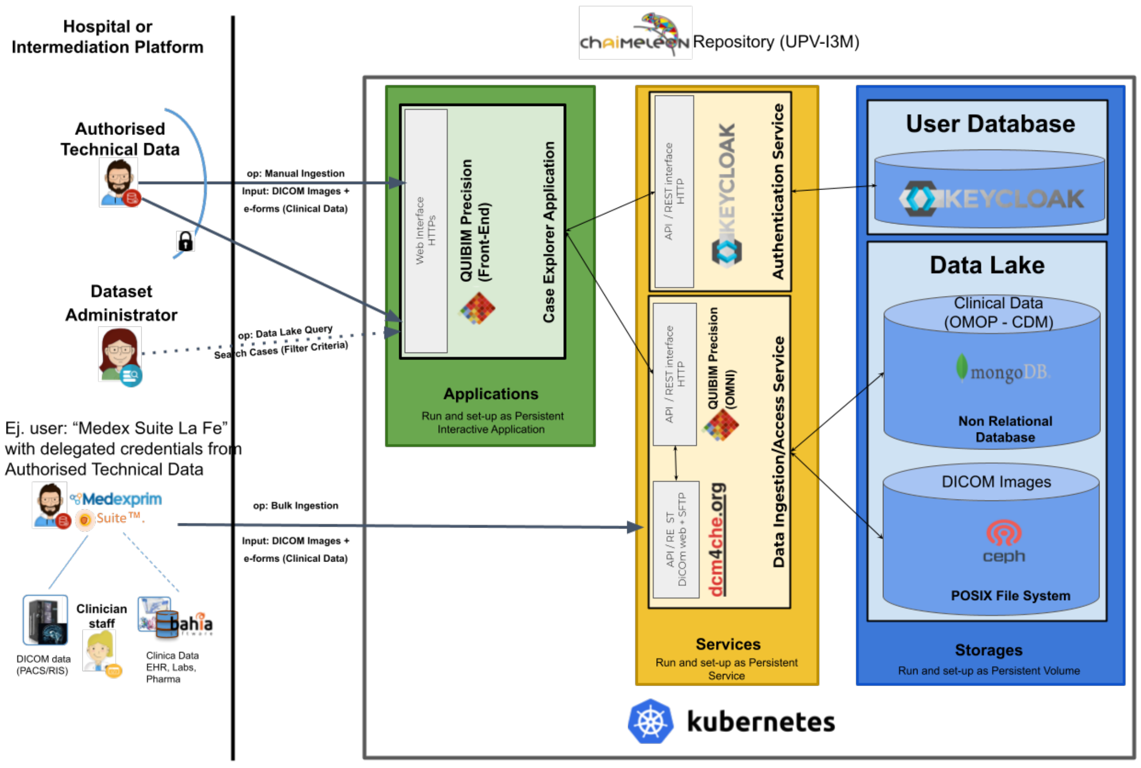

Hospital or Intermediation Platform
Repository (UPV-I3M)
Authorised Technical Data
Dataset Administrator
Ej. user: "Medex Suite La Fe" with delegated credentials from Authorised Technical Data
Medexprim Suite™
Clinician staff
DICOM data (PACS/RIS)
Clinica Data EHR, Labs, Pharma
bahia
op: Manual Ingestion
Input: DICOM Images + e-forms (Clinical Data)
op: Data Lake Query
Search Cases (Filter Criteria)
op: Bulk Ingestion
Input: DICOM Images + e-forms (Clinical Data)
Web Interface HTTPs
QUIBIM Precision (Front-End)
Case Explorer Application
Applications
Run and set-up as Persistent Interactive Application
API / REST interface HTTP
KEYCLOAK
Authentication Service
API / REST interface HTTP
QUIBIM Precision (OMNI)
API / RE ST DICOm web + SFTP
dcm4che.org
Data Ingestion/Access Service
Services
Run and set-up as Persistent Service
User Database
KEYCLOAK
Data Lake
Clinical Data (OMOP - CDM)
mongoDB
Non Relational Database
DICOM Images
ceph
POSIX File System
Storages
Run and set-up as Persistent Volume
kubernetes

Figure 49. *Diagram of the Interim CHAIMELEON Architecture for Data Lake Management.*

7.2.3 Dataset Management

Figure 50 shows the main components that will be implemented in the architecture of the *CHAIMELEON Repository* for Dataset management including interactions with *Users* and among components.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant

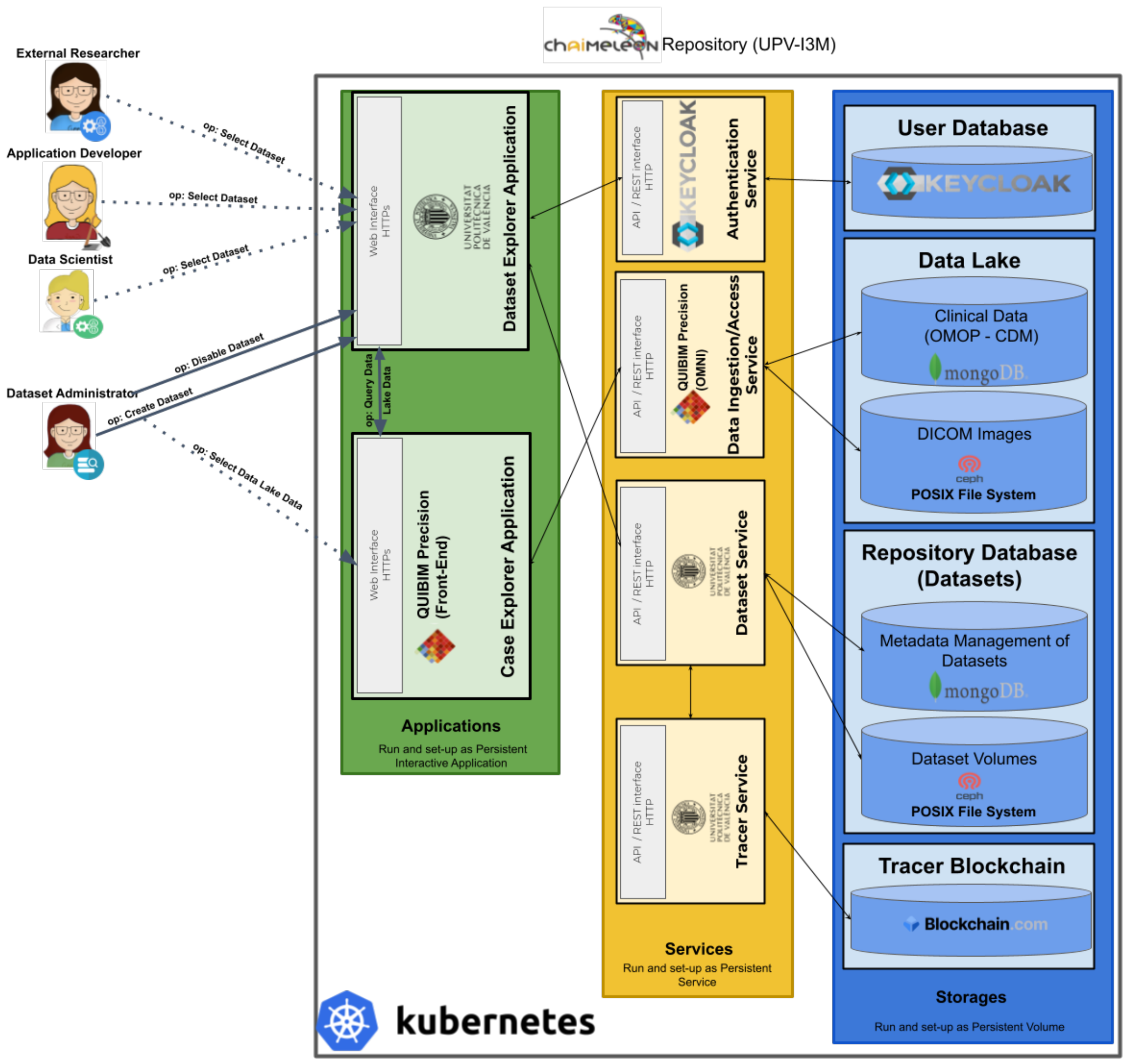


Figure 50. *Diagram of the Interim CHAIMELEON Architecture for Dataset Management.*

7.2.4 Standalone Application Management

Figure 51 shows the main components that will be implemented in the architecture of the *CHAIMELEON Repository* for *Standalone Application* management including interactions with *Users* and among components.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant

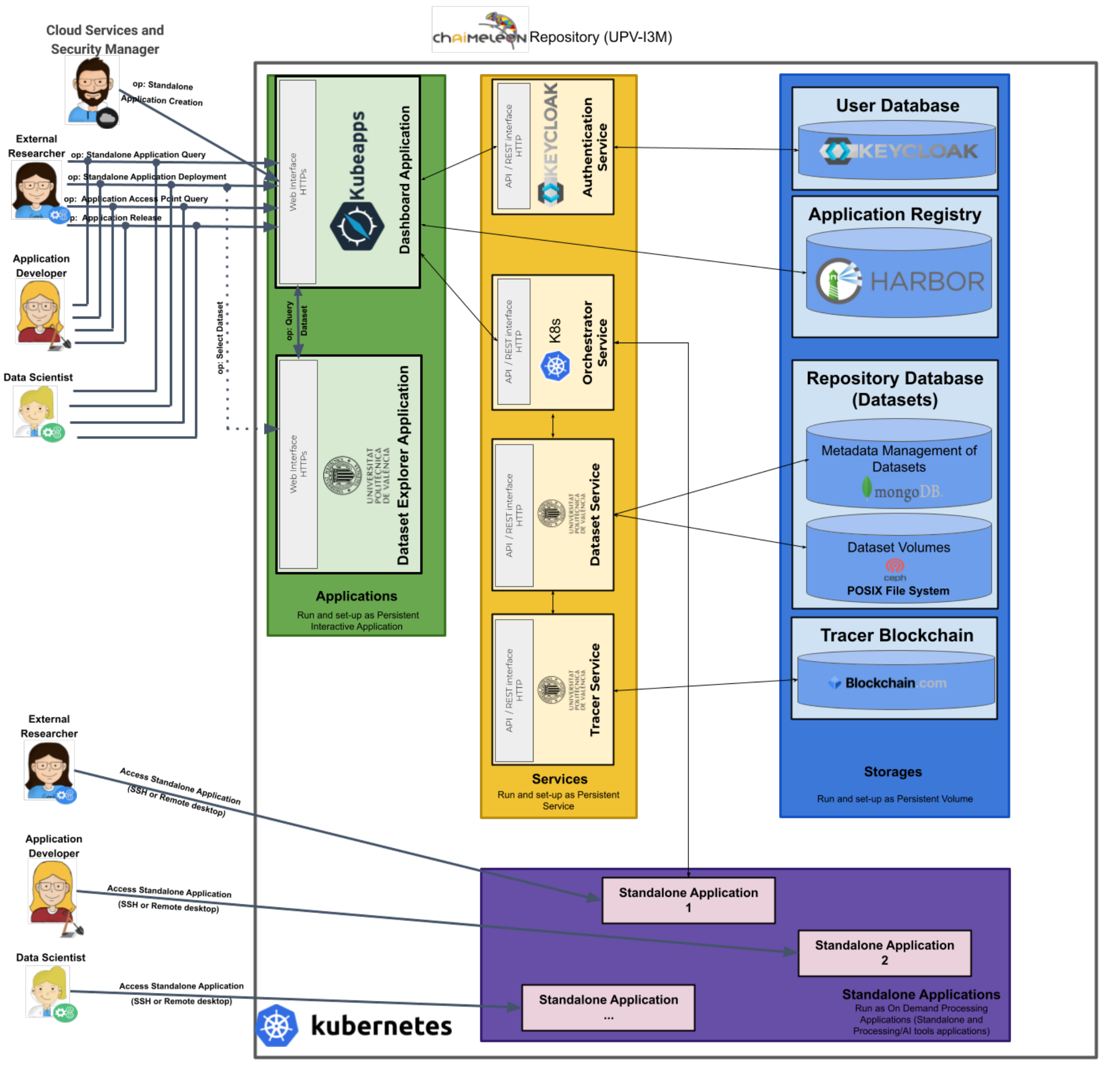

Cloud Services and Security Manager
op: Standalone Application Creation
Repository (UPV-I3M)
External Researcher
op: Standalone Application Query
op: Standalone Application Deployment
op: Application Access Point Query
op: Application Release
Application Developer
Data Scientist
op: Select Dataset
op: Query Dataset
Web interface HTTPs
Kubeapps
Dashboard Application
Dataset Explorer Application
Applications
Run and set-up as Persistent Interactive Application
API / REST interface HTTP
KEYCLOAK
Authentication Service
K8s
Orchestrator Service
Dataset Service
Tracer Service
Services
Run and set-up as Persistent Service
User Database
KEYCLOAK
Application Registry
HARBOR
Repository Database (Datasets)
Metadata Management of Datasets
mongoDB
Dataset Volumes
ceph
POSIX File System
Tracer Blockchain
Blockchain.com
Storages
Run and set-up as Persistent Volume
External Researcher
Access Standalone Application (SSH or Remote desktop)
Application Developer
Access Standalone Application (SSH or Remote desktop)
Data Scientist
Access Standalone Application (SSH or Remote desktop)
Standalone Application 1
Standalone Application 2
Standalone Application ...
Standalone Applications
Run as On Demand Processing Applications (Standalone and Processing/AI tools applications)
kubernetes

Figure 51.*Diagram of the Interim CHAIMELEON Architecture for Standalone Application Management.*

7.2.5 Case explorer Management (Marketplace)

Figure 52 shows the main components that will be implemented in the architecture of the *CHAIMELEON Repository* for *marketplace* management including interactions with *Users* and among components.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant

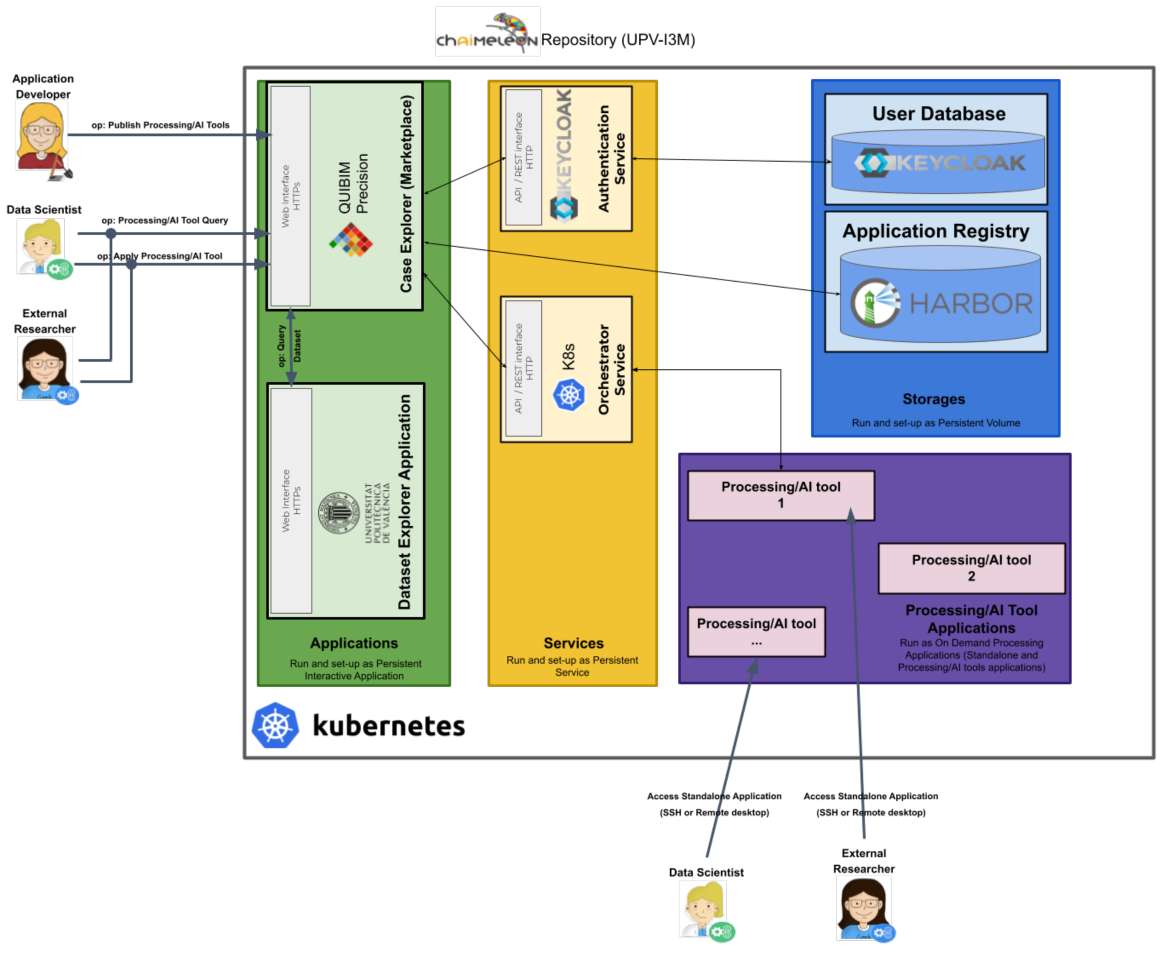

Repository (UPV-I3M)
Application Developer
op: Publish Processing/AI Tools
Data Scientist
op: Processing/AI Tool Query
op: Apply Processing/AI Tool
External Researcher
Web interface HTTPs
QUIBIM Precision
Case Explorer (Marketplace)
op: Query Dataset
Dataset Explorer Application
Applications
Run and set-up as Persistent Interactive Application
API / REST interface HTTP
KEYCLOAK
Authentication Service
K8s
Orchestrator Service
Services
Run and set-up as Persistent Service
User Database
KEYCLOAK
Application Registry
HARBOR
Storages
Run and set-up as Persistent Volume
Processing/AI tool 1
Processing/AI tool 2
Processing/AI tool ...
Processing/AI Tool Applications
Run as On Demand Processing Applications (Standalone and Processing/AI tools applications)
kubernetes
Access Standalone Application (SSH or Remote desktop)
Access Standalone Application (SSH or Remote desktop)
Data Scientist
External Researcher

Figure 52. *Diagram of the Interim CHAIMELEON Architecture for Marketplace Management.*

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant

8. Conclusions

An Interim version of the platform has been designed through a systematic and extensive process that considered the information on standards and technologies of previous deliverables and performed a thorough identification of the expected functionality of the repository with the participation of the whole consortium. It is important to state that the main two concepts are: The *CHAIMELEON Repository*, as a platform to store and process the data, and the *Dataset*, as a subset of anonymized and annotated data that has a Persistent Identifier.

This document is a guideline for the implementation of the repository and datasets, especially relevant for WP4, but also for the other WPs to find the way they could interact with the repository. As the repository will be mainly developed using Open-Source tools, 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 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. Those 8 *User Roles* have defined a total of 15 *User Stories*, which define narratively complete interactions with the repository. These *User Stories* have been the basis for the definition of 24 *Use Cases*, covering 5 areas: Authentication and authorization, Data Lake management, Dataset Management, Standalone Application Management and Marketplace Management. In this analysis, the CHAIMELEON repository identified 5 *Platform Applications* as Persistent applications that expose the functionality (User Registration Application, Case Explorer Application, Marketplace, Dataset Explorer Application and Application Dashboard), 5 *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 83 requirements covering 13 transversal requirements which are applied to all *Storages*, *Services* and *Applications*, 28 requirements applied to *Storages*, 18 for *Services* and 24 for *Platform Applications*.

One of the key aspects of the repository is traceability. Despite the use of anonymized data, the repository restricts the processing of the data to its local environment, so the data are not downloaded outside of the platform. Even in this case, the repository annotates the main research objects and actors (datasets, 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 interim 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