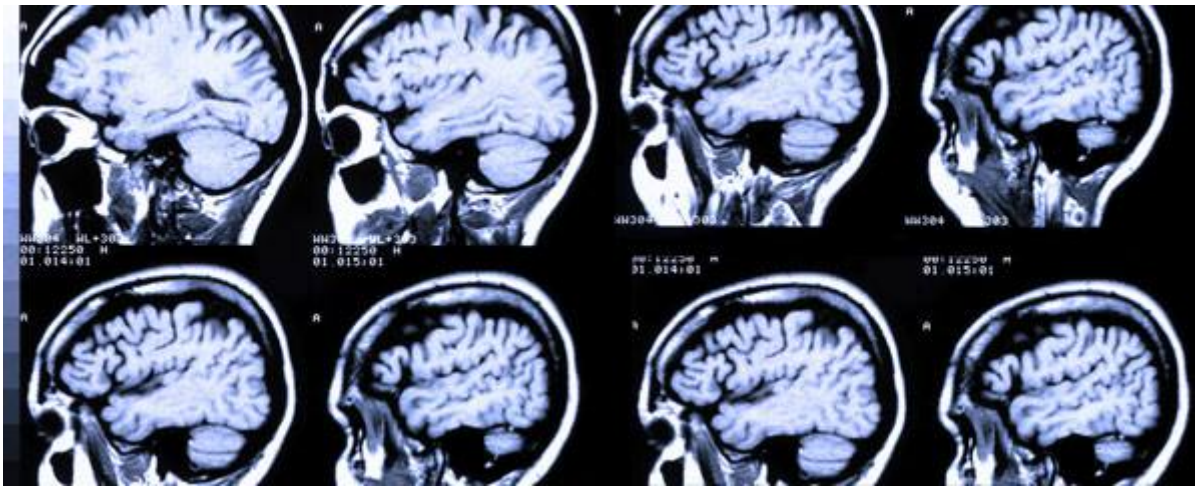
Department of Electronics, Telecommunications and Informatics
University of Aveiro

Informatics Project
LEI
2022

Osvaldo Rocha Pacheco
António Neves

MI4Web

Technical Report



16th June 2022

MI4WEB
Informatics Project

LEI
University of Aveiro
June 2022

Team:

Marta Fradique	98629
André Freixo	98495
Daniel Figueiredo	98498
Pedro Sobral	98491
Eva Bartolomeu	98513

Coordinators:
Augusto Silva
Joaquim Madeira

Keywords

DICOM - Digital Imaging and Communications in Medicine is a standard for handling, storing, printing, and transmitting information in medical imaging.
PACS - Picture Archive and Communication System is used to store and digitally transmit electronic images and clinically-relevant reports.
OHIF - Zero-footprint, open source and web-based medical imaging viewer.
VTK - *VTK.js* is a software system for *3D* computer graphics, modeling, image processing, volume rendering, and *2D* plotting.
Cornerstone - *JavaScript* library with a lot of relevant features like: rendering stacks of volumes viewports, manipulate Images, make annotations in the images, segmentation of 3D images in the volume viewports and synchronization between multiple viewports [12].

Acknowledgments

Firstly we would like to thank our supervisors Augusto Silva and Joaquim Madeira, for their support, advice and availability. Secondly, we want to acknowledge Rui Lebre for his help on the virtual machines configuration and some deployment issues. Thirdly, we praise the help of our colleague Vasco Regal with docker configurations. Also, we want to thank André Serralheiro for his help in the 3D opacity functions. Lastly, we are grateful to our friend Sebastião Ladeiras for his help editing the video DEMO of the project.

Index

1.Inception Phase

1.1 Context

Nowadays Medical Imaging is a very important component for medical practice, since it helps the healthcare professionals get a better understanding of the human body.

Medical Imaging ecosystems rely upon networked hardware and software components that ultimately govern the heavy dataflows between image sources and image consumers. Also, the technology PACS(Picture Archive and Communication System) is used to store and digitally transmit electronic images and clinically-relevant reports. In addition, Digital Imaging and Communications in Medicine (DICOM) is a standard for handling, storing, printing, and transmitting information in medical imaging.

Using the technologies described previously and a web-based, medical imaging viewer such as OHIF it is now possible to set up a minimal medical imaging networked platform.

1.2 Problem

Nowadays, the zero-footprint fully web-based visualization is making its way and gaining wider acceptance within the professional community, offering a great solution since most medically approved diagnostic visualization clients are still rather resource consuming vendor-locked software applications. Since the level of acceptance is growing up, the necessity of upgrading and building new functionalities and features is mandatory, so that it can continue to gain recognition and grow.

One of these applications is OHIF, a zero-footprint, open source and web-based medical imaging viewer, that gives us the ability to add and configure extensions, which makes it very expandable. Therefore, this platform will be the starting point of our project

1.3 Goal

Even though OHIF already has a lot of functionalities, it would be helpful for the healthcare professionals to have some additional features such as :

- Basic transactions for medical image storage and retrieval ;
- DICOM import and export;
- Multiplaner image display functionalities;
- 3D Display tools as VTK plugins;
- Image Annotation Edition and Storage;
- Creation of Medical Reports;
- Admin Interface to manage the application equipment and the access.

Overall, our main goal is to develop these new features and new interfaces.

1.4 Task List

In order to monitor the tasks and have a good team work environment, we used a tool called Jira. In which we defined the main modules of the project, listed the tasks in each module and assigned each task and module to one or more elements of the group.

1.5 Expected Results

If all our goals are accomplished we will have a platform on the web that allows us to visualize image studies using a viewer client in conjunction with a server.

Also, the services will have to run on the web and in a client-server structure.

Moreover, the visualization of the images will have to allow canonical views and three-dimensional object views and the implemented system will have to be distributed.

Finally, we plan to have a platform capable of supporting different users and an admin to manage the users and the application.

1.6 Related work

At this moment we can have access to some open source tools that support the PACS servers. Such as, OHIF, an open source, web-based, medical imaging viewer, DICOOGLE an open source PACS and ORTHANC an open source DICOM server.

1.7 Communication Plan

- Project Plan: Jira;
- Reports and presentations: Google Docs and Google Slides;
- Repository: Github;

1.8 Team Roles

- Product Owner: Marta Fradique
- DevOps Master: Daniel Figueiredo
- Architect: Pedro Sobral
- Architect: André Freixo
- Team Manager: Eva Bartolomeu
- Advisers: Augusto Silva and Joaquim Madeira

1.9 Project Calendar

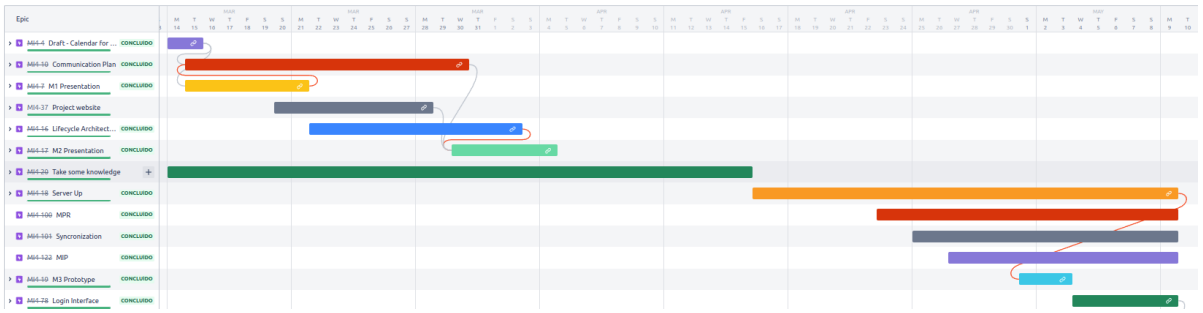


Fig 1- Project calendar 14 March - 9 May

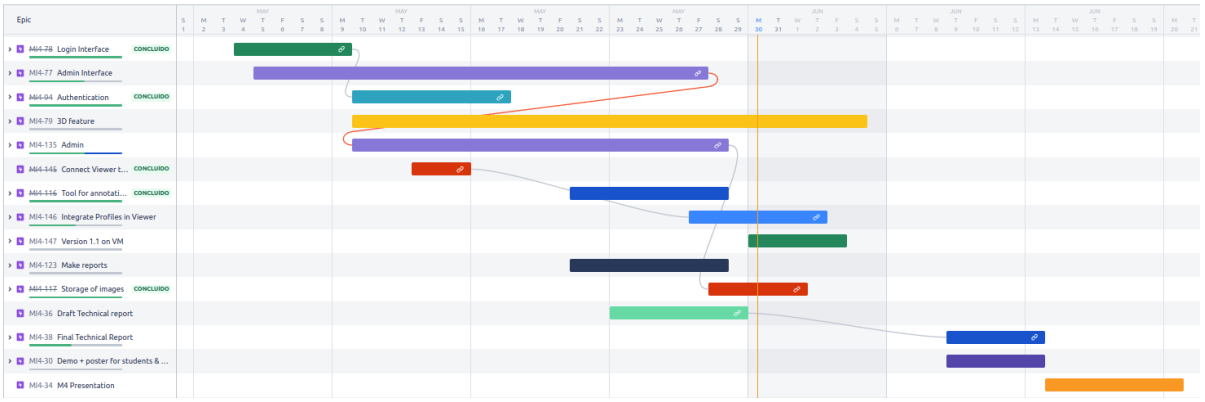


Fig 2- Project calendar 14 March - 9 May

2. Elaboration Phase

2.1 Requirements Gathering

To have a complete comprehension of the problem, we had several meetings with our coordinators, and read documentation (books and presentations) about medical images and how to handle images of this kind. Also, we learned about the existing technologies around this problem.

Firstly, we decided to make important decisions about how some features will be implemented on our project, what will be the system boundaries, the main objectives of the system and what users the system will be able to handle.

Secondly, we did research about the state of art in this area, and we discovered that some projects similar to ours already exist (Ambra Health, IBM Clinical Development, and some solutions that Siemens Healthineers presents). However all of them are always proprietary software, so it's not simple to have this expensive software, and in most cases it is necessary to have special licenses to have them. Despite that, this research improved our requirements and our insight about the project.

In the third part, using all the information that we gathered and with the help of our coordinators, and their experience in the area of the medical images, we discussed some points and its importance in the project and the time we have to implement them.

Therefore, with all of this information, we were able to make some settlements about the project, such as:

- The user interface has to be simple, with elementary processes and operations.
- It is very important to have a secure system, in order to maintain all the sensible data in the right hands.
- The users of the system.
- Determinate the technologies and the main tools to implement the proposed objectives.

In order to access the system, we decided to implement a web application, so that the users can access the platform at any time. The users have different types of tools and permissions. All the information, in this case the medical images, will be stored on a PACS server, and will be used by a service that truly enables core medical imaging data flows over the internet.

2.2 Context and State of The Art

2.2.1 Context

In an environment related with medical imaging there is the need to use technologies in order to govern the heavy dataflows between image sources and image consumers.

In the development of this project we aim to create a Web platform that grants us the ability to visualize medical image (DICOM) studies, with the use of a visualizer based on OHIF in conjunction with ORTHANC, which implements PACS system. Our main goal is to develop extensions capable of adding functionalities to treat and visualize medical images, both in 2D and 3D.

2.2.2 State Of The Art

There are some works that were already developed that are related to our project.

OHIF - The open health imaging foundation is a medical imaging viewer that is open-sourced and web-based which can be customized in order to connect to image archives that support DicomWeb, helping to map proprietary API formats.

It enables the deployment of imaging applications on premise or in the cloud without requiring installation of custom software on the user's computer.

It has many benefits, like decreased costs and information technology support requirements, as well as improved accessibility across sites.

It can also be modified in order to support site-specific workflows and accommodate evolving research requirements.

DICOOGLE - It is an open source PACS archive system that has enabled its use in research and the healthcare industry, by covering a wide variety of use cases without changes to the core system.

Some of the features of DICOOGLE are:

DICOM - Compliant implementations of the Storage and Query/Retrieve service classes;

A set of REST web services for user-facing applications to interface with Dicoogle;

A web application that enables its configuration and usage;

A plugin-based backbone for loading extensions in deployment time;

A common library for developing plugins named Dicoogle SDK.

ORTHANC - This provides a simple and powerful standalone DICOM server and is designed to improve the DICOM flows in hospitals and to support research about the automated analysis of medical images [7].

It hides the complexity of both the DICOM format and DICOM protocol and lets the users give attention to the content of the DICOM files. One unique aspect of ORTHANC is that it supplies a RESTful API, which allows to drive Orthanc from any computer language.

Lastly, it features a plugin mechanism that adds new modules which extend the main capacities of its REST API.

2.3 Actors

The target user for the application is a person that wants to see medical images and has the facilities needed (permissions), whose knowledge on technology must be at least the minimum needed so it can work well with the interface presented. The user of the application will commonly be a member of a clinical staff and has the permissions to check exams of patients.

The levels of expertise required to use the system are little:

- For everyone that uses the system, as the web application experience is similar to using any regular website, anyone familiar with web browsing is skilled enough.

The main actors are described in the following list:

- Guest: Represents an user of the system that is not logged in, therefore they have the access to the same features as the referring imaging staff, although they only have access to a reduced number of studies.
- Clinical Imaging Staff: Represents a person that works in the health area and has permissions to check medical images from patients, meaning that it is an authenticated user in the web server, that has full access and full permissions to access images and create annotations on those images and create reports.
- Referring Imaging Staff: Same thing as the Guest user although has access to all medical images.
- System Manager: Represents an administrator of the system, and is able to manage the access to the application, network DICOM node control, manage the extensions and check the data and statistics of the application.

2.4 Use Cases

Figure 3 and 4 presents the use case diagram related to the web application and its different actors (clinical imaging staff, referring clinical staff, system manager and guest).

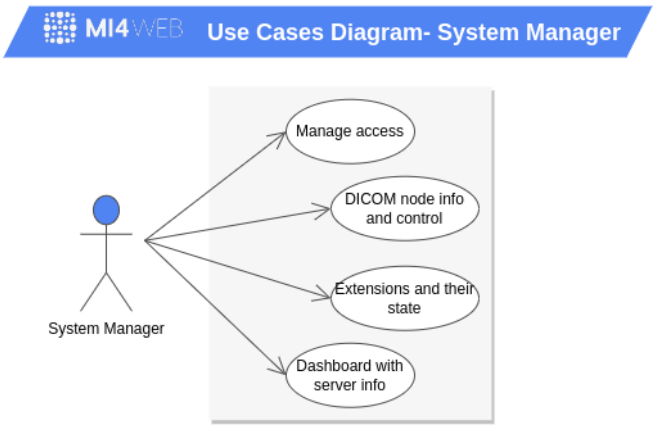


Fig 3- System Manager Use Cases Diagram

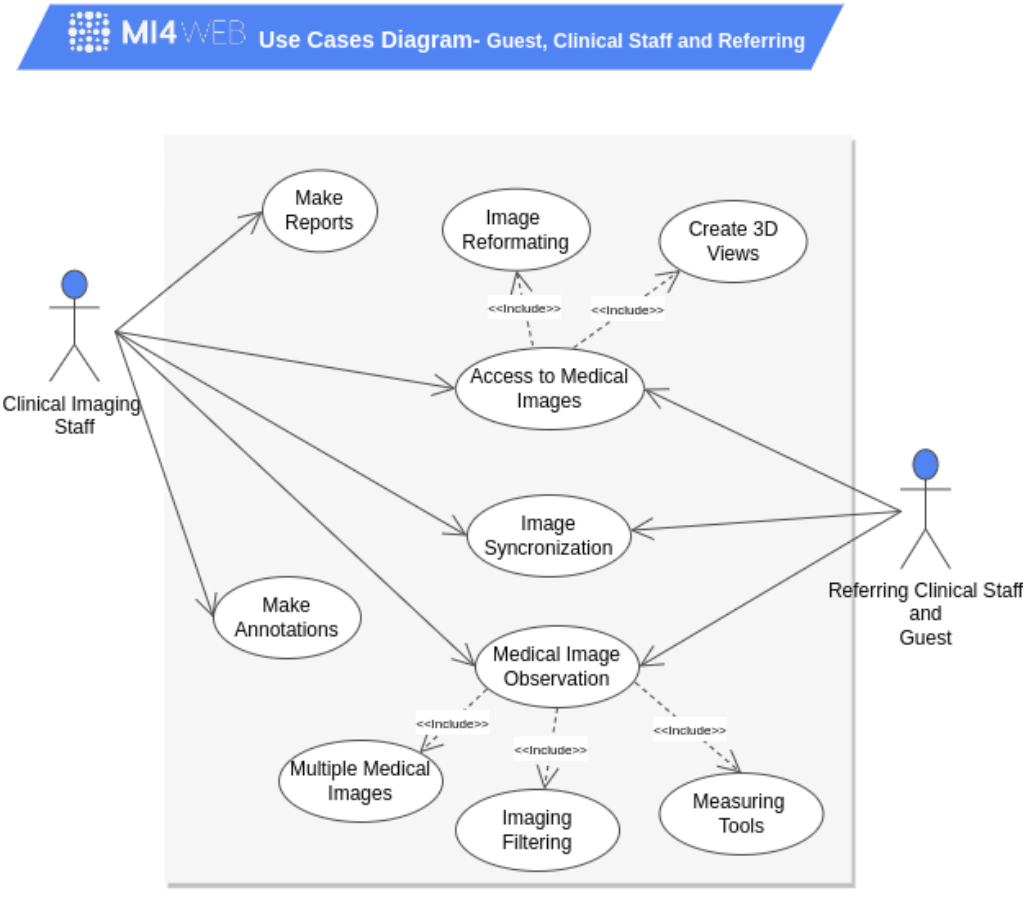


Fig 4-Guest, Clinical Imaging Staff and Referring Clinical Staff Use Cases Diagram

2.4.1 System Manager

- **Manage access**

  It's possible to manage the access of the system features.

  Priority: High

- **Extension Installation**

  The admin can see all the extensions of the application, check if they are being used , connect them and disconnect them.

  Priority: Medium

- **Network DICOM node model**

  The admin can analyze the DICOM nodes connected to the network, and check the information about each of them.

  Priority: Medium

2.4.2 Clinical Imaging Staff

- **Multiple medical images**

  Allows the display of several medical images at the same time.

  Priority: High

- **Image Synchronization**

  It is possible to observe multiple images concurrently, we can also synchronize those images to scroll at the same time.

  Priority: High

- **Image reformatting**

  Create 2D views in addition to axial views, which means synchronous sagittal views and coronal views.

  Priority: High

- **Create 3D views**

  Allow to create 3D images with different views, and it is possible to move the 3D images around so it's easier to observe them.

  Priority: High

- **Make Annotations and Reports**

  Create annotations on DICOM studies and make a report on those images.

  Priority: High

2.4.3 Referring Imaging Staff and Guest

- **Image Synchronization**

As it is possible to observe multiple images at the same time, we can also synchronize those images to scroll at the same time.
Priority: High

- Image reformatting
  Create 2D views in addition to axial views, which means sagittal views and coronal views.
  Priority: High
- Create 3D views
  Allow the creation of 3D images with different views, and it is possible to move the 3D image around so it's easier to observe the image.
  Priority: High
- Multiple medical images
  Allows to display several medical images at the same time.
  Priority: High

2.5 Requirements

2.5.1 Functional Requirements

1. Business Rules:
   - Our applications differentiates between Referring, Imaging Staff, Systems Manager and Guest, both in business logic and client applications.

2. Administrative functions

   - Our application has a Systems Manager responsible for controlling accesses, control of DICOM nodes and extensions management.

3. Authentication and authorization levels

   - Since security is very important in the health industry, to have access to their privileges, systems manager, referring and imaging staff, should be logged-in.
   - Imaging Staff makes annotations and reports of the DICOM studies(is for example an radiologist).
   - Referring has all the visualization tools but can't make annotations in the system (for example it can be a doctor or a physiotherapist),
   - Systems Managers are responsible for controlling accesses and other tasks that were already mentioned before.
   - The guests can only try a demo of the app, being able to try the tools in one studie.

4. Historical data

- Our application is based in OHIF, an open-source web image platform, which is the base and starting point of our project.

5. Regulatory Requirements

- Users are able to login into their accounts .
- The application is able to perform requests to the ORTHANC server through the DICOMWeb API.
- Our application supports DICOM image import and export.
- All the images that are retrieved from the server can be accessed through the application.
- It's possible to use image segmentation tools.
- The application has an admin interface.
- Users are able to edit and make annotations about the images.
- Our application supports 3D display tools such as plugins and 2D display functionalities.
- The users in the Referring category are able to write reports and download them.

2.5.2 Non-Functional Requirements

1. Usability

This system is easy to use and learn, since our platform is open to anyone. Users may not have a technological background, reinforcing the idea that the processes must be easy to understand.

Priority: High.

2. Reliability

Since this is an application where we are dealing with various health related data, it's very important for this system to run flawlessly for a certain period of time. Moreover, reliability is a fundamental requirement to minimize the chances of failure. In addition, it is also relevant to have a good error handling, if an error occurs, the application is responsive so that the user doesn't misunderstand, or leave the system.

Priority: High.

3. Security

Security is extremely necessary for this platform, as we are dealing with health related data.This application has data that will only be authorized to a certain entity, or entities, and this is where the authentication and authorization part comes in. Given this, an authentication module was implemented, so that each user is identified. Besides that, the authorization must also be visible, so that each user can access their private data.

Priority: High.

4. Interoperability

Interoperability is an essential requirement, which will facilitate the sharing of information and data with other systems and external hardware, such as the PACS server with a viewer.

Priority: High.

5. Portability

The Web based platform is accessible in different web browsers, and whichever browser must work the same.

Priority: Medium.

6. Scalability

In this project the server is able to handle more than one client at the same time, which brings the necessity of being a distributed system. Consequently, we have a good horizontal scalability, that is, adding more machines to the resource pool. In addition, we also have to scale it vertically, that is, add resources and energy to a machine (CPU, GPU, RAM, storage,...).

Priority: Medium.

7. Performance:

The software system is fast in terms of user actions, and in the view. Because, many of the features that the users have are based on image manipulation, which needs a quick response so that there is no misunderstanding with the system.

Priority: High.

8. Availability

Availability can also be one of the requirements. Since this platform on the Web is public for anyone, it's very important to always be available to the users.

Priority: High.

9. Maintainability

Maintainability is one of the requirements that may be included. It grants easiness of later correction and alterations of a component of the system, improving that component.

Priority: Medium.

2.6. System Architecture

In this section we will explore the details to have a good implementation of the system, as well the pros and cons to choose a technology instead of another. So all the architecture will be extensively analyzed.

The next figure presents the architecture defined for our system. In a first view we can separate the architecture into four main modules: Frontend, Backend, Persistence and *PACS Server*. In the frontend module, since the *OHIF viewer* was the starting point of the

project, the technology selected was *React*. In the backend module, we have to deal with many aspects, an *API* is necessary to communicate with the frontend and the server, to generate and process the *3D* images it is important to have an image processing component. Also, It is extremely important to have a secure system, because of that an authentication protocol was implemented. In the Persistence Module, we look forward to a simple database, *MySQL*, to store user profiles mainly.

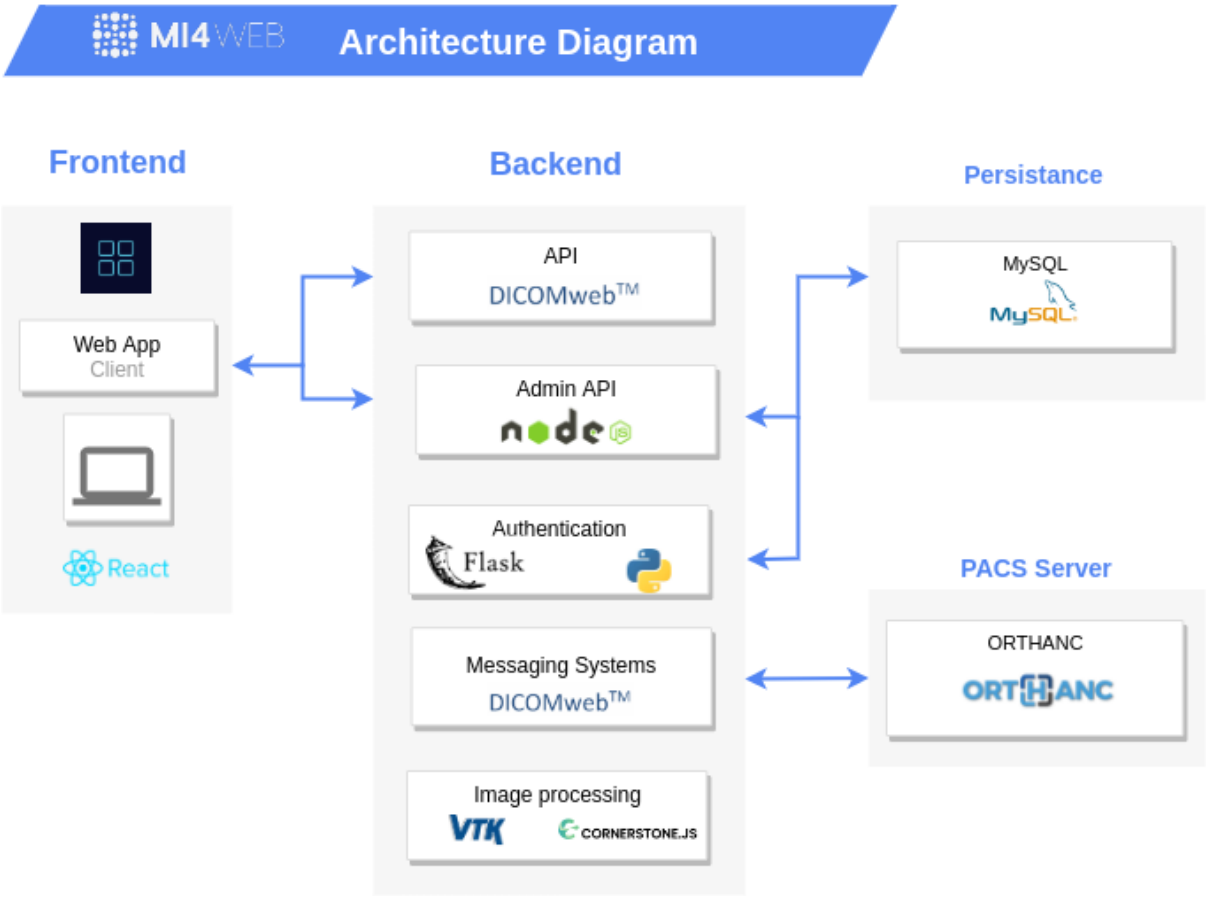


Fig 5- Architecture Diagram

2.7.1 Frontend

As referred before this module is built using the *OHIF* viewer, and as this viewer works with *React*, it will be used as well in our project.

- *React*: Is a *JavaScript framework* that is very popular nowadays, with *React* it is possible to create complex pages in a facilitated way since some components needed for it were already developed. The main reason why we used *React* was because the application OHIF was already built in *React*, therefore, we had to work in consonance with what was previously done before.

- *OHIF*: The Open Health Imaging Foundation is a medical imaging viewer that is open-sourced and web-based which can be customized in order to connect to image archives that support *DicomWeb*, helping to map proprietary *API* formats. It has so many advantages, such as:
    - It enables the deployment of imaging applications on premise or in the cloud without requiring installation of custom software on the user's computer.
    - It decreased costs and information technology support requirements, as well as improved accessibility across sites.
    - It can also be modified in order to support site-specific workflows and accommodate evolving research requirements.

A disadvantage that we saw is that it can be a little bit complicated, at first to understand all the viewer code, but with the help of the following documentation (https://docs.ohif.org) we passed this difficulty.

2.7.2 Backend

The backend is divided into different components: *API*, Authentication and Image Processing.

- ***API***: We use two APIs for different purposes, one to work with medical images, the DICOMWeb, and another created by us with node.js to manage various services in our platform.

    - The *DICOMWeb*, is a *RESTful DICOM* service, it is able to send, retrieve and query for medical images and related information. It is a light-weight way to access the images, and it was built to be easy to implement by developers who have minimal familiarity with the *DICOM* standard [10].

    - The API is made with nodeJS with Express, and is able to handle with access control, display information in the admin interface

- **Authentication**: Security is a very important factor in our project, we are dealing with medical images, and the security of the data and the privacy of the patient are very relevant. In order to make our web application secure, an authentication system based on a *E-CHAP* protocol will be used, so only the authorized have access to their information.

2.7.5 Image Processing

One of the main features of the project is to develop ways and tools to see medical images in different ways (views), to make *3D* images in the different images and views orientation. In order to implement those features, we started by creating an extension, as we said before OHIF was built to be flexible and extensible enough to support any workflow,

therefore it's easy to add extensions that provide new functionalities and new behaviors.

There are a few core extensions available out of the box and maintained by the OHIF community, two of them are VTK and Cornerstone which we will use in this project.

Moreover, *VTK.js* is a software system for *3D* computer graphics, modeling, image processing, volume rendering, and *2D* plotting. So there are many *VTK.js* features that are relevant for us, so that we will use this library.

As for *Cornerstone.js*., we used it in order to implement a synchronization between multiple viewports. This *JavaScript* library has a bunch of relevant features like: rendering stacks of volumes viewports, manipulate Images (zoom, scroll), make annotations in the images (measure distances, write important things, supports multiple viewports), segmentation of 3D images in the volume viewports; and synchronization between multiple viewports.

2.7.3 Persistence

In this module a database was implemented, with the objective of storing the user information of the system, mainly logins and tokens to identify users. As we already work with *MySQL* in a previous project, we decided to use this database. It is light, it is easy to scale and offers good protection in terms of security.

2.7.4 PACS Server

Since we need a server to communicate and store our images, we used a *PACS* server called *Orthanc*. *Orthanc* is an open-source, lightweight *DICOM* server for healthcare and medical research. It is designed to have the following objectives:

- To ease *DICOM* scripting for clinical routine;
- To ease data management for clinical routine and medical research;
- To bring *DICOM* images to the Computer Vision systems;

It hides the complexity of *DICOM* format and *DICOM* protocol and lets the users give more attention to the content of the *DICOM* files. A very good thing about this server is that it supplies a *RESTful API,* which allows to drive *Orthanc* from any computer language, becoming a very good server to implement on our project.

2.8 Information Model

To build the Information Model, firstly, we created the table user, which has two descendents, the table not_accepted, which are the users that are yet to be accepted by the administrator, and the table staff, which are the users that were already verified and accepted by the administrator. There is also the table admin, which defines the administrator of our website, having the responsibility of checking and accepting users, and managing everything. Lastly, we also have the table chart_info, which provides the information about the number of studies carried out per day.

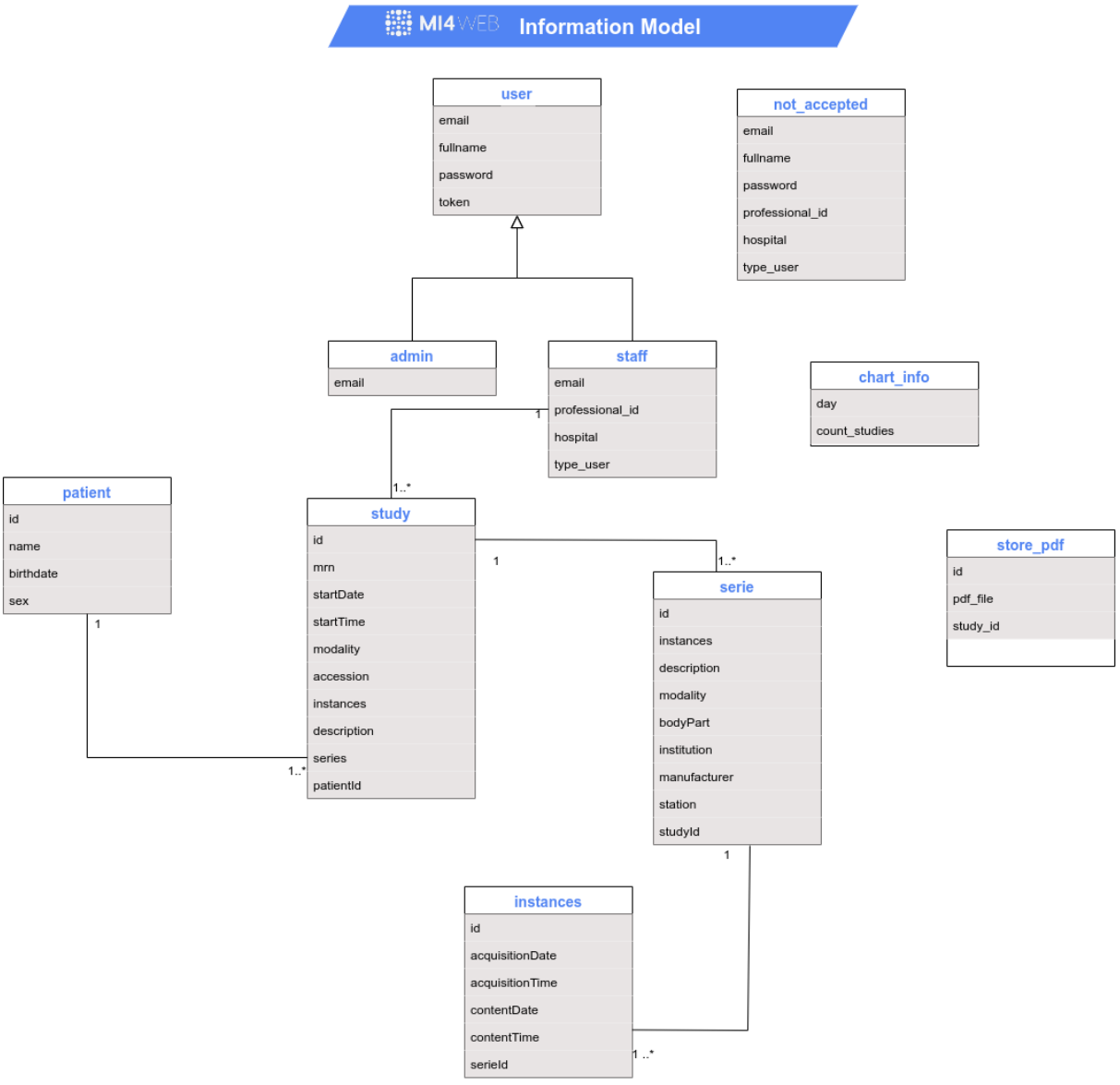


Fig 6- Information Model DIagram

2.9 Deployment Diagram

The following deployment diagram shows the respective components that compose our system and how they will be deployed and communicated.

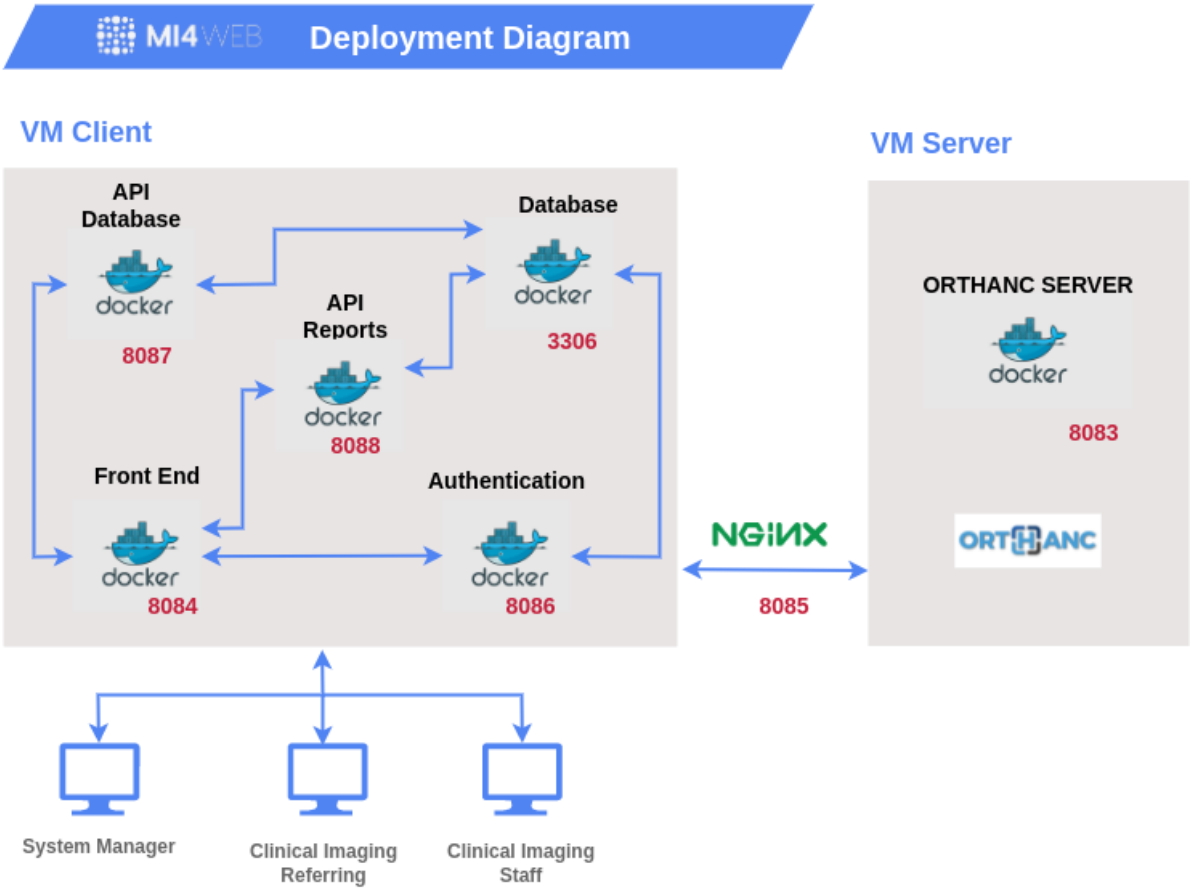


Fig 7- Deployment Diagram

We have two virtual machines in our deployment, one as Server, and other as Client.

The Server has a docker container on 8083 port to up the PACS Server, in that case, the Orthanc Server that will store the medical images.

The Client have five docker containers:

- Front-end on port 8084,
- Authentication serve on 8086 port,
- Mysql database on 3306 port,
- API - Database on port 8087,
- API - Reports on port 8088.

The communication between the machines have a nginx configuration, in order to have a reverse-proxy to not have CORS problems, when the viewer makes requests to the Orthanc server.

In each virtual machine we have 8 ports that can be used, and these ports have a different mapping to be reached from outside; in the VM Client side we have ports between 8082-8089 open, mapped through mednat network with the ports between 8752-8759, in the VM Server side we have ports between 8082-8089 open, mapped through mednat network with the ports between 8762-8769.

All the connections have to be made with the UA network or with the UA VPN.

The connection to the viewer is made through the following link: http://mednat.ieeta.pt:8754

It is possible to see every step to deploy the full application, in the following link: https://github.com/TheScorpoi/mi4web/blob/main/devops/README.md

2.10 Key Components

2.10.1 Extension Overview

Extensions are configurable through an extension object. An extension is a plain JavaScript object that has an id property, and one or more modules and/or lifecycle hooks.

```
4   export default {
5     /**
6      * Only required property. Should be a unique value across all extensions.
7      */
8     id: 'example-extension',
9
10    /**
11     * LIFECYCLE HOOKS
12     */
13
14  >  preRegistration({…
19    }) {},
20
21  >  /**…
24  >  getViewportModule() {…
26    },
27  >  getSopClassHandlerModule() {…
29    },
30  >  getPanelModule() {…
32    },
33  >  getToolbarModule() {…
35    },
36  >  getCommandsModule(/* store */) {…
38    },
39  };
```

Fig 8- Extension Skeleton

- **getViewportModule** returns the viewport component associated with the extension.
- **getToolbarModule** should return an object which contains UI buttons to be put on the main OHIF toolbar.
- **getCommandsModule** provides a list of commands which may either be bound to toolbar buttons in the toolbarModule or to hotkeys.

Extensions can also add custom UI components to either of OHIF side panels through the panelModule.

2.10.2 Module: Toolbar

ToolbarModule is a "plain old JavaScript Object" with some key words OHIF recognises and consumes. It constructs the UI that allows execution of commands from the commandModule (Fig. 9).

```
20  const TOOLBAR_BUTTON_TYPES = {
21    COMMAND: 'command',
22    SET_TOOL_ACTIVE: 'setToolActive',
23    BUILT_IN: 'builtIn',
24  };
25
26  > const TOOLBAR_BUTTON_BEHAVIORS = {…
31  };
32
33  /* TODO: Export enums through a extension manager. */
34  > const enums = {…
37  };
38
39  const definitions = [
40    {
41      id: 'StackScroll',
42      label: 'Stack Scroll',
43      icon: 'bars',
44      //
45      type: TOOLBAR_BUTTON_TYPES.SET_TOOL_ACTIVE,
46      commandName: 'setToolActive',
47      commandOptions: { toolName: 'StackScroll' },
48    },
49    {
50      id: 'Zoom',
51      label: 'Zoom',
52      icon: 'search-plus',
53      //
54      type: TOOLBAR_BUTTON_TYPES.SET_TOOL_ACTIVE,
55      commandName: 'setToolActive',
56      commandOptions: { toolName: 'Zoom' },
57    },
58    {
59      id: 'Wwwc',
60      label: 'Levels',
61      icon: 'level',
62      //
63      type: TOOLBAR_BUTTON_TYPES.SET_TOOL_ACTIVE,
64      commandName: 'setToolActive',
65      commandOptions: { toolName: 'Wwwc' },
66    },
67    {
68      id: 'Pan',
69      label: 'Pan',
70      icon: 'arrows',
71      //
72      type: TOOLBAR_BUTTON_TYPES.SET_TOOL_ACTIVE,
```

Fig 9- Example of a toolbarModule (cornerstone extension toolbarModule).

The definitions object defines buttons to be placed in the toolbar. This includes the icon, label and command to be executed, as well as options to pass that command.

Buttons of type **setToolActive** define the buttons as a toggleable tool, which lets OHIF know to untoggle other buttons of the same type to signify a change in tool.

Command executes the given command from the command module.

```
265  export default {
266    definitions,
267    defaultContext: 'ACTIVE_VIEWPORT::CORNERSTONE',
268  };
269
```

Fig 10- Example of a toolbarModule (cornerstone extension toolbarModule).

The toolbarModule consists of the definitions and the defaultContext (Fig. 10). This is the context in which the defined buttons will appear in the toolbar.

Here this is when the context is ACTIVE_VIEWPORT:CORNERSTONE, which means these buttons will only be displayed when the active viewport is a CORNERSTONE.js viewport.

2.10.3 Module: Commands

CommandsModule is composed of actions and definitions and the defaultContext (Fig. 11).

```
329  const definitions = {
330    jumpToImage: {
331      commandFn: actions.jumpToImage,
332      storeContexts: [],
333      options: {},
334    },
335    getNearbyToolData: {
336      commandFn: actions.getNearbyToolData,
337      storeContexts: [],
338      options: {},
339    },
340    removeToolState: {
341      commandFn: actions.removeToolState,
342      storeContexts: [],
343      options: {},
344    },
```

Fig 11 - Example of a commandsModule (cornerstone extension commandsModule).

The definitions object defines commands which can be used by the toolbarModule or bound to hotkeys.

storeContexts can be passed to commands, such that the commands may have access to data from within the redux store.

```
23  const commandsModule = ({ servicesManager }) => {
24    const actions = {
25      rotateViewport: ({ viewports, rotation }) => {
26        const enabledElement = getEnabledElement(viewports.activeViewportIndex);
27
28        if (enabledElement) {
29          let viewport = cornerstone.getViewport(enabledElement);
30          viewport.rotation += rotation;
31          cornerstone.setViewport(enabledElement, viewport);
32        }
33      },
34  >   flipViewportHorizontal: ({ viewports }) => {…
42      },
43  >   flipViewportVertical: ({ viewports }) => {…
```

Fig 12 - Example of a commandsModule (cornerstone extension commandsModule).

The actions object defines the actions associated with each definition (Fig 12). These are functions that may change the active tool, perform an operation, or change the state in some way.

2.10.4 react-vtkjs-viewport

react-vtkjs-viewport [21] is an open source component library containing React components for easily displaying DICOM data with VTK.js.

The react-vtkjs-viewport contains two major elements: View2D and View3D React components. A set of maintained interactor styles and widgets.

The interactor styles and widgets are not deeply integrated with the viewport components, such that consumers of the library can choose which tools to attach, and to allow extensibility in the future.

3. Implementation

In this section we will explain how the main features of this application were implemented.

3.1 Authentication

As mentioned before the Clinical Imaging Staff, Referring Imaging Staff and System Manager must be logged in to have access to the application, and since our application has sensitive data about the patients health it was very important to implement a safe protocol to protect the client data.

To implement this we decided to use the protocol Enhanced challenge-response authentication protocol (E-CHAP) learned in the previous semester in the subject SIO. This protocol consists of instead of running the authentication protocol one time, running it N times instead, and providing in each protocol the minimum set of information (1 bit) in each response. Then, both participants only get convinced about the benignity of the other after N correct answers. If the server receives a wrong answer, instead of aborting, it sends random responses, this way, if the other one is an impostor, it will no longer benefit from the information received. More information about this protocol can be accessed in this link.

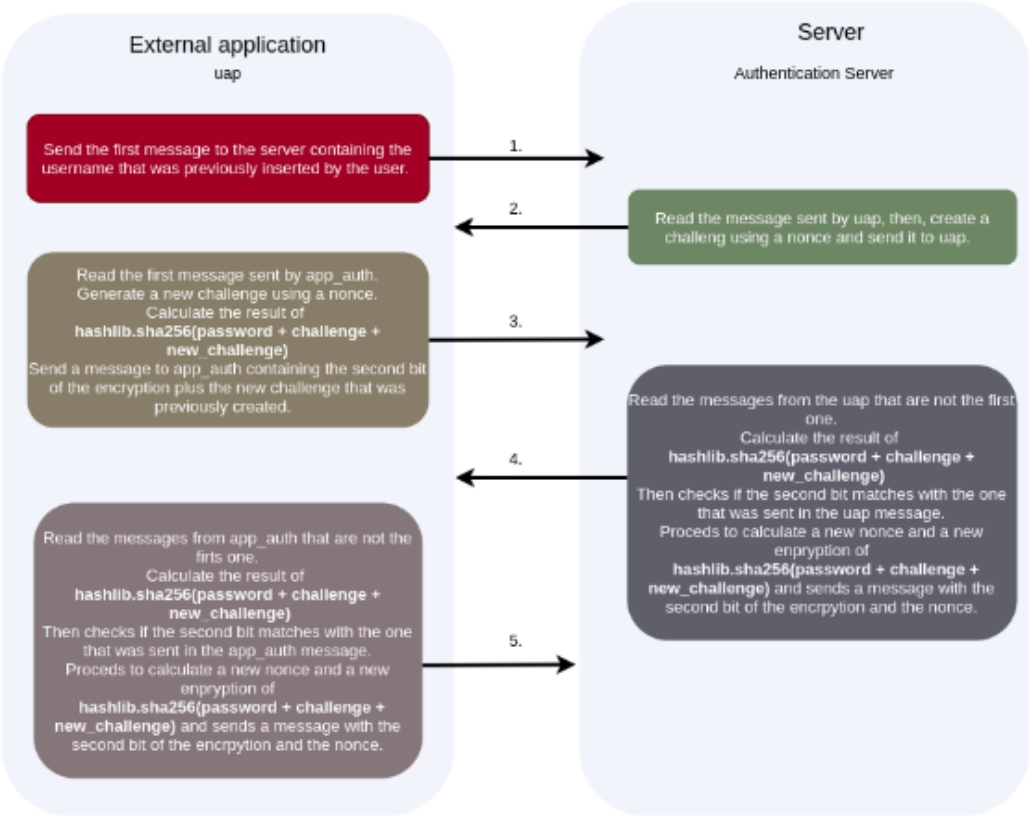


Fig 13- Authentication Protocol Diagram

Also, the Clinical Imaging Staff and Referring Imaging Staff should register to use the application with all the features. To build this system and save all of the accounts we used a database in *MySql,* each user has a field corresponding to which category he belongs to,

then, when the users sign into their accounts it will lead to the application with the authorization level of his category(Clinical Imaging Staff or Referring Imaging Staff or System Manager).

Fig 14- Login Page

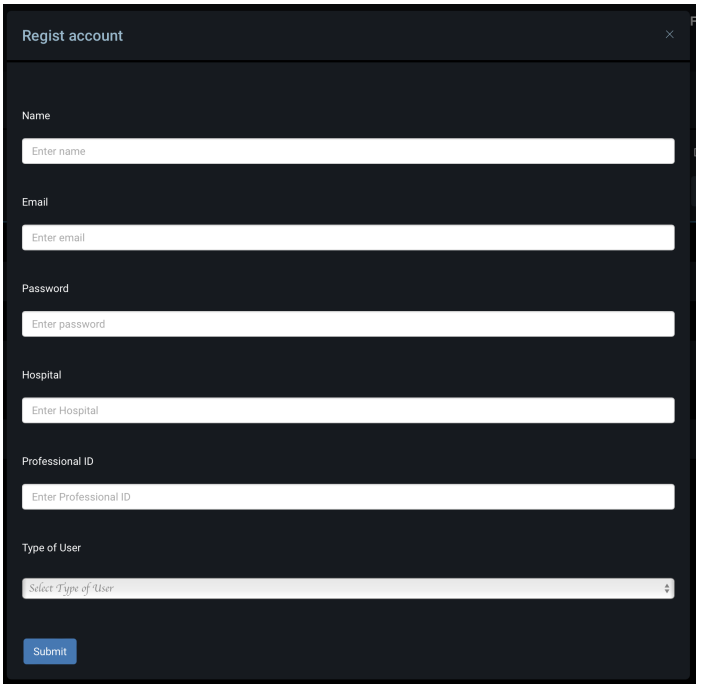


Fig 15- Register Page

3.2 Admin Interface

Here we can see all of the admin interfaces. All of the interfaces were built in React so it would be in consonance with the *OHIF* viewer.

3.2.1 Dashboard

In the Dashboard page, we can see all of the important information about the *ORTHANC* server. To fetch these data, we used the *REST API* of *ORTHANC*.

It is possible to see specific information about the server like Server and Database version, number of images, patients, series, and total memory (disk) used. There is a graphic that displays the number of studies presents on the the server by time.

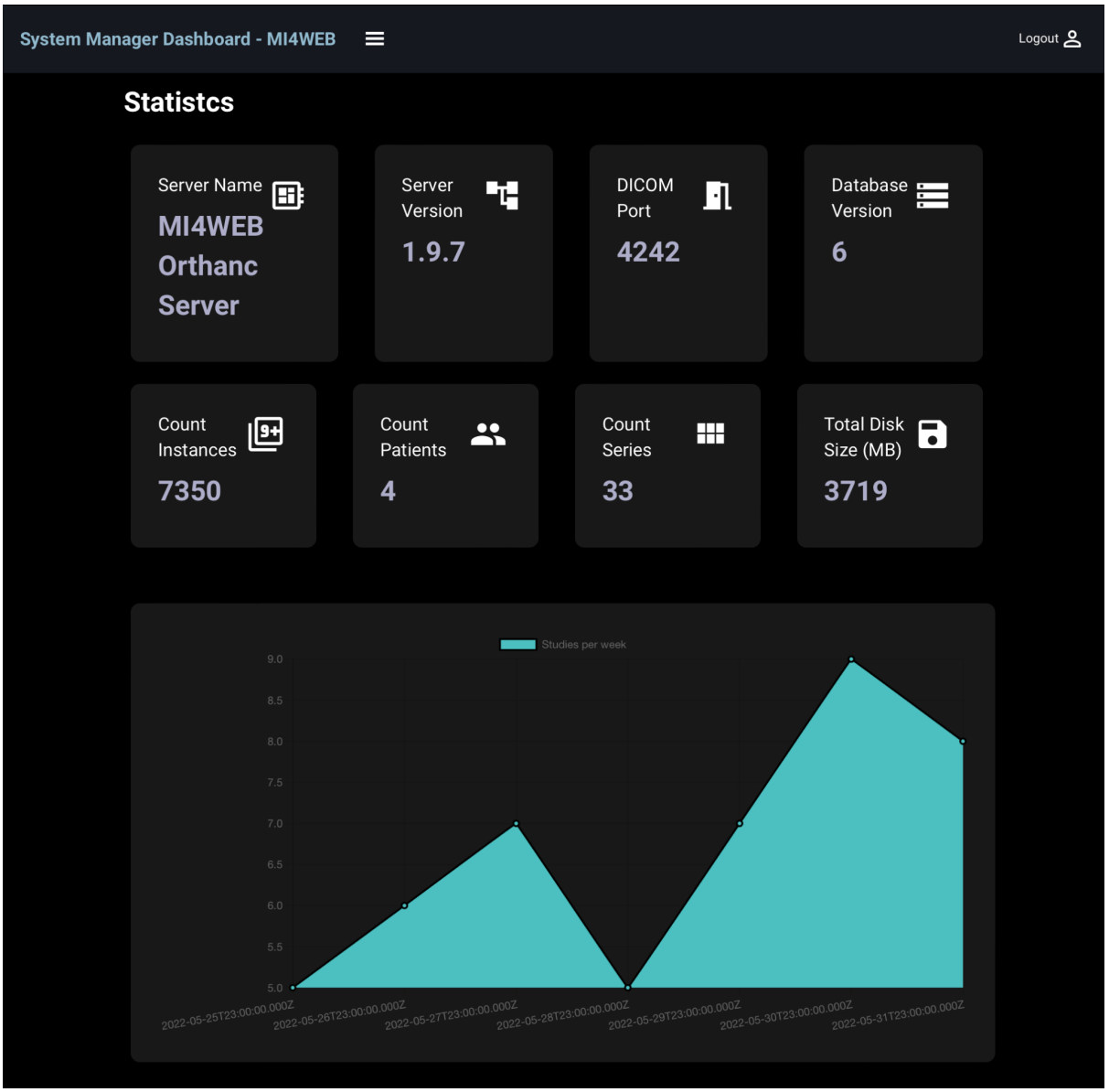


Fig 16- Admin Dashboard

3.2.2 Access Control

In this section the admin can control the accesses to the application, decide if the new users that registered should be accepted and if some users should get their privileges revoked and accounts eliminated.

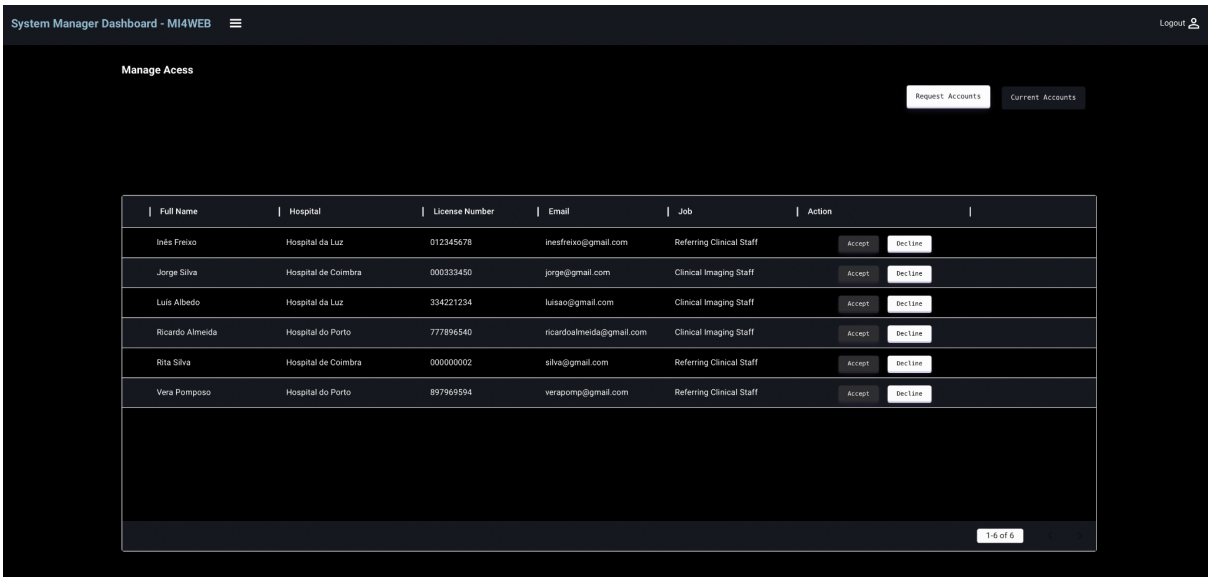


Fig 17- Accept new Users

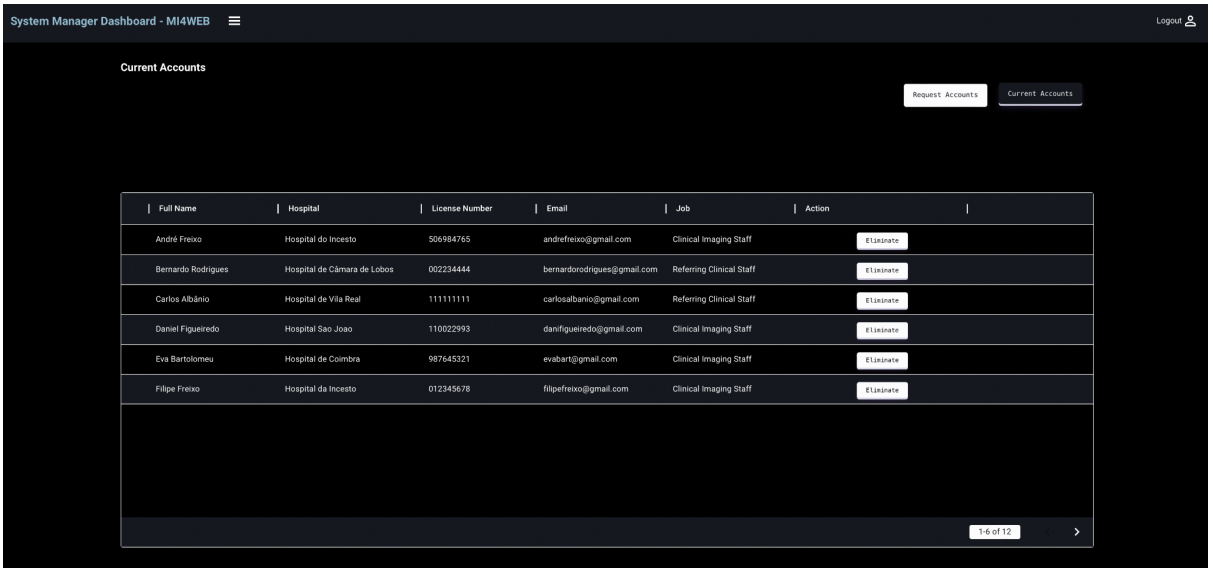


Fig 18-Eliminate Users Accounts

3.2.3 DICOM Nodes

Here, the admin can examine all the information of the DICOM nodes connected into our Network.

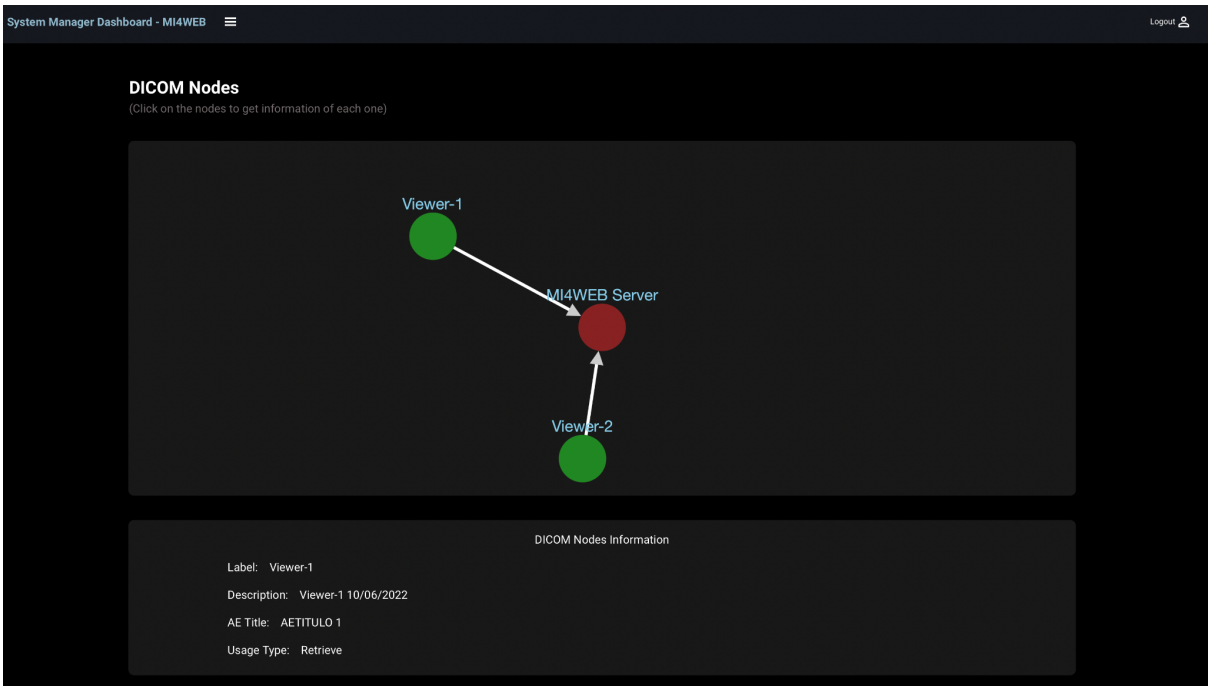


Fig 19- Admin Access Control

3.2.3 Extensions Management

In the Extensions Management the admin can see all of the extensions on the viewer and see their state(if they are active or not).

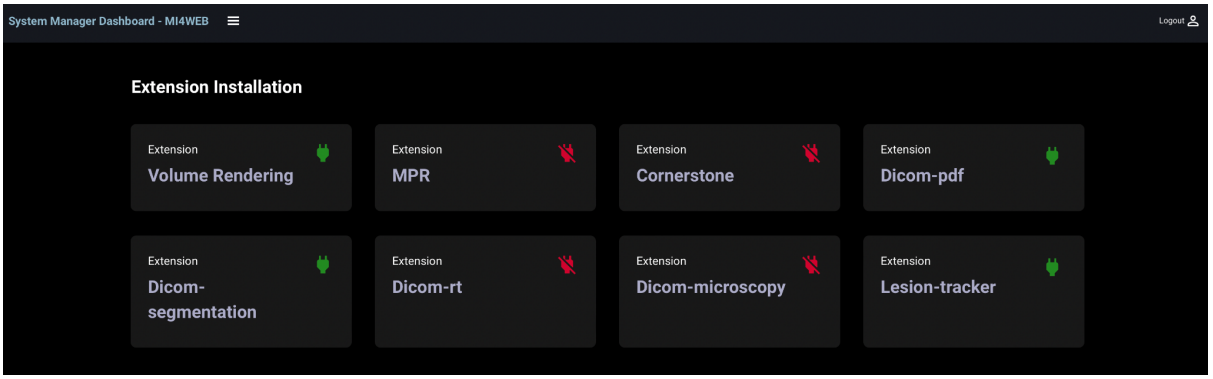


Fig 20- Admin Extensions Management

3.3 Import of DICOM images

In order to get studies of medical images available on our website, we used the medical image archive "The Cancer Imaging Archive (TCIA)" https://www.cancerimagingarchive.ne to have access to the studies. Then, using the NBIA data retriever application we downloaded the studies, consequently we used ORTHANC to upload the studies to the server. The last step was to use a configuration file so that the OHIF platform can fetch the medical images from the server.

In case the user wants to upload studies to the OHIF viewer, all he has to do is go to the ORTHANC interface and upload them, in our case the Orthanc interface can be accessed in http://mednat.ieeta.pt:8765

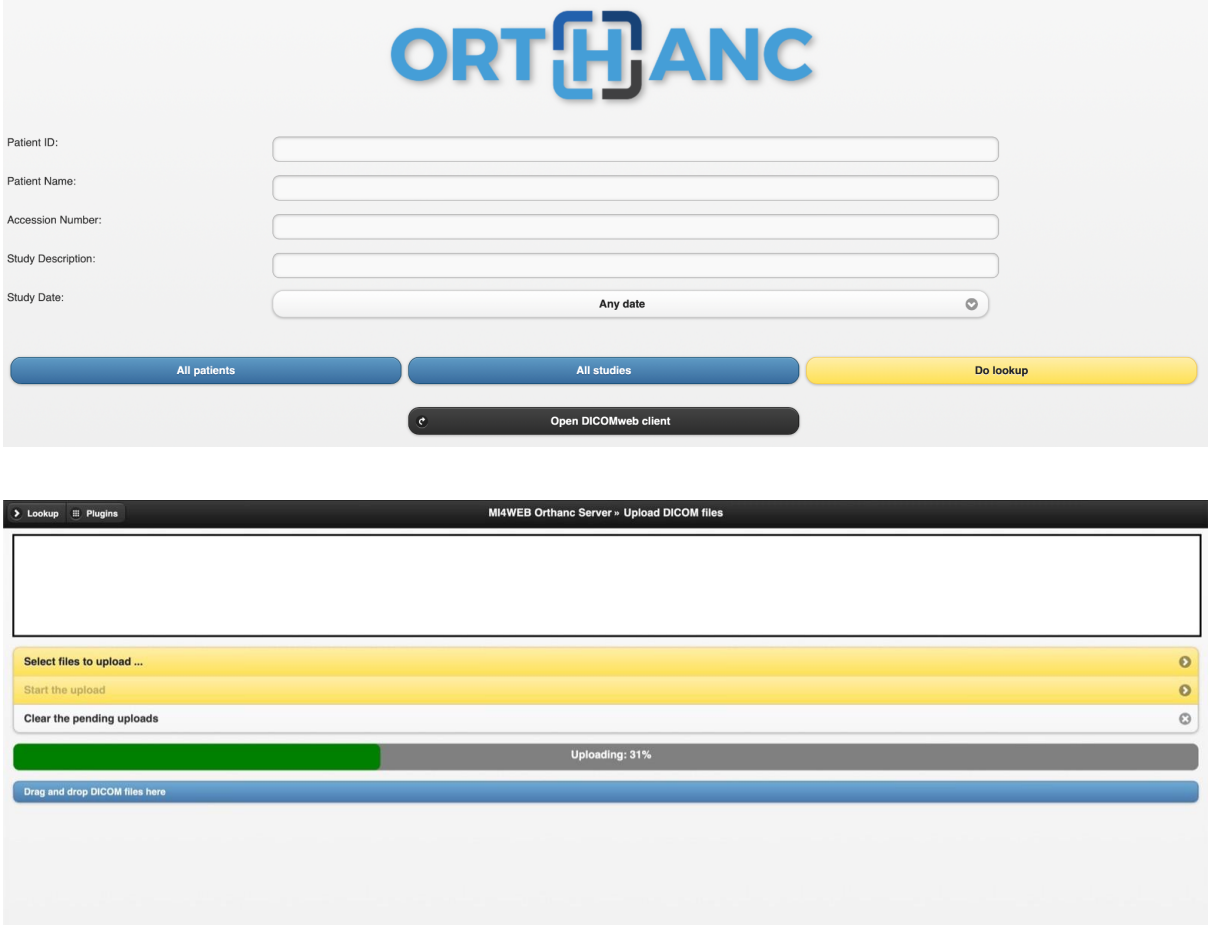


Fig 21- Upload Studies to the ORTHANC Server

3.4 Measurements Storage

In our viewer it is possible to make annotations and measurements in the DICOM images, this is only if the user has the permissions of a Clinical Imaging Staff. In case a Referring Imaging Staff or a Guest tries to save a measurement on our system an error message will be displayed.

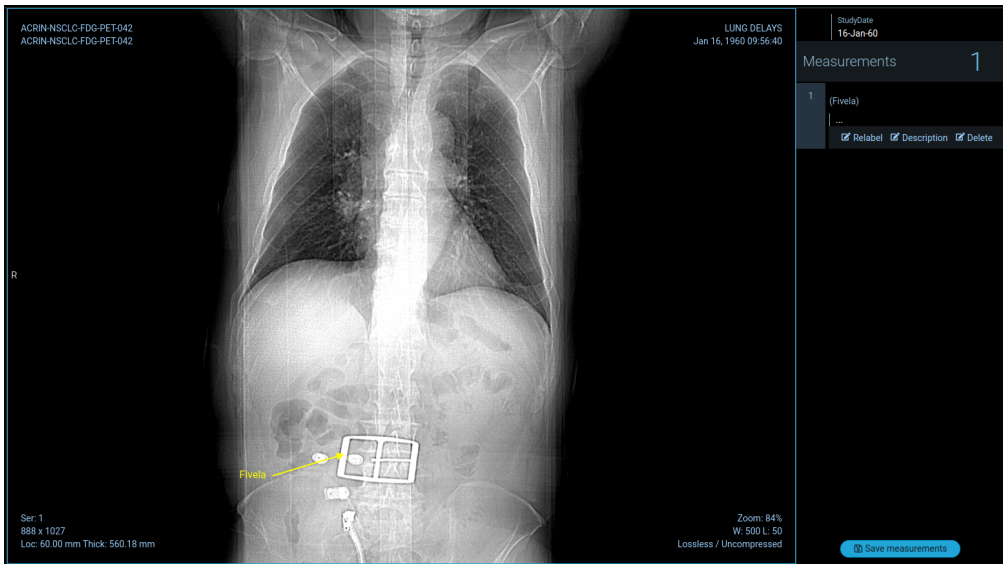


Fig 22- Annotation on DICOM image

As we can see, in the figure 22, there is an annotation on the DICOM image, that is displayed in the measurements tab. If the user tries to store the measurement and a server error occurs (for example), an error message will appear (Fig. 23).

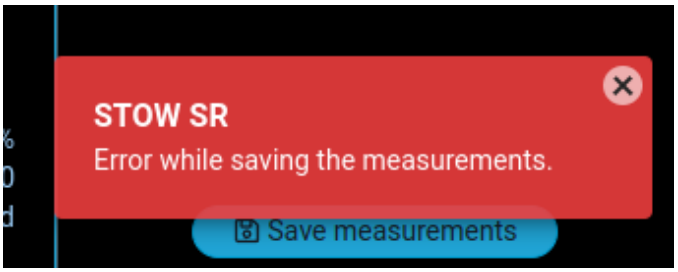


Fig 23- Server error

In case of a user that tries to store a measurement and does not have the necessary permissions, another error message will appear.

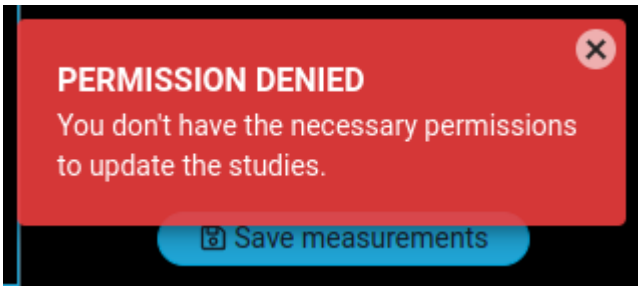


Fig 24- Permission denied error

In case of a successful measurement storage (Clinical Imaging Staff made the request), a success message will appear (Fig. 25).

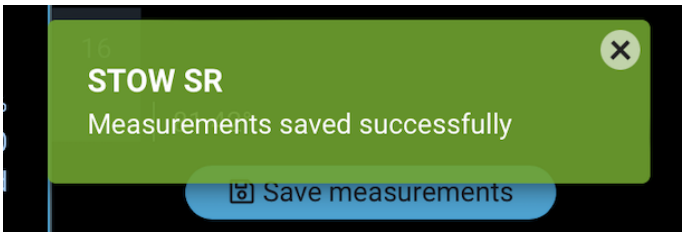


Fig 25- Successful measurement storage

3.5 Image reformatting

In our project, image reformatting is made using the VTK extension that we built, which possesses a set of functions that return objects to perform specific tasks and functionalities.

To make this extension, firstly we track some Github issues about this subject and we find some information that we find useful [22].

To create this extension we started by creating a command for MPR in the VTK extension commandsModule and then we created a button (MPR) that calls the MPR command.

Note that 2D MPR mode is only available for series that form a consistent volume (Fig. 26) (that is, all pixel spacings equal, slices equally spaced, and so on, which means you will not see the MPR button for such series as time-resolved MRI, as an example (Fig. 27)), this also happens with the 3D mode.

Fig 26- Toolbar with 2D MPR and 3D mode.

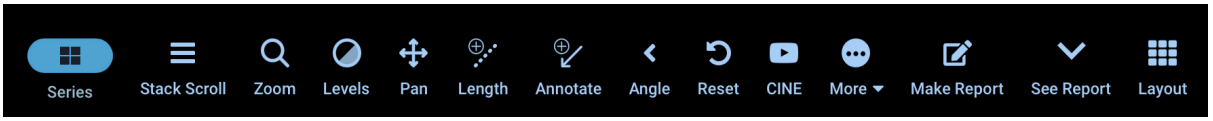


Fig 27- Toolbar without 2D MPR and 3D mode.

When the MPR button is clicked it will change the toolbar with the features that the MPR command has implemented (Fig. 28).

Fig 28- MPR features

The “Crosshairs” button is supposed to be activated when we click on the MPR button, and with this button we can move the planes on each image that is displayed, being able to synchronize the three different views (Fig. 29).

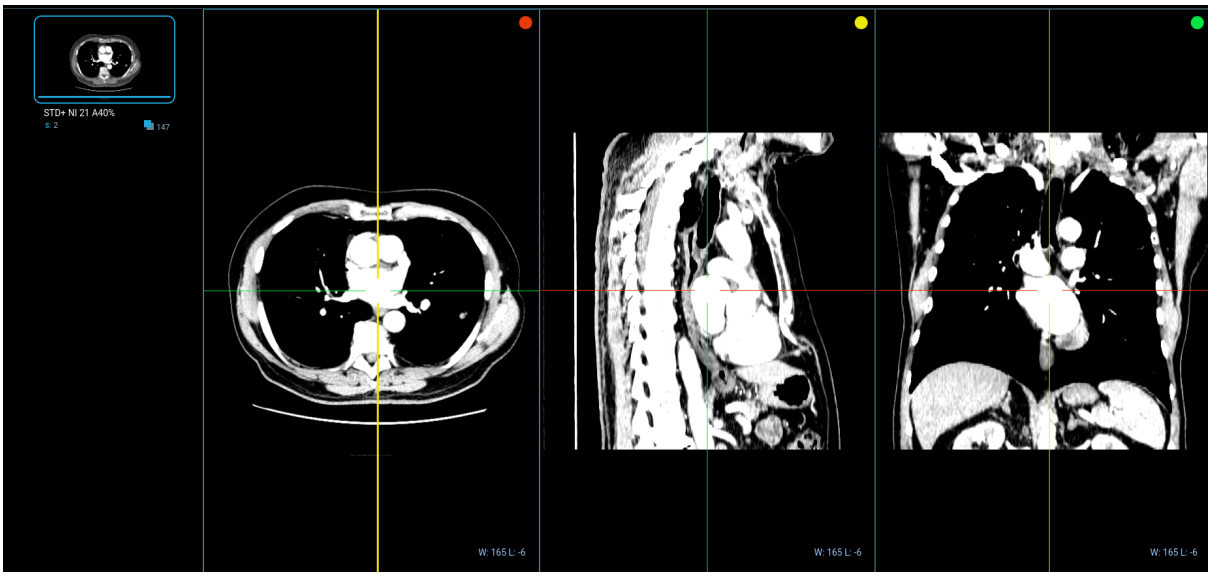


Fig 29- MPR Viewport

In the case of the “WWWC” button, this makes the image receive more or less light intensity.

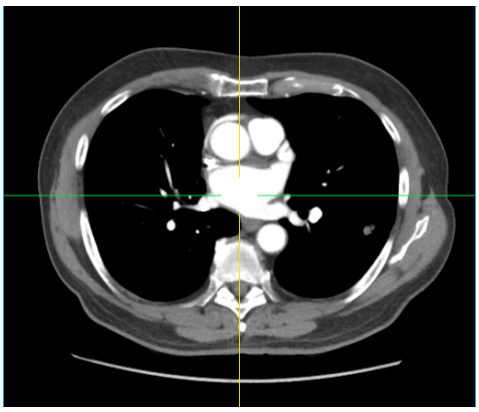


Fig 30- WWWC not applied

Fig 31- WWWC applied

In the case of the “Slab Thickness” and the select that contains the text “MIP”, this feature is supposed to work as a method that allows to find all hyperdense structures in a certain volume, in the case of MIP, and a method that enables detection of low-density structures in a given volume, in the case of the MinIP.

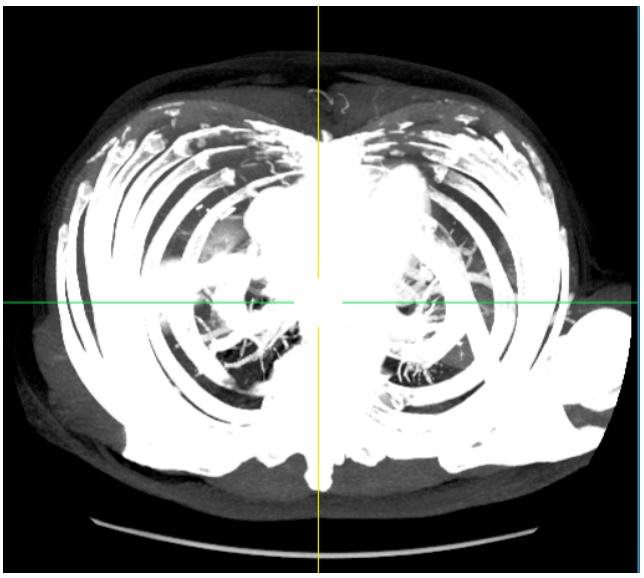


Fig 32- MIP

In this extension we create a viewport component, whose name is OHIFVTKViewport. The OHIFVTKViewport is a light wrapper around the react-vtkjs-viewport.

In the OHIF Viewer, volumes are represented by displaySets, which are collections of 2D frames fetched and decoded from a PACS or other backend source by cornerstoneWADOImageLoader [23]. These frames are decompressed in parallel using web workers.

When a displaySet is opened in a VTK viewport these 2D images are progressively loaded into a vtkDataArray, cached, and displayed in the viewport.

During loading appropriate scaling functions are applied as provided by the DICOM headers (e.g. Hounsfield Units for CT or Standard Uptake Value for PT).

The OHIFVTKViewport just exposes an API for interacting with the react-vtkjs-viewport state via OHIF commands.

In terms of the implementation of MPR command, we started by creating the viewport space, so it was possible to have three different views (axial, sagittal and coronal).

```
const viewportProps = [
  {
    //Axial
    orientation: {
      sliceNormal: [0, 0, 1],
      viewUp: [0, -1, 0],
    },
  },
  {
    // Sagittal
    orientation: {
      sliceNormal: [1, 0, 0],
      viewUp: [0, 0, 1],
    },
  },
  {
    // Coronal
    orientation: {
      sliceNormal: [0, 1, 0],
      viewUp: [0, 0, 1],
    },
  },
];
```

Fig 33- MPR Viewport Properties

Next up, we call a function called setMPRLayout with the viewportProps as one of the arguments, and what this function does is that it will display the available space in three different views.

To implement this feature we use the View2D component of the react-vtkjs-viewport library. With this component we achieve both orthogonal and oblique MPR by putting the volume in the shader on the GPU and moving the camera through the volume using a small clipping range.

We can achieve MIP by increasing the slab thickness being sampled and rendering each pixel with the maximum value in the line of sight.

Then, when the viewport is successfully activated, the next step was to implement the crosshairs so it was possible to implement the synchronization between the different views. In the development of crosshairs we used an SVG crosshair widget from react-vtkjs-viewport.

We made a wwwc interaction style for window/level, also through the react-vtkjs-viewport library.

The Viewer uses VTK.js for multi-planar reformatting. We use thin-slice volume rendering to implement MPR in the OHIF Viewer. In order to support PET/CT image fusion with overlaid segmentation label maps, we need to implement multi-volume rendering in VTK.js.

3.6 Create 3D views

We decided to implement 3D within the VTK extension, together with MPR, since it facilitated the creation of new toolbars, new commands and also because the MPR extension only appears in the toolbar for images that support this type of functionality, so this condition too it would be easier to implement in the 3D features because it is already done in MPR.

So we start by creating a command for 3D in the VTK extension commandsModule settings.

Then we create a 3D button on the VTK extension toolbar, and a button with a dropdown of various transfer functions, we will discuss these functions in more detail later (Fig. 34).

Fig 34- 3D toolbar.

Next, we define the action associated with the 3D command, this is where we use volume rendering to produce the 3D views of the volume. This with the help of the View3D component of the react-vtkjs-viewport library.

First, we created a promise from all instances of the study and series in question.

We create an actor and a mapper for the respective imageData, apply the CT-Bones transfer function to the image and finally render it (Fig 35) [3].

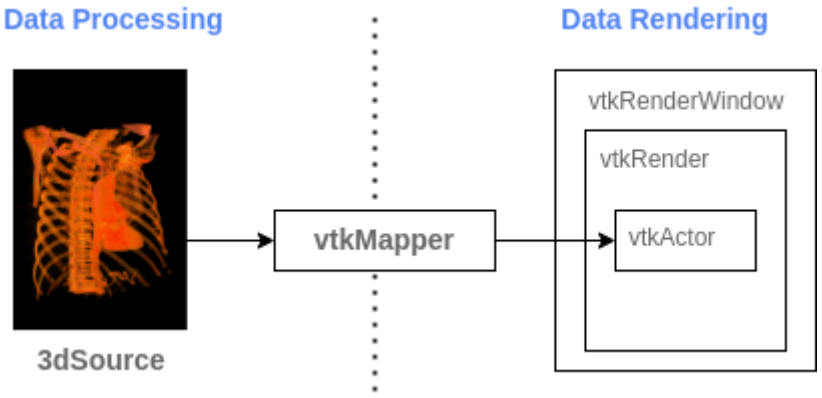


Fig 35- 3D rendering pipeline.

The 3D volume is made up of values whose name is called voxel. The Color transfer function determines which voxels map to which color, while the opacity function determines which voxels are transparent (Fig. 36).

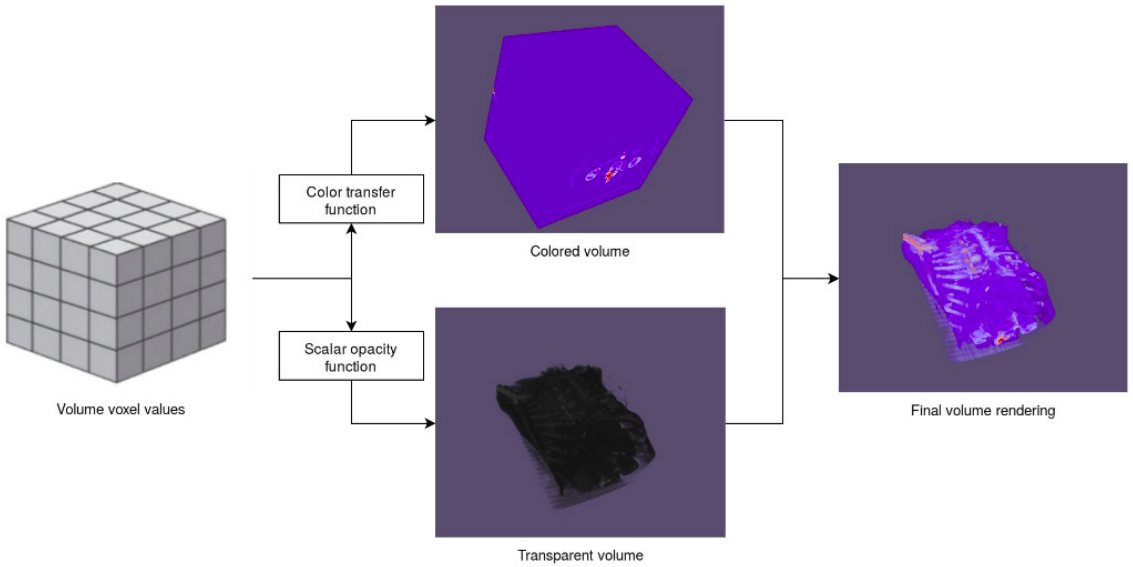


Fig 36- Volume Rendering: Opacity and Color.

In order to control the Color transfer functions of the 3D volume, we created a toolbar dropdown with various functions, as mentioned before.

The transfer functions that we defined as well as their characteristics (opacity gradient, shadow,...) are available in the presets array.

```
{
  name: 'CT-AAA',
  gradientOpacity: '4 0 1 255 1',
  specularPower: '10',
  scalarOpacity:
    '12 -3024 0 143.556 0 166.222 0.686275 214.389 0.696078 419.736 0.833333 3071 0.803922',
  id: 'vtkMRMLVolumePropertyNode1',
  specular: '0.2',
  shade: '1',
  ambient: '0.1',
  colorTransfer:
    '24 -3024 0 0 0 143.556 0.615686 0.356863 0.184314 166.222 0.882353 0.603922 0.290196 214.389 1 1 1 419.736 1 0.937033 0.954531 3071 0.827451 0.658824 1',
  selectable: 'true',
  diffuse: '0.9',
  interpolation: '1',
  effectiveRange: '143.556 419.736',
},
```

Fig 37 - Example of a transfer function (CT-AAA).

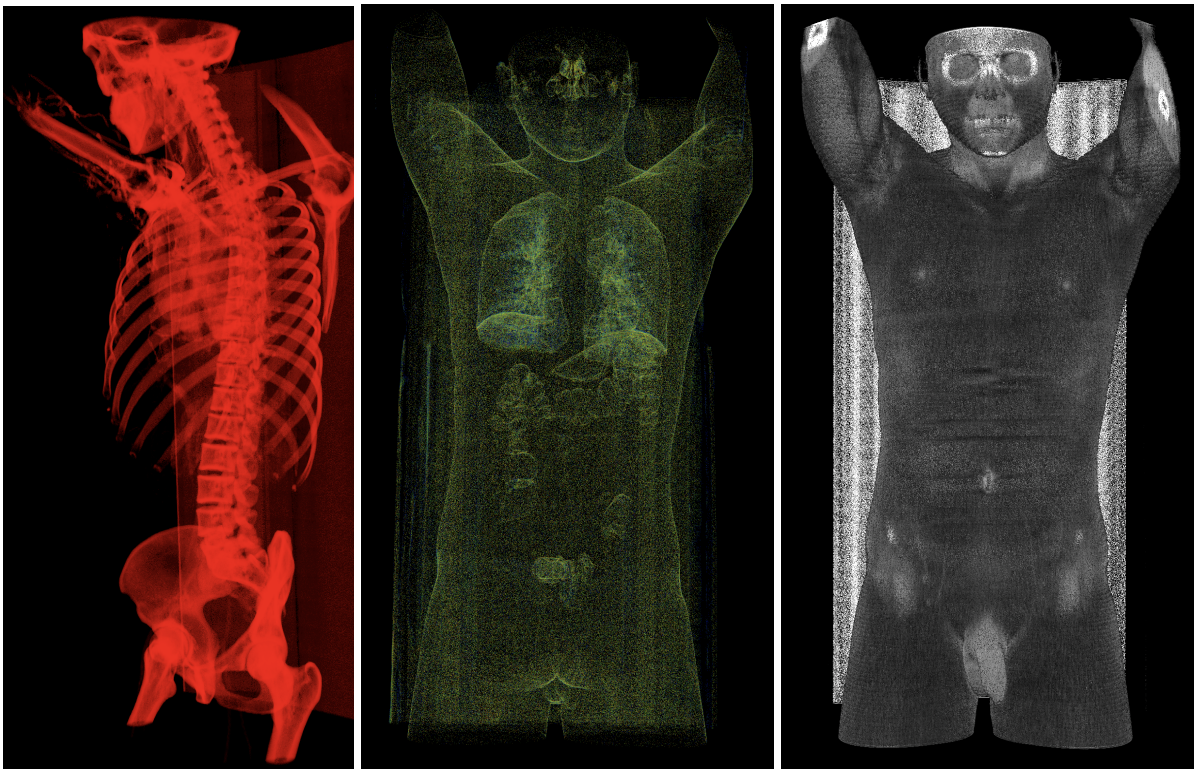


Fig 38 - Example of a series in 3 different transfer functions (CT-Bones, CT-Lung, CT-Soft-Tissues).

We also created a widget to iteratively edit the 3D volume opacity function, through multiple Gaussian functions [4].

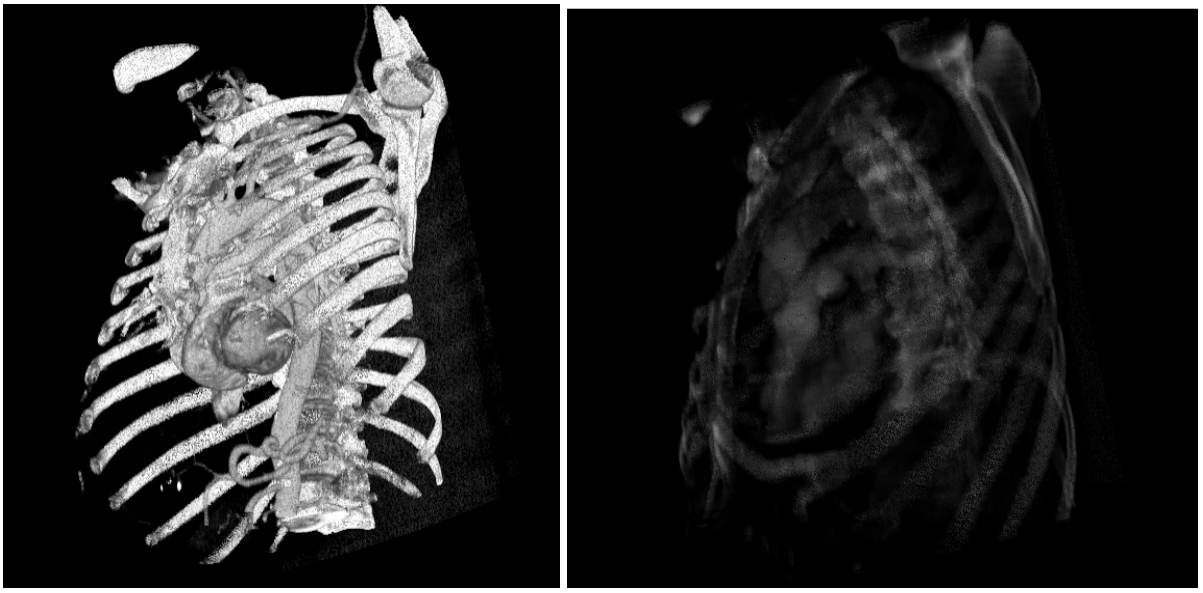


Fig 39 - Example of a series with different opacity functions.

3.7 Creation of Reports

The Clinical Imaging Staff can write reports about the medical images studies, this functionality is only available for them. If the Referring Imaging Staff presses the icon to write a report it receives a message saying that he doesn't have the permissions to perform that action. The interface for this functionality has the fields for the user to fill, and using them it creates a structured medical report in pdf format.

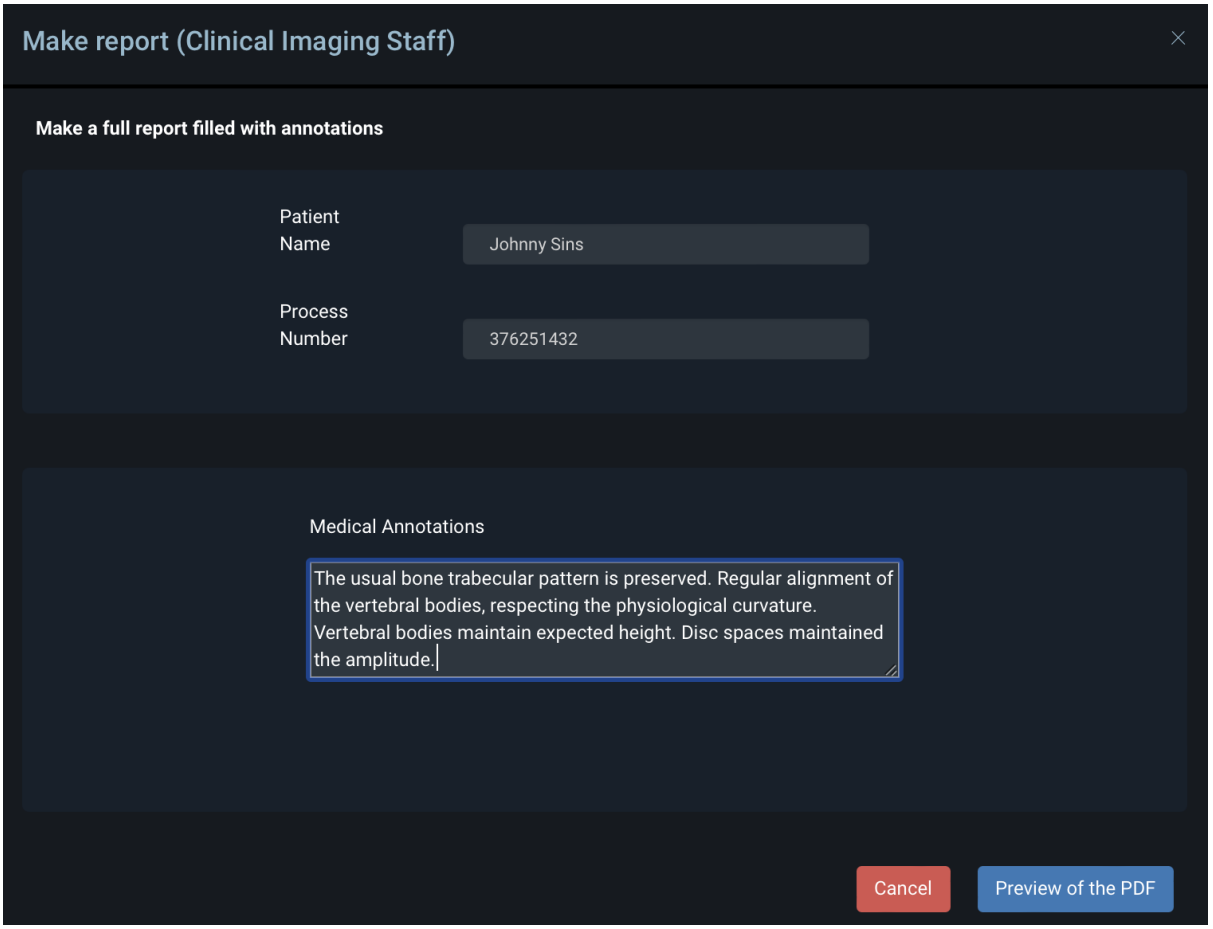


Fig 40- Write the report information

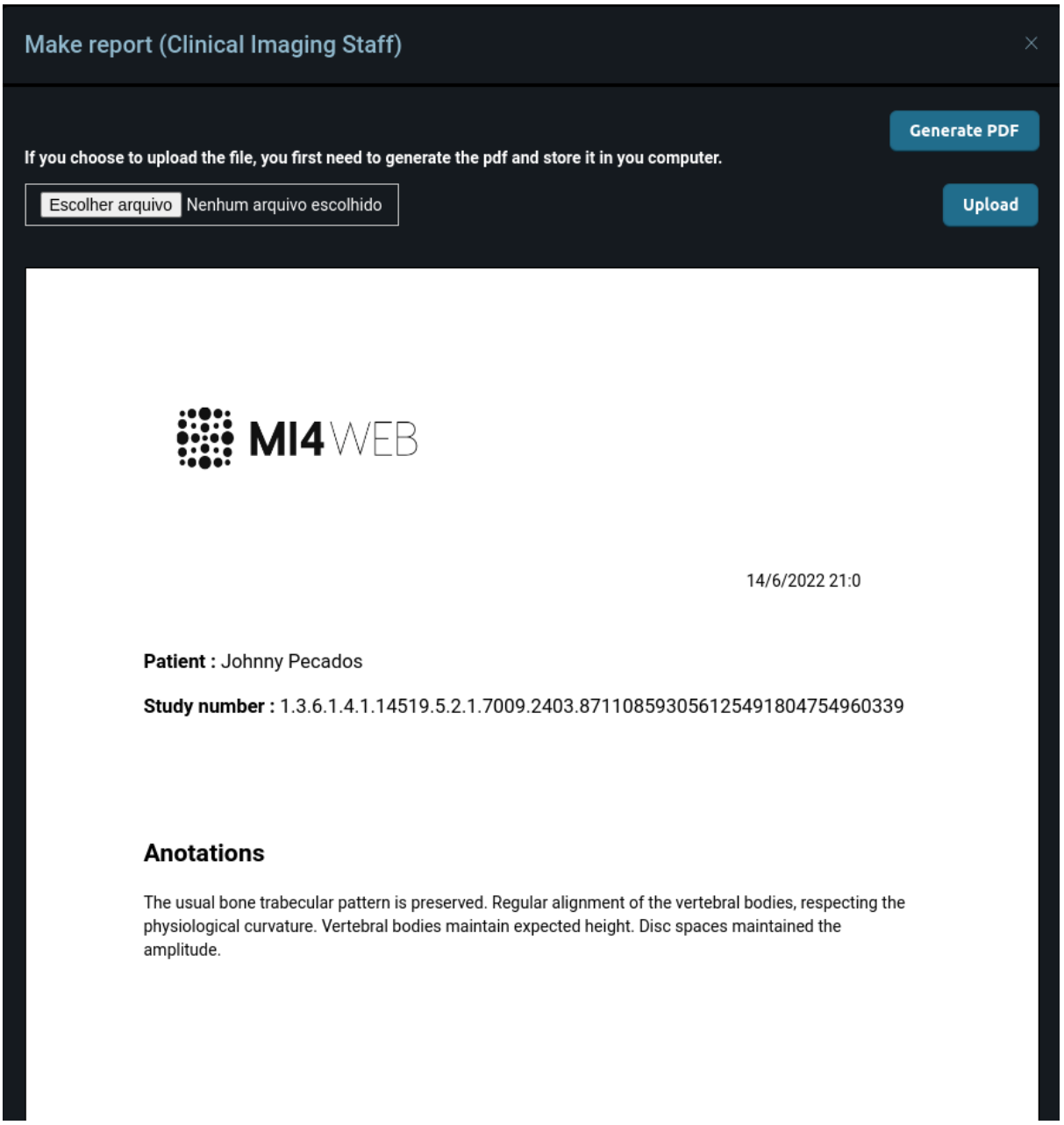


Fig 41- Generated report

In case of the Clinical Imaging Staff clicking on the “Generate PDF” button, this will trigger the download of the pdf file. If the Clinical Imaging Staff needs to send the generated pdf to another doctor, it will be easier for him to just upload this pdf in our system using the “Upload” button. If this button is clicked the report is saved in a database linked to the id of the DICOM study [2], subsequently the reports can be viewed by the Referring Imaging Staff. Using this way, it is more simple for the Referring Imaging Staff to see the files associated with this study and download them.

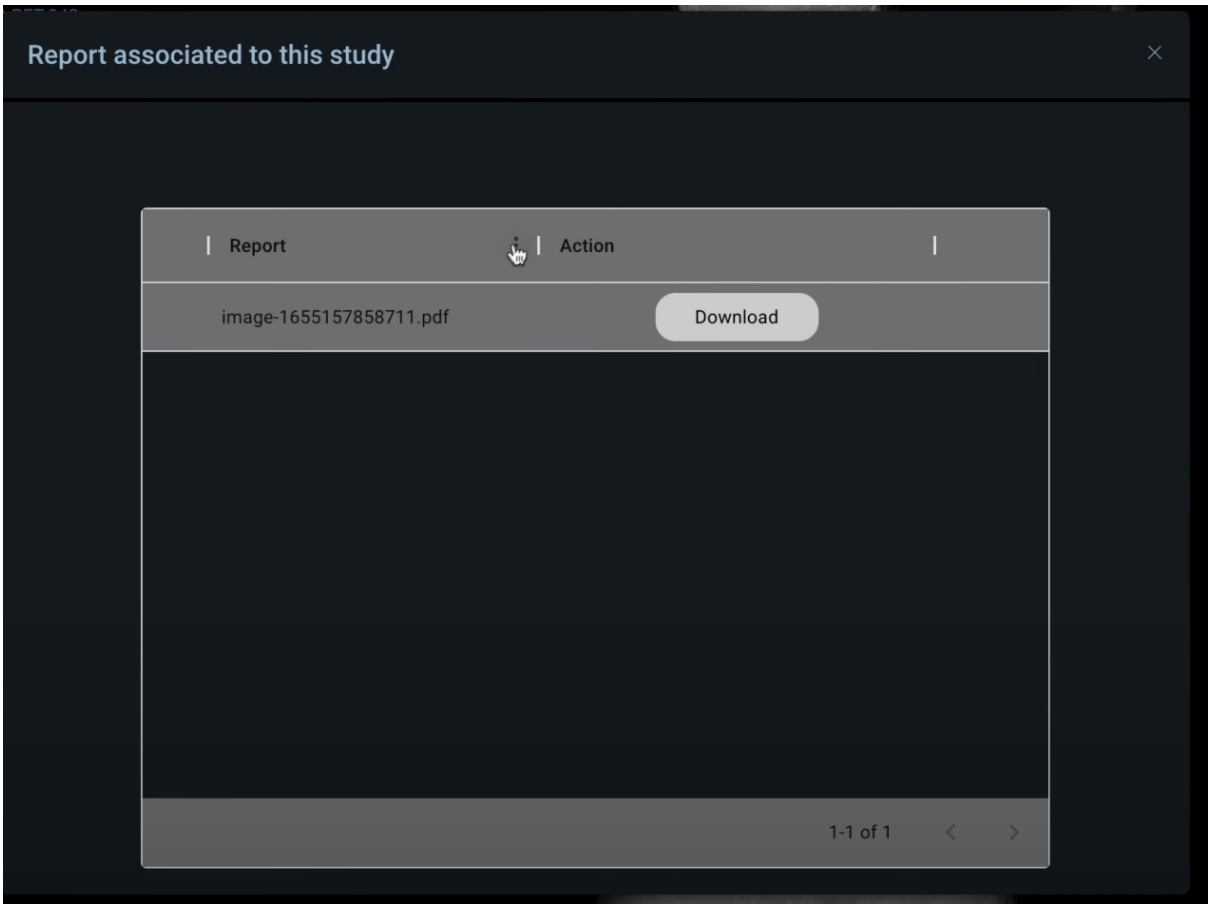


Fig 42- See reports of the associated study

4. Results and discussion

4.1 Usability tests with students

4.1.1 Summary

In order to verify if it's easy for the users to use our application we performed a set of usability tests with the help of some colleagues and family members. They were asked to perform some tasks which gave us some idea on how others understand how they can accomplish those tasks.

4.1.2 Methodology

We started by giving an introduction about our project to the users, followed by us asking them to perform the tasks chosen by usa dn, in case of not knowing what to do, tak to a member of our group.
The tasks that we choose are the following:

- Use mpr tool to visualize different views of medical images and rotate each one of them.
- Use mip to see hyperdense structures.
- Upload a report that was made.
- See and manipulate 3D images.
- Find information about our server in the admin.

4.1.3 Results

<u>Use mpr tool to visualize different views of medical images and rotate each one of them.</u>

Was the task completed successfully?

Was the task completed successfully?	Answer
YES	100%
NO	0%

Did you require help to complete the task?

Did you require help to complete the task?	Answer
YES	60%
NO	40%

Difficulty of the task(1-5)

Difficulty of the task	Answer
1	20%
2	20%
3	20%
4	40%
5	0%

Use mip to see hyperdense structures.

Was the task completed successfully?

Was the task completed successfully?	Answer
YES	100%
NO	0%

Did you require help to complete the task?

Did you require help to complete the task?	Answer
YES	40%
NO	60%

Difficulty of the task(1-5)

Difficulty of the task	Answer
1	0%
2	0%
3	20%
4	40%
5	40%

Upload a report that was made.

Was the task completed successfully?

Was the task completed successfully?	Answer
YES	100%
NO	0%

Did you require help to complete the task?

Did you require help to complete the task?	Answer
YES	80%
NO	20%

Difficulty of the task(1-5)

Difficulty of the task	Answer
1	0%
2	60%
3	20%
4	0%
5	20%

See and manipulate 3D images.

Was the task completed successfully?

Was the task completed successfully?	Answer
YES	100%
NO	0%

Did you require help to complete the task?

Did you require help to complete the task?	Answer
YES	0%
NO	100%

Difficulty of the task(1-5)

Difficulty of the task	Answer
1	40%
2	20%
3	40%
4	0%
5	0%

Find informations about our server in the admin pages.

Was the task completed successfully?

Was the task completed successfully?	Answer
YES	100%
NO	0%

Did you require help to complete the task?

Did you require help to complete the task?	Answer
YES	40%
NO	60%

Difficulty of the task(1-5)

Difficulty of the task	Answer

1	0%
2	40%
3	40%
4	20%
5	0%

5. Future Work

- Implement transfer functions on 2D views.
- On the make report feature, if the doctor needs, put an option to add fields.
- On the see report feature, being able to preview the file instead of downloading it right away.
- Escalate the ORTHANC server.
- Clipping.

6. Conclusion

In conclusion, when we started working on this project our main goals were to enrich the OHIF platform, add new features and interfaces in order to help healthcare professionals.

Looking at the final product, the teams and our coordinators are satisfied with our work, since our main goals were completed and all the main features are working.

While we were building our website our main adversity was to create 3D views, since we had never worked with *VTK* before.

Overall, all the members of the group think this project was very important because of the amount of new knowledge we gained and the new technologies we learned or perfect, such as *reactjs*, *docker*, *Vtk*, Cornerstone, etc…

7. References

[1] “OHIF Documentation”, <https://docs.ohif.org/>
[2] “Upload Image In MySQL DB using Node js Express + Multer”, Tuts Make, <http://www.tutsmake.com/upload-image-in-mysql-db-using-node-js-express-multer>
[3] “Volume Rendering”, Kitware Inc., <https://react-vtkjs-viewport.netlify.app/volume-rendering>
[4] “PiecewiseGaussianWidget”, Kitware Inc., <https://kitware.github.io/vtk-js/examples/PiecewiseGaussianWidget.html>
[5] “PostgreSQL vs. MySQL: what you need to know”, Fivetran Inc., <https://www.fivetran.com/blog/postgresql-vs-mysql>
[6] “Mysql - the world’s most popular open source database.”, Oracle <https://www.mysql.com/>
[7] “Orthanc in a nutshell”, Sébastien Jodogne, Department of Medical Physics, University Hospital of Liège, Osimis S.A, ICTEAM UCLouvain, <https://www.orthanc-server.com/static.php?page=about>
[8] “React - A JavaScript library for building user interfaces”, Meta Platforms, Inc., <https://reactjs.org/>
[9] “Extensible web imaging platform for oncology“, Open Health Imaging Foundation, <https://ohif.org/>
[10] “**DICOMweb™**”, The Medical Imaging Technology Association (MITA), a division of NEMA, serves as the DICOM Secretariat., <https://www.dicomstandard.org/using/dicomweb>
[11] “VTK Overview”, Ken Martin, Will Schroeder, <https://vtk.org/about>
[12] “Cornerstone-Overview”, Open Health Imaging Foundation, <https://www.cornerstonejs.org/docs/getting-started/overview/>
[13] “8 Advantages of Using MySQL”, Techstrong Group, <https://devops.com/8-advantages-using-mysql/>
[14] [Forrest Li, Stephen Aylward] (Kitware), Steve Pieper (Isomics), [Trinity Urban, Gordon J. Harris] (MGH and OHIF), [Erik Ziegler, James A. Petts, Danny Brown] (OHIF), OHIF + VTK.js [PowerPoint slides].
[15] [Forrest Li, Stephen Aylward] (Kitware), Steve Pieper (Isomics), [Trinity Urban, Gordon J. Harris] (MGH and OHIF), [Erik Ziegler, James A. Petts, Danny Brown] (OHIF), vtk.js Architecture and Tooling [PowerPoint slides].
[16] [Forrest Li, Stephen Aylward] (Kitware), Steve Pieper (Isomics), [Trinity Urban, Gordon J. Harris] (MGH and OHIF), [Erik Ziegler, James A. Petts, Danny Brown] (OHIF), Developing with vtk.js [PowerPoint slides].
[17] [Forrest Li, Stephen Aylward] (Kitware), Steve Pieper (Isomics), [Trinity Urban, Gordon J. Harris] (MGH and OHIF), [Erik Ziegler, James A. Petts, Danny Brown] (OHIF), Introduction to VTK.js for OHIF [PowerPoint slides].
[18] RSI @ UA –2014/15, DICOM COMMUNICATIONS [PowerPoint slides].
[19] RSI @ UA, Carlos Costa, Augusto Silva, DICOM Standard [PowerPoint slides].

[20] KEITH J. DREYER, DO, PHDDAVID S. HIRSCHORN, MDJAMES H. THRALL, MDAMIT MEHTA, MD (2006). PACS: A Guide to the Digital Revolution, SECOND EDITION, Springer.

[21] “react-vtkjs-viewport”, OHIF, <https://github.com/OHIF/react-vtkjs-viewport>

[22] “VTK Extension: MPR mode should have free rotate option #563”, GitHub, Inc., <https://github.com/OHIF/Viewers/issues/563>

[23] “cornerstoneWADOImageLoader”, cornerstonejs, <https://github.com/cornerstonejs/cornerstoneWADOImageLoader>