

Documentation & Project Diary

Innovation Lab 2

Year 2023

Project: **OpenPACS**

Team: **07**

# General Information

**Project name:** < *OpenPACS >*

**Supervisor:** < *<DI Dr. techn. Matthias Blaickner,* [*matthias.blaickner@technikum-wien.at*](mailto:matthias.blaickner@technikum-wien.at) *>*

Innovation Lab < *2, summer term 2022/23 >*

## Projectteam:

Florea Gheorghe-Casian,[if22x003@technikum-wien.at](mailto:if22x003@technikum-wien.at), Adam Alexandru-Mircea,[if22x001@technikum-wien.at,](mailto:if22x001@technikum-wien.at) Reynaud Cade, [if21b252@technikum-wien.at,](mailto:if21b252@technikum-wien.at) Islamovic Armin, [if22b144@technikum-wien.at](mailto:if22b144@technikum-wien.at)

## Management Summary of the Project

* *Implementing a more efficient external viewer with a better UI(Weasis/Ohif)*
* *Improving the security and privacy of the patient files*
* *Allowing the patient to have a more varied folder(different data types)*
* *Making the interaction with the data more accessible (annotations)*
* *Implementing a cloud storage system to the already existing database for better control over the patient files*

## Framework Conditions and Project Environment

In this part of the project our team will deal with the necessary improvements of the basis we created last semester. For this we will still need to use the Storage SCU Emulator for standard file implementation into the dcm4chee-arc-light interface but to also add on top of it another external viewer for ex. Ohif or Weasis. Additionally for better performance and security our team will have to improve the performance and security of the database by enabling some sort of cloud storage system with the help of Dropbox/ICloud/Google Storage etc. while trying to keep the costs at a minimum and storage capacity at its maximum.

## Semester-Roadmap

*The first step in achieving what we planned for this semester is o gather as much information and to do as much research into the topic as we can. Here each member of the team will do individual documentation work so that we are all on the same page when stage 2 will begin. Moving on to the second step which is the implementation of the external viewer for the database. In this phase of the project our team will have to create a way for the viewer to bind to the already existing database and to be able to view, edit and save files. The 3rd step of the project will be about working towards a cloud storage system for the files which will improve the security and efficiency of the database. The last step will be in trying to bind all the steps together, meaning that the files viewed and edited in the external viewer to get also stored in the cloud storage system.*

## Collaboration & Tooling

[DCM4CHE(https://www](http://www.dcm4che.org/)).dcm4che.org/)

DCM4CHE is a set of free and open source tools and programs for healthcare. For performance and portability, these applications have been built in the Java programming language so that they can be deployed on JDK.

[DICOM(https://www.orthanc-server.com/download.php)](http://www.orthanc-server.com/download.php))

The industry standard for communicating and managing medical image data and associated data is referred to as Digital Imaging and Communications in Medicine (DICOM). Most commonly used for the storage and transmission of medical images, DICOM enables the integration of medical imaging equipment from many manufacturers, including scanners, servers, workstations, printers, network devices, and PACS. DICOM is widely used in hospitals and is also successfully deployed in smaller facilities such as dental offices and doctors' offices.

PACS

The PACS server, to which both a short-term and a long-term archive are connected, forms a PACS system. are connected, forming a PACS. The PACS server not only provides a connection to the associated imaging modalities, but also provides data to viewing and

post-processing systems. A connection to the Radiology Information System (RIS) is usually also provided. Larger PACS installations can consist of additional servers and archives that can be interconnected over long distances.

OHIF viewer

The OHIF (Open Health Imaging Foundation) Viewer is a free, open-source, web-based viewer for medical imaging data. It is designed to be highly customizable and easily integrated into existing healthcare workflows. The OHIF Viewer can display a wide variety of medical imaging data, including DICOM (Digital Imaging and Communications in Medicine) images, as well as non-DICOM formats such as JPEG, PNG, and TIFF. It is compatible with most modern web browsers and can be accessed from any device with an internet connection.

Weasis

Weasis is a free medical DICOM viewer that can be used as a standalone or web-based application. Weasis can be integrated with other systems that handle DICOM data, such as PACS (Picture Archiving and Communication System), RIS (Radiology Information System), HIS (Hospital Information System), or PHR (Personal Health Record). Weasis can display images from different modalities, such as X-ray, CT, MRI, ultrasound, etc. Weasis enables measuring distances, angles, areas, and volumes on the images and also enables annotating the images with text or graphics.

# Brief Description of the Project

Our project is about improving health care with the help of our ﬁeld of study, computer science. The goal is to create a working environment in which various patient examinations from the FHTW database can be viewed, edited, stored and forwarded. For this purpose, we will convert the FHTW health database into DICOM. Starting with standard information like name, age(HL7 ﬁles), to more detailed and complex information like an X-ray or ultrasound examination (DICOM ﬁles). All these ﬁles should be accessible to a specialized person individually or simultaneously in order to make the correct diagnosis. For this purpose, our team got familiar with the working environment of dcm4che, a Java-based program that processes these types of ﬁles. We connected the FHTW health database to a PACS client system to access and process the current data from the database. Additionally, we plan to add more types of ﬁles that can be viewed in the OpenPACS system, such as blood samples, which can be critical in diagnosing certain conditions. To achieve this, we will explore various ﬁle formats, such as HL7 and FHIR, and investigate the necessary steps to integrate them into the system. We also recognize the importance of cloud storage in modern healthcare systems, and we plan to incorporate cloud storage capabilities into the OpenPACS system. This will allow for easy storage, sharing, and access to patient data across multiple devices and locations, while ensuring the security and privacy of the data. Overall, our project aims to leverage the power of computer science to improve healthcare outcomes by creating a comprehensive and user-friendly platform for accessing and processing patient data. By incorporating an external viewer, expanding the types of ﬁles that can be viewed, and implementing cloud storage capabilities, we hope to create a versatile and robust system that can meet the evolving needs of healthcare professionals and patients alike.

# Specification of the Solution

Similar to last semester, our work began with a lot of research for the planning and execution of our ideas. Unlike last semester, we had encountered a lot more problems with our implementation because of faulty documentation on the web, so most of our time was spent on debugging and finding alternatives.

The viewer was a big part of our project so the implementation wasn’t going to be easy either. There were a couple of components we had to install in order to properly configure and bind the viewer to our database. First component was ‘Portainer’, a web manager for Docker containers, which allowed us to easily create new volumes, assign images and access their configuration files. When configuring the Volume that was going to store the necessary files for the viewer we had to assign a specific version of the Ohif image, the latest having some issues with viewing some files. After mapping the network port to 3000 we moved onto the actual binding of the dcm4chee database with our viewer. This took place in a specific configuration file that was found in /var/lib/docker/volumes/ohif/\_data/app-config.js

After configuring the viewer we were finally able to access it via localhost:3000

System boundaries:

* User interface: allows users to interact with the viewer and perform tasks such as loading, viewing and manipulating medical images.
* Extensibility: Ohif is designed to allow developers to customize and extend its functionality. Provides support for additional plugins, tools and many other features
* Image displaying and manipulation: Ohif provides the functionality to render and display medical images in any web browser, allowing the user to perform multiple operations on the images at the same time

Features:

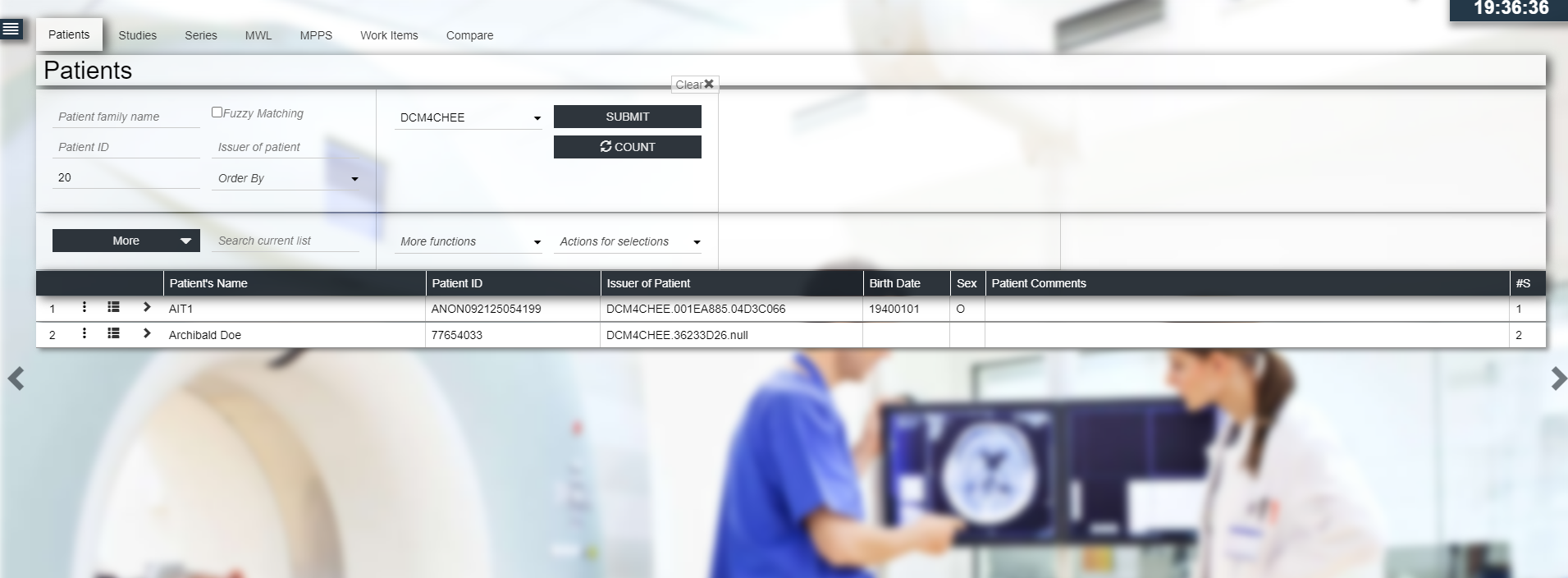
* Cross-Platform Accessibility: Ohif is a web based viewer, meaning it can be accessed from any device that has a browser and an internet connection being able to run on Desktops, phones, tablets or laptops
* 3D Visualization: Ohif supports the rendering and display of 3D images, such as volumetric data from CT or MRI scans. It enables users to explore and interact with 3D reconstructions and view different slices or projections of the data
* Multi-Modality Support: The viewer supports various medical imaging modalities, including X-ray, computed tomography(CT), magnetic resonance imaging (MRI), ultrasound and many more
* Measurement and Annotation Tools: Users are able to perform operations such as distance measurement, angle measurement, area calculations and many more. It also provides support for adding annotations in form of text, arrows, markers on specific regions of interest
* Image manipulation: Users can zoom in and out, adjusting brightness and contrast and applying different windowing settings to highlight specific anatomical details

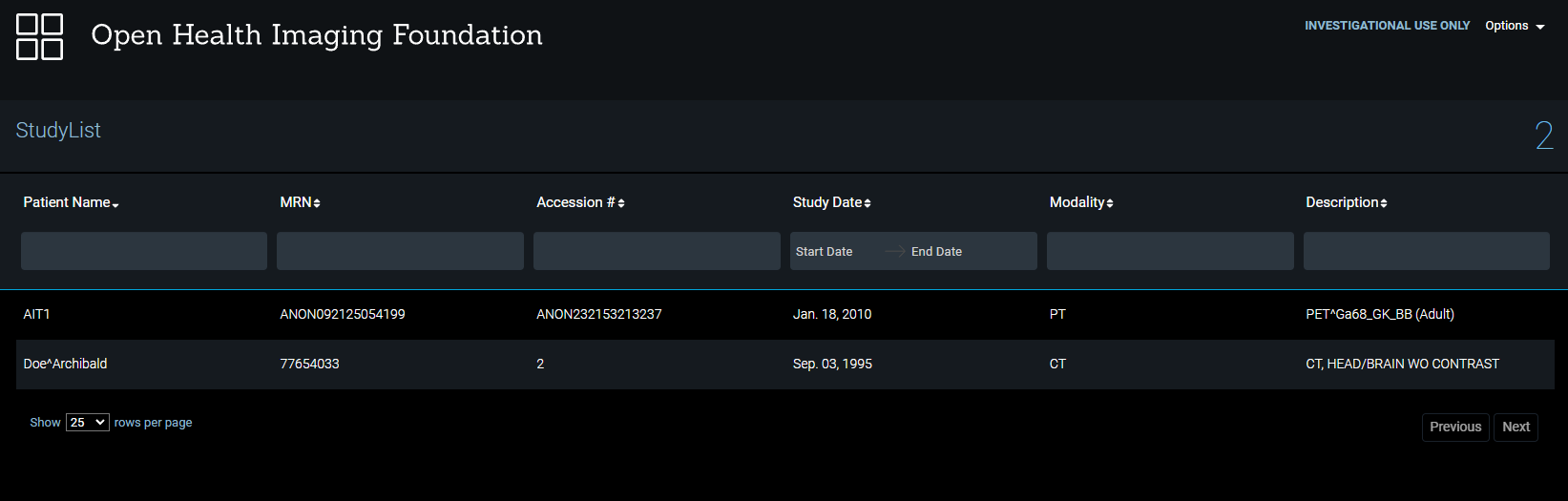
Interfaces:

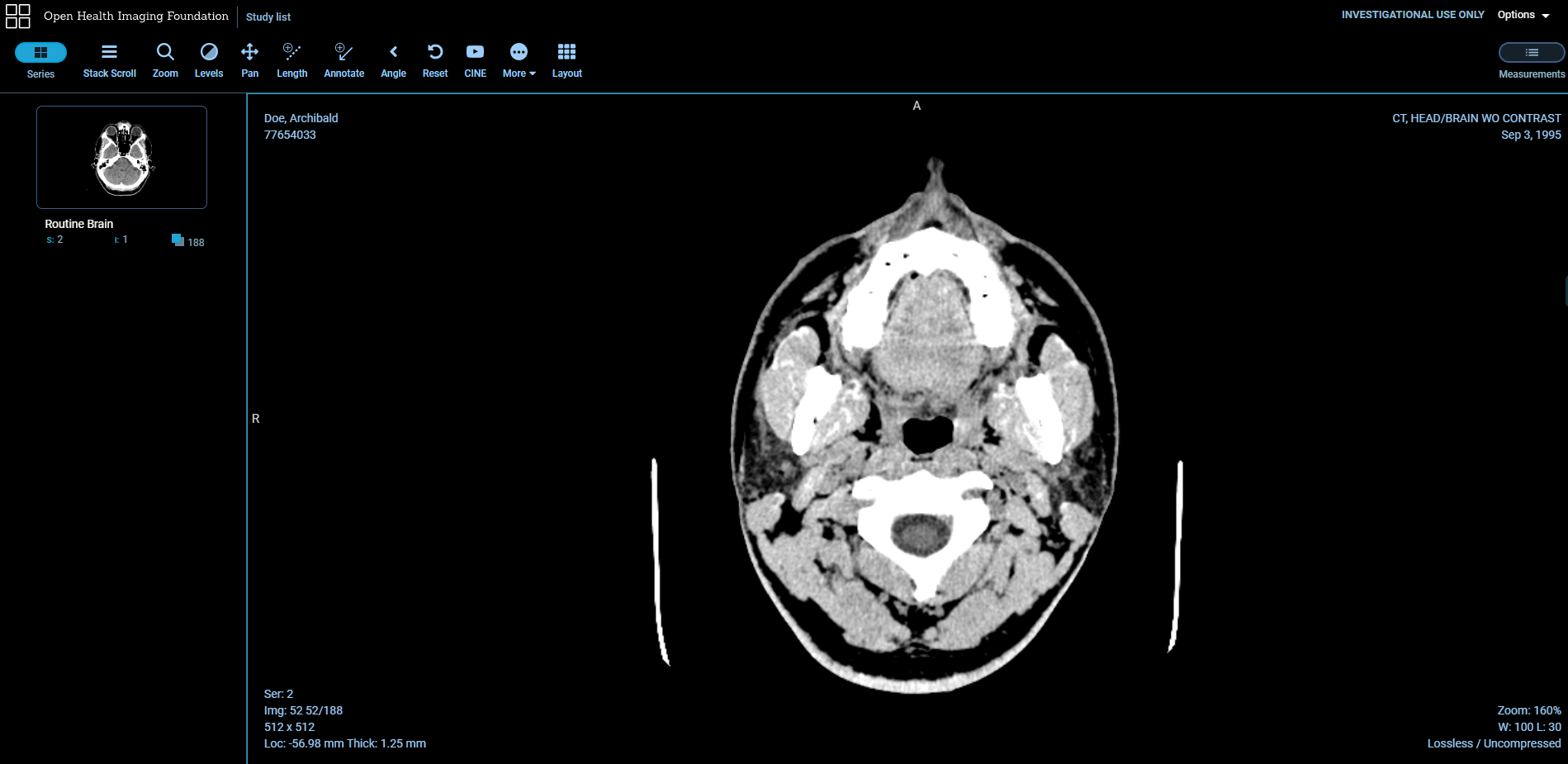
* Study list: The viewer includes a study list where you can see a panel with the patients that get imported from the dcm4chee database. A study includes the relevant information about the patient such as name, ID, study dates and other descriptions
* Customizable layout: Users can arrange panels such as image viewer, study list, annotations and measurement tools according to their workflow and viewing preferences
* Intuitive design: The interface is designed to be user friendly and intuitive, being easily understood for both healthcare professionals and software engineers
* Contextual menus: Provides additional functionality based on the user’s interactions. For example: right clicking on an image may bring up a menu with options for specific settings

Non functional requirements:

* Performance: The viewer should provide fast and responsive performance, allowing users to load and manipulate medical images efficiently. It should be able to handle large amounts of datasets and maintain smooth interactions, even with complex 3D reconstructions
* Scalability: It should be able to handle the growing demands of a healthcare environment without sacrificing performance or responsiveness
* Security: It should implement security measures such as encryption of data in transit, access controls and authentication mechanisms to protect sensitive data
* Compatibility: The viewer should be compatible with a large range of web browsers and operating systems to ensure broad accessibility
* Reliability: The viewer should be available whenever the user needs it without complications
* Extensibility: The viewer should be able to access data from an external database
* Usability: The viewer should have a user friendly interface with an easy to navigate source code for configuring different features and options







# Effort Estimation

Research work:

The research part will take place before each implementation phase, meaning that it will be an ongoing process for as long as the project takes. Using the Delphi Method, we estimated between 25-35 hours with the most probable estimation of 30 in total.

Incorporation of cloud storage:

Having to take into account all the stages of this phase which consist of: integration, planning, configuration, testing and implementation, our team considered this the longest phase of our project. This part was estimated to take between 60-90 hours but most likely 80 hours, by using the Delphi method.

Incorporation of an external viewer:

For more complex operations on the images we will need to implement an external viewer such as OHIF or Weasis. First of all we need to identify the essential features and functionalities of the external viewer. Then define the compatibility and integration requirements with the existing system. We must determine the desired user experience and interface design. Using the Delphi Method, we estimated between 30-40 hours with the most probable estimation of 35 hours in total.

# Delivery

List of components

* Docker
* DCM4CHEE DICOM Archive
* OHIF viewer
* Storage SCU Emulator

Final Solution or Solution Components Including Source Code

Our final solution includes the following primary components:

1. Docker:

We saved all of our containers and images used for the project in Docker

2. DCM4CHEE DICOM Archive:

Here are saved all of our patients and all the other types of files, like blood samples.

3. OHIF Viewer:

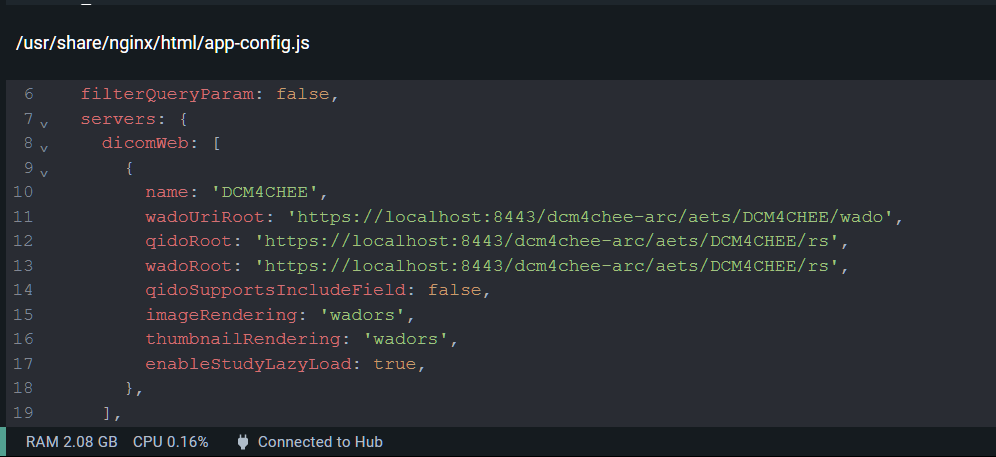
An open-source, web-based medical imaging viewer. Implemented this semester, it allows the viewing and manipulation of a wide range of image types, including DICOM.

We encountered some problems with the connection between DICOM and OHIF, but thanks to the docker update that enables us to search and modify files in Docker we managed to solve that problem. In the OHIF container following the path „/usr/share/nginx/html/” you should modify the app-config „dicomweb” section with the following:

wadoUriRoot: 'https://localhost:8443/dcm4chee-arc/aets/DCM4CHEE/wado',

qidoRoot: 'https://localhost:8443/dcm4chee-arc/aets/DCM4CHEE/rs',

wadoRoot: 'https://localhost:8443/dcm4chee-arc/aets/DCM4CHEE/rs',

like shown in the picture below:

System Architecture and Data Storage

Our solution is built on a modular, microservices architecture. Each component (Docker, DCM4CHEE DICOM Archive, OHIF Viewer, Storage SCU Emulator) is encapsulated in its own service, allowing for easy scaling and management. Data is stored in the DCM4CHEE DICOM Archive, which can handle a variety of image types and has robust support for DICOM standards.

List of Any Required Licenses and Information about Copyrights

All components of our solution are open-source. Docker is licensed under the Apache 2.0 license, DCM4CHEE and the Storage SCU Emulator are licensed under the Mozilla Public License 2.0, and the OHIF Viewer is licensed under the MIT license.

Hardware specification

Our solution should run on any modern hardware capable of supporting Docker. Specific hardware requirements will depend on the expected load and usage.

Description of the installation process

Last semester we installed DCM4CHEE and here is the explanation:

Two files in the same folder:

- docker-compose.env: STORAGE\_DIR=/storage/fs1 POSTGRES\_DB=pacsdb POSTGRES\_USER=pacs POSTGRES\_PASSWORD=pacs WILDFLY\_ADMIN\_USER=admin WILDFLY\_ADMIN\_PASSWORD=admin

IID\_PATIENT\_URL=../../weasis-pacs-connector/weasis?&patientID={}&cdb&target=\_self IID\_STUDY\_URL=../../weasis-pacs-connector/weasis?&studyUID={}&cdb&target=\_self

- docker-compose.yaml:

version: "3" services:

ldap:

image: dcm4che/slapd-dcm4chee:2.6.3-29.0 logging:

driver: json-file

options:

max-size: "10m" ports:

- "389:389"

env\_file: docker-compose.env volumes:

- /var/local/dcm4chee-arc/ldap:/var/lib/openldap/openldap-data

- /var/local/dcm4chee-arc/slapd.d:/etc/openldap/slapd.d db:

image: dcm4che/postgres-dcm4chee:14.5-29 logging:

driver: json-file options:

max-size: "10m" ports:

- "5432:5432"

env\_file: docker-compose.env volumes:

- /etc/localtime:/etc/localtime:ro

- /etc/timezone:/etc/timezone:ro

- /var/local/dcm4chee-arc/db:/var/lib/postgresql/data arc:

image: dcm4che/dcm4chee-arc-psql:5.29.0 logging:

driver: json-file options:

max-size: "10m" ports:

- "8080:8080"

- "8443:8443"

- "9990:9990"

- "9993:9993"

- "11112:11112"

- "2762:2762"

- "2575:2575"

- "12575:12575"

env\_file: docker-compose.env environment:

WILDFLY\_CHOWN: /opt/wildfly/standalone /storage WILDFLY\_WAIT\_FOR: ldap:389 db:5432

depends\_on:

- ldap

- db volumes:

- /etc/localtime:/etc/localtime:ro

- /etc/timezone:/etc/timezone:ro

- /var/local/dcm4chee-arc/wildfly:/opt/wildfly/standalone

- /var/local/dcm4chee-arc/storage:/storage

From this folder, in the Ubuntu terminal, run the command

docker-compose -p dcm4chee up -d (Docker must be installed for this).

For the installation process there were 2 options. First option was by installing everything via a command line and the second approach was with Portainer. Our team chose the Portainer approach because it was safer, where we had to firstly register an account on the official Portainer website and then add the container in our Docker from the command line. After installing the container we accessed it via localhost:9443 and created a new volume for our viewer. In that volume we attached the official Ohif image with an older version because the latest had some issues. Before deploying the container we had to create a new network map for our computers. After deploying the container, all that was left was to bind it to our dcm4chee database. In order to do this we had to navigate to a specific configuration file located in the container. At first it was hard to find it but after a Docker UI update where the feature was added to navigate the containers files directly from the Docker, we were able to access and edit the configuration file, enabling us to properly bind the viewer to our database. To test the viewer we simply navigated to the localhost port 3000. There, we were able to see every study form our database, access it and do all the operations that were supported standardly by Ohif.

# Our Project Diary

Sprint 1:

The problem we had is that we were not sure which medical viewer we wanted to implement, the choice was between the ohif and the weasis viewer. We already tried implementing them both last semester but failed so we needed to do some more research to be able to implement one of these into our project.

Sprint 2:

This was the most labor intensive sprint of this semester because thanks to a new docker update we were able to access a configuration file and edit it which allowed us to implement the ohif viewer, which was previously not possible in this way. We also discovered portainer which allowed us a simpler way of accessing and monitoring our docker images. We also found an issue with the ohif viewer where the page would black out and not show any information on most browsers but we were not able to solve this problem yet.

Sprint 3:

The big problem for this sprint is that we set a goal of uploading this project on a cloud storage, but after discussing it with the project supervisor we realized that this is not an improvement on the project but rather creates more security issues and wastes resources on unusable features.

Sprint 4:

To achieve this sprint´s goal, we researched ways of uploading text files to dcm4 databases. We did not find any, so we thought about setting up another health database, running it simultaneously with the dcm4 database and have them send information to each other. We found OpenMRS as a health database that supports text files and set it up.

Sprint 5:

This sprint made us realize that connecting the 2 databases was a lot harder than previously anticipated. We found the program HL7 Soup and thought it would allow us to easily achieve this. However, after installing and using HL7 Soup, we saw that to connect the 2 databases would take a lot more time than we thought and so was not a realistic implementation for this project. After discussing this together with our supervisor, we looked into more ways of uploading text files to the database with the idea of converting them to dcm files.

Sprint 6:

We converted pdf files to dcm files but found some new problems while trying to upload them to the database. While looking for a solution to this problem, we found out how to upload pdf files directly to the database making the conversion pointless. We also found the solution for the ohif page blanking out which was due to some security issues.

All meetings took place online, so we don't have photos together. With Whatsapp we planned a meeting and voted on important decisions. One meeting consisted of discussing a problem and trying to solve it together. When a member of our group succeeded in solving the problem, he would explain the solution so that we could all move forward at the same pace. One big difficulty we had to adjust to is the fact that there is little guidance or documentation on exactly how to work with open PACS.Another disadvantage was the implementation of the project itself. We had many failures, but in the end, we all shared the steps taken and came up with a clear plan and overcame the problem as a team. The best part of the project was that we built trust and could rely on each other to achieve the goals we set at the beginning of the semester.