Universidade de Aveiro
2020

Departamento de Eletrónica, Telecomunicações e Informática

Rui Filipe Ribeiro Jesus

Pathology PACS - A centralized web platform for pathology medical imaging management

PACS de Patologia - Uma plataforma centralizada para a gestão de imagem médica de patologia

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Mestrado de engenharia informática, realizada sob a orientação científica do Carlos Manuel Azevedo Costa, Professor auxiliar do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro

o júri

agradecimentos

Aos meus pais, por me terem suportado durante estes cinco anos, apesar de todas as dificuldades encontradas pelo caminho. Este percurso não teria sido possível sem esse apoio.

Aos meus colegas de curso e amigos, que fizeram deste percurso um percurso cheio de boas memórias que ficam para a vida.

Ao meu orientador, Professor Doutor Carlos Costa, pelas oportunidades que me deu durante a licenciatura e mestrado e pelo suporte neste trabalho e anteriores, que enriqueceram a minha experiência académica bem como a minha experiência profissional.

Ao meu orientador na BMD Software, Doutor Luís Bastião, pela possibilidade do estágio profissional e pelo suporte e apoio na realização desta dissertação.

Aos restantes colegas de trabalho na BMD Software, pelo acolhimento que facilitou a integração na empresa e pelos contributos dados neste trabalho.

A todos, um muito obrigado.

palavras-chave

Patologia digital, visualizador digital de lâminas, Dicoogle, PACS, DICOM, análise de imagem

resumo

A área de Patologia clínica digital ainda se encontra a dar os primeiros passos no desenvolvimento de soluções interoperáveis que permitam a aquisição, arquivo e visualização distribuída da imagem, incluindo ferramentas de suporte ao diagnóstico. Os atuais cenários de revisão à distância usam aplicações proprietárias que não são interoperáveis com a norma DICOM. Isto deve-se ao facto de a tecnologia não estar suficientemente madura para apoiar a prática clínica numa área em que uma imagem digitalizada pode atingir vários giga-pixels, requerendo novas soluções de engenharia para suportar grandes volumes de dados, da ordem de gigabyte por estudo, que necessitam de ser consumidos remotamente em tempo real.

Esta dissertação teve como objetivo estudar e desenvolver tecnologias e sistemas de informação que permitam a aquisição digital da imagem de patologia, o seu arquivo centralizado, a partilha e revisão colaborativa com ferramentas de suporte à decisão. O resultado é uma solução Web inovadora, de elevada produtividade, diagnóstico seguro e baseada em processos e protocolos normalizados. Um Web-browser comum foi transformado numa estação de trabalho capaz de aceder ao arquivo em qualquer altura e qualquer lugar, independentemente do sistema operativo, computador ou dispositivo móvel.

keywords

digital pathology, WSI viewer, digital slides, Dicoogle, DICOM, PACS, image analysis

abstract

The clinical area of digital Pathology is still giving its first steps in the development of interoperable solutions that enable the distributed acquisition, storage, and visualization of medical images, including diagnostic support tools. Nowadays, digital management solutions use proprietary image formats and communication protocols that are not compatible with the DICOM standard. Moreover, the available technologies are not mature enough to support the practice of medicine in an area where scanned images can reach several gigapixels, requiring new engineering approaches to support huge volumes of data, in the order of gigabytes per study, that need to be consumed in real time.

This dissertation aims to research and develop new technologies and associated information systems, capable of supporting the digital acquisition of pathology images, their centralized archive, sharing, and collaborative review with decision support tools. The result is an innovative web solution, focused on increasing productivity, with safe diagnostics and based on normalized protocols. A common web browser was transformed into a professional workstation that is able to access the image repository at any place and time, regardless of the operating system and without any prior installation.

Index

Figures Index

Table Index

List of acronyms

AMD	**A**synchronous **M**odule **D**efinition
API	**A**pplication **P**rogramming **I**nterface
CAD	**C**omputer **A**ssisted **D**iagnosis
CLI	**C**ommand **L**ine **I**nterface
DIA	**D**igital **I**mage **A**nalysis
DICOM	**D**igital **I**maging and **C**ommunications in **M**edicine
GB	**G**iga**b**yte
GDPR	**G**eneral **D**ata **P**rotection **R**egulation
HTTP	**H**yper **T**ext **T**ransfer **P**rotocol
H&E	**H**ematoxylin-**E**osin
IHC	**I**mmuno**h**isto**c**hemistry
JSON	**J**ava**s**cript **O**bject **N**otation
JVM	**J**ava **V**irtual **M**achine
KB	**K**ilo**b**yte
MB	**M**ega**b**yte
MVC	**M**odel **V**iew **C**omponent
MVP	**M**inimal **V**iable **P**roduct
OLTP	**O**nline **T**ransaction **P**rocessing
OME	**O**pen **M**icroscopy **E**nvironment
PACS	**P**icture **A**rchiving and **C**ommunications **S**ystem
QIDO	**Q**uery based on **I**D for **D**ICOM **O**bjects
RBAC	**R**ole **B**ased **A**ccess **C**ontrol
REST	**R**epresentational **S**tate **T**ransfer
RML	**R**egional **M**edical **L**aboratory
ROI	**R**egion **o**f **I**nterest
SDK	**S**oftware **D**evelopment **K**it

STOW	**S**tore **O**ver the **W**eb
TCP	**T**ransmission **C**ontrol **P**rotocol
UI	**U**ser **I**nterface
URL	**U**nique **R**esource **L**ocator
VM	**V**irtual **M**achine
WADO	**W**eb **A**ccess to **D**ICOM **O**bjects
WSI	**W**hole **S**lide **I**mage

1. Introduction

The use of digital images in pathology rises with determination, making it urgent and timely to develop solutions that support the diagnosis and a distributed revision of cases. While the radiology field has been using teleradiology platforms for over two decades, clinical pathology is now beginning to take the first steps in digitization, particularly in the use of specialty standards.

The main objective of this dissertation was to create a cloud-based platform for archiving, visualizing, and reviewing medical imaging studies of pathological anatomy, designated as PathoBox. It is a centralized platform that serves multiple institutions and usage scenarios, providing universal access through a pure web application designed for uploading, sharing, and reporting of cases. This document presents the research done in technologies and advancements in digital pathology and how they contributed to the project proposed here. It is proposed a cloud-ready architecture, the addressed market requirements, the supported use cases, followed by a detailed description of its implementation, and the results obtained. Finally, it is discussed how the platform can be expanded and improved in future work.

1.1. Scenario

The presence of digital technologies in the healthcare industry has become more prevalent due to the advantages that it brings over traditional ones. Improvements in workflow speed, quality of the diagnosis and management of the medical data were made possible by storing the medical exams, and scans in a digital format. These improvements increased the desire from pathologists for digital pathology solutions [1].

The storage of medical scans, in areas like radiology or pathology, was made possible because of the appearance of new technologies like digital scanners, that are capable of scanning, specifically in the pathology branch, pathological samples in a single digital slide. This is named Whole Slide Imaging (WSI) and allows digital slides of pathological samples, that usually have big dimensions, to be seen through a device like a computer screen [2].

Digital pathology is defined as the acquisition, management, sharing and interpretation of pathology information through a digital environment. This encompasses the usage of digital slides, obtained through a scanning device, and viewed using whole slide imaging technology, in the practice of medicine, instead of the conventional physical slides and light microscopes.

There are many advantages in digital pathology when compared to the conventional approach, mainly in two segments: professional and educational.

In the professional segment, or production environment, the institutions no longer need storage space and trained personnel to conserve and manage the physical slides, reducing this way the operation costs. Moreover, the digitally stored slides do not suffer from degradation, unlike their physical counterpart, that loose quality over time [3]. The integrity of the slides remains constant. Finally, due to the tools available in a digital environment, such as intelligent algorithms to identify regions of interest or powerful editing tools that can interact directly with the image, the practice of medicine improves, since doctors have at their disposal more and better diagnostic tools [4]. The speed of diagnosis and work ergonomics can also be improved with digital pathology [5]. Due to the possibility of concurrent access by distinct doctors, remotely, to a digital slide, digital pathology opens doors to a collaborative environment where teleconsultation and collaborative diagnoses are made possible. Because text annotations and voice annotations can be saved directly with a slide, the diagnostic reports are enhanced, and because the reports themselves are digital, their storage and sharing are also improved.

In the educational segment, the pathology teaching process is much agile with digital slides, that are easy to access in a concurrent way and shared with the community, including collaborative annotations done over the same images. Similarly, to the production segment, the existence of more editing tools and intelligent algorithms, allows students to do more with pathological samples. The better graphical interface and the possibility of interacting with other students results in a faster way of learning the concepts of pathology [6], [7].

Both students and researchers can explore new working processes with the introduction of digital pathology. This new reality has the potential for providing unlimited access to samples of slides repository, any time, and anywhere. Moreover, the slides themselves are easier to find, due to the presence of search filters and thumbnails to aid the search [8]. Annotations can also be saved along with the image, which allows users to take personal notes on a slide, and in a collaborative environment, view annotations made by other colleagues [9]. All these aspects make the teaching of pathology more convenient, which ultimately results in better education [10].

There are however many problems in the digital pathology field that have been delaying its adoption, like the price of the digital scanners and the lack of software that effectively takes advantage of digital slides [11], [12].

The first digital scanners for pathology slides were expensive, making the adoption of these scanners by small medical facilities hard. In order to get their slides digitalized, they had to send them to bigger institutions and wait for their return. This process is slow and involves the transportation of the physical slides which goes against the principles of digital pathology, which ultimately wants to dispense the physical slides in any workflow phase. The increase of market offer and the continuous adoption of this technology by medical institutions, associated with the natural technologic evolution, has made digital scanners more affordable, thus becoming more accessible, even for the small and medium medical facilities. Accompanied by advancements in the image analysis software, that proved to be as efficient as microscope viewing [13], the concept of digital pathology became more and more a reality. However, market maturity is far away from the nowadays reality of radiology. The lack of accessible digital scanners, or even microscopes with direct digital output, persists, and not every facility possesses pathologists capable of performing diagnostics of pathological slides.

Pathology medical specialty lacks human resources for supporting the nowadays requests of service. Therefore, even if clinical centers had a digital scanner and a software workstation to visualize digital slides, they would not take advantage of that, since they have no one physically available to perform the diagnosis. It is in this context that this project integrates itself. It aims to create a collaborative web platform for interconnecting the pathologists with distinct medical facilities, that produce digital slides with distinct equipment and formats. A platform of this nature brings significant advantages to the pathology field, such as flexibility of diagnostic; easy management of resources; automation of the doctors' work, increasing productivity, and even the quality of diagnostics, due to the availability of better analysis tools [2], [4].

1.2. Motivation

This dissertation aims to give a contribution to the above-described problem related to network archive and distribution of digital pathology studies. The idea and work development were done in cooperation with BMD Software, a company specialized in the development of medical software. The main goal of this project is to answer the market need for a cloud-based pathology management system able to satisfy the functional requirement of distinct scenarios, including teaching and the telepathology that provides remote diagnosis of cases. The company already has similar products in the areas of radiology and cardiology, and the market starts to request a similar solution in the pathology branch.

This project intends to make a contribution to image acquisition, storage, distribution, and remote visualization in the area of digital pathology, improving the practice of medicine, including teaching and telepathology.

1.3. Methodology

The development of this dissertation followed the work methodology described in the following steps:

- Literature review of existing platforms, with the intent to define the best set of functional and non-functional requirements. The search for existing platforms was made using relevant keywords. The goals of the research were to both find advantages and disadvantages of the existing platforms and understand the fundamental features that PathoBox needs to have;
- Study of the work previously done by BMD Software, that could be incorporated onto this project, namely, the platforms CardioBox and PACScenter. These platforms share common features with PathoBox, but in a different medical context (cardiology and radiology respectively). A collaborative image viewer prototype developed at Aveiro University [9] was also studied since it also is a web-based WSI viewer. The research done had the goal to understand which concepts could be imported into PathoBox as well as the possible improvements that could be fixed in the development of PathoBox;

- Select the best technologies to be used in the project implementation. Because of some similarities with other company products and in order to maintain some consistency with them, the technologies used in those projects had a preference. Nevertheless, analysis and discussions were carried out to raise the advantages and disadvantages of each platform.
- Getting acquaintance with the technologies used in other projects of the company and their internal structure, to define PathoBox's structure.
  - Study of the PlayFramework, a Java Web Framework;
  - Study of Handlebars, a templating tool for HTML pages;
  - Study of the Model View Controller (MVC) model developed for the frontend application.
- Development of a minimal viable product (MVP), with all essential viewer functionalities and a basic management user interface (UI), to validate the project structure, the technologies used, the proposed architecture and to be used for internal testing by the project collaborators;
- Collection of pathologist's feedback, to validate the UI choice and most importantly, to validate the viewer, its performance, usability, and tools.

1.4. Structure of the dissertation

The present dissertation is divided into six chapters. The first chapter, the present one, is the introduction. It is discussed the current context of digital pathology, presenting the motivation and need for the proposed solution. The second chapter will go into detail about the background of the solution, explaining key aspects of digital pathology and digital medicine in general, describing key components previously developed that will be used in the project. It will also be explored similar solutions to the proposed one that exist on the market, as well as a brief presentation of the current state of image analysis in pathology. The third chapter presents PathoBox proposal. It presents the system requirements, the system architecture, the technologies used in the project and finally the class and deployment models. The fourth chapter presents the PathoBox implementation and explains in detail the design of the proposed platform, how the requirements were implemented and the user interactions with the platform. The fifth chapter will present the results of the project, with a detailed description of the solution implemented, how the user can use the platform and what he can do with it, and how well it answers the requirements raised at the beginning of the project. In the sixth chapter, the conclusion, a brief summary of the main contributions of this work are presented, the main problems during development, the results obtained and what was left to accomplish. It is also explored possible future lines of work on the platform. The document will end with the bibliography used in its production.

2. Background

In this section, it will be presented the concepts and state of the art technologies in the domain of the proposed platform, including the relevant advancements in digital pathology systems.

2.1. PACS-DICOM

PACS (Picture Archiving and Communication Systems) refers to the software and hardware that are required to scan, store, and view medical images through a digital network. This system of image archiving first appeared in radiology [14], and eventually was adopted by other medical branches when the digital slide viewers and the scanning hardware evolved, as it was the case with pathology. PACS is what allows a medical platform to provide remote access to case studies and their associated images.

Initially, the PACS concept simply referred to an image repository, but as technology evolved, so did the requirements of a PACS system. The need to store patient data along with the image, as well as remote access through a web protocol and necessity to share data with other institutions led to the development of DICOM (Digital Imaging and Communications in Medicine) standard that defined the image format, communication processes, query and retrieval, for each of the medical branches [15].

DICOM is one of the most popular standards in medicine [16], having been originally created due to the difficulty that professionals in the health care industry had in dealing with distinct formats of modalities, the usual designation of the equipment that produces images. The different modalities on the market each had their own proprietary format, resulting in incompatibilities between different medical facilities when they had to share a case study. Because of the lack of a standard, different imaging viewing software could not handle images from different equipment than the one it was built for, thus resulting in many roadblocks when facilities had to share the scans between them. It was then that the DICOM standard was created. It developed a set of rules and protocols to handle medical images, in order to facilitate the communication between devices and ease the work with the images themselves. The DICOM standard continued to evolve, supporting more medical branches. The development of technologies in the microscopy area, including WSI viewers, led also to its adoption in the pathology. The DICOM standard, before the arrival of WSI, was focused on the patient and not on the image, however, due to the needs of WSI viewers, the standard was adopted to support data centered around the image rather than the patient. Nowadays, DICOM includes rules and protocols for the preparation, acquisition and handling of whole slide images [17], [18].

The DICOM files are layered and can be considered as two main blocks: the header, that contains the metadata of the file and codifying structure, and the body with a set of data elements (or attributes) that includes a very diversified kind of information elements like, for instance, patient demographics, equipment, clinical team, radiation doses, and the pixel data that contains visual information (Fig. 1)[19]. Attributes in a DICOM file are identified by tags, each with a well-defined meaning and purpose. The DICOM standard divides the attributes into two groups: the mandatory attributes that must be present in a correctly structured DICOM file and private attributes that can be used by vendors to add proprietary data to the object but cannot be interpreted by the other manufacturers.

The DICOM support for the WSI was established with supplement 145 which simply took what already existed in WSI image viewing and turned it into a standard. The supplement was adapted from radiology ones, taking advantage of existing workflow steps. Just like in radiology, the supplement 145 states that a glass slide can be scanned along with its label. Its information is sent to the laboratory information system. Upon retrieval of this sample, all the patient information comes along with the digital slide [20]. This allows information systems to display, alongside the digital slide, relevant patient information so that pathologists can easily identify the case study. This supplement allows the image to be saved to a PACS repository and compliant viewers will be able to directly access the slides.

Alongside the supplement 145, the DICOM supplement 122 was created to identify the specimen at an image level. Unlike other imaging fields, where the connection between patient and image was supported by DICOM, in pathology, a more robust mechanism was needed to specify attributes at an image level. This supplement identifies the specimen, the container where the specimen resides, and each of its components, the specimen collection sampling and the processing, and finally when needed to better understand the image, the specimen ancestors [21]. The specimen and container have different identifiers, for maximum flexibility in information systems.

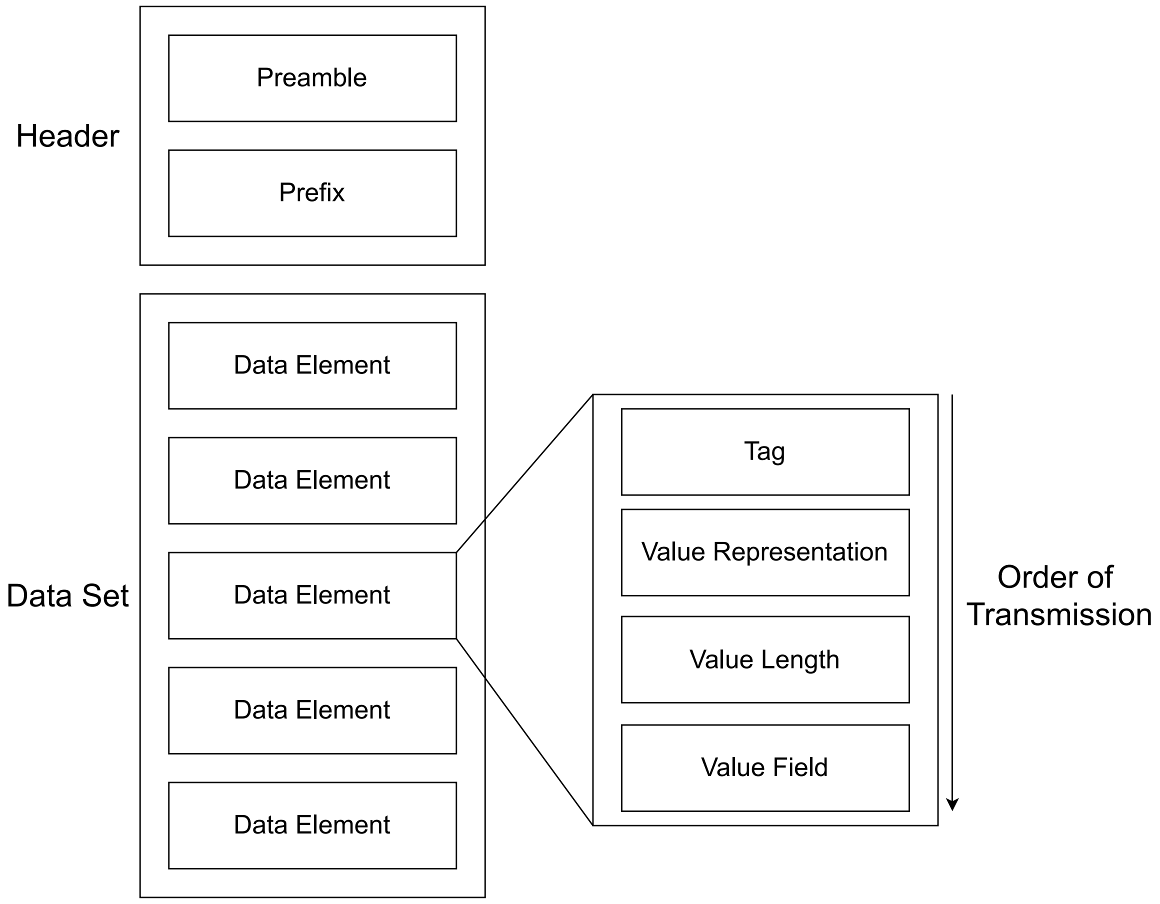


Fig. 1 - Overview of the structure of a DICOM file, adapted from [19]

2.2. Dicoogle

As PACS became more prevalent in the healthcare industry, new usage paradigms emerged that forced the development of new architectures and functionalities to satisfy the new requirements. However, the development process is too slow to satisfy modern standards of software engineering. In this area, testing and integration of new features is very complex, and the development, deployment and testing of these new features is not fast enough. Moreover, each facility ended up having their own version of a PACS repository, with some unique features needed for improving their workflow, which difficulted the share of data between distinct facilities [15]. To meet those needs, new technological approaches started being requested to allow the easy test and development of new PACS repository features.

Dicoogle is an open source PACS that uses the DICOM standard to store and retrieve medical images. It was built using a plugin architecture (Fig. 2), which allows easy development of new features or changes to its core functionality. The core services that can be modified are storage, indexing, query, and web service. Dicoogle exposes an interface to each of those services, so that they can be controlled by the developed plugins. This architecture allows Dicoogle to bridge the gap between researchers and DICOM networks, allowing an easy deployment and testing of new ideas. The development of new features is facilitated by the Software Development Kit (SDK) present in the framework, allowing another programmer to easily add new modules to Dicoogle without changing its core functionality. It also serves as a bridge between the PACS repository and third-party applications [15], which is where the project presented here integrates itself. Dicoogle is a research project of our group at Aveiro University. It is very used by the research community and in production environments, in different medical branches like radiology and cardiology, and is also used as a repository for PathoBox. The performance of Dicoogle was tested in a digital pathology setting, and it proved to be an efficient solution for the handling of WSI images [22].

Finally, because Dicoogle works with the DICOM standard, sharing data between institutions is facilitated, even if they do not use Dicoogle as their PACS repository. As long as they use the DICOM format or a format that Dicoogle can transform into DICOM (and even if it cannot, a plugin can be developed for it), institutions can share their medical data, strengthening the concept of telemedicine.

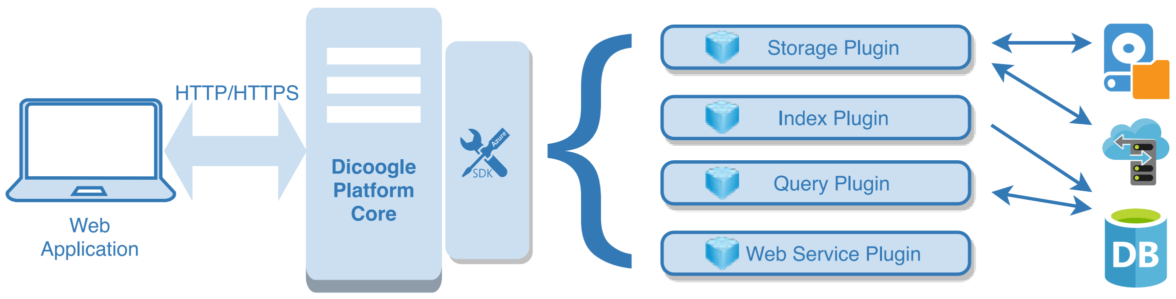


Fig. 2 - Dicoogle architecture, adapted from [15]

2.3. Whole Slide Images

Whole Slide Imaging (WSI) refers to the representation of digital slides that are produced by the scanners of microscopic slides. The usage of WSI viewers has been tested and approved by pathologists. The technology proved to be as efficient as the light microscope while retaining the image quality. Because of the great detail necessary to view these images with the necessary quality, the scans of these slides, especially in pathology, can reach considerable sizes even in a compressed format (in the order of several dozens of gigapixel) [13].

These constraints prove to be considerable challenges to web-based viewers. If the viewer takes too long to load the images, the user experience will be poor, but most importantly, the system needs to be fast so that it can compete with the traditional microscope. Due to the big size of the slides, it is not expectable that the user stores them on his personal device, because of the space limitations that usually are present in personal computers, thus it is required to store the slides on a remote server, which requires remote access to the images by the client application [23].

To solve the latency problem, keeping the quality of the images, the visualization of digital slides has to be made through a pyramid system (Fig. 3), where the slide is represented by many layers, each layer being composed of many frames (that can come in many formats, like JPEG or PNG). The bottom layers of the pyramid are more complex, having more frames thus resulting in more detail of the section being viewed. However, the user will see a smaller section of the image. This system allows the response times to remain fast enough, as the system will only load a small number of images at once, without compromising the quality of the slide. The DICOM supplement 145 specifies how whole slide images layers must be organized. Each layer of the image is composed of image tiles which in turn are stored by row. Each tile represents a portion of the image (Fig. 4).

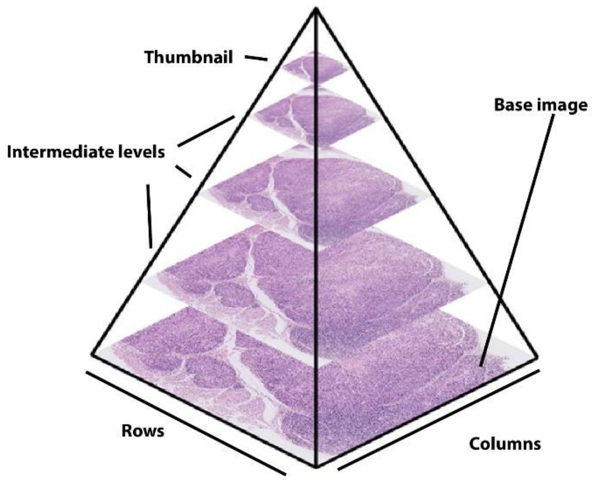


Fig. 3 - Whole slide image pyramid visualization, according to the supplement 145 of the DICOM standard [24]

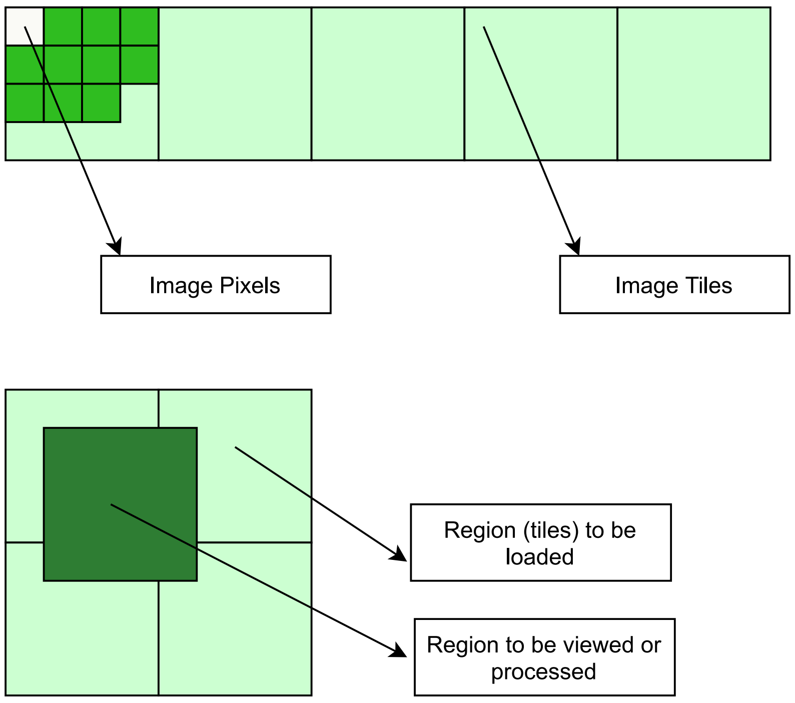


Fig. 4 - Data organization of a pyramid layer, according to the supplement 145 of the DICOM standard [24]

In the image, the dark green square represents a section of the image to be viewed or processed. To access the dark region, the four green light squares must be loaded.

The tile sizes can range in size and can be configured in the DICOM file. Larger tile sizes mean less tiles to be loaded but more data will be loaded overall. The size of the tiles can range from 240x240 pixels (172 Kilobytes uncompressed) to 4096x4096 pixels (50 Megabytes uncompressed).

To reduce the tile size improving the performance of the system, compression algorithms can be used to reduce the size of the WSI data. The most frequently used types are JPEG and JPEG2000. Pathologists found that with these two methods there is no loss of diagnostic information. JPEG2000 yields higher compression rates and fewer image artifacts however is more resource intensive [24].

To navigate through the pyramid, the user uses a zoom functionality to ascend or descend in the pyramid and the pan action to move inside a layer. Other applications, like Google Maps, already use this type of visualization for viewing maps.

2.4. *Whole Slide Image Viewers*

There are a lot of different aspects to take into consideration when discussing whole slide image viewers. Due to the heterogeneity of pathology cases, a different set of requirements is needed for each different pathology [25]. This led to a wide variety of WSI viewer solutions available. Currently, there are proprietary viewers, that dominate the professional segment [25], that come included in digital scanners solutions, as is the case with Leica scanners, and opensource viewers, that are meant to be used by the general public and try to address one specific need or problem in digital pathology. The viewers can be desktop applications or web applications that can be used through a browser. Furthermore, the viewers may or may not support the DICOM standard.

A research was conducted to understand what the current solutions on the market offer and to find out if there are any systems that integrate WSI viewers in medical platforms destined for diagnosis and report of case studies.

The search was made, using many combinations of the following terms: digital, centralized, pathology, telepathology, PACS, WSI, viewer, DICOM. The research found desktop applications and web applications, that either supports their own proprietary formats or can work with multiple image formats, including support for the DICOM standard. The research also found a centralized pathology PACS system like the one proposed on this work. The following subsections describe in detail some of the applications found and compare them to PathoBox, to understand the differences between the applications and how PathoBox distinguishes itself from the other solutions.

2.4.1 Aperio ImageScope

Aperio ImageScope is the proprietary WSI viewer from Leica. Of the products evaluated, this one is among the most complex when it comes to functionalities. This viewer only has the full capability with “eSlides”, their proprietary image format. Because this viewer was so complete in terms of what it can do, it was used as a basis to stipulate the most important tools a pathology viewer must have. Leica has been on the market for a long time, which gives credibility to their product because of the time it had to evolve and mature and the feedback collected from its users.

Aperio ImageScope was designed to be used in combination with their aperio scanners and is available for free to download on their website. The viewer accepts some formats, but some features are disabled on third-party formats, like image analysis that only works with their proprietary format.

Some features of this viewer that are interesting:

- The user can crop out parts of the slide using the annotations tool. These crops can be exported to an image format like JPEG, can be sent via email to another user and can be analyzed using the image analysis algorithms that the viewer possesses;
- Detailed annotation window, where a user can alter the annotation, navigate to said annotation on the viewer, run the analysis software, and see its results;

- The capacity of re-defining the image resolution. The user can create a ruler and specify its length, altering the image resolution accordingly. Important when the image comes with the wrong dimensions;
- Different UI configurations. The user can specify a “clinic” mode where only the relevant tools for clinic work are shown in the toolbar;
- Capacity to measure the distance between two annotations. Not to be confounded with the ruler annotation that measures the distance between any two points of the image;
- Advanced color filters destined for research purposes.

2.4.2 Omero.iviewer

Omero.iviewer is a solution that was designed around the slide repository from Omero and can be used through a web browser or as a desktop application. Omero is an image server, that converts proprietary formats, like Leica’s, in their own proprietary format, using Bio-Formats from OME (Open Microscopy Environment), to be used in other applications. Bio-Formats[1] is a standalone java library to read and write life sciences image formats. It is similar to Dicoogle, for being an image repository that can be programmed and extended for support of new features or new image formats. However, it goes beyond light microscopy and supports modalities like electron microscopy and matrices/tables of biological data. While Dicoogle focuses more in being a medical image repository, Omero places itself as a data management tool, that connects multiple repositories and databases in order to aggregate many types of biological data under a common application programming interface (API) [26].

The OMERO platform supports the import of data through the application, through the command-line interface (CLI) and automatically through a defined directory that automatically uploads the files with the supported formats; management of data through a permission system, controlling the ownership of resources through user groups, that allows the sharing of data safely between different users; image analysis tools that can be extended by the community through scripting, with integration with third-party image analysis solutions like ImageJ;

Despite supporting multiple image formats, including basic DICOM images, Omero does not support DICOM pyramid images, which a pathology DICOM compliant viewer needs to support. For this reason, Omero.iviewer does not fit in the project requirements despite being close in terms of functionality to the desired product, for being an image viewer that can be used through a web browser and possesses an annotation system similar to the one pretended while providing the user with simple to use analysis tools.

[1] https://docs.openmicroscopy.org/bio-formats/6.3.1/about/index.html

2.4.3 Open Source Applications – QuPath, Cytomine and Orbit

This section will briefly mention some relevant open source projects, pointing out their main features:

- QuPath is a WSI viewer built using java and Open Slide (section 2.4.4). It is available for Windows MacOS and Linux and it stands out for being a lightweight viewer with powerful and intuitive analysis tools. It possesses dedicated tools for H&E (Hematoxylin and eosin) and IHC (Immunohistochemistry) workflows, with the possibility to manually configure steps for other workflows. Another interesting point is its data model, where everything in the viewer is treated as an object. This allows the viewer to expose an API to be used in scripting tools, approach used in their image analysis software [27]. However, this viewer does not support remote access through a web browser, as it is only available as a desktop application and the application is not integrated with a DICOM-PACS repository;
- Cytomine, unlike QuPath, is a web-based WSI viewer, with support for collaborative features. Its main application, as described by the authors, is to perform image analysis on large datasets of digital slides, with support for collaborative environments. The literature however does not mention support for the DICOM standard, and the editing tools on the viewer are not very extensive. It is not mentioned as well the capacity to control the viewer through an API [28];
- Orbit is a desktop application that supports connection with the Omero repository. It is mainly targeted at researchers and investigators, providing advanced analysis tools with built-in artificial intelligence and deep learning algorithms. This viewer has support for DICOM, however it is not built for clinical/educational purposes. It makes use of spark, to execute the image analysis software in a distributed way through the image tiles [29].

2.4.4 Other Software

Other software solutions for pathology that are worth mentioning are: Open Slide with OpenSeadragon, a web viewer that combines the OpenSlide library with the web-based technology OpenSeadragon to view pyramid images [30]. OpenSlide is a library for whole slide image reading and processing implemented in C. It serves as a foundation for other WSI viewers like QuPath and Cytomine but is also used to test image analysis algorithms. Its performance and lightweight resource consumption make it a strong choice for handling WSI. The literature does not mention support for the DICOM standard which is a disadvantage of this product; PMA.start is a desktop application designed to support the development of extensions and image analysis workflows through its API [31]. It has integration with ImageJ which provides its image analysis capabilities. Moreover, it supports the DICOM format; Hamamatsu viewer[2], like Leica's viewer, is a pathology viewer available for free that supports their proprietary format and some others including DICOM.

[2] https://nanozoomer.hamamatsu.com/eu/en/Software/NDPView.html

2.4.5 Results

Table 1 presents a summary of all the relevant features each viewer has. The main takeaway is that each viewer reviewed possesses image analysis capabilities, to some degree. This feature is essential on any digital slide viewer given the benefits it brings to professionals and researchers, as it cuts down the manual work they are required to do.

Despite all viewers having image analysis tools, only Orbit and QuPath integrate artificial intelligence and deep learning in their analysis algorithms. In both viewers however the user is required to train the system with training datasets first in order to use the intelligent tools. Normal image analysis tools are ready to use out of the box in all viewers mentioned, although not all expose an interface to expand those capabilities.

Another important aspect is that only Leica Viewer and Cytomine support collaborative tools. Despite the advantages of such modality, its uses in diagnostic and reporting of pathology studies is still limited.

Table 1 - Summary of the reviewed viewer's features

Viewer/Feature	Web-Based	Image Analysis	AI / Deep Learning	Extensible	DICOM - WSI	Collaborative Tools
Leica Viewer	✗	✓	✗	✗	✗	✓
QuPath	✗	✓	✓	✓	✓	✗
Cytomine	✗	✓	✗	✗	✗	✓
Orbit	✗	✓	✓	✓	✓	✗
Omero.iviewer	✓	✓	✗	✗	✗	✗
PathoBox	✓	✓	✗	✓	✓	✗

2.5. Centralized Medical Reporting Platforms

When it comes to centralized reporting platforms, there are available solutions on the market for many clinical areas like, for instance, cardiology, pathology, and radiology. Sectra Medical is a company that specializes in imaging IT solutions in the clinical areas mentioned. Sectra Digital Pathology PACS[3] is their solution for digital pathology workflows that supports the analysis and revision of pathology studies.

The pathologists can remotely view the digital slides associated with a patient and perform the diagnosis through their WSI viewer. The viewer possesses an annotation system where users can mark areas of interest or define Regions of Interest (ROI) where they can perform image analysis. The product specifically mentions KI-67 assessment, used in the diagnosis of breast cancer. The pathologists can edit the analysis´s results, improving the algorithm via its feedback loop input. Furthermore, the viewer possesses multi window view where pathologists can open up to four study images in the viewer, useful to analyze the same tissue sample stained with different types of stains. Finally, the application exposes a standardized API that facilitates integration with other image analysis products such as ImageJ.

There are however two main differences between Sectra's platform and PathoBox. Firstly, Sectra PACS is a desktop application instead of a web application. Secondly, the focus of the application is on the management of studies and not on the reporting concept proposed by PathoBox, that is, the organization of the user interface and its capabilities differ from the ones in PathoBox, not answering the raised requirements fully.

CardioBox is a centralized system for managing ECGs (Electrocardiograms) with remote diagnostic tools. This platform served as a base of inspiration for PathoBox since it has the same requirements. The only difference between the two applications is that one deals with ECGs while the other deals with digital slides, so the viewing and diagnostic tools will differ.

CardioBox uses the concept of organizations, that are composed of facilities that "produce" ECGs for analysis. These are uploaded to the platform and connected to a study. The studies are assigned to a doctor registered in the platform. The doctor can review the case study, producing a report (diagnostic) to be sent back to the facility that uploaded it. This platform shares some of its concepts with PathoBox since its use cases are very similar.

[3] https://medical.sectra.com/product/sectra-digital-pathology-solution/

2.6. *Collaborative Whole Slide Image Viewing*

Collaborative image viewing in pathology brings many advantages, both in an educational and professional environment [3], [6], [7]. A parallel platform, developed in combination with another collaborator at University of Aveiro, proposed an architecture for a fully web-based collaborative platform where users can create working sessions and share them with their peers, with role-based control at a resource and at a session level [9], [32], [33]. The goal of the project was to assess the usability of a collaborative WSI viewer, whilst presenting possible use cases in the after mentioned areas.

A working session is a live document where users can interact with a whole slide image at the same time. Changes on the image are replicated across all users in the session. The changes in the image creates actions that are stored in the database, to persist the session. Users can logout of the session and return to it later, keeping all the progress done.

The sessions can be joined via a link or via the platform if the users are registered. The sessions keep track of the users that joined them so that the same user does not join the session twice.

The sessions support the assignment of a configurable set of permissions to control user actions on the viewer. The permissions can be changed live by the session moderators. This feature allows moderators in the session to control the permissions of the participating users live. This modality strengthens the collaborative concept of the platform since it prevents sessions with many users from becoming unmanageable and chaotic (in the event of many users trying to pan to different areas at the same time, for example).

The viewer also supports the replication of a session's actions in a separate replay viewer. This modality allows users to review in order all the actions performed in a session. This feature is important when the user needs to review a session. To illustrate, a user that missed an online class can review it with this feature. The user has tools available that allow him to control the speed of the replication or pause it.

The usability of the viewer can be assessed in an educational and professional segment. In the educational segment, the sessions can be used to give open lectures about a topic. The professor can create user groups and give online classes, with the possibility to enforce a schedule to attend. Students can review classes with the replay tool. This tool can also be used by the professor to evaluate a student's knowledge by creating exams and reviewing them with the replay tool. In the professional segment, the platform can be used for collaborative diagnosis of a case study, teleconsultation by inviting a doctor to a session and review diagnostics performed during a session with the replay tool.

2.7. *Data Privacy and Security in Medical Systems*

Medical imaging software handles personal patient archives, therefore dealing with personal and private data. With the rise of digital medical systems, the threat of information breaches resulting from cybernetic attacks is a rising concern for medical facilities [34]. Moreover, medical information systems need to be compliant with the GDPR (General Data Protection Regulation) for operation in Europe [35]. Therefore, to block the access of unauthorized users to an organization's medical repositories, a security system needs to be integrated within the platform. Traditionally, users within a PACS environment only operate in one domain, having access to all resources within it. As the concept of digital medicine evolved, new use cases created new necessities that challenged the conventional security designs.

It became necessary to have different ownership domains in the same server instance, to enable the practice of medicine in a centralized PACS system shared between distinct facilities for example. Such is the case with PathoBox platform. One doctor should only have access to the data from patients from his facility. Another use case example is allowing access to certain resources within an organization for academic purposes like tele-research and education.

For this matter, an access control mechanism is needed to control the access to resources inside a domain. A RBAC (Role Based Access Control) system, like the one demonstrated in [36] can be used in this scenario to protect the patient data from unauthorized access. A RBAC system works with the concept of user roles. A user role defines which resources in the platform a user with this role has access to. The resources can be data in a database or simply UI elements in the application.

Fig. 5 demonstrates a simple example using RBAC to control the permissions of each type of user to the case study resource. The doctor, that has the doctor role, has full permissions over the studies in his domain. He can view, edit, and delete studies. Meanwhile the student only has permission to view the studies.

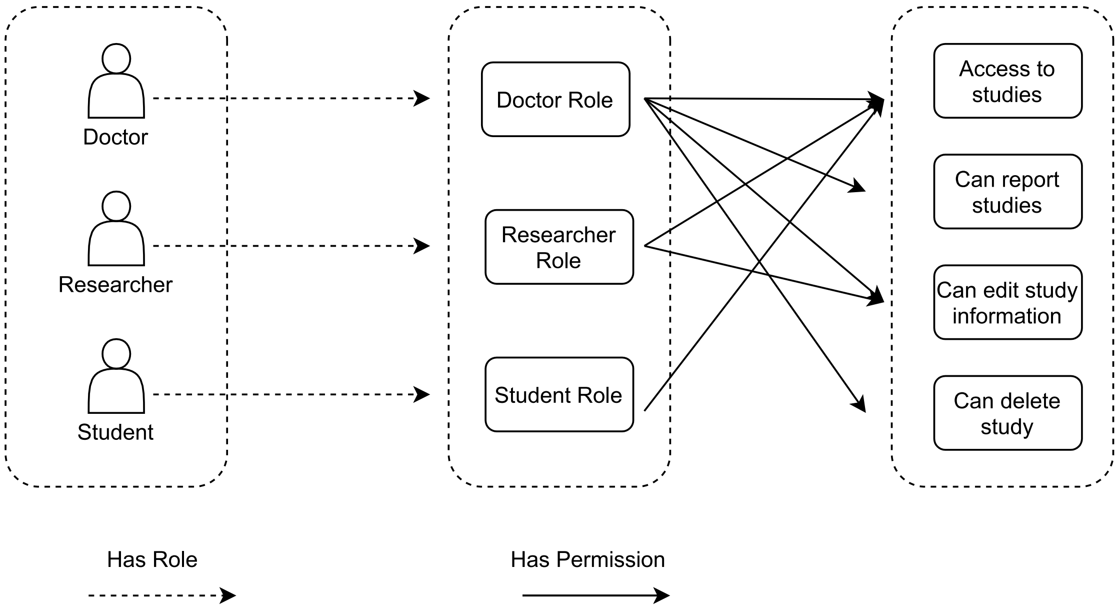


Fig. 5 - Simple RBAC example with three users and three roles

Another example, an institution has facilities and those facilities have doctors. Doctors from facility A cannot access data from Facility B because the roles are not shared between the facilities. However, if the doctor is given a role by the organization instead of the facility, it now has access to both facilities' A and B data, since facilities A and B belong to the same organization. The roles can be edited, extending their permissions to other entities and facilities. The roles of the doctors and facilities need to be managed by the system administrators.

The data also needs to be protected against security breaches and possible catastrophes, given its nature and importance to the underlying systems that are dependent on it. If for example, the data becomes inaccessible due to its corruption, then the system is no longer in compliance with the GDPR regulations and the professionals that are using it are also affected, resulting in possible delays in the diagnosis which can lead to harmful consequences for the patient in care. To address these issues, reliable authentication mechanisms need to be integrated. The system also needs to contemplate service and data availability which can be achieved with data backups and system architectures that enable high availability scenarios.

2.8. *Image analysis in Pathology*

With the rise of WSI technologies for image viewing in pathology, Computer Assisted Diagnosis (CAD) became a reality and it is currently used to provide support to the pathologist in the diagnosis process [37].

The analysis of a glass slide is an arduous task that involves manually scanning the whole slide, taking a considerable time in the reporting process. Even in digital pathology, with the enhanced navigation tools, this process is still a dispendious task taking up a lot of time from the pathologist. Moreover, image analysis by pathologists in many cases are based on educated guesses that might be biased, leading to errors in the diagnosis [38]. In the analysis of breast cancer, the most common type of cancer in women, around 80% of the studies analyzed are benign. This means that the pathologist spends a lot of time and effort diagnosing slides that do not pose any threat to the patient. Pathologists recognized these gains and have been using image analysis tools in the diagnose of pathology slides [37].

There are two branches in pathology that focus on the analysis of specimen under the microscope, Histology and Cytology (Fig. 6)

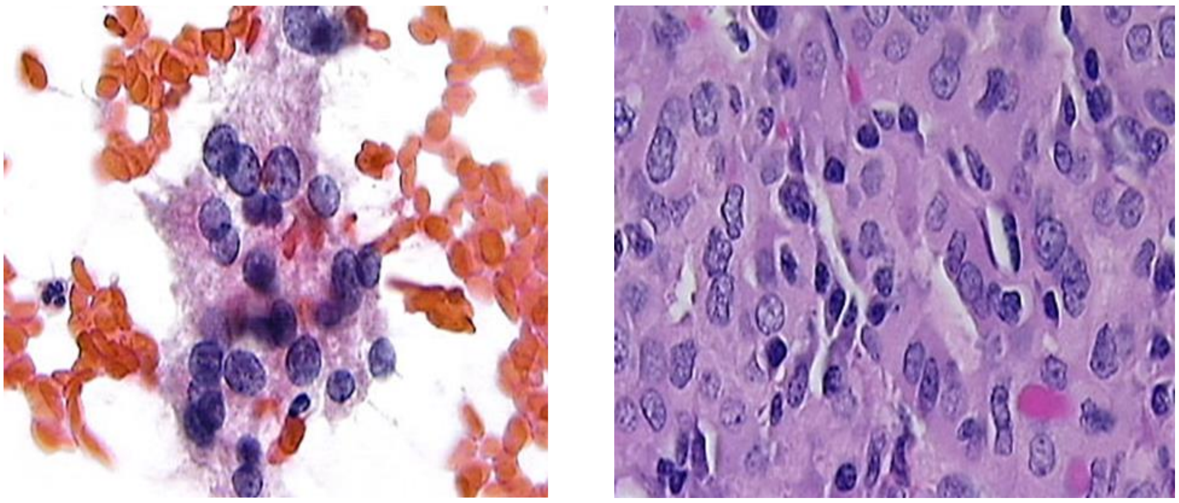


Fig. 6 - Comparison between Cytology and Histology slides for the same type of tissue sample

Histology is the study of biopsies or surgical specimen under the microscope to find signs of disease. The samples are stained with the aim of identifying cellular components. Counter stains are used to provide contrast. Currently, the preferred staining method used by pathologists is Hematoxylin-Eosin (H&E) and it is believed that it will continue to be the preferred staining method for the next 50 years [39]. Immunohistochemistry (IHC) is another staining method that derives from Immunostaining and is frequently used in the detection of breast cancer.

Cytology on the other hand focuses on the cell structure, function, and chemistry. It is the result of less invasive biopsies. It is the most found type of imagery for both disease screening and biopsy purposes, given its efficiency in analysis, cost reduction and is less invasive [37], [40]. The advantages of Cytology in image analysis are reflected in the complexity of the tissue samples. Usually, cells and cell clusters are well-defined and isolated and the absence of complex structures such as glands makes samples easier to analyze, for both pathologists and analysis software.

Despite the advantages of Cytology, it is not as accurate and reliable as Histology [41]. Pathologists agree that Histology is the gold standard in the diagnose of several diseases, including almost all types of cancer [42], due to its reliability in the diagnosis, so both are concurrently used in pathology diagnosis.

2.8.1 Importance of Image Analysis Software in Pathology

Digital Image Analysis (DIA) refers to the usage of quantitative image analysis algorithms to calculate relevant medical data for the diagnosis of studies. It helps the pathologist evaluate whole slide images, improving accuracy, reliability, reproducibility and productivity [43]. It has been in use in other medical branches like in radiology and pathology and it is a well-developed field with lots of documentation and literature.

Current solutions to address DIA in Histology and Cytology samples focuses on quantitative image analysis to calculate biomarkers. For example, for the detection of breast cancer on stained tissue by IHC, the pathologists try to determine the estrogen receptor (ER), progesterone receptor (PR) and human epidermal growth factor receptor 2 (HER2) biomarkers to assess if the patient has breast cancer [43]. Image analysis tools have been developed that specifically focus on the calculation of these biomarkers on IHC stained samples. Different pathologies with different staining methods require different biomedical data and thus different algorithms. Quantitative image analysis is also used for image segmentation with the intent to create active counters around nuclei cells or to highlight regions of interest.

Another approach in DIA relies on the usage of artificial intelligence to grade and detect cancer in tissue samples. Unlike quantitative image analysis, where the algorithm only calculates the biomarkers and the classification is done by the pathologist, with the usage of artificial intelligence (AI) the software also takes care of the classification of the slides.

The usage of AI in sample classification requires a training set to calibrate the software, where the samples are provided along with their diagnosis. The algorithm will extract features from the image and will be able to detect patterns in the data that allows it to classify the samples. The intent of AI tools is not to replace pathologists but rather help them in classifying the studies. These tools have the potential to become more accurate, faster and reliable than human pathologists in the diagnose of cancer and other pathologies [43]. However, the adoption of intelligent tools in pathology has had major setbacks, for its poor performance in intensive usage scenarios, lack of labeled data for training sets, and lack of approval by the health organizations for the usage of AI in pathology clinical environments [44].

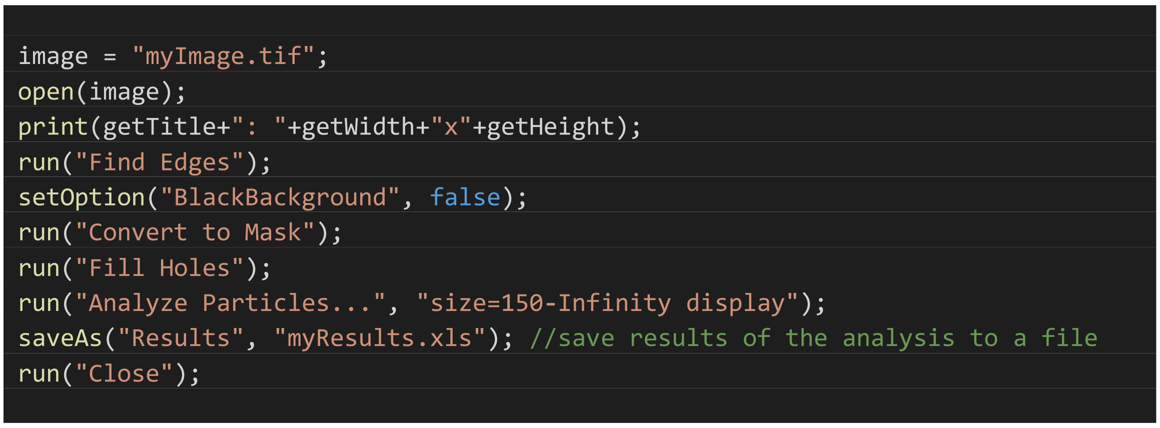
2.8.2 Image Analysis Solutions

This section talks about some image analysis tools commonly used in pathology workflows. It presents their advantages and the problems they tackle in digital medical image analysis.

ImageJ

ImageJ is an opensource application developed in Java that comprises a library of tools and algorithms for biomedical image analysis, amongst which there are tools specific for digital pathology.

The tool is extensible with plugins and scripts and allows developers to implement new features into ImageJ with relative ease. The scripts can be written in many languages like Python, Javascript or Macro language, which is a specific language created for ImageJ [45]. This language behaves like other programming languages. Users can define variables and functions, making use looping and if/else statements. It is a simple language because the user can execute functions from plugins or execute commands on the UI directly from Macro scripts with simple function calls, as demonstrated in Fig. 7. The main advantage of this language is its simplicity. However, since it is not a fully-fledged programming language, it has some drawbacks in its usability and versatility.



```
image = "myImage.tif";
open(image);
print(getTitle+": "+getWidth+"x"+getHeight);
run("Find Edges");
setOption("BlackBackground", false);
run("Convert to Mask");
run("Fill Holes");
run("Analyze Particles...", "size=150-Infinity display");
saveAs("Results", "myResults.xls"); //save results of the analysis to a file
run("Close");
```

Fig. 7 - Simple ImageJ Macro script example

Thanks to the contributions of the open source community, ImageJ grew into a large ecosystem that supports many fields of biomedical image analysis with the addition of new plugins [46]. It is possible in ImageJ to use a series of plugins to improve the results of another plugin. For instance, image segmentation can be improved by first applying the Contrast-Limited Adaptive Histogram Equalization (CLAHE) plugin which will make the borders of cellular structures more distinct [45].

ImageJ works with many image formats including DICOM (with non-compressed tiles). However, due to its plugin structure, it is easily adaptable to support different image formats. The Bio-Formats plugin for example implements support for almost all kinds of DICOM images.

The application is a simple toolbar where the user can import his images and use plugins or run macro scripts on them. The application was designed to be simple to use given its target demographic. It has a macro recorder that allows a user to record a set of actions he performs on the UI, generating a macro script in the end. The user can execute this macro through the UI. The software can be run in a headless mode for usage in different contexts.

Since the beginning that the development of ImageJ contemplated integration with other platforms, such as MATLAB. Biologists often need to use multiple tools to acquire and analyze data and it is crucial that the applications can communicate with one another. This extended ImageJ functionalities, further solidifying its place as a top tier image processing and analysis tool.

Fiji[4] is a distribution of ImageJ that encompasses a lot of plugins to facilitate scientific image analysis. The goal was to provide a platform with everything already included, that was free to download and unpack [46]. It is the recommended distribution of ImageJ for beginners and alike given the convenience of not having to search and download the plugins they need. Fiji is just ImageJ with bundled plugins so the user can still include more plugins that he needs and program new Macros.

ImageJ was not natively designed to work with whole slide images. Thus, a plugin named SlideJ was developed to support tile-based processing, bringing WSI support for ImageJ. The plugin depends on the Bio-Formats plugin to handle DICOM files and other WSI formats. It takes as input the files to analyze and a Macro script to execute on each tile. The user can use other plugins and algorithms as he would for a normal image. SlideJ takes care of splitting the image into tiles and applying the Macro script [23]. The authors point out however that the plugin is not meant for production environments where efficiency is necessary. Moreover, the plugin does not work with compressed DICOM WSI file. This is an obstacle to the usage of this plugin in PathoBox since DICOM support is required.

The main disadvantage of ImageJ is its integration with a third-party application such as PathoBox. ImageJ and its plugins were developed to work with a graphical user interface. Implementing this tool in an headless environment creates many issues with the plugins that assume an UI is available. ImageJ2, a new version of ImageJ, is currently being developed to address these issues. This new version however does not change the plugins behavior so many plugins have yet not been adapted to a headless environment.

QuPath

QuPath besides being an image viewer, it is also an image analysis tool with support for biomarker analysis/ IHC quantification and tumor analysis on H&E. It was designed specifically to work with WSI images. It works with a hierarchical object structure where each object represents an area in the image. Objects can be created manually by the user or automatically by the image analysis algorithms. Objects can be linked together with parent-child relations and can be assigned classifications and measurements. This model allows QuPath to scale well with a large number of objects. These objects can be queried in an efficient way and are used to interactively train object classifiers using machine learning algorithms [27].

[4] https://imagej.net/Fiji

QuPath is apt to exchange data with ImageJ and MATLAB. This relation can be exploited to use the image analysis capabilities of ImageJ on a whole slide image and vice-versa.

QuPath can be adapted to work outside of the designated areas through custom extensions and workflows that developers and users can create. For example, it was used to provide an in-depth analysis of progressive optic nerve degeneration (ON) in rats. The research programmed a workflow in QuPath that allowed to classify objects as axons or gliotic elements, used in the detection of ON [47].

CellProfiler

CellProfiler is a MATLAB based free opensource software, designed to analyze and batch process cells in biological images. It was designed to be easy to use, with most tasks available by clicking on the UI.

CellProfiler works with the concept of pipelines and modules. A pipeline is a series of modules that execute in order. A module is a series of image operations. A typical pipeline consists of loading the images, correcting for uneven illumination, identifying the objects, and then taking measurements on those objects. The users can mix and match modules and adjust parameters depending on the intended use. It is possible to assess the performance of each module as they run [48], [49].

Objects in CellProfiler correspond to components on the image and follow a hierarchical structure. Primary objects such as cell nuclei will serve as base objects for cells, which will be consisted by the nuclei and the surrounding objects. For object identification, the following features are used: size, shape, color intensity, degree of correlation between colors, texture (smoothness), and number of neighboring objects. CellProfiler comes with built-in algorithms for image analysis but researchers can create new ones and include them in the software.

CellProfiler can be used to count and segment cells in the image, to calculate their distribution, size, intensity, texture of fluorescent stains and other cell related metrics [49]. M. R. Lamprecht et al. [48] used the software to count yeast colonies, assess wound healing and other visible quantifiable assays.

Another important aspect of CellProfiler is the capacity to run a pipeline in a distributed environment. The software contains instructions on how to setup the environment. Once setup, the pipeline runs normally as it would on a single machine. This allows CellProfiler to run image analysis on multiple images at once, drastically reducing processing time for large batches of images [49].

Orbit

Orbit is a whole slide image analysis tool that uses context-based structure classification to compute several image features that are fed to a Support Vector Machine (SVM) for machine learning algorithms. In contrast to deep learning methods, an SVM classifier is easier to train, requiring only a few training regions, allowing users to rapidly create new classification models [29].

Orbit uses a generic map-reduce based tile processing mechanism which enables the application to use complex image analysis algorithms designed for small images to work with WSI. Orbit can be extended for example with third-party tools such as TensorFlow to create more complex classification models. Other examples of software integrations are CellProfiler for more detailed cell analysis on each tile and ImageJ for other analysis algorithms that do not come bundled in with Orbit.

The main difference between QuPath and Orbit is that the latter was designed to work with existing image analysis algorithms and tools, with its tile-based image processing. Aside from that, both applications can work with WSI images and allow the user to create his own classification models. Orbit, like QuPath, is also extensible through user made scripts and modules [29].

Orbit possesses two ways of defining ROIs. The first is through manual annotations done by the user and the second is through a trained model that automatically identifies the ROIs. The two methods can be used concurrently, with the manual method having priority.

Summary

Table 2 sums up the discussed tools, presenting a comparison between the main features of each analysis tool. It is concluded that all tools besides CellProfiler support WSI and trainable user models for ROIs. ImageJ has support for these features thanks to plugins developed by the community while QuPath and Orbit natively integrated that functionality thanks to their data model.

Every tool allows the user do create his own plugins and scripts to respond to a specific need. Image analysis in pathology is a vast field and no software out of the box can respond to every possible need, hence the necessity for user made scripts and plugins to extend its functionality.

Table 2 - Summary table of the image analysis tools presented

Feature / Tool	Fiji/ImageJ	QuPath	CellProfiler	Orbit
WSI	✓	✓	✗	✓
DICOM-WSI	✓	✗	✗	✓
Plugins	✓	✓	✓	✓
User Scripts	✓	✓	✓	✓
Trainable Models	✓	✓	✗	✓

3. PathoBox Proposal

This section describes in detail the PathoBox proposal, presenting its functional and non-functional requirements, architecture, technologies, class model and deployment model.

3.1. System Requirements

The development process of PathoBox started with the clarification of the system's functional and non-functional requirements. From the similar reporting platforms PACScenter and CardioBox and the pathology web viewer designed in [9], [33], it was possible to have a strong basis as to what the fundamental requirements of the platform were. The next step consisted in the literature review, presented in Section 2.4, about existing WSI pathology viewers. Past experiences of the company with clients that also have services in pathology, also helped to understand the standards that the users have regarding this type of platforms. The next subsections describe the functional and non-functional requirements of the platform.

3.1.1 Non-functional Requirements

To provide a good user experience and to stay competitive with other products on the market, PathoBox needs to address the following non-functional requirements:

Performance

One of the most important aspects of a digital slide viewer is its efficiency and performance when opening and navigating the visual data. It needs to compete with the traditional light microscope to provide a good user experience. Pathologists are very experienced in maneuvering the light microscope so, if the digital application falls behind in speed, they will not be interested in using the solution anymore.

Likewise, the UI of the application needs to be fast and responsive. The application is meant to speed the reviewing of case studies so the user must feel satisfaction using the platform for long periods of time. If the application has constant stutters or takes a few seconds to load the tiles each time the contents of the page change, it will cause discomfort in its use, resulting in a bad user experience which will result in resilience to its adoption.

Security

As mentioned in section 2.7, an important aspect of medical platforms is the security and privacy of patient data. The access to patient records and their digital slides needs to be protected. Only authorized pathologists should have access to these records.

The application must also be able to protect the data in the case of system failure, that is, the integrity of the data must be assured. A loss of data in a medical platform is especially critical since it might negatively affect patient care. However, the security mechanism cannot compromise the platform performance and usability.

The acquisition of the slides and its related metadata is a costly procedure and the diagnostic times cannot be compromised.

Modifiability/Extensibility

The application needs to be modifiable and extensible to fit as much as possible the requirements and needs of the medical facilities. Each institution might have different use cases for the platform, so it is important that the addition of new features or edition of existing ones does not become an expensive task. Digital pathology is not a stagnant field therefore the application needs to keep up with the changes to stay relevant.

The WSI viewer in particular needs to be easily modified to support new usage environments such as integration with third-party platforms and it needs to be easily extended to support new image tools.

Usability

Finally, the application needs to be easy to use and to understand. The target demographic of this application is accustomed to manual workflows and is not tech savvy. It is important then that the users get familiar with the application in a short time, for a shorter adaptation period. Since the application was designed to improve the reporting workflow, the UI needs to reflect those needs by being efficient in its usage.

3.1.2 Functional Requirements

The set of functional requirements were obtained from the literature review of existing platforms as well as an analysis of PACScenter and CardioBox.

There are three main actors that will be using the platform: the organization administrators; the health technicians and the pathologists. The organization administrators are responsible for managing an organization. This includes managing its users, facilities, authentication providers and access roles. It is required from the administrator's basic knowledge about the platform to operate the UI. The health technicians represent the facilities and will manage the case studies. They are responsible for assigning the case studies to the pathologists and managing the mailboxes that will receive the reports. Unlike the organization administrators, this entity does not need prior knowledge about the platform in order to use it. Finally, the pathologists are the entities responsible for analyzing and reporting the case studies. Like the health technicians, pathologists do not need to be knowledgeable about the platform's systems to use it.

Organizations and Facilities Use Cases

The diagram depicted in Fig. 8 shows the main use cases of both organizations and facilities. The organizations represented by their administrators need to manage the facilities, users, authentication providers, access roles and organization details through PathoBox's management page. More concretely, regarding the users and access roles, the organizations must be able to create new users or edit existing ones including their roles. The administrators can create new roles defining their set of permissions in the platform and assign them to users. It is the responsibility of this actor to define the doctors and health technicians in the platform. Regarding the facilities, the administrators need to configure their users, mailboxes, and provenances. Mailboxes are used to automatically collect the diagnostics from the pathologists and are a crucial element in the optimization of the reporting workflow.

The main concern of the facilities represented by the health technicians is the support of a variety of different image formats, including DICOM, by the system to be compliant with the modalities they have. Thus, the upload of case studies needs to adapt to the facility by supporting its image formats. Users must be able to upload the studies both through the UI and when supported, through the scanning devices that will automatically perform the upload. In this case they need to be registered as provenances in the platform. Independently of the method used to upload the studies, the health technician will be able to assign the studies to the doctors registered in the facility.

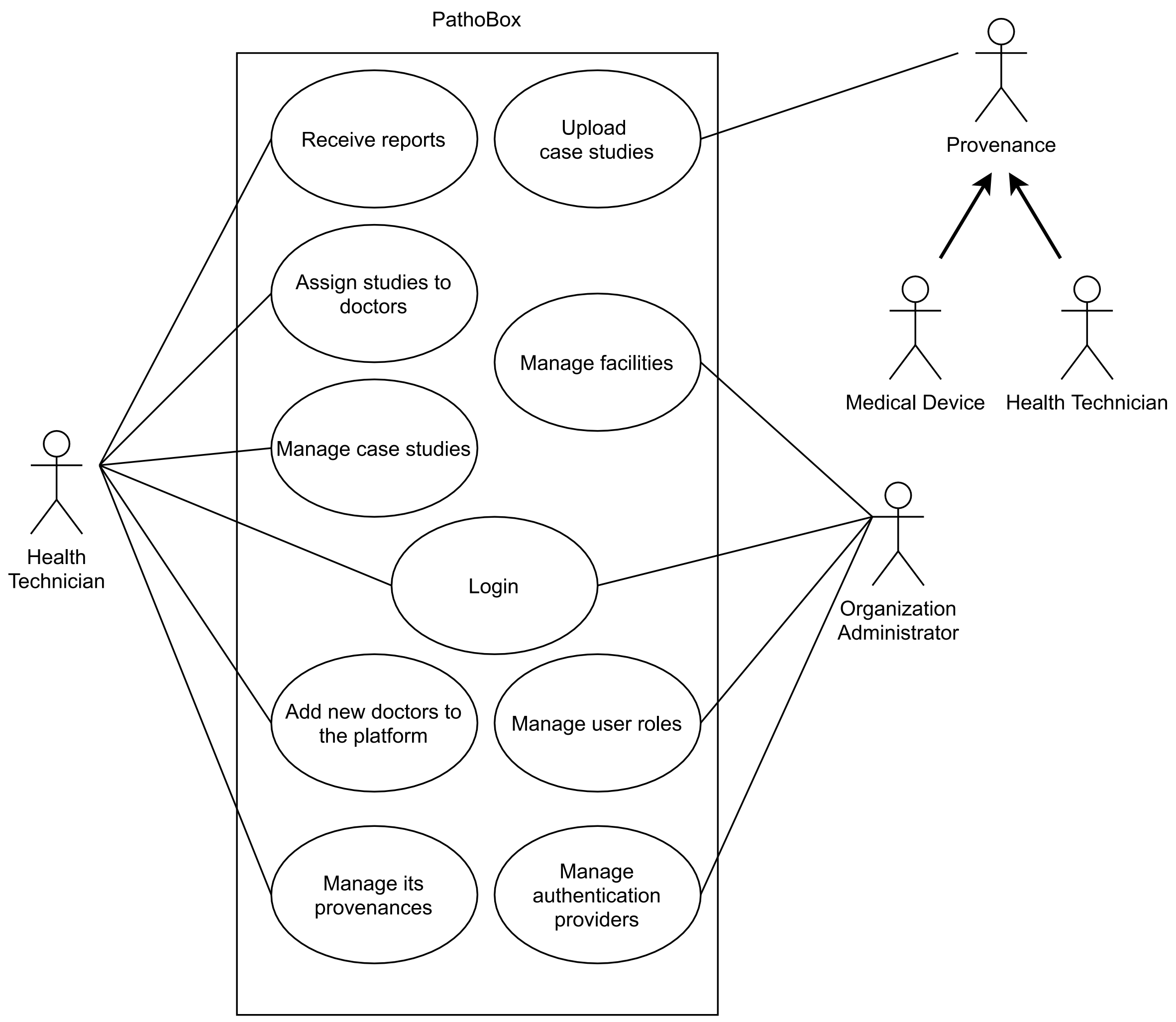


Fig. 8 - Organization and Facility use case diagram

Pathologists Use Cases

The pathologists interact with two different sections of the platform, the case study inbox and the image viewer, unlike the previous actors that mainly use only one section.

In the inbox (Fig. 9), the pathologists need to have a view of the studies they have to report, with the possibility to filter them by priority, group and provenances. From the inbox, they need to open the studies that will be displayed in the viewer. For example, if the pathologist orders the studies by priority, then opens the first study in the list, the viewer needs to save that ordering so that the next study opened is the second one in the list of priority studies. From the inbox, the pathologist needs to be able to preview a report and deliver it to the correspondent facility that uploaded it.

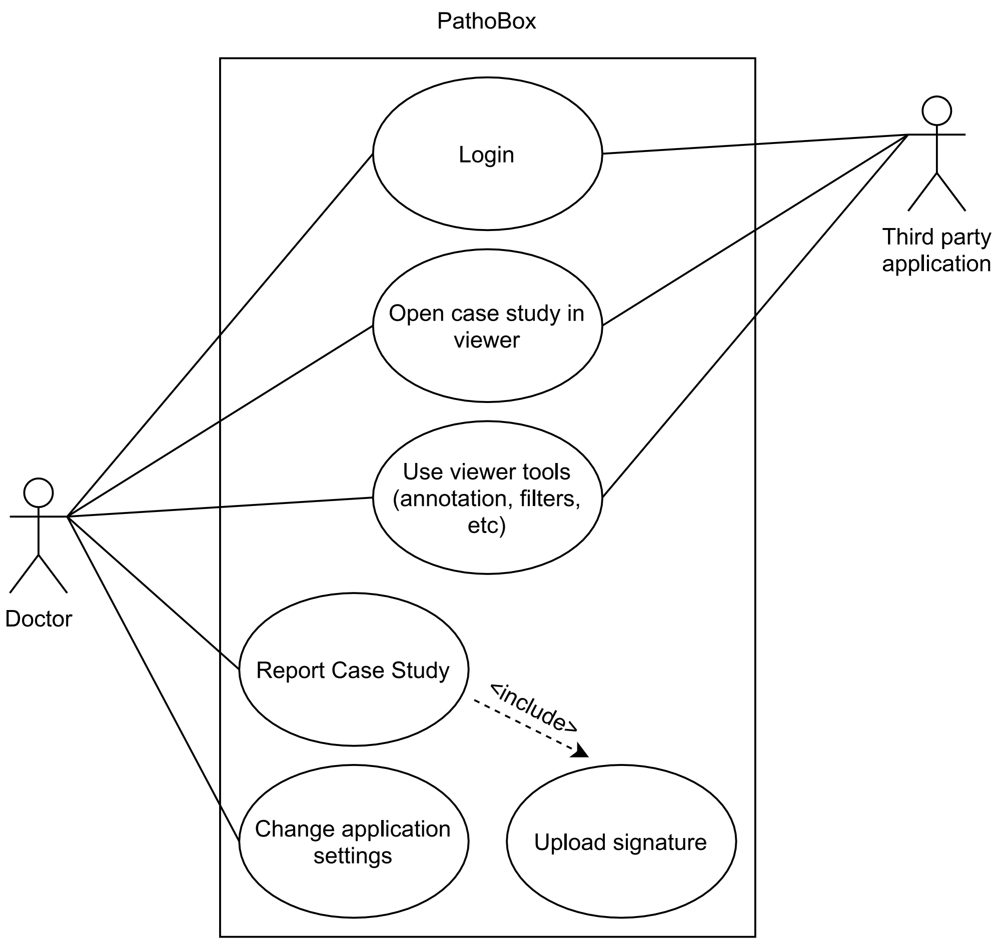

PathoBox
Login
Open case study in viewer
Use viewer tools (annotation, filters, etc)
Report Case Study
<include>
Change application settings
Upload signature
Third party application
Doctor

Fig. 9 - Doctor and third-party application use case diagram

In the viewer (Fig. 10), the pathologists need tools to both analyse the images and produce the diagnostic reports.

To analyse a study, the pathologist needs to have good navigational instruments that allow him to quickly explore portions of the slide. Similarly to the light microscope, the viewer must be able to pan through different locations in the image and apply different magnification levels to view in better detail portions of the image. The pathologist needs to assess rapidly its location in the slide and the current magnification level for a better experience navigating the slide.

Another important instrument in the analysis of a slide is the capacity to annotate the study with different types of visual annotations. These annotations must be capable of denoting regions of interest in the image, marking analysis steps, and measuring distances or angles in the image. A grid overlay with customizable dimensions is another important tool to help the pathologist referencing areas in the image and have a better idea of the region of interest dimensions.

Automatic image analysis is another important component in pathology workflows. The assessment of biomarkers is often required in the diagnosis of a pathology and that process cannot be done by the human eye. Thus, the viewer must provide ways for the pathologist to perform certain image analysis workflows on portions of the image (denoted by annotations) or over the whole image to calculate relevant image measurements for the patient diagnosis.

Finally, the writing of the report must be a semi-automatic task where minimal input from the pathologist is required. The export of image annotations and image measurements into the document needs to be supported to automate the process, requiring only from the pathologist the writing of the patient diagnosis in the report. To aid in the process, the pathologist will want to preview the final appearance of the report, so that option needs to be present in the viewer as well.

For a better experience visualizing the slide, the UI of the viewer needs to be adaptable to the pathologist's preferences as well as possible. Examples of this are the capacity to hide certain windows or reposition certain UI elements on the page.

The viewer needs to display the information of the study currently opened, namely the patient information and facility that uploaded the study. A window displaying all images associated with this study needs to be available so that the pathologist can switch between the different images of the study. It is common practice in pathology to produce different slides from the same tissue sample to help the pathologist in the diagnosis.

The viewer needs to have tools to navigate between different studies to report. The pathologist can open many studies from the inbox, so the viewer needs to be able to navigate between them.

To use the viewer in other contexts, such as integration in learning platforms, or integration of collaborative features for professional/educational segments, the viewer needs to expose an interface that controls some of its components. In a learning environment, for example, it should be possible to hide certain image annotations based on the user role (student or professor).

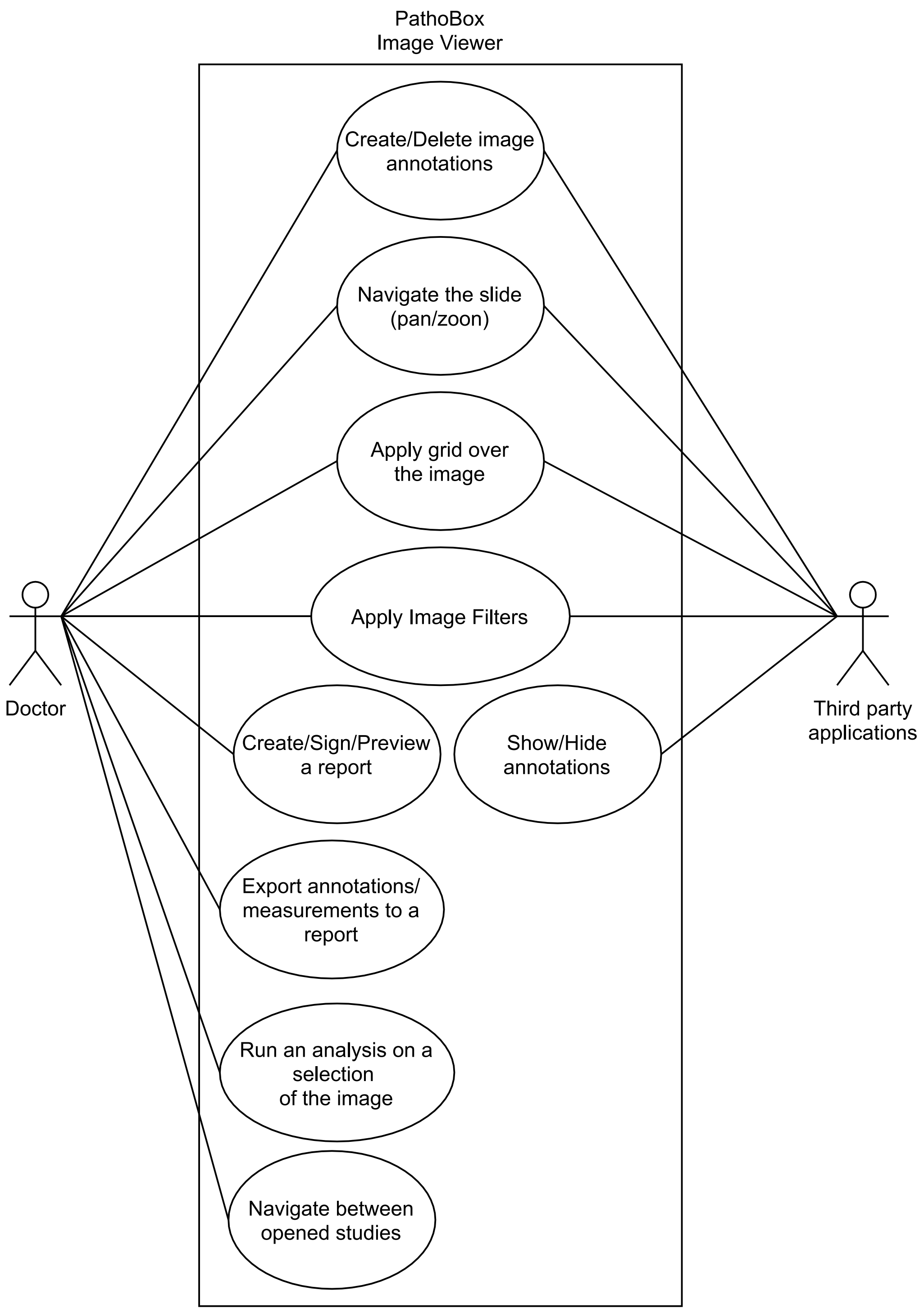

PathoBox
Image Viewer
Create/Delete image annotations
Navigate the slide (pan/zoon)
Apply grid over the image
Apply Image Filters
Create/Sign/Preview a report
Show/Hide annotations
Export annotations/ measurements to a report
Run an analysis on a selection of the image
Navigate between opened studies
Doctor
Third party applications

Fig. 10 - Image Viewer use case diagram

3.2. System Architecture

The architecture of PathoBox consists of three main entities: the PathoBox web client application, the PathoBox server application, and the Dicoogle PACS server (Fig. 11).

The system follows a client-server architecture. The webapp is the application the users interact with. It communicates with the server app to request its webpages, services, and data. The server app is where all the business logic is contained. It leverages the communication between the webapp and all third-party services including Dicoogle. Finally, Dicoogle is where the WSI images are stored. It includes a database for images metadata and provides an interface of services for the query and retrieval of DICOM files.

One instance of PathoBox can connect to different instances of Dicoogle, installed on different machines. The communications can be configured to work via a representational state transfer (REST) API or through a TCP (Transmission Control Protocol) socket. This modality facilitates archive sharing between different PathoBox installations and potentiates scalability and reliability since the application can connect to more than one archive.

The webapp communicates solely with the server app, even when retrieving the digital slides from Dicoogle. This creates a layer of abstraction between the user and the archives and allows the server to have full control of the communications, protecting the access to these images with the authentication and role systems in place.

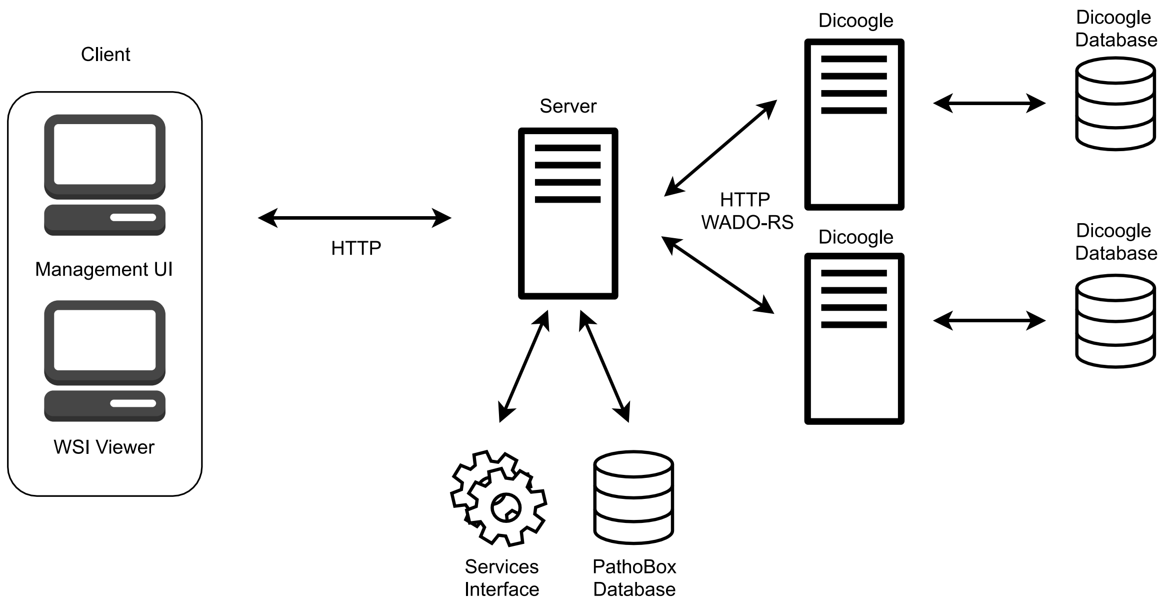


Fig. 11 - System architecture diagram

Both the frontend and backend applications follow the MVC (Model View Controller) pattern. This allows a direct mapping between the frontend and backend modules, helping with the project's organization.

The MVC pattern (Fig. 12) consists in a View, the component responsible to show the UI, the controller that is the component that controls the view and the model, that is the logic layer that connects the controller to the database [50].

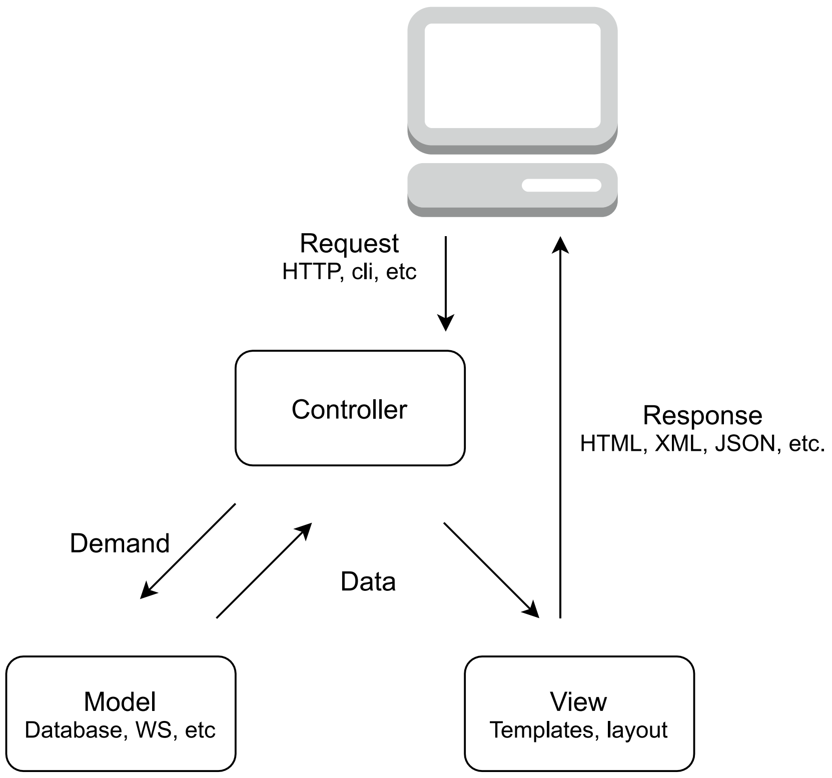


Fig. 12 - MVC pattern adapted from [50]

This structure allows a modular design that facilitates the edition of components without affecting other components. The development process took advantage of this feature to develop the project by modules, focusing on developing and testing one module at a time and connect them afterwards.

The component diagram depicted in Fig. 13 presents the layers of the system as well as its components. There is the client layer, the server layer that receives requests from the client layer through the router and finally the data source layer where the databases used in the system are present and the image analysis providers layer where the image analysis services are present.

Every request is routed internally to the right controller. It is at this step where the application verifies the user authentication and permissions to check if he has access to the requested route. The controllers are responsible for handling the user request. They can consume multiple services each with a very well-defined interface. For example, the ReportController uses the PdfService to generate a PDF document of a report from a case study. The database services are used to insert, retrieve, delete, and query data from the database. They must implement the JPAService interface to have access to the JPA entity manager that is responsible for the mapping of the database entries into Java objects. The ArchiveService is used to access the Dicoogle repositories, to insert and retrieve studies as well as to query metadata from Dicoogle.

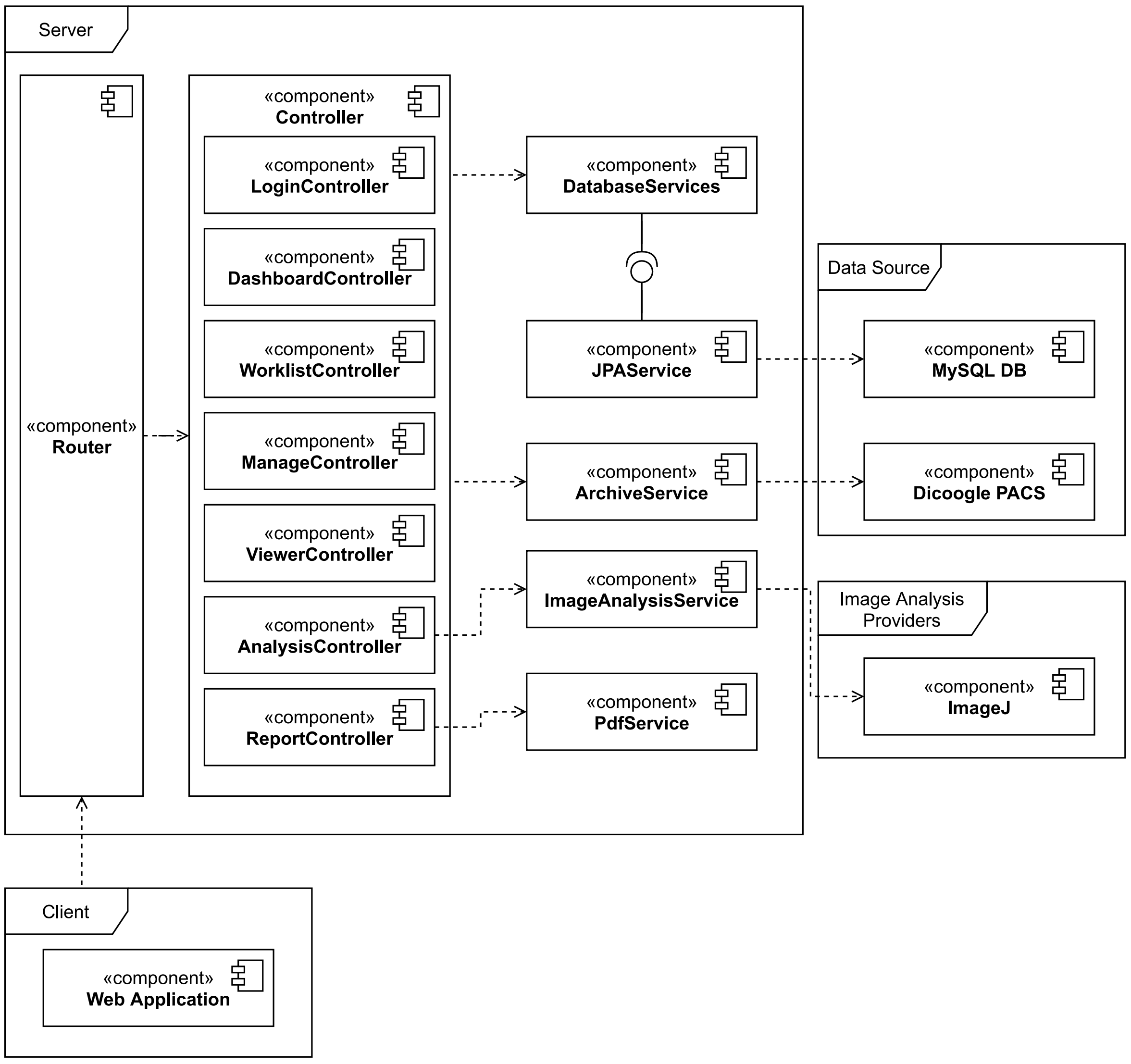

Server
«component» Router
«component» Controller
«component» LoginController
«component» DashboardController
«component» WorklistController
«component» ManageController
«component» ViewerController
«component» AnalysisController
«component» ReportController
«component» DatabaseServices
«component» JPAService
«component» ArchiveService
«component» ImageAnalysisService
«component» PdfService
Data Source
«component» MySQL DB
«component» Dicoogle PACS
Image Analysis Providers
«component» ImageJ
Client
«component» Web Application

Fig. 13 - System component diagram

3.3. System Technologies

This sub section presents an in-depth review of the main technologies used for the web application and server application.

3.3.1 Web Client Application

The choice of technologies for the web application was conditioned by three variants: performance, consistency with the products from BMD Software and design pattern.

It is known from previous works that the application needs to be fast, to provide good user experience, despite bringing gains for being a digital solution. If the application takes more than a few seconds to perform an action, the user experience will be poor and thus the clients will not be interested in the solution. It is also important to maintain some consistency with other projects from BMD Software so that future maintenance and development by new members to the project is made easier. It also helps with the re-usage of code and libraries if the projects use the same basis. Finally, the web application needs to be compliant with modern design patterns, given the advantages they bring to the development process and application performance.

With those three requirements in mind, a simple frontend framework based on RequireJS was used. RequireJS is an open source module and file loader, that provides the application with asynchronous module loading capabilities. This is done through the usage of the asynchronous module definition (AMD) specification. Each module declares the dependencies it needs and RequireJS takes care of loading those dependencies. Modules are loaded as they are requested by the client allowing the application to stay performant and simple even when a lot of plugins and libraries are used across the different modules [51]. This keeps loading times low and prevents the application from becoming confusing and entangled in Javascript libraries, keeping the code complexity from scaling too highly allowing future development on the platform to stay relatively simple. It also eases the developer’s work in maintaining and updating libraries, since they all need to be configured in a RequireJS configuration file. This gives easy access to all dependencies and libraries in use by the application.

The frontend application consists of a group of modules and controllers. A module is a group of controllers that interact with the views and the models, in accordance with the MVC pattern. All components on the frontend application are defined as RequireJS modules so that each component can declare its dependencies and be imported by other components. This allows components to be re-used across the application and can be asynchronously loaded. The definition of the modules and stylesheets used on each page is defined at the server level. The base HTML pages served by the server state which stylesheets and controllers it requires. Once a page is requested, the web application will read the HTML file to detect which modules and controllers to load. Once these modules are loaded, they can instantiate other modules that they need. RequireJS takes care of assembling the dependencies declared on the loaded components.

Native Javascript was used as the programming language. Handlebars was used to provide HTML templating functionality. CSS stylesheets and the bootstrap template AdminLTE[5] based on Twitter Bootstrap were used for styling the HTML pages.

Handlebars is an extension of Mustache[6]. It is a simple templating engine that works either in the client side or the server side. PathoBox compiles the templates server side and the data is only populated at the client application, since their content is dynamic. Handlebars works with the concept of context to refer to objects in the templates. It supports a basic set of directives such as if/else blocks and looping blocks. One of the most important aspects of Handlers is the capacity to define custom helpers that allows developers to extend Handlebars' functionality [52]. For example, a helper function can be created to format date objects that takes as an argument the type of format (short format, long format). The developer can use this helper to format date objects in the HTML automatically.

By combining these two technologies it was possible to implement the MVC pattern in a simple and performant way in the application. The framework can be easily extended by the developers given its modular architecture and the acquaintance the development team has with the code.

Angular framework was another option discussed for the frontend application. Despite being more advanced, this framework also introduces more liabilities and performance issues since the programmer does not have full control over the application's behaviour and might wrongfully apply certain Angular concepts. A wrong system architecture and poor implementation of certain Angular directives can also contribute to worse application performance [53]. Moreover, recent versions of Angular require a server to host the web application. This increases the complexity of the solution. Not only that, the chosen technologies keep the platform in line with the other platforms developed at BMD Software.

Angular was also considered since it was the application used in the development of the collaborative viewer presented in [9] so there was already some experience with the framework. It is also a very powerful tool that cuts down development time with features like two-way-binding and dependency injection. There is also plenty of literature and documentation about angular which makes it easy for new and unexperienced developers to create fast prototypes and simple applications. Angular grew to be one of the most popular solutions currently on the market for web development [54].

As for the web viewer, there are two main components that were used. OpenSeadragon for the visualization of the digital slides and Fabricjs to handle the construction of the image annotations.

OpenSeadragon is an opensource web viewer for zoomable images of high resolutions, implemented purely in Javascript, that works in both browsers and mobile phones. This technology allows the visualization of whole slide images thanks to its tile-based processing that allows only certain parts of the image to be downloaded at a time. OpenSeadragon can be controlled through an extensive API and can be extended through user made plugins [55].

[5] https://github.com/ColorlibHQ/AdminLTE
[6] https://mustache.github.io/

The fetching of the tiles is possible due to a custom tile source, implemented by our research group at University, that uses WADO-RS (Web Access to DICOM Objects REST Services) to fetch the tiles from the DICOM repository. OpenSeadragon will only load the tiles the user is currently viewing. As he pans to different zones of the image or zooms to different levels, the application will asynchronously load the new tiles. This provides a fluid experience with minimal delay in the loading of the tiles.

Fabricjs is an opensource canvas library. This component was used for the viewer annotations due to its natural integration with OpenSeadragon thanks to the OpenSeadragonFabricjsOverlay plugin, its simple API to create and control objects in the canvas and finally its hierarchical object based architecture that provides the tool with object oriented capabilities like sub classing [56].

Fabricjs' support for sub classing was a deciding factor for this technology, as it allows the extension of its base objects to have custom behaviours and controls. As an example, the Polygon object takes a set of points and draws a polygon on the canvas. The default behaviour does not allow edition of one single point, instead the user can only change the x and y dimensions of the whole shape. To address this issue, a custom class that extends the Polygon class can be designed to enable the edition of singular points while retaining all the characteristics of the base class.

The OpenSeadragonFabricjsOverlay plugin binds the Fabricjs canvas to the OpenSeadragon canvas. This means that when OpenSeadragon changes its canvas size, for instance in a zoom event, the Fabricjs canvas will be updated accordingly. This component was designed for an outdated version of Fabricjs. So, in order to take advantage of the new features and improvements of Fabricjs, as well as keeping the dependencies as updated as possible, an effort was made to alter the plugin to include support for the most recent version of Fabricjs without affecting the plugin's performance.

3.3.2 Server Application

One more time, the choice of technologies for the server-side application was affected by system performance and compatibility with other products from the company. It was important to select a Java framework for the server application given that Dicoogle is also written in Java and many other libraries of reference, that deal with DICOM files and medical image analysis, are also only available in Java.

Play Framework ended up being the choice, since it was already used in other projects at BMD. It is based on a lightweight, stateless, and web-friendly architecture in Java, built with Akka. Akka is a toolkit written in Scala that facilitates the development of distributed and concurrent software on the Java Virtual Machine (JVM) and provides applications with good and predictable scalability and minimal resource consumption.

Play Framework can compete with Spring Framework, another popular Java Framework, in terms of performance, but the easier setup of Play applications, live code reloading and being an open source project, makes it more appealing to use from a developer point of view [57].

MySQL was used as the database for this project for being a relational database that is both lightweight and powerful. It is free to use and resource efficient, meaning it can run well on a low-end server with moderate load [58].

Previous experiences with MongoDB [9], a document-oriented database, demonstrated that they work well for fast prototyping and simple applications. The natural integration of JSON (JavaScript Object Notation) objects is also a nice feature since it allows objects in the database to be directly loaded into the HTML templates without much work needed. However, the database is not very practical for complex data models since it was not built to handle complex aggregation queries and searches for non-key attributes. In products with a rigid and well-defined schema that do not benefit from the scheme-less feature of MongoDB, a relational database is the better choice [59].

Column-oriented databases are not applicable in this context either. The main advantage of such databases is the scalability with large amounts of data for reading operations. PathoBox is not a big data application, so those advantages are nullified. Moreover, PathoBox is an OLTP (Online Transaction Processing) system, so it requires fast writing speeds which a column-oriented database system cannot provide [60]. The consistency and reliability of a relational database plus the fact it can handle more complex data models and queries, as well as having a natural and seamless integration with Java's JPA framework makes relational databases the best choice. This does not exclude however the possibility of using secondary databases for answering specific requirements that relational databases cannot handle, like an in-memory cache database to speed up access to some data in the database.

To deploy the application, the Docker technology was used. It is an open platform for developing shipping and running applications. It uses Docker containers to encapsulate applications, abstracting them from the host machine where they will be installed. A Docker container is nothing more than a package with the application code and all its dependencies that is independent from the computing environment, allowing the deployment and running of applications in different environments with little effort needed. A Docker container is similar in functionality to a virtual machine.

The abstraction between the container and the computing unit is achieved through an extra layer that Docker adds that is responsible for the communication between the host machine's kernel and the container. While this extra layer might affect performance, Docker proved to be better than the other competing solutions. Its easiness of use plus the quality attributes it brings to the solution like portability and deployability made it one of the most popular and widely adopted solutions for application deployment [61].

The containers are created from a Docker image that specifies the contents of the container. The images usually are created from base images of operating systems or environments like the Java platform. Docker images can be hosted in a Docker registry, that is a Docker image repository. Images can be pushed or pulled from a single source.

3.4. Deployment Model

The deployment of PathoBox was done using a Docker network on a Linux virtual machine (VM) (Fig. 14). This is a simple deployment with only one instance of MySQL and Dicoogle meant for easy prototyping and testing of the solution.

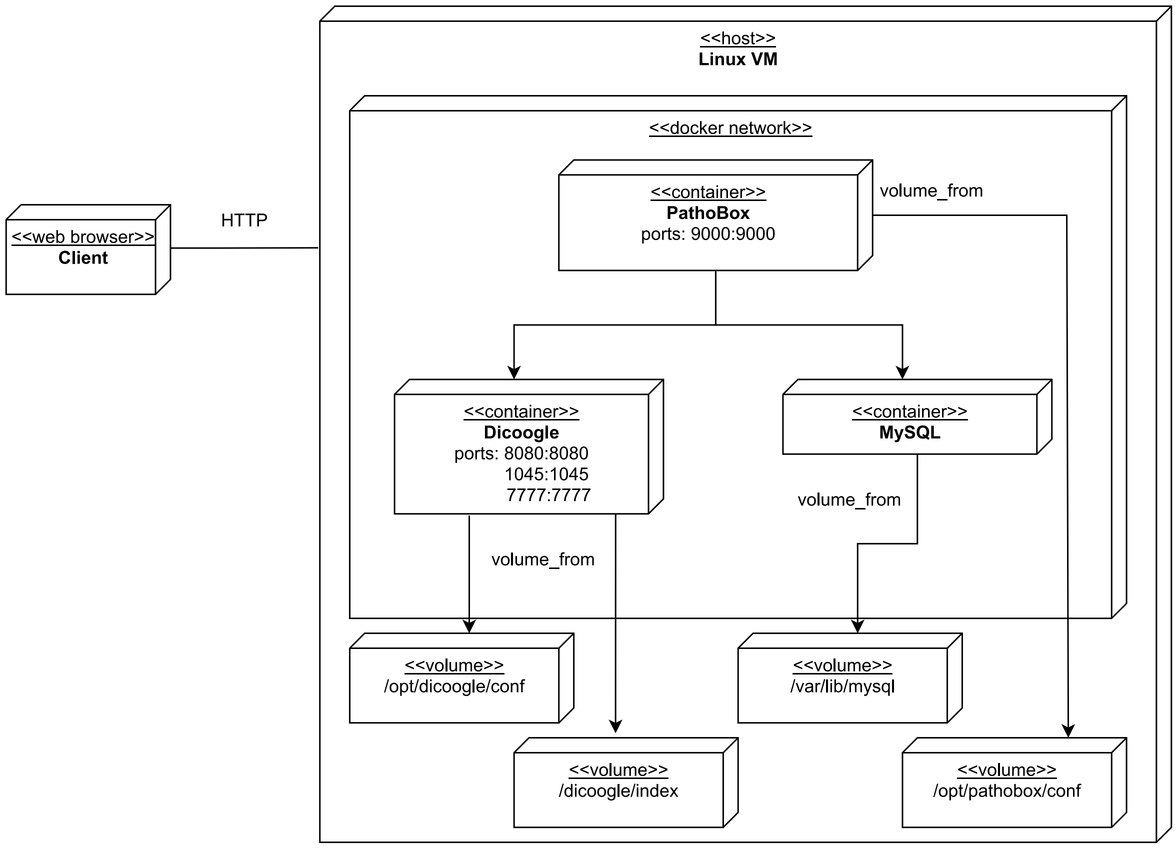


Fig. 14 - PathoBox local deployment diagram

The application was split into three containers: a container holding the PathoBox application, another for the Dicoogle PACS archive and finally one container for MySQL database engine. Each Docker container runs as if it was an isolated application, so it cannot communicate with other containers unless communication bridges are established. A Docker network was used to create a common environment for running containers. This architecture keeps the PathoBox application separated from its dependencies, making the update process of each individual component easier without affecting the portability and deployability of the system. For example, under this architecture, to deploy an update to PathoBox, all it is needed is to recreate its container with the new code. The other containers stay untouched and continue to operate as usual.

In a Docker network, dependencies can be created between containers. In this case, PathoBox needs to be able to establish a connection with both Dicoogle and MySQL so by declaring these two containers as dependencies, Docker makes sure that the two containers are instantiated first in the network and only then PathoBox.

To establish connections with the client, the connection ports of PathoBox and Dicoogle were exposed in the Docker containers. This creates a link between the host machine's ports and the Docker container ports. The client, by accessing the connection port on the host machine, in this case, the port 9000, is redirected to the PathoBox container where the web server is located.

The configuration files required for each container are mounted on folders located in the host machine. While this process creates an extra dependency on the host machine, reducing the application's deployability, it in turn provides flexibility in the configuration of the containers, allowing the setup of multiple machines each destined to different tasks. For instance, one machine can have production configurations while the other has testing configurations. This prevents the developer from keeping two different versions of the configuration files when deploying updates to the application. It is also a benefit of the capacity to change the configuration files without having to rebuild the containers. Moreover, in the case of MySQL container, by having the data mounted onto the host machine, it is possible to restart the container or even delete it without losing the data in the database. This is useful in continuous integration environments.

3.5. Class Model

The class model of PathoBox represented in the following class diagram (Fig. 15) can be categorized into five groups: Organization, User, RBAC, Case Study and DICOM.

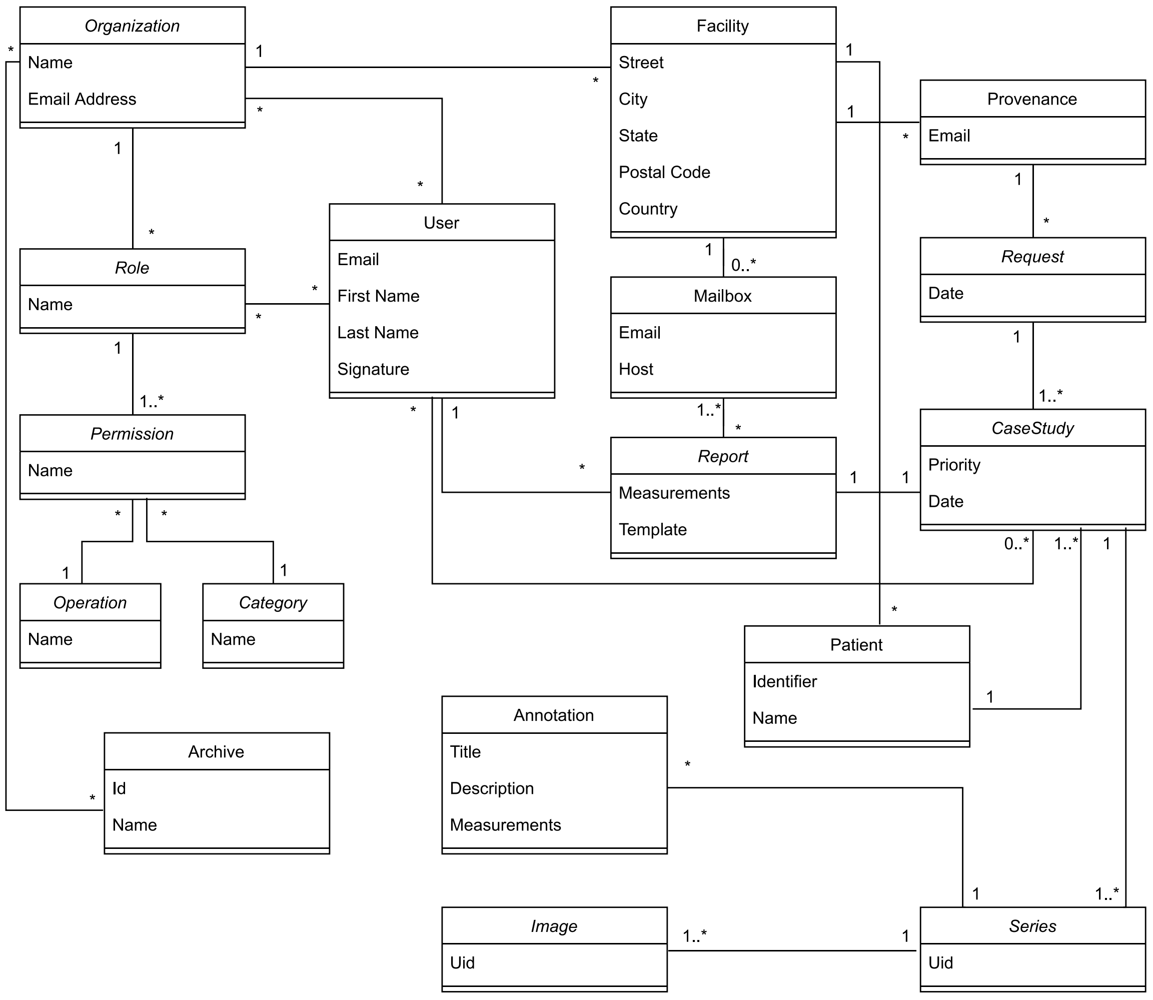


Fig. 15 - Simplified system class diagram

The Organization group refers to all the entities involved in the management of organizations and facilities as well as the upload and assign of case studies. The Organization entity represents a group of facilities that will use the platform. A Facility belongs to an organization and represents hospitals and clinics. Each facility has provenances that produce the case studies to report. These provenances can be digital scanners in the facility or health technicians that manually uploaded studies to the platform. Each facility has a group of mailboxes that can be used to collect the study reports produced.

Organizations can have one or more archives, that model Dicoogle instances. They are defined by a set of configurations that are used to determine the type of access to the archive (web services or TCP socket). The archive fields are described with detail in Table 3. If the access to Dicoogle is to be made via web services, then the qido wado and stow fields need to be set. Otherwise, if the connections are to be made via TCP sockets, then the IP and port addresses of the query storage and move services need to be set. If both these types of configurations are set, the system will use the TCP socket method.

The User group represents the platform users. The actions available to them in the platform are defined by their roles. They can assume the role of administrators of an organization, pathologists, health technicians or any combination of these roles. They can belong to a facility as a doctor or as a health technician and to an organization as an administrator of that organization. A user can belong to different organizations and different facilities, assuming different roles in each. By belonging to a facility, the user forcefully belongs to that facility's organization.

The RBAC group refers to the entities used to build the roles the users have in the platform. The roles are composed of a set of permissions. A permission is defined by a category and an operation. There are four basic operations: Create, Edit, View and Remove. As for categories, there is one category for every resource that needs to be protected. To name a few, there is Study category, Organization category, Report category, etc. The idea is to create as many categories as needed. The available Operations and Categories are defined in a configuration file.

The roles are created by organizations and are only applicable to users inside that organization. This means that a user that does not belong to an organization cannot have roles from that organization. A user can have many roles from different organizations however the access to resources of one organization is only defined by the roles the user has in that organization. A role is unique within an organization and must have a name and a list of permissions.

The CaseStudy group consists of all the entities used to represent and manage case studies in the system. The CaseStudy class represents a DICOM study. All studies belong to a request object. The requests are performed by provenance from a facility upon the upload of new studies to the platform. Each case study has one report linked to it. Users can be assigned case studies, gaining access to write/edit its report. To note, however, that those users cannot edit or delete the case studies that they were assigned to. A different set of permissions is necessary for those actions.

Finally, there is the DICOM group that refers to the entities present in a DICOM file. Each case study has a set of series that represent the many different slides that one study may have. Each series has a set of images that represent the layers of the WSI pyramid, and a set of image annotations created in the viewer (Table 4). The patients are defined in the DICOM files and can be linked to several studies. The patients are registered in the facility that uploaded their study.

Table 3 - Archive model fields

Field	Feature	Description
Name	Name	Name of the Archive
qrAETitle	AET Title	Title of the application entity responsible for the query retrieval service
qrIP	IP address	IP address of the query retrieval service
qrPort	Port address	Port address of the query retrieval service
qido	Endpoint	Endpoint for the QIDO service
wado	Endpoint	Endpoint for the WADO service
moveAETitle	AET Title	Title of the application entity responsible for the move service
moveIP	IP address	IP address of the move service
morePort	Port address	Port address of the move service
storageIP	IP address	IP address of the storage service
storagePort	Port address	Port address of the storage service
stow	Endpoint	Endpoint for the stow service

Table 4 - Annotation model fields

Field	Feature	Description
id	Identifier	Unique database identifier
series_id	Identifier	Identifier of the series this annotation belongs to
user_id	Identifier	Identifier of the user who created this annotation
title	Title	Annotation title
annotationDescription	Description	Descriptive text of the annotation
controller	Annotation Controller	JSON string that contains the necessary annotation details to draw the annotation in the viewer
color	Color	Annotation Color
measurements	Image Analysis Measurements	JSON string that contains measurements from image analysis procedures
attachReport	Attach to Report	Flag to import this annotation to the report
attachImage	Attach Image	Flag to import image selection to the report
selection	Image Selection	The pixel data of the cropped image from the viewer

4. PathoBox Implementation

This section describes in detail how the many components and features of the platform were developed. The objective of PathoBox is to answer the market needs and create a management platform for digital slide diagnosis, based on web technologies, that supports the DICOM standard and allows facilities to upload case studies for being reported by pathologists registered in the platform.

4.1. User Authorization and Authentication

Users in the platform can be created in two ways: through a configuration file or through the web application by a user with the necessary permissions. The application assumes that at least one user, the moderator, will be configured initially. Both through the configuration file and through the web application, an email and authentication provider need to be set. By default, the application supports a simple local authentication provider with a password and email. Different authentication providers can be configured through the configuration file or web application. Fig. 16 showcases a usual login through an authentication provider.

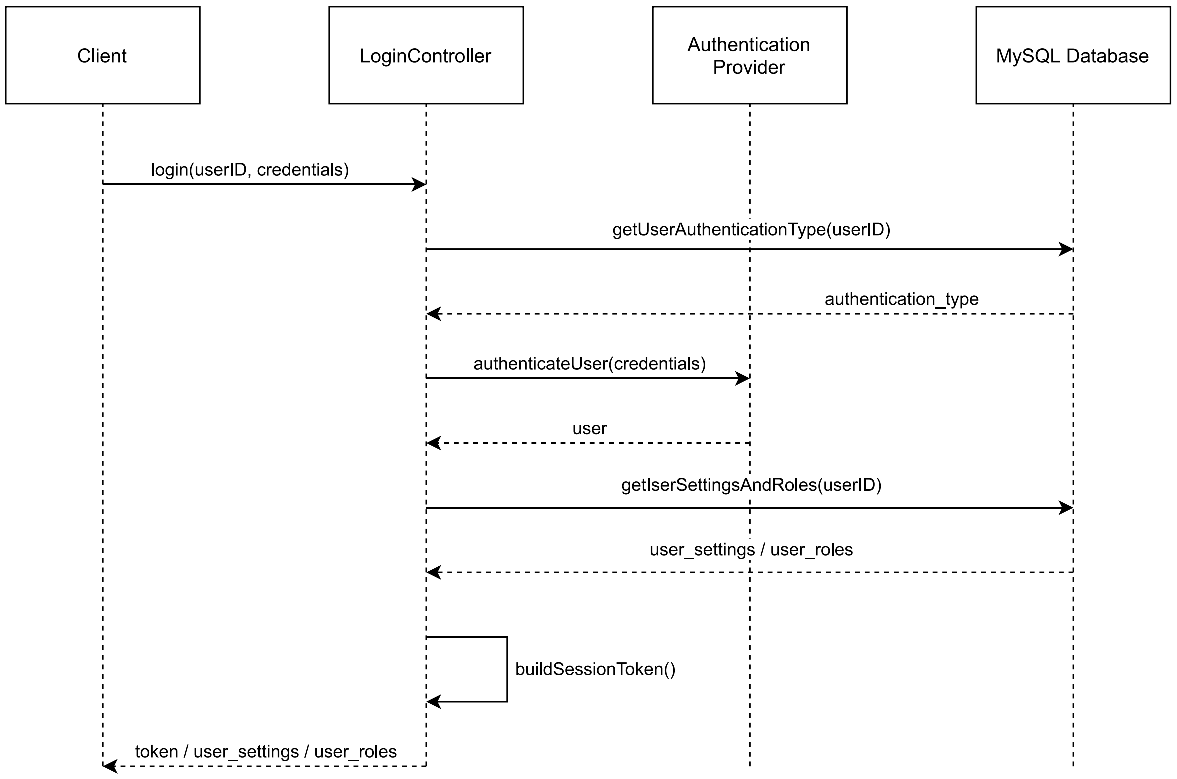

Client
LoginController
Authentication Provider
MySQL Database
login(userID, credentials)
getUserAuthenticationType(userID)
authentication_type
authenticateUser(credentials)
user
getIserSettingsAndRoles(userID)
user_settings / user_roles
buildSessionToken()
token / user_settings / user_roles

Fig. 16 - Sequence diagram for user login

Each organization has its own authentication providers and they can assign to each user one of those providers. This allows organizations to use their own providers with little effort, which in turn allows the platform to stay compliant with the organization's protocols.

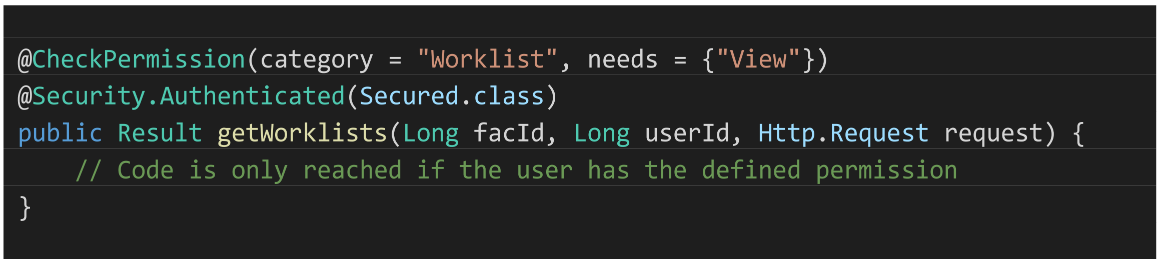
The default local provider takes the user password and email and compares it to the corresponding password hash in the database. In case the user was assigned a different provider, the server simply sends a request and waits for the validation of the user credentials.

All routes in the application are protected and require the user to perform a login in order to be used. Once logged in, the user session is handled by the framework. It is a stateless session that uses cookies to verify the logged in user. The session expires after a configured time and the user will be prompted to refresh his session.

4.2. User Roles, Settings and Preferences

Once the user logs in, the server sends to the web application his user roles and settings. The data is stored in memory until the user closes or leaves the application. The user roles will determine which parts of the application the user has access to while the settings control the look and feel of the UI. The user roles are used at two different levels: at the request level and at the client level.

At a request level (Fig. 17), the roles are used to deny the user access to a specific request. It can be a GET request for data or a HTML page or a POST request to store some data in the server. Before the route is executed, the application checks if the user has the specified access permission. In the example, route “getWorklists” requires the user to have the operation “View” under the “Worklist” category. The application checks the user roles and their permissions and checks if this combination exists in any of them. If it does not have the necessary permission, the application throws an exception and logs out the user, redirecting it to the login page. This system allows the protection of resources at a server level.



```
@CheckPermission(category = "Worklist", needs = {"View"})
@Security.Authenticated(Secured.class)
public Result getWorklists(Long facId, Long userId, Http.Request request) {
    // Code is only reached if the user has the defined permission
}
```

Fig. 17 - Example of role permission blocking access to a GET request

To reflect the access permissions at the server level, these are also applied at the client level, by controlling the presence of the affected elements. For example, if the user needs permissions to access the Worklist page but does not have them, then it does not make sense showing a link in the navbar to this page only for the user to be logged out once he tries to access that page (Fig. 18).

Some pages of the application contain actions that affect more than one category. In the worklist page, for instance, the user has access to a list of studies to assign, with the possibility to delete them. In this case, it is not enough to protect the route “getWorklists” but is also necessary to protect individual elements on the page like, for instance, the delete buttons. So, even if a user has access to a page, it does not mean it will have access to everything that is available on that page.

```
<li class="nav-item" view-needs="Worklist-View">
    <a href="{{routes.worklist}}" class="nav-link">
        <i class="fas fa-list"></i>
        {{messages.[navbar.worklist]}}
    </a>
</li>
```

Fig. 18 - Example of a role permission preventing a UI element from being rendered

Effectively, for most actions in the platform, the user roles are checked twice. Once at the UI level, that controls access/privileges to the application components and once at the request level when the web application requests access to a resource or operation, such as uploading a study, that is protected.

The roles, just like users, can be created through a configuration file (Fig. 19) or through the web application by a user with the necessary permissions. In the illustration, the role is named “Doctor” and has one permission object. The permission object has the category CaseStudy and possesses all available operations. Each user that is assigned this role will have full access to all actions regarding case studies.

```
{
    "name": "Doctor",
    "permissions": [
        {
          "category": "CaseStudy",
          "operations": [
            "Create",
            "Edit",
            "View",
            "Remove"
          ]
        }
      ]
}
```

Fig. 19 - Simple example of a role that defines access permissions to the Case Study resource

The permissions of a role are additive, that is, if a user has two roles, where one grants him permission to edit reports and the other grants him permission to delete them, they add up granting the user permissions to edit and delete reports.

While the names of the roles must be different, their permissions can be repeated. If one role has create, edit, and delete permissions over a certain resource and another role has only edit and delete permissions, one user can still be assigned these two roles, retaining all three permissions mentioned. This system allows flexibility in the creation of roles, allowing for mix and match of different roles to create many different privilege profiles.

The user settings are used to control certain aspects of the application like the position of elements in the web page, the application language and whether to show or hide some UI components. The settings were designed to allow the user to customize the UI so that the application can better suit his preferences and make him feel more comfortable with the application which in turn increases his productivity. The settings are loaded into the session data upon user login. The system was designed to easily implement new settings to expand the customization of the platform. The user settings assume default values when new users are created. The settings can be changed by the user on the platform.

The user also has access to an account page where he can change his login credentials, upload a signature to the platform and check his privileges in the platform. The signature is used when the user signs the case study reports. One is needed to edit/deliver the report.

4.3. Upload and Assign of Case Studies

The upload of case studies is a complex procedure in the platform since it interacts with a lot of different components. Fig. 20 showcases a flowchart diagram for file upload.

In the first iteration, the upload is manual requiring the user to log in into the platform and select which cases he wants to upload. The user can upload a zip file or the files individually. The application accepts files from a list of compatible formats, including DICOM. The upload has no size limit since studies can reach considerable volumes of data, ranging from hundreds of MB(Megabytes) to some GB(Gigabytes) and the user can upload more than one file at once.

Large file uploads over the internet pose many issues for the user. If he temporarily loses connection, the upload might fail or get corrupted. If the upload fails, the user has to re-upload the file and given their size, unless the user has a fast internet connection, the long waiting times will result in frustrating user experience. If the upload for some reason gets corrupted, the application will have to reject the file since the data is unusable, thus requiring its re-upload.

To support all these scenarios, the web application uploads the files to the server over the HTTP protocol by chunks when they exceed a certain size threshold. The plugin Dropzone.js was used to take care of the file chunking. It takes as arguments the size threshold, used to decide whether a file should be split into chunks or not, the number of chunks to upload in parallel and the number of retries for each chunk if the upload fails. The plugin also supports several callbacks to deal with the many states of the upload. Namely, a callback for when a chunk is uploaded with success and when all chunks were uploaded. This information is used to display the upload status, i.e. to give feedback of the upload process to the user. The user can upload the file(s) via drag and drop or he can choose the file(s) through the file explorer.

Once the user selects a file to upload, the application will check its size do decide whether it needs chunking or not. If the file needs to be split into parts, the application will add a special header to the request used to identify the upload as a chunked one. When all chunks have been uploaded, the application will send a message to the server, with the identity of the file, informing it of that fact.

If the file needs to be chunked, the application will query the server, before upload, to ask if the file already had any uploaded chunks. This allows the application to keep track of the upload progress of each individual file when the upload is interrupted. If a file is split into ten chunks for upload, and the upload is interrupted whilst uploading the fourth chunk, the application will not have to upload the first three chunks again. The user needs to manually restart the upload if it was interrupted.

The server, depending on the headers that come with the uploaded data, will be able to determine if the upload is a chunked file or not. If the file uploaded is a chunked one, the server stores this file in a temporary folder, under a unique name. The name of the folder consists of the name of the file, the timestamp when the upload started and the name of the user who uploaded the file. The username is added to support the pause and cancel operations. The name of the file is a hash combined with its index in the upload.

Once the server receives confirmation that the last chunk of the file was sent, it triggers an action that will merge all the chunks in a new file that will be uploaded to Dicoogle. If the file uploaded is not a DICOM file, Dicoogle will “dicomize” it, i.e. it will transform the file into a DICOM file. This operation is not supported for all formats, so the application cancels the upload if the user tries to upload a file format that is not supported.

To allow the user management of its internet connection bandwidth, pause/resume actions were implemented in the upload so that the user has some control over the process. He can stop the upload preventing it from consuming all network resources and can resume it when the user no longer has that concern. The user can also cancel the process if some mistake occurs with file selection or simply wants to abort the operation. If the upload of a chunked file is canceled, the server will delete all chunks uploaded up until that point.

Once the upload is complete, the application will query Dicoogle for the resulting metadata to build the necessary entities. Each upload will correspond to a request from a provenance in the database. Each request has one or more case studies to review. If the user uploads two case studies at once, then a request with two studies will be created. In this case, since the upload is manual, the provenance will correspond to the logged in user. However, it could be a scanner that automatically puts the results of the scan in a folder to upload to the platform.

From the DICOM file, the application will fetch the patient’s data, relevant study fields like modality and time of the scan and will fetch the study series and their images. The application will create a new patient in the database if the patient in the metadata does not yet exist. Otherwise, it will just add the uploaded studies to the patient’s records. The reason the entities are added to the database only after being uploaded to Dicoogle is to guarantee that when the study appears in the application, the images are ready to use in Dicoogle.

Finally, the user will receive a status message once the operation is completed and the uploaded studies will be available for assigning.

With the upload complete, the user will have available in the worklist page under the unassigned section the newly uploaded studies. The user can select one or more studies and choose the doctors he wants to assign these studies to. Once the assigning is complete, the server will add the studies to the selected doctors’ inbox so that the next time they log in, they will be available for reporting. The user can unassign the studies from doctors if the study has not already been delivered. In this scenario, the report in progress and its associated data is not deleted.

One study may be assigned to more than one doctor at a time, despite only having one report. The report keeps track of a version number that indicates how many times it was changed and only keeps one signature, from the latest edition. This feature is meant to give the possibility for facilities to have more than one doctor working on a study.

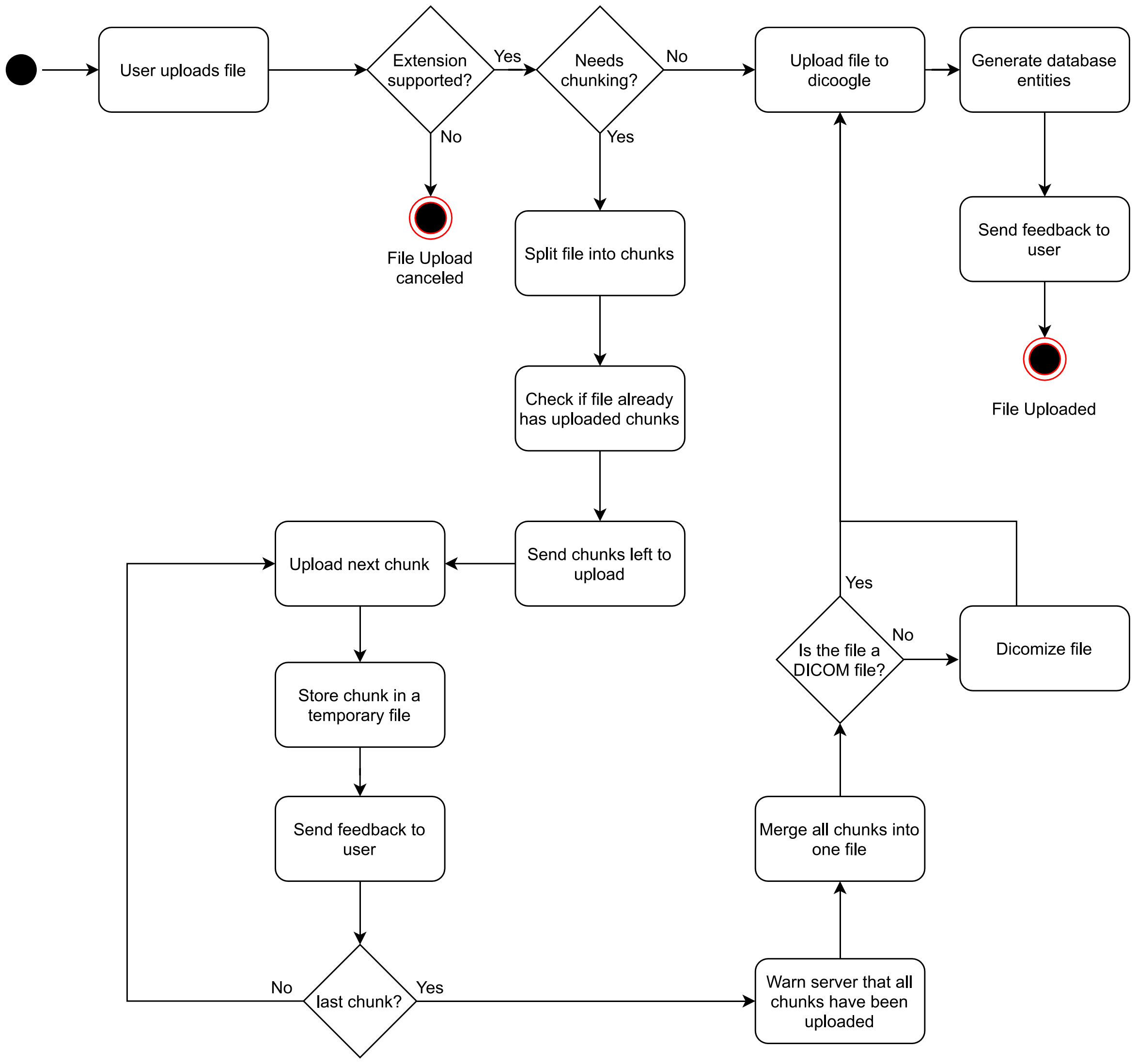


Fig. 20 - Flowchart diagram for a file upload

4.4. Case Studies Inbox and Archive

The inbox page contains the studies to report, assigned to the logged in user. The search page allows users to search for case studies uploaded to the platform. Both pages present the results in a tabular format where it is displayed information about the study and the patient.

The inbox only presents studies that have yet not been reported. This classification is done through a report state that determines its current state. The report may not exist yet, representing the not created state, may already be created but not delivered, representing the not delivered state and finally the report may be created and delivered, representing the delivered state. Studies whose report's state is delivered will not show up in this page. These states are used to determine which actions are available in the inbox page table.

Studies in the inbox can be grouped by priority, request, or provenance. The user can switch between facilities to display their respective requests if he belongs to multiple facilities. The priority filter allows the user to see the requests ordered by its priority, with the highest ones on the top of the list. Their priority can be changed by the user from the table.

Fig. 21 shows the sequence diagram when the user opens the inbox page. The application sends to the server the facility id (from the facility dropdown list) to know which facility to fetch the studies from. First, the server assesses the permissions of this user over the reports from identified facility. This is done by querying the roles from the organization that the facility belongs, to check if the user has a role from that organization that allows him to control the reports. Next, the server fetches the available facility requests, from the database, which contain the studies to review. The studies are filtered by user id so that only the studies assigned to this user are returned. After that, the application will iterate over the studies to assess the report state. If the report has not been created or delivered yet, the study is added to the response. The interaction finishes with the server returning the studies to the application.

The actions available, for each study, in this platform allow the user to open the study and deliver, preview, or delete its report (if one exists). If a report is deleted, its state returns to the not created state. When a study is opened in the viewer, the application will store in memory the current list of studies displayed in the inbox. This action allows the viewer to keep the order of the studies displayed to the user. For instance, if the user filters the studies by the highest priority and then orders them by name, the display order in the viewer will keep those settings, opening the studies with only the highest priority and in alphabetical order. This feature enhances the usability of the application by allowing the user to have full control over the way studies are opened and displayed in the viewer. The viewer is also aware of the index of the opened study. If the user opens the third study in the list, the viewer will allow the user to go back to the second study or proceed to the fourth.

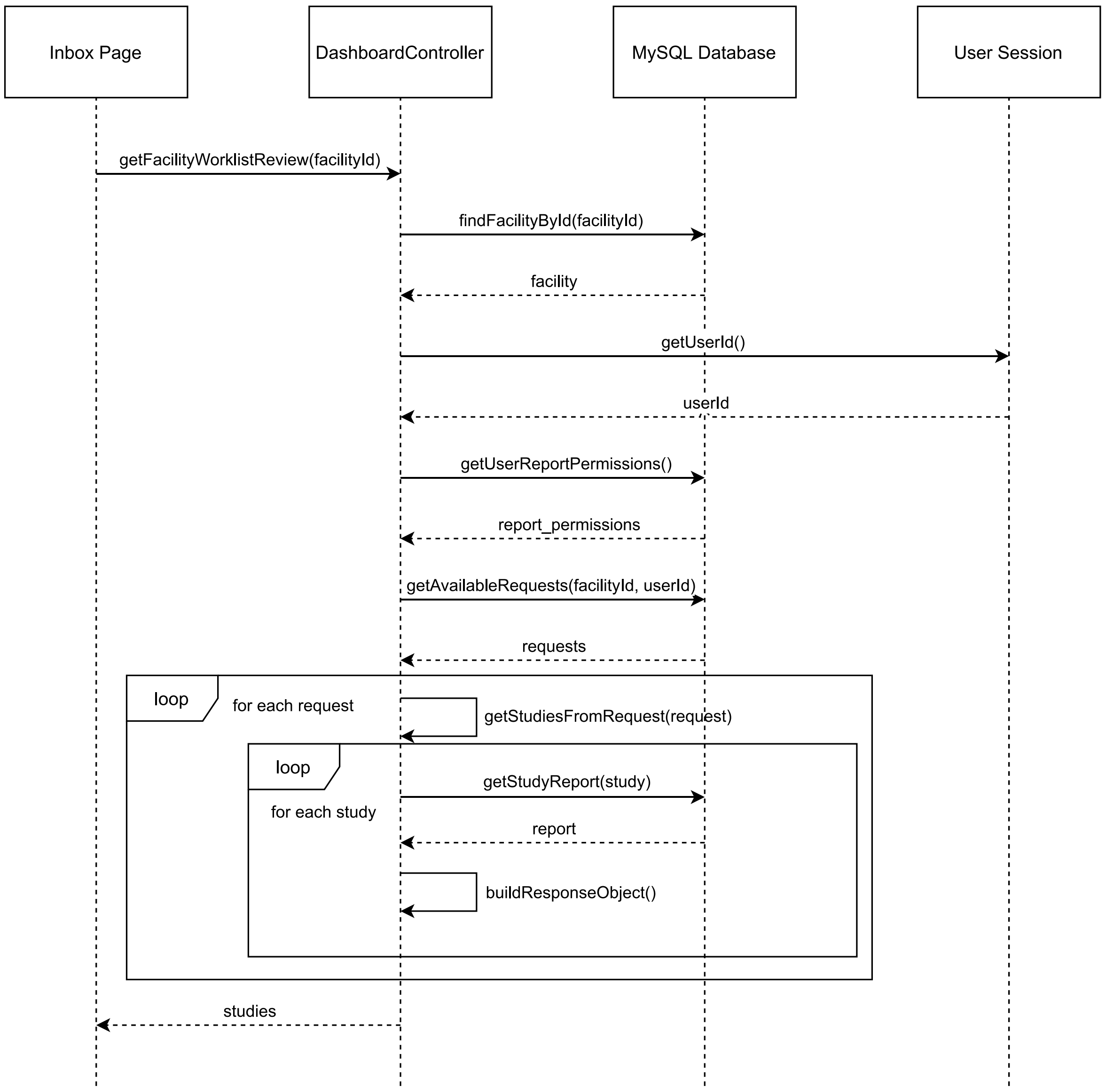


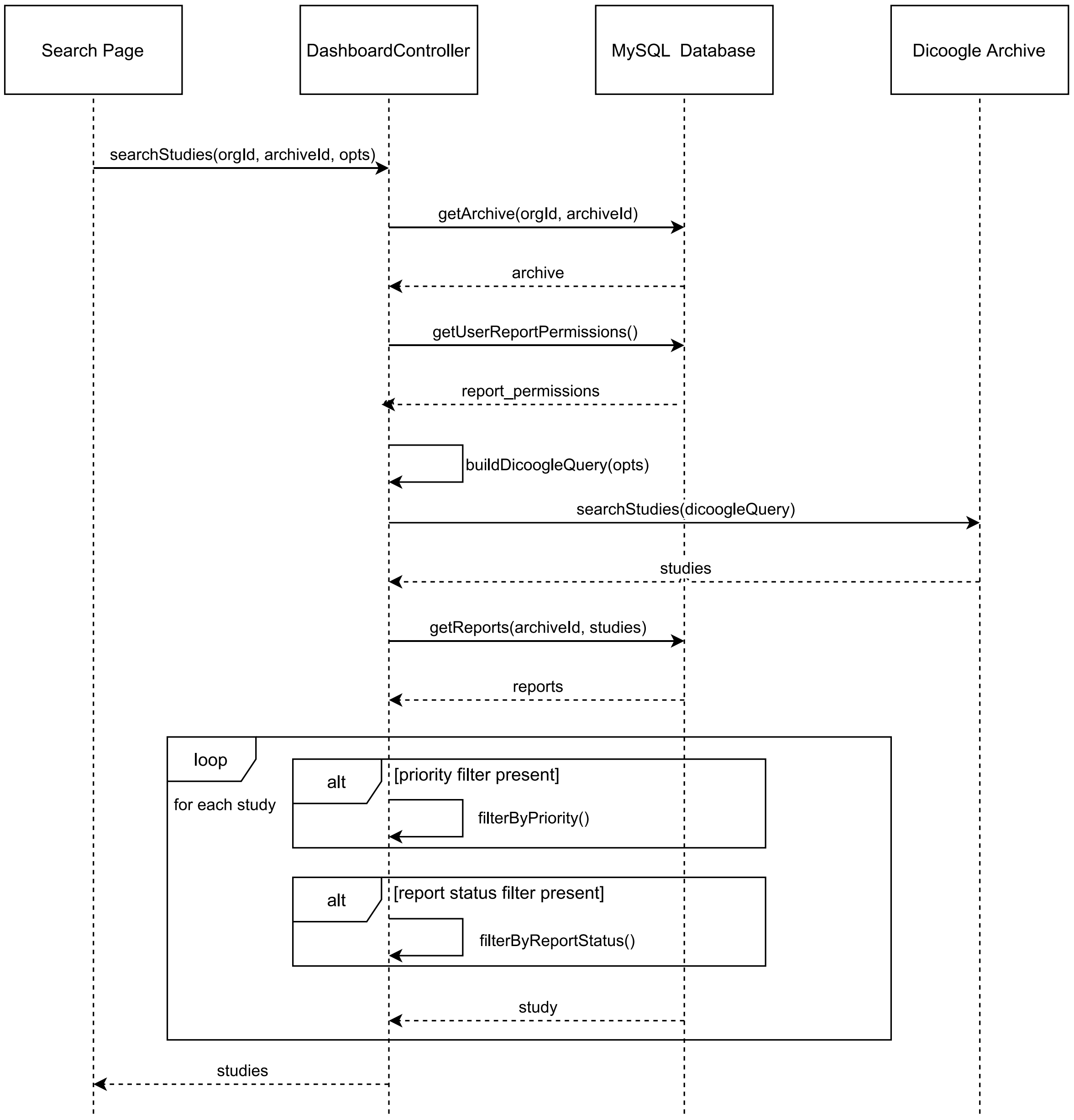
Fig. 21 - Case Studies Inbox sequence diagram

In the archive page, the user has access to different filters to search for studies, like patient name, the time interval of the upload/creation of the study, study fields, and more. The user can open a study from this page in the viewer but cannot make changes to its report if the study is already reported.

The following sequence diagram (Fig. 22) refers to a search action. The application sends to the server the organization id, the archive id to use in the search, and an options object containing the fields of the search. The server will get the corresponding archive that will have the necessary methods to connect to its physical counterpart, a Dicoogle installation. The search fields are formatted to match the DICOM formats. Once that process is done, the server will query Dicoogle for the studies that match the given query. A list of DICOM tags is sent along with the query to specify the response fields, presented in the response. Dicoogle will add those fields if they exist in the DICOM files metadata. to the response. The search for the studies is done at

Dicoogle since PathoBox does not store all metadata that can be present in a DICOM file. It is possible with this structure to easily extend the search tool to include more search fields.

Once the server gets the response, it iterates over the studies to filter by the remaining search fields, like priority and report state. Since Dicoogle does not have access to this information, this process needs to be done on the server. The user permissions need to be assessed for each study because the search page also contains actions to control the reports. It is important to block the sign/deliver actions on this page to avoid user access to studies that are not assigned to him. The interaction finishes with the server returning, to the web application, the list of studies that matched the search criteria.



Search Page
DashboardController
MySQL Database
Dicoogle Archive
searchStudies(orgId, archiveId, opts)
getArchive(orgId, archiveId)
archive
getUserReportPermissions()
report_permissions
buildDicoogleQuery(opts)
searchStudies(dicoogleQuery)
studies
getReports(archiveId, studies)
reports
loop
for each study
alt
[priority filter present]
filterByPriority()
alt
[report status filter present]
filterByReportStatus()
study
studies

Fig. 22 - Search of case studies sequence diagram

4.5. *Web Viewer*

The Web viewer is a workstation where the user can view and report case studies. As mentioned, if the user accesses the viewer from the inbox page, he will be able to navigate through the studies to review within the viewer. This improves the reporting workflow since it reduces to a minimum the interactions with the application. All actions are one click away from the user.

4.5.1 Viewer Architecture

The web viewer, represented in Fig. 23 component diagram, is split into multiple components and services each with their own responsibilities. The study window is the component responsible for displaying the study information, the study report as well as its actions and the list of series of this study. This component interacts with the viewer service to open new series or new studies in the viewer through the navigation keys that allow the user to go to the next or previous study in the list and with the image analysis service to perform analysis over the whole image.

The viewer service is the central piece of the viewer responsible for orchestrating the interactions of the components. It stores in memory the list of studies received as input, a reference to the PathologyViewer component, that contains the OpenSeadragon viewer and the toolbar, and a reference to the study currently opened. Once the viewer service opens a new study, the viewer will instantiate a canvas overlay that the user will interact with to draw image annotations. This overlay is also used by other components to add any sort of graphic element to the viewer, like a grid overlay. This canvas is updated by the viewer whenever its viewport changes, upon a pan or zoom action. All the annotations on the canvas are updated accordingly to reflect the new viewport. The size of the annotations must match the current image resolution.

The canvas overlay communicates with the annotation service to manage the annotation logic. It is through this service that annotations are stored and retrieved from the database. The service also takes care of updating the annotations, for example, when the user changes their color. The canvas overlay is also responsible for managing the annotation window.

This window is responsible for displaying the annotation data of the opened annotation. When this window is opened, the viewer enters a "drawing" state, disabling interactions with the viewer and enabling interactions with the annotations. It is through this window that the user has access to commands specific to annotations, such as attaching the annotation to the report or run an image analysis procedure. Once the user is done with the annotation, the component relays back to the annotation service the changes made that stores them in the server. The service in turn communicates the changes to the canvas overlay, updating the look of the annotations. The annotation window communicates directly with the image analysis service when the user intends to run an analysis on the annotation. This service has no logic associated with it and acts as an interface for the available backend image analysis services.

Finally, the report service is used by the study window and acts as an interface for the backend services. This service is used by the application to sign a report, edit its contents, or delete it.

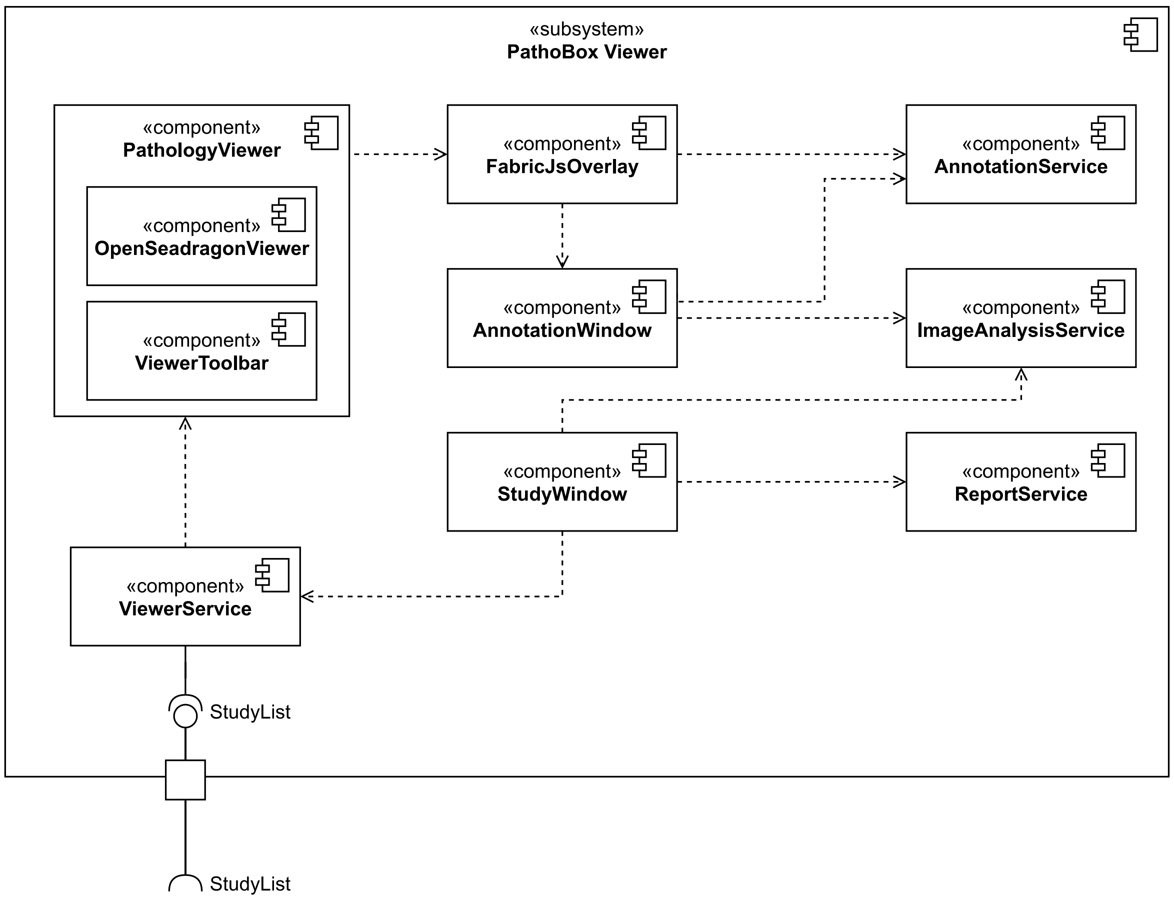


Fig. 23 - Web Viewer component diagram

4.5.2 Viewer Integration

The viewer was designed with third-party integration in mind. As the component diagram depicts, the only dependency is the study list input that determines the studies to open in the viewer. This list can be passed to the viewer programmatically or through the Unique Resource Locator (URL) used to open the viewer page. In the latter case, only one study can be passed as an argument and a list of one study will be created to serve as input. The integration of the viewer in this first stage was designed to be used in conjunction with a web socket. Native implementation of the viewer in other web applications will be limited by the available technologies. The current architecture makes it possible to package the viewer in a standalone package, to be used as a Javascript plugin. However, some extra work is required to adapt the Handlebars HTML templates used. Fig. 24 shows how the integration using Iframe and a web socket would take place.

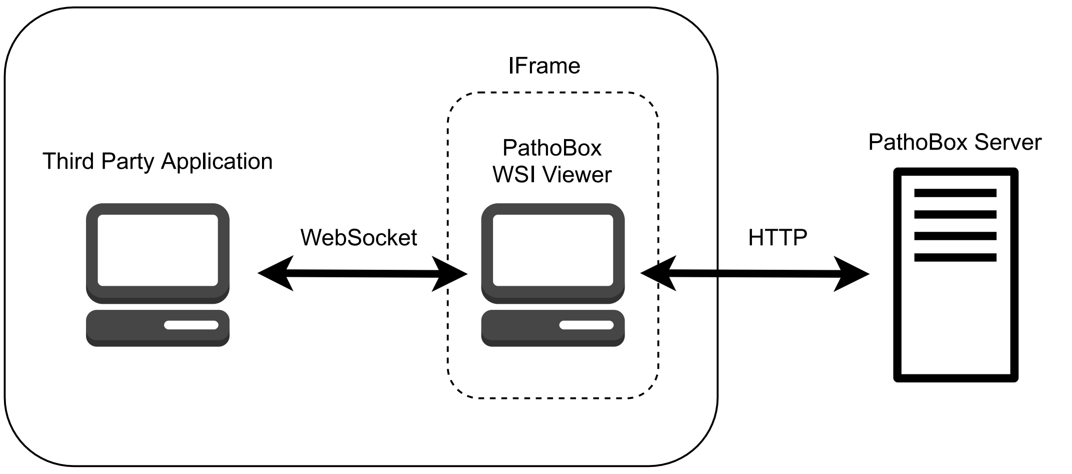


Fig. 24 - Example of viewer integration through Iframe

4.5.3 Viewer Annotations

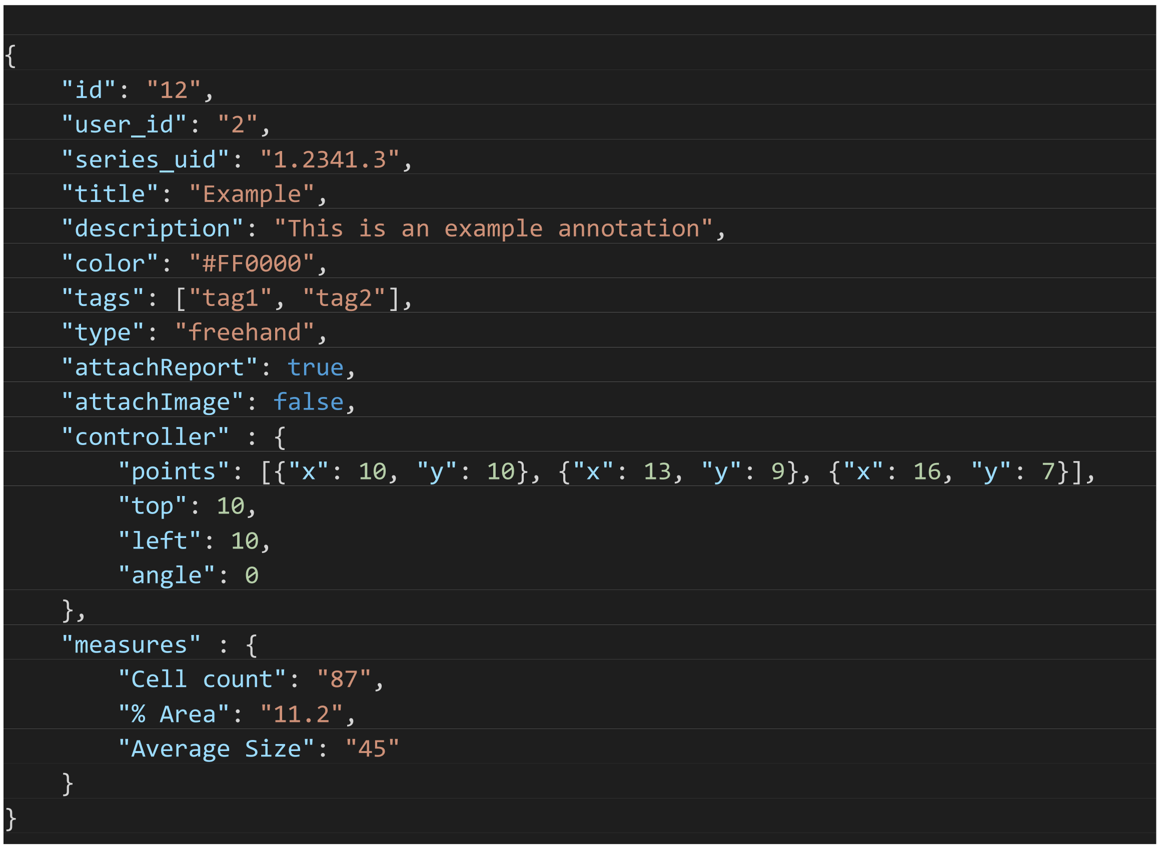
The annotation tools allow the user to create persistent image annotations on the study. The annotations done in the images are stored at a series level, which means other users will be able to see them. A study with two series will have two distinct sets of annotations, one for each series.

The following annotations were implemented:

- Measure annotation: This annotation allows the user to create measures on the image. It provides a ruler over the image and is able to calculate the distance between two points. This annotation is important because, for instance, it allows doctors to measure lesions on the tissue or even measure other annotations. There are also angle annotations that allow users to measure angles on the image;
- Shape annotation: This annotation allows the creation of an area in the image. The purpose of this annotation is to allow doctors to select an area of the image to either crop it to a report or run an analysis algorithm over the selected area. For example, this type of annotation can be used to delimit an area of the image where a lesion is located. That area can be exported to the report and the pathologist can append a short description. Currently, the following shapes where implemented: rectangle, freehand polygon, and freehand line;
- Counter annotation: This annotation allows the user to place a counter on the image. This annotation is useful, especially in a collaborative or teaching environment. It allows the user to mark steps on the image so that they can be followed. For example, to mark the steps of a diagnosis, the doctor can use this annotation to mark the order in which he analyzed the image and created his annotations. All the counter annotations on an image have an order, and if one of those annotations is deleted, the remaining counter annotations are updated accordingly;
- Marker annotation: This annotation is similar to the counter annotation, but its usage serves a different purpose. A marker, unlike a counter annotation, does not have an order. The purpose is to denote regions of interest that are not necessarily related. Because the markers have a unique shape, they are

easily identified in the image. These annotations are added to the navigator window as well, for easy recognition.

All annotations share a base class that defines their common attributes, a controller object that specifies the information unique to that annotation and a measures object that holds the results of an image analysis procedure. (Fig. 25). The common attributes are saved in the database as unique fields while the controller and measures object are saved as a JSON string in the database.



```
{
    "id": "12",
    "user_id": "2",
    "series_uid": "1.2341.3",
    "title": "Example",
    "description": "This is an example annotation",
    "color": "#FF0000",
    "tags": ["tag1", "tag2"],
    "type": "freehand",
    "attachReport": true,
    "attachImage": false,
    "controller" : {
        "points": [{"x": 10, "y": 10}, {"x": 13, "y": 9}, {"x": 16, "y": 7}],
        "top": 10,
        "left": 10,
        "angle": 0
    },
    "measures" : {
        "Cell count": "87",
        "% Area": "11.2",
        "Average Size": "45"
    }
}
```

Fig. 25 - Simple example of a freehand´s annotation JSON representation.

The figure displays a simple example of a freehand annotation, where the controller object was heavily simplified for demonstration purposes. All annotations possess three ids that identify the annotation, the user who created the annotation and the image where this annotation was made.

The controller object defines the annotation itself, containing all the information necessary to draw the annotation in the viewer. The controller always has information to determine the position of the annotation in the image, its rotation, and size. In the case of a rectangle annotation, only the top, left coordinates, width, and length of the annotation are necessary to draw it in the viewer. The polygon annotation, for example, only requires a list of points to be drawn. The points use the absolute coordinates of the image as a reference. This makes annotations responsive to the current image position and magnification level, becoming independent of the viewport state they were initially drawn in. Moreover, this makes it possible to associate the annotations to the study at a series level.

Unlike the other fields, the controller is stored as a JSON string. This decision was made since the controller does not have a fixed structure between different annotation types. To support multiple types of annotations and, in order to support a transparent and abstract API to control them, storing this object as a string allows annotation controllers to take any shape they want. The downside of this approach is that the application loses the capacity to query for specific controller fields and can no longer validate its data. However, such requirements were not crucial since the data can be validated on the frontend, and there is not much interest in querying fields from the controller. The same reasoning applies to the measurements object. An alternative solution to the problem would be to create a different database model for each type of controller. However, this would severely limit the flexibility of these objects. Moreover, it would hinder the development of future new annotations and it would add to the complexity of the solution.

Another approach considered was the use of a secondary scheme-less database like, for instance, MongoDB to store the controllers. The main database would keep a reference to the record in the secondary database. While this solution would keep the complexity of the annotation model down, the added system requirements, increased input/output operations and increased complexity in system architecture outweighed the gains.

The user can configure any of the common attributes through the annotation window component. He can write a title, a short descriptive text, associate tags to the annotation and lastly choose its color. The purpose of these attributes is to allow the user to identify the annotations in whatever way he wants. All these attributes are optional, as the viewer keeps track internally of the different annotations through the automatically generated ids. The goal of these attributes is to both help the user producing the medical reports and to help with the communication between different users in a multi-user environment. In the reporting process, the user can import some of the annotation attributes like title description and tags into the report. These elements enhance the conventional pathology reports since visual elements can be appended to the image, richening the report's content for a better understanding of the diagnosis by other doctors or even by the patients.

The attributes "attachReport" and "attachImage" are used to mark the annotations that will be imported onto the report and, in the case of a shape annotation, if the selected region is to be cropped. An image can only be attached to the report if both the "attachReport" and "attachImage" are set to true.

To crop the image, the web application maps the image points that represent the annotation to viewport points in order to create a canvas over the image from which the application will perform the crop. The cropped image will always correspond to the visible area that the annotation represents. Since the annotations have variable sizes that correspond to the current magnification level of the viewer, one annotation at different magnification levels will represent different images that can be cropped. The crop only happens once the user marks the annotation as such, so the user can control the magnification level of the viewer before attempting to crop the image. If the user tries to crop an annotation bigger than the viewport or tries to crop a section too big in size, the application will warn the user accordingly. The cropped image is sent to the server as a base64 string. A max limit of 10MB was configured for the upload of cropped images. However, on a 1920x1080 screen resolution, crops representing nearly the full area of the viewport did not reach the 100 Kilobyte (KB) in size. The higher cap is meant for higher resolution monitors where the viewport will be bigger. Once it arrives at the server, the base64 string is converted into a byte array and stored as a medium sized blob on the MySQL database.

The diagram in Fig. 26 depicts the creation of a shape annotation. The user starts by selecting from the toolbar the annotation he wants to create. Once the user finishes drawing the annotation, the application will create the annotation model and open the annotation window with all associated details. The user can fill in the fields and save the annotation, which updates its data, or can simply close the window since the annotation was already saved in the database. The option to attach a crop is only available in shape annotations. If the user marks the annotation as such, the system will generate the crop and upload it into the platform asynchronously. The user can close the annotation window and proceed as normal while the cropped image uploads in the background.

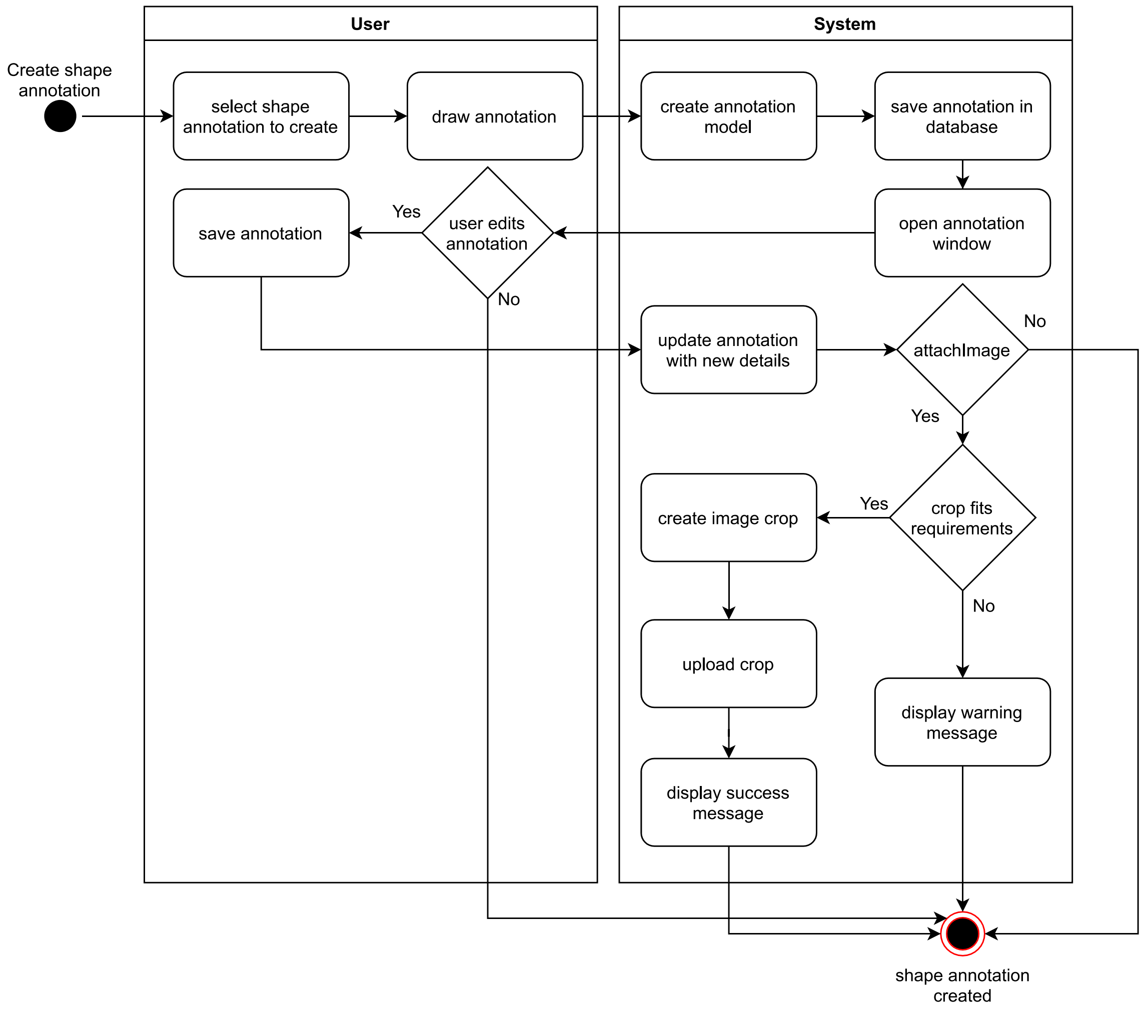


Fig. 26 - Flowchart diagram for the creation of a shape type annotation with image crop

4.6. Study Report

The main goal of PathoBox is to provide an alternative solution to the conventional microscope analysis in the pathology branch, with the main benefit of speeding up the process of diagnosis of a slide. The reporting tools developed for the viewer workstation aim to achieve that objective. Pathobox provides a set of tools that allow doctors to produce reports on a digital slide in a semi-automatic way, with very little input required.

The generation of reports is done live by the server application. The sequence diagram presented in Fig. 27 depicts a report preview request done by the user. It sends the report id used by the server to query for the necessary entities to build the report, namely, the patient data, doctor in charge of the study, the facility responsible for the study, and the annotations of the study. A byte array with the pdf's contents is generated and sent to the web application where it will be displayed in a pdf viewer.

Pathology reports may vary depending on the type of sample that they were based on. However, all reports share a common base structure. It is usually presented a diagnosis section that includes the conclusions taken from the analysis, a gross examination section where pathologists describe the sample as seen with a naked eye, and a microscopic examination section where the report goes into detail about the metrics of the slide [62]. Fig. 28 presents an example of a standard pathology report.

Some reports might also include snapshots of the slide where the diagnosed problem is located (when digital viewers are used). Generic information like, for instance, patient name and slides date, is also usually present. Because the reports assume very similar structures, despite its different contents, the application can use the same base template for distinct types of pathology reports.

In the viewer, the doctor is expected to use the annotation tools to mark areas of interest, to be included in the report. He can fill in the annotation fields like title and description to also be included in the report. While the process of making the annotations and populating them with data is manual, their insertion in the report is automatic, unless the doctor chooses not to. Shape annotations can generate an image snippet in the report, while the remaining annotations will only add their attributes to the report.

The doctor can use the image analysis tools to run a specific task over a selected part of the image, or over the whole image, to include in the report. The results of the analysis can appear in the diagnosis area or be linked to the exported annotations. Moreover, they can be accessed from the viewer and the user can edit the results or delete them.

Once the doctor finishes the report, he can sign and move on to the next study on the list. Signing the report does not deliver it, and that action can only be done through the inbox page. The doctor is able to make changes to the report before delivering it.

The reports are associated with the last doctor that signed them. If more than one doctor is assigned to review the same study, all be able to make changes to the report but only one signature will be present in the final document. The report keeps track of the number of times it was changed before being delivered.

The first page of the report is divided into four sections. The first section is the header where the information regarding the patient, study, and facility is present. The next three sections are respectively, the diagnosis, gross examination, and microscopic examination. The next pages of the report are used to present the annotations. If there are no annotations to attach, the report will only consist of one page. The application will create as many pages as necessary to accommodate all annotations. Before adding an annotation to the page, it is checked if the cropped image fits within the boundaries of the page. If it does not, a resize operation will take place, keeping the aspect ratio of the image, until the image fits on the page. If the image can be rotated to fit in the page without the need for a resize, the application will choose that method. However, this behavior can be disabled.

When checking the size of the annotation, the algorithm looks at both the width and height to check if they fit in the boundaries. It then rescales the image by the dimension that is not in accordance. If both dimensions do not fit in the page, the algorithm first rescales the image horizontally and then if needed, rescales it again vertically. Finally, the combined height of the image and the text are checked before adding the annotation to the page. If the annotation exceeds the available height, a new page is created. This means that when annotations are rescaled, they are rescaled to fit in an empty page. The application will not try to rescale an annotation to fit in with another annotation. This decision was made to preserve as much as possible the detail and dimensions of the original annotation.

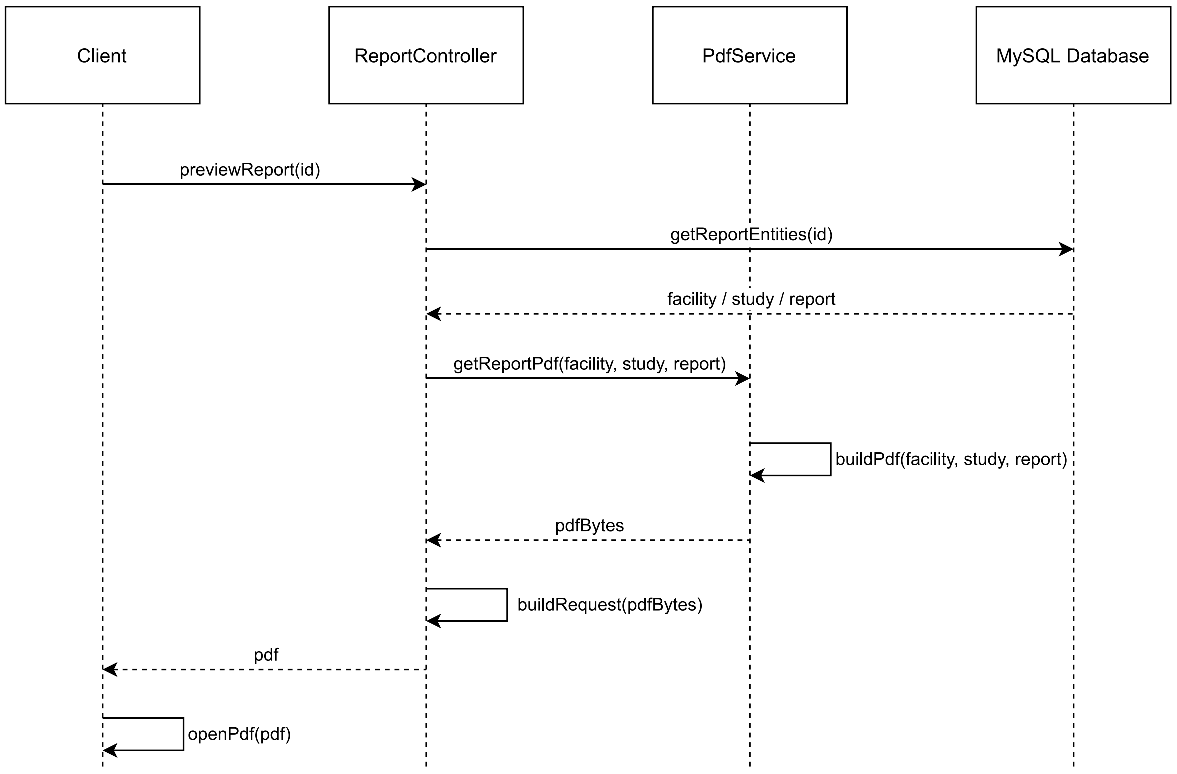
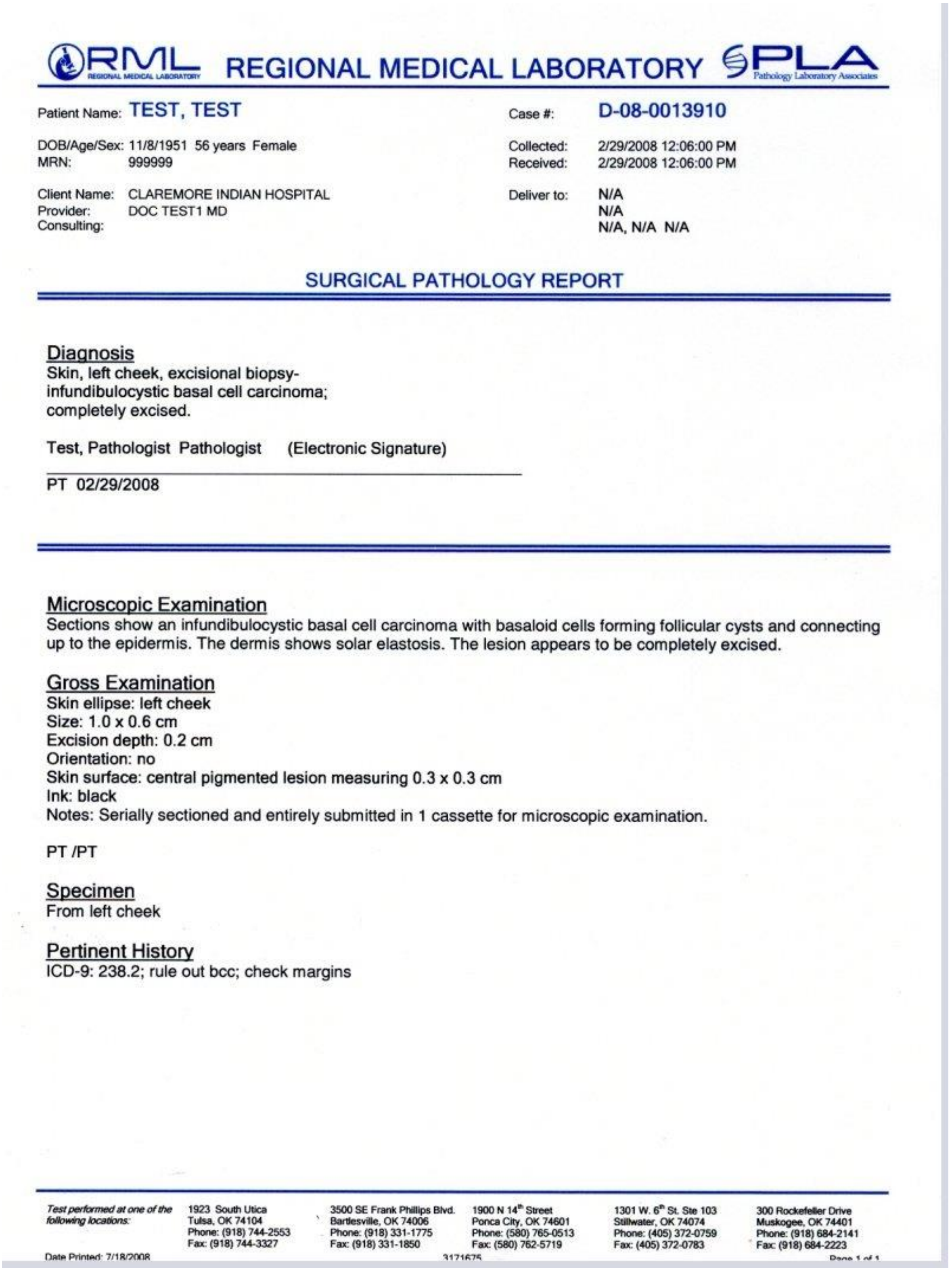


Fig. 27 - Sequence diagram for the generation of a report



RML REGIONAL MEDICAL LABORATORY — REGIONAL MEDICAL LABORATORY — PLA Pathology Laboratory Associates

Patient Name:	TEST, TEST	Case #:	D-08-0013910
DOB/Age/Sex:	11/8/1951 56 years Female	Collected:	2/29/2008 12:06:00 PM
MRN:	999999	Received:	2/29/2008 12:06:00 PM
Client Name:	CLAREMORE INDIAN HOSPITAL	Deliver to:	N/A
Provider:	DOC TEST1 MD		N/A
Consulting:			N/A, N/A N/A

SURGICAL PATHOLOGY REPORT

Diagnosis
Skin, left cheek, excisional biopsy-
infundibulocystic basal cell carcinoma;
completely excised.

Test, Pathologist Pathologist (Electronic Signature)

PT 02/29/2008

Microscopic Examination
Sections show an infundibulocystic basal cell carcinoma with basaloid cells forming follicular cysts and connecting up to the epidermis. The dermis shows solar elastosis. The lesion appears to be completely excised.

Gross Examination
Skin ellipse: left cheek
Size: 1.0 x 0.6 cm
Excision depth: 0.2 cm
Orientation: no
Skin surface: central pigmented lesion measuring 0.3 x 0.3 cm
Ink: black
Notes: Serially sectioned and entirely submitted in 1 cassette for microscopic examination.

PT /PT

Specimen
From left cheek

Pertinent History
ICD-9: 238.2; rule out bcc; check margins

Test performed at one of the following locations:	1923 South Utica Tulsa, OK 74104 Phone: (918) 744-2553 Fax: (918) 744-3327	3500 SE Frank Phillips Blvd. Bartlesville, OK 74006 Phone: (918) 331-1775 Fax: (918) 331-1850	1900 N 14th Street Ponca City, OK 74601 Phone: (580) 765-0513 Fax: (580) 762-5719	1301 W. 6th St. Ste 103 Stillwater, OK 74074 Phone: (405) 372-0759 Fax: (405) 372-0783	300 Rockefeller Drive Muskogee, OK 74401 Phone: (918) 684-2141 Fax: (918) 684-2223

Date Printed: 7/18/2008 3171675 Page 1 of 1

Fig. 28 - Example of a pathology report that follows the format proposed in [62] *produced by the Regional Medical Laboratory (RML)*

4.7. Diagnostic Support Tools

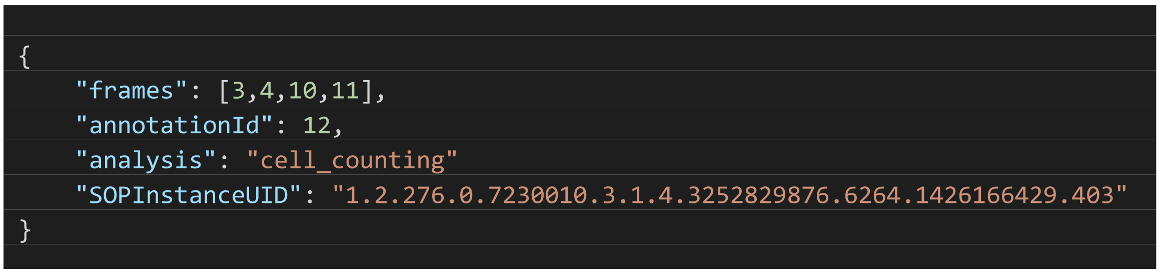
As mentioned in section 2.8, there are many different workflows in pathology and different tissue samples, from distinct pathologies, require different biomedical data to be diagnosed. Determining ROIs also varies from sample to sample according to their type and staining method used. This heterogeneity makes it difficult for the application to automatically select, based on the image type, an analysis algorithm to calculate relevant medical data for the diagnosis of the slide or classify the sample with some medical condition.

The reviewed solutions always relayed the classification of the pathology and choice of analysis workflow to the user. In an automatic analysis modus operandi, the pipelines usually focus on one type of pathology, and is up to the user to choose the correct pipeline to use in the classification of the slide. In order to accurately determine the type of sample at hand, data regarding the pathology needed to be present in the metadata of the study or a large dataset of the different types of pathologies would be needed to train classification algorithms capable of assessing the pathology of the slide.

The proposed solution, following the steps of other solutions, leaves that choice up to the user. This prevents the inevitable scenario of a wrong classification being given by the system and computational resources being wasted in analysis procedures that are not necessary. The downside is that it does take away some of the automation from the reporting and analysis process.

To perform an analysis, the user must first choose an ROI to be used. It can be the whole image or only a section of it. Depending on the type of analysis the user chooses, the application will use different layers of the image. Cell counting or biomarker assessment might present better results with the higher resolutions available at the lower levels of the WSI pyramid. Other analysis procedures might require lower resolutions, thus higher levels of the pyramid can be used. This choice is done by the system and not the user. This selection is important as it allows the application to control the computing resources used in the image analysis. Higher levels of the image have less frames to process. The choice is done by a simple mapping of analysis procedures to preferred resolutions. The user can choose a different resolution level if the chosen one is not ideal.

When an ROI is used, the client application will translate the image coordinates into the corresponding image frames. Since the annotations use the absolute image coordinates, it is possible to determine the frames contained within one annotation at a certain resolution level. Once the frames are assessed, it is sent to the server a list with the index of the frames to analyze, the annotation id respective to the ROI used, so that the application can save the results in the database and link them to the annotation, and finally the unique image identifier that is going to be used to extract the frames. When the analysis is performed over the whole image, this step is skipped since all the frames are going to be used (Fig. 29).



```
{
    "frames": [3,4,10,11],
    "annotationId": 12,
    "analysis": "cell_counting"
    "SOPInstanceUID": "1.2.276.0.7230010.3.1.4.3252829876.6264.1426166429.403"
}
```

Fig. 29 - Example of a request for an analysis procedure sent by the client application

The processing done on the image is tile-based, to control memory usage. Even lower resolution layers can be images too large to fit in memory, especially when the concurrent analysis of images is a possibility. The default tile size in whole slide images is usually small. The DICOM files used to test the system had the dimensions 512x512 pixels. Using this method, it is possible to process large images while controlling the memory used by the application. Distributed processing is also enabled since the tiles can be processed in parallel.

In the first implementation, for quick prototyping and testing of the solution, it was decided to have the analysis being done server side by the server application, where the image frames are requested from Dicoogle. The ideal scenario is having Dicoogle, through a custom plugin, performing the image analysis. This does not invalidate the mechanisms and data models designed, as they can be imported onto the plugin. Having the processing being done by Dicoogle is the optimal solution since it avoids the transmission of the frames between the two components. If Dicoogle is installed locally, this procedure is not very taxing. However, assuming a remote installation, this method is not practicable.

The image analysis is done by third-party tools such as ImageJ. The goal was to reutilize the efforts done in this area. Given the different interfaces of each analysis tool, an abstraction layer was needed to wrap the analysis services under a common endpoint for the application. Moreover, it is not guaranteed that the results will come with a standard format. To support these scenarios and make future integration of different analysis tools easier, a helper class that follows the façade design pattern was created to serve as a wrapper for the different image analysis service providers. It takes as an argument the analysis to perform and the image to process. Internally, the component, depending on the type of analysis requests, will connect to the analysis tool that provides that service, sending it the image and any other necessary fields.

The sequence diagram in Fig. 30 presents the workflow of an image analysis request. The client application, once the ROI and analysis procedure are set, sends to the server application a request for an image analysis procedure. The AnalysisService is the component responsible for interpreting the user request and initiating the analysis procedure. Once the analysis is complete, the analysis service will translate the results obtained into the normalized JSON object. Next, it stores the results in the database and sends them back to the client application. Independently of the analysis type, the results are transmitted to the frontend in the same way.

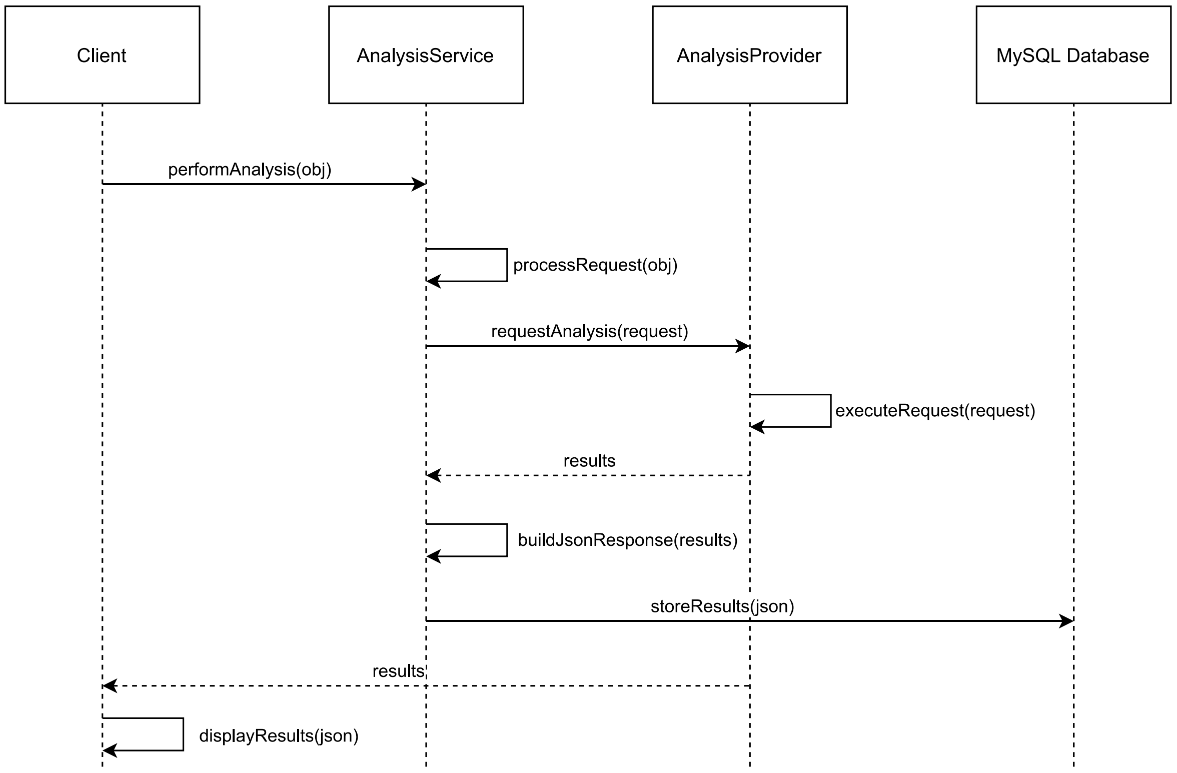
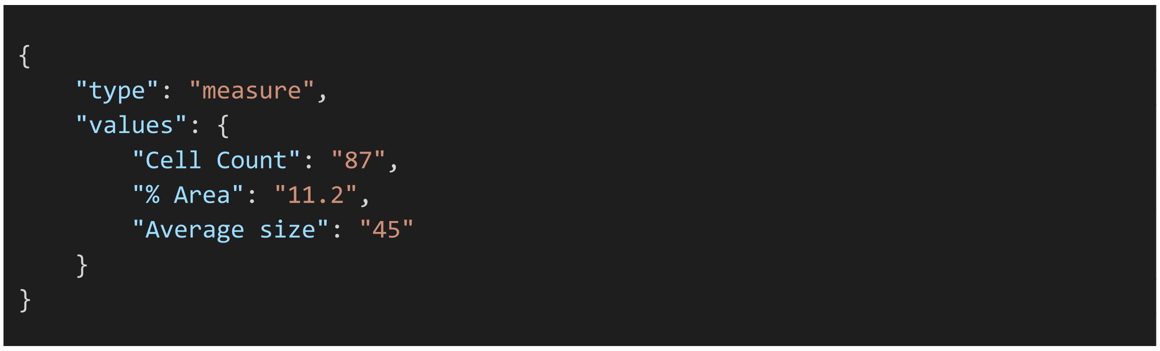


Fig. 30 - Image analysis sequence diagram

The analysis will return different types of results according to the algorithm used. ImageJ, for example, stores the results in a comma separated value (CSV) file. Once the analysis is complete, the system converts the CSV file into a JSON file to send it back to the user (Fig. 31). Usually, results will come in two ways, either as a list of biomedical data or regions of interest (i.e. areas) in the image. By converting the results to a JSON file, the application guarantees flexibility in the storage and acquisition of results. An extra field is added to identify the type of results. This field is necessary to let the client application know how to display those results.

Biomarkers can be listed as a simple table, while ROIs or areas can be defined as a list of points. The frontend application, according to the type of data it gets, will either present the biomarkers to the user in a tabular format or in the case of an ROI, it will draw it using the annotation tools available in the viewer.

The biomedical data can be associated with the whole image or only a portion of it, depending on how the user chose the area of analysis. If the data refers to the whole image, the results are presented in a separate window. If the results refer to only a section of the image, then they will be associated with the annotation performed by the user to represent the ROI. In both scenarios, the user can export these results onto the report. An intended workflow for the pathologist consists of designating an ROI for analysis, wait for the results and, once they are available, the pathologist confirms the results and decides if he wants to add them to the report or discard them. The pathologist can edit the results as well.



```
{
    "type": "measure",
    "values": {
        "Cell Count": "87",
        "% Area": "11.2",
        "Average size": "45"
    }
}
```

Fig. 31 - Example of a result returned from an analysis procedure

In this first iteration, the application relies solely on ImageJ’s capabilities to perform image analysis. While its functionality can be greatly extended through custom plugins and Macros, it can only do so much. It is expected that, as the platform grows, more tools will need to be integrated. Most analysis tools have native integration with ImageJ so it is expected that the effort in integrating more analysis tools will be reduced. Nevertheless, the system was designed to facilitate the integration of new image analysis modules.

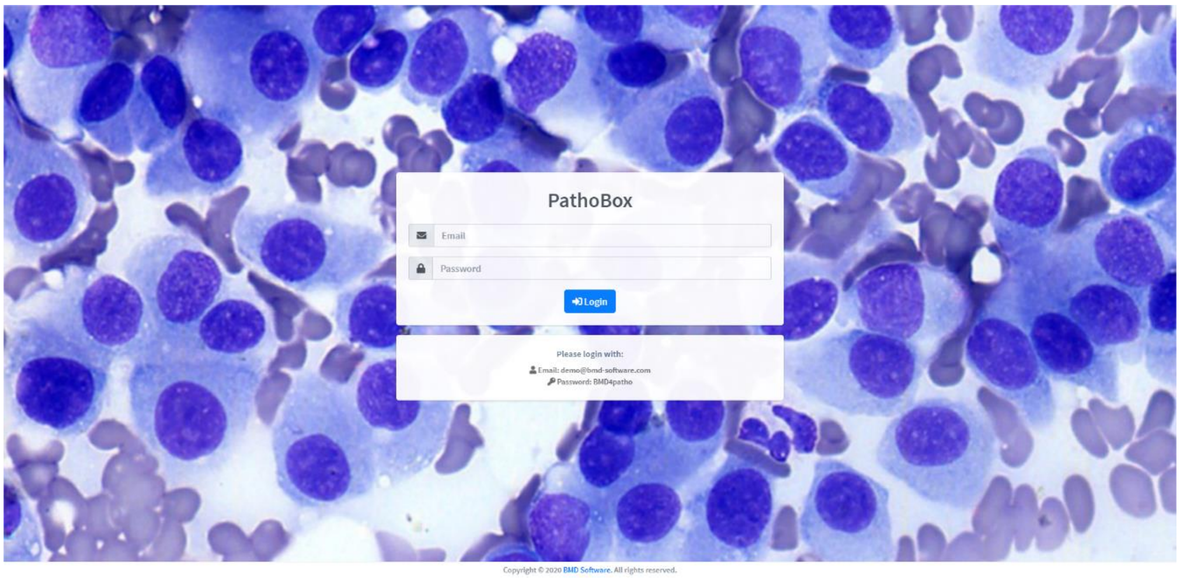
5. Results

This section presents the results obtained and aims to prove that the chosen technologies and the proposed architecture are able to answer the raised requirements. It is presented with screenshots of the most relevant sections of the application, describing the user interface and its functionalities.

5.1. PathoBox Application

Login Page

This is the login page (Fig. 32) where the user can input his credentials to access the platform. The server will verify the credentials and, if the user is logged in with success, he is redirected to the home page.



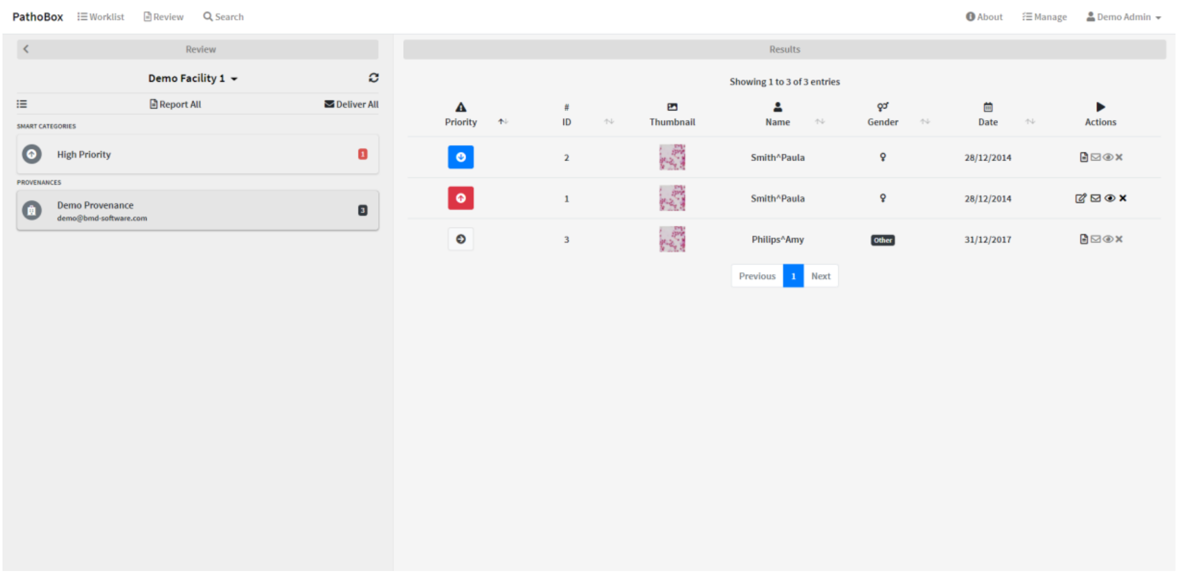
Please login with:
Email: demo@bmd-software.com
Password: BMD4patho

Copyright © 2020 BMD Software. All rights reserved.

Fig. 32 - Screenshot of the login page

Case Study Inbox

The home page of the application is presented in Fig. 33. The user is redirected to this page upon logging in and, within the application, can click on the brand logo at the top left corner to always return to this page.



Showing 1 to 3 of 3 entries

Priority	# ID	Thumbnail	Name	Gender	Date	Actions
	2		Smith^Paula	♀	28/12/2014	
	1		Smith^Paula	♀	28/12/2014	
	3		Philips^Amy	Other	31/12/2017	

Fig. 33 - Screenshot of the case study inbox page

The navigation bar at the top of the page is present on all pages of the application, except the login page. Its appearance changes according to the permissions of the logged in user. In the presented example, the logged in user has the administrator role that grants him access to all resources of the platform. Usually, the links to the Manage page and Worklist page would not be displayed.

On the left side of the screenshot, there is a collapsible sidebar that controls the right table's items. The sidebar displays either a list of requests or a list of provenances. Each request is associated with a list of studies. In the example, the requests were grouped by provenance and, for the selected one, the table shows all the service requests of that provenance assigned to the logged in user. When grouped by request, it is presented on each item the name of the provenance responsible for this request, its email, the date of the request and the number of studies in it. When grouped by provenance, only the provenance name, email and the number of studies is presented. Right above the request list, there is a high priority filter that shows only the studies identified with the high priority flag.

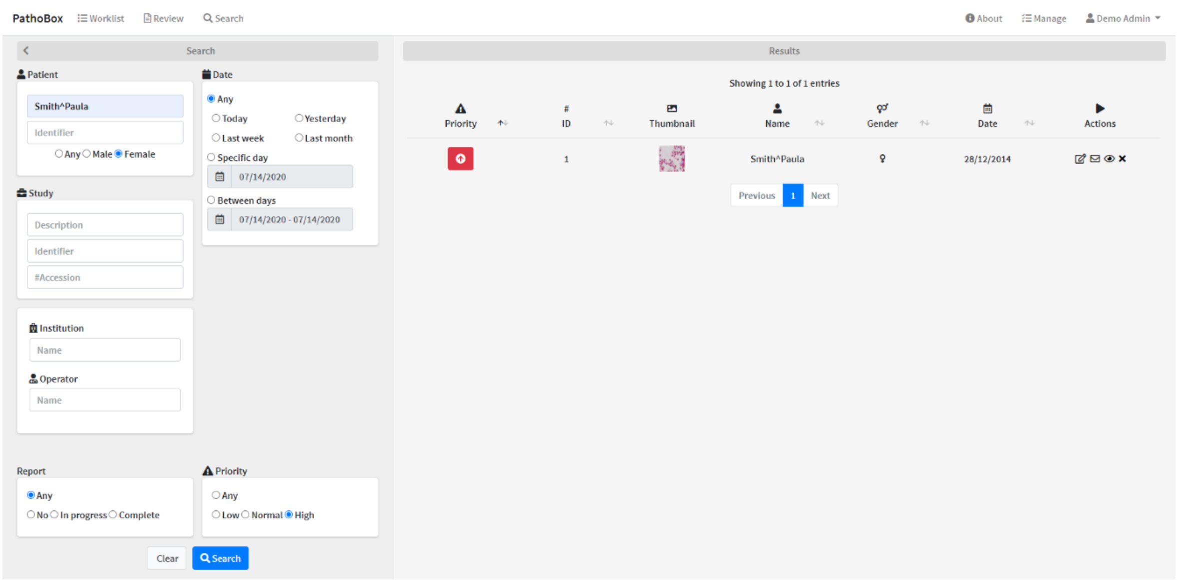
At the top of the sidebar, there is the facility dropdown list, that allows the user to switch the facility. When he switches the facility, the sidebar and table contents are updated accordingly. Below the facility dropdown, there is a set of buttons, a "group by" button that groups the studies by request or by provenance, a "report all" button that acts as a shortcut to open all studies left to report in the viewer and, finally, a "deliver all" button that is a shortcut to deliver all reported studies that are pending in the outbox.

The case studies table presents the studies from the item selected on the sidebar. It shows the priority of the study, its identifier, a thumbnail of the first series of the study, the patient name that this study belongs to, the gender of the patient, the date of the study and finally the study actions. These are respectively, open study, deliver report, preview report, and delete report actions. The state of these actions is controlled by the state of the report. If a report has not yet been created for this study, the deliver preview and delete buttons will be disabled. In the example, the second study in the table has a report already created so it has different actions available when compared with the other studies. The user can also open the study by clicking directly on the row of the study. The user can also sort the table to control the presentation order in the viewer.

The priorities are color coded. Red for high, blue for low and white for normal. When the priority is changed, the behaviour of the high priority filter is updated accordingly. To change the priority, the user can use the button on the table.

Search Page

The search page is presented in Fig. 34. It has the same base layout as the inbox page. A left sidebar that controls the studies that appear on the right-side table. In this page the user can search for studies by patient information, study information, study date, study priority and report status. Unlike the inbox page, the search page displays studies that might already have its report delivered and studies that were not assigned to the logged in user. When a user opens a study, it will only open that study and not the other ones from the table. In the example, a search was made for studies marked with priority "high", with female patients named "Smith^Paula".



Search

Patient
Smith^Paula
Identifier
Any Male [✓ Female]

Date
[✓ Any]
Today
Yesterday
Last week
Last month
Specific day
07/14/2020
Between days
07/14/2020 - 07/14/2020

Study
Description
Identifier
#Accession

Institution
Name
Operator
Name

Report
[✓ Any]
No In progress Complete

Priority
Any
Low Normal [✓ High]

Clear Search

Results

Showing 1 to 1 of 1 entries

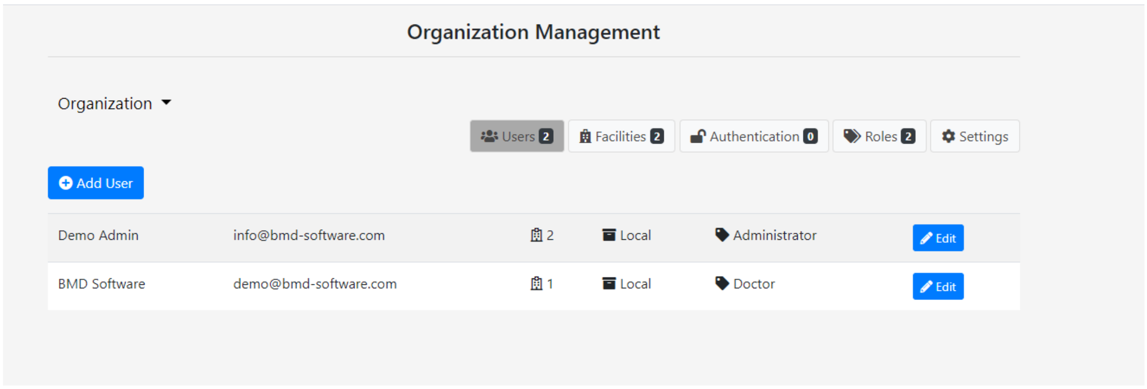
Priority	# ID	Thumbnail	Name	Gender	Date	Actions
🟥	1		Smith^Paula	♀	28/12/2014	

Previous 1 Next

Fig. 34 - Screenshot of the search page

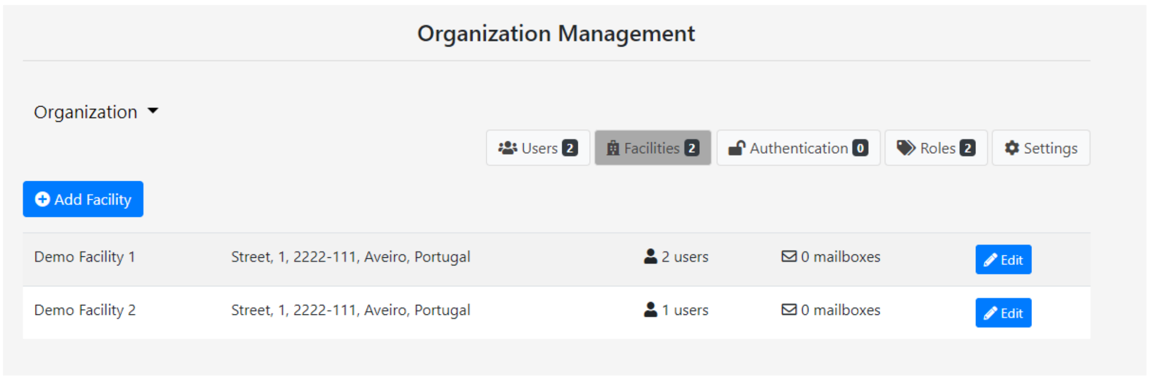
Manage page

The management page (Fig. 35) is where users with management permissions can check the configurations related to organizations and facilities. This page is only accessible if the user has the manage permissions. The page displays, in tabular format, information about the organization, namely, its facilities, users, roles, and authentication providers. The user can create, edit, or delete the entries.



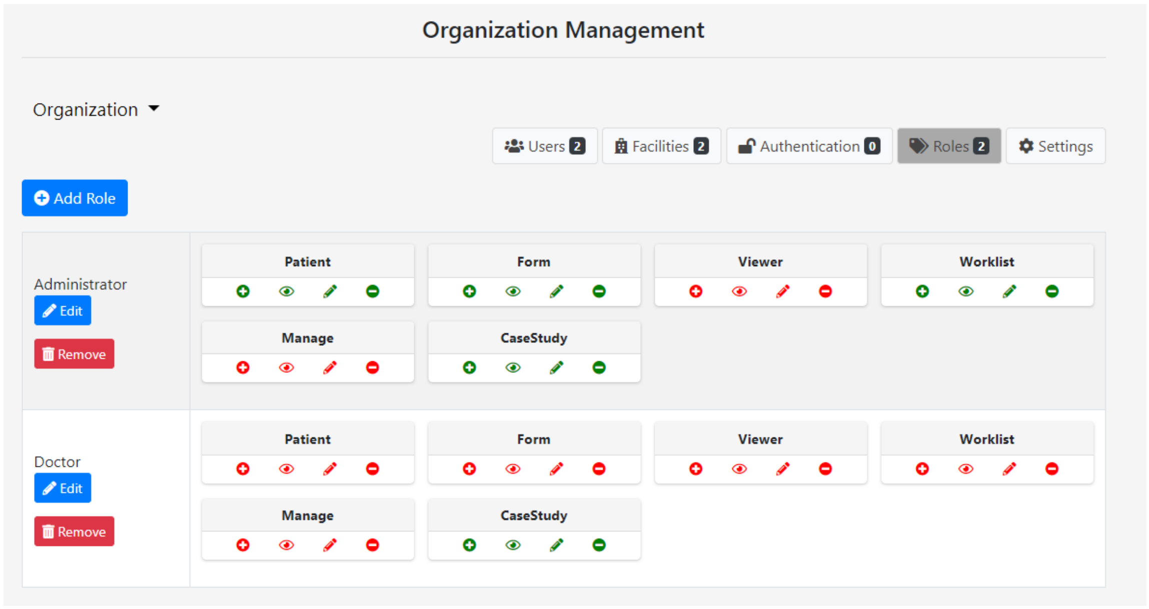
Demo Admin	info@bmd-software.com	2	Local	Administrator	Edit
BMD Software	demo@bmd-software.com	1	Local	Doctor	Edit

a)



Demo Facility 1	Street, 1, 2222-111, Aveiro, Portugal	2 users	0 mailboxes	Edit
Demo Facility 2	Street, 1, 2222-111, Aveiro, Portugal	1 users	0 mailboxes	Edit

b)



	Patient	Form	Viewer	Worklist	Manage	CaseStudy
Administrator	🟢 🟢 🟢 🟢	🟢 🟢 🟢 🟢	🔴 🔴 🔴 🔴	🟢 🟢 🟢 🟢	🔴 🔴 🔴 🔴	🟢 🟢 🟢 🟢
Doctor	🔴 🔴 🔴 🔴	🔴 🔴 🔴 🔴	🔴 🔴 🔴 🔴	🔴 🔴 🔴 🔴	🔴 🔴 🔴 🔴	🟢 🟢 🟢 🟢

c)

Fig. 35 - Screenshot of the various sub sections of the manage page: a) Users table; b) Facilities table; c) Roles table

In the list of facilities, the user can select one facility to view it in more detail. He will have access to the facility's users, provenances, and mailboxes. Just like before, the user can create new entities or edit the existing ones.

This section is for administrators that might want to change the configurations of an organization/facility. All the data on this page can be inserted through the configuration file, however, by having a user-friendly UI, it becomes possible for the end user to change those configurations himself. It allows the election of platform administrators, that do not need to have deep background knowledge about the platform details to manage it.

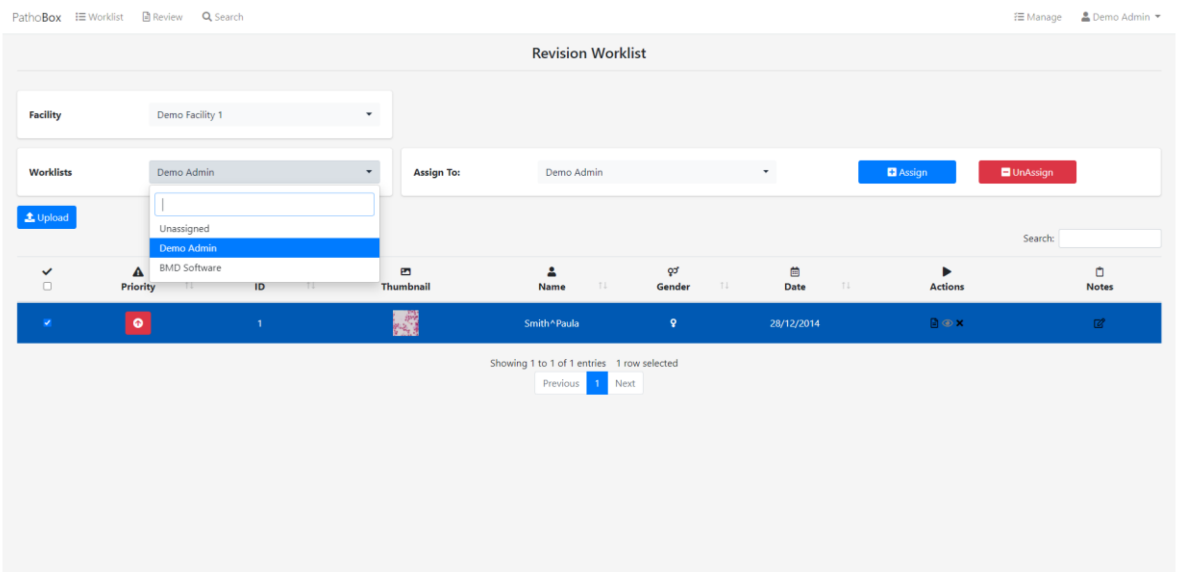
Worklist page

In the worklist page (Fig. 36), the users can assign/unassign studies to/from doctors. This page is only accessible if the user has the worklist permissions.

The first dropdown on the page allows the user to select the facility. The opened dropdown contains a list of users for that facility and one extra option, "Unassigned" which refers to the studies that are not assigned to any doctor. The last dropdown is the same as the opened one but without the "Unassigned" option.

The user can select multiple studies from the table to assign to one doctor or more. Similarly, to the first table, the actions on this table allow the user to open the study in the viewer, preview its report and lastly delete the study. The user can also add notes to studies from this window and change their priorities.

Finally, the upload button on the image is where the user can upload studies to the platform. It opens a modal with a drag and drop window.



Revision Worklist

Facility: Demo Facility 1

Worklists: Demo Admin

Assign To: Demo Admin

✓	Priority	ID	Thumbnail	Name	Gender	Date	Actions	Notes
✓		1		Smith^Paula		28/12/2014		

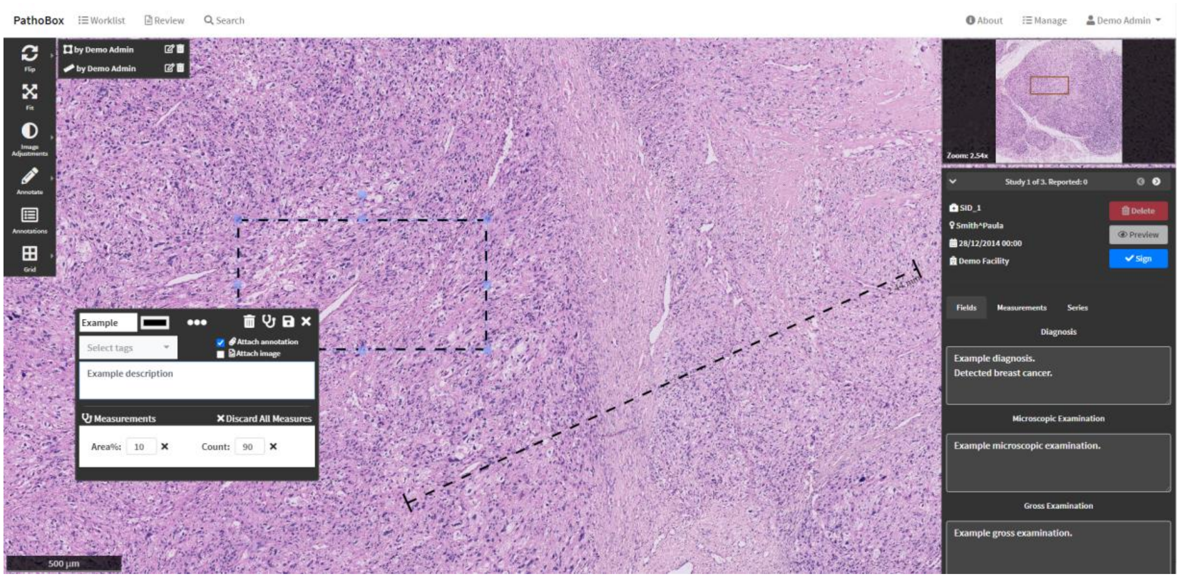
Showing 1 to 1 of 1 entries 1 row selected

Previous 1 Next

Fig. 36 - Screenshot of the worklist page

Whole Slide Image viewer

The whole slide viewer page is shown in Fig. 37.



Example

Select tags

Example description

Area%: 10

Count: 90

SID_1

Smith^Paula

28/12/2014 00:00

Demo Facility

Example diagnosis.
Detected breast cancer.

Example microscopic examination.

Example gross examination.

Fig. 37 - Screenshot of the whole slide image viewer

In this page the user can view the digital slides from a study, perform some analysis on the image and produce a report. The toolbar at the top left corner is where the tools to edit the image are located. The user can change its rotation, apply image filters like increased saturation or brightness, draw annotations and apply a grid overlay. The user can also open a window containing the annotations present in the image, for easy access to all the annotations without having to search through them in the image area. This window is right next to the toolbar in the screenshot.

At the bottom left corner, there is a ruler that represents the scale of the image. Its size automatically adjusts according to the magnification level of the image. It helps the user to have a notion of the size of the visible area.

At the top right corner, there is the navigator window. It is composed of a static outer window that shows a thumbnail of the slide and an inner window that represents the portion of the image the user is currently watching. This window allows the user to rapidly navigate between sections of the image without getting lost. The navigator shows the magnification level at the bottom. The color of the inner window changes according to the magnification level following the standard color coding of microscope lenses [63] represented in Table 5.

Table 5 - Magnification color coding according to the specifications defined in [61]

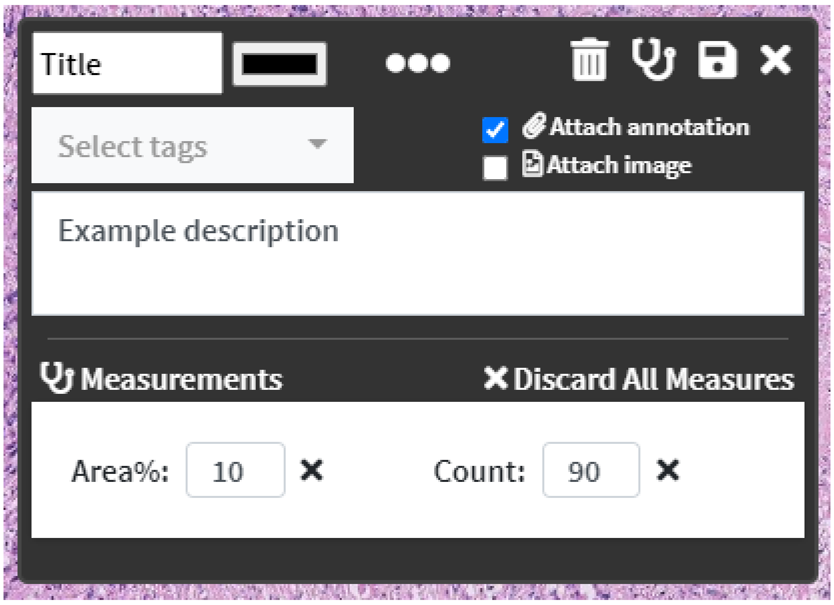
Magnification	Color Code
1-1.5x	Black
2-2.5x	Brown
4-5x	Red
10x	Yellow
16-20x	Green
25-32x	Turqoise
40-50	Light Blue
60-63x	Cobalt Blue
100-250x	White

On the right-side bar, just below the navigator, is the case study selector. The user can go back and forth between the studies he opened in the inbox. Right below the user has access to three tabs, namely, the report examination fields, the global image measurements and the study series. In the report examinations tab, the user can write notes in each section that will be included in the report. In the measurements tab, the user can order an image analysis to run over the whole image and edit the resulting measures. Lastly, in the series tab, the user has access to all series that compose the study. It is presented the number of images of that series (the levels of the pyramid) and the date of the series. The example only has one series, but it is common in pathology for one study to possess multiple series, that correspond to different cuts of the same tissue or different staining methods.

Fig. 38 displays the annotation window for a rectangle annotation. On the left side are the fields the user can populate to be included in the report. On the top right side are the annotation actions. The user can save the annotation, updating with the new data, can run an analysis algorithm (this option is only available for shape type annotations) and can delete the annotation. Right below the user has two checkboxes that he can use to attach the annotation to the report and to crop its respective image to the report. Once again, this last checkbox is only available for shape type annotations.

At the bottom of the window, the user has access to the measurements of this annotation. The user can edit or delete any of the fields or can discard all of them. The changes to an annotation are only saved if the user explicitly presses the save button. So, if he alters some part of the annotation and then closes the window, without first clicking on the save button, the changes will not be saved in the database.

Fig. 39 showcases all the implemented annotations. The annotations scale in size according to the magnification level of the viewer. In the example, a magnification of 15x was used so the annotations are well visible.



Example description

Area%: 10

Count: 90

Fig. 38 - Screenshot of an annotation window for a rectangle annotation

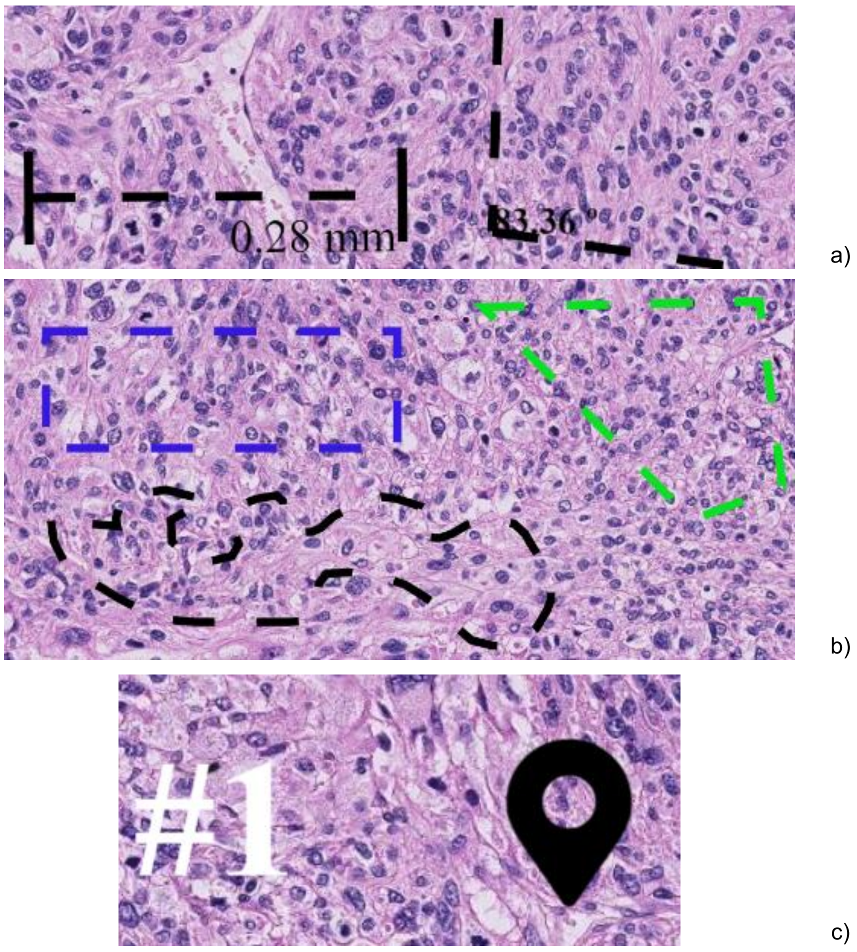
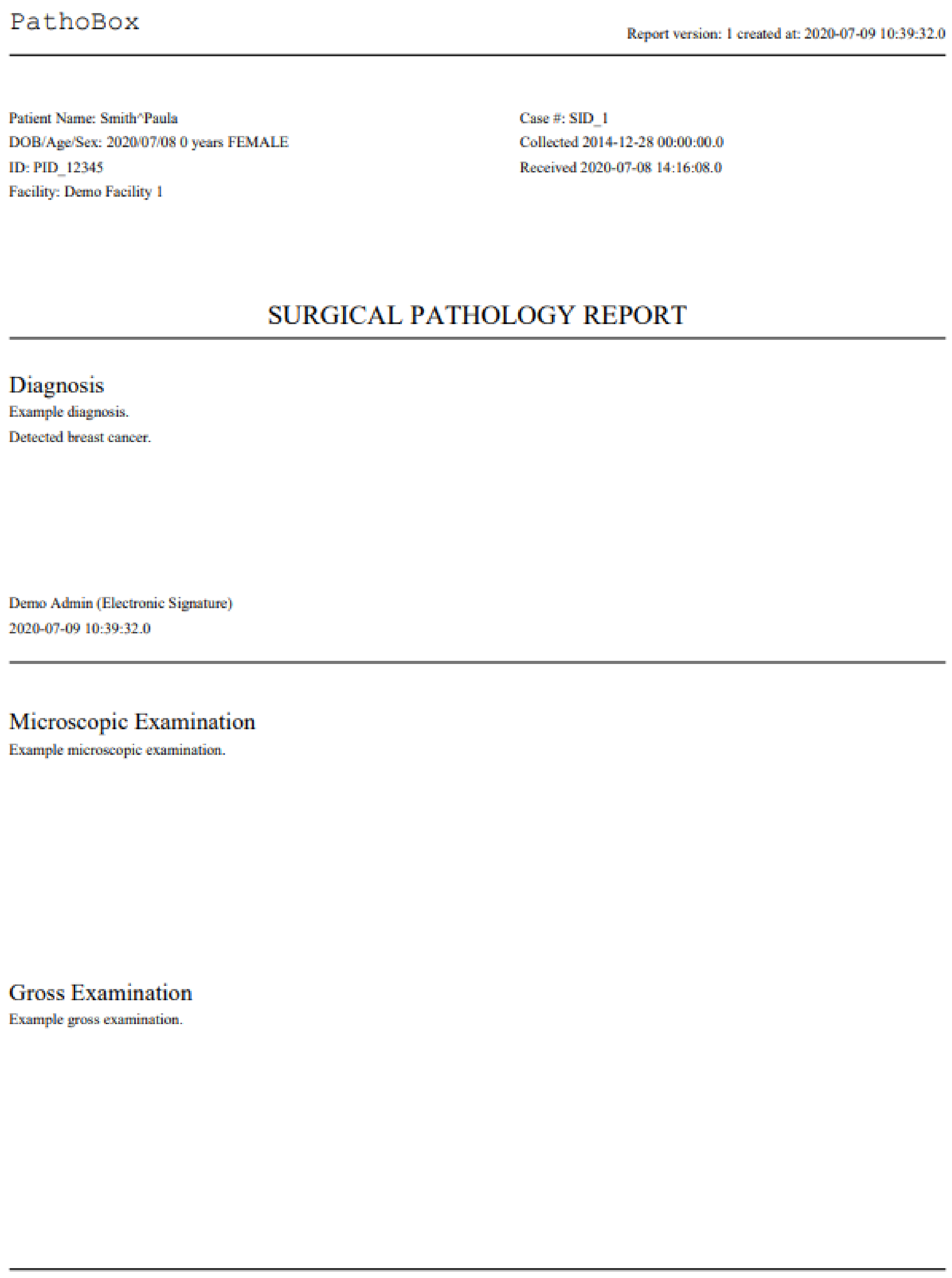


Fig. 39 - Screenshot of the implemented viewer annotation: a) Ruler and Angle Annotation; b) Rectangle, Polygon and Freehand annotation; c) Counter and Marker annotation

Fig. 40 and Fig. 41 display respectively the first page of a report generated for a case study and the annotations page with one exported annotation. The header displays the brand on the left and, on the right, the version of the report alongside its creation date. The first section of the report contains information about the patient, study and facility that uploaded the study. Finally, the next three sections refer to the diagnosis and examinations performed by the pathologist.



PathoBox

Report version: 1 created at: 2020-07-09 10:39:32.0

Patient Name: Smith^Paula	Case #: SID_1
DOB/Age/Sex: 2020/07/08 0 years FEMALE	Collected 2014-12-28 00:00:00.0
ID: PID_12345	Received 2020-07-08 14:16:08.0
Facility: Demo Facility 1	

SURGICAL PATHOLOGY REPORT

Diagnosis

Example diagnosis.

Detected breast cancer.

Demo Admin (Electronic Signature)

2020-07-09 10:39:32.0

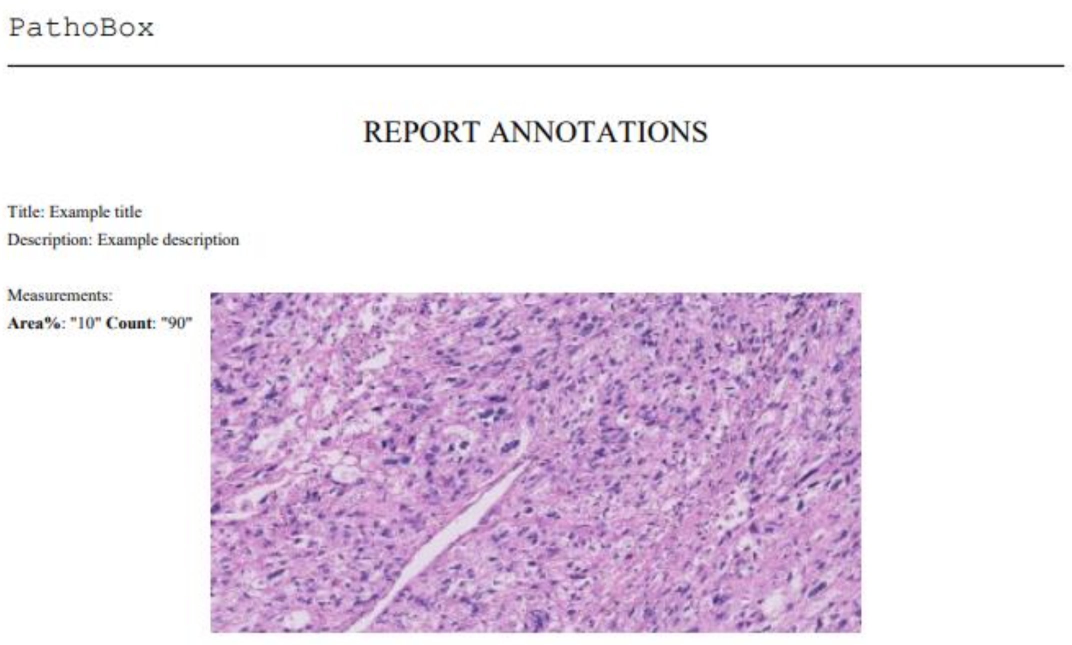
Microscopic Examination

Example microscopic examination.

Gross Examination

Example gross examination.

Fig. 40 - Screenshot of the pathology report for a case study



PathoBox

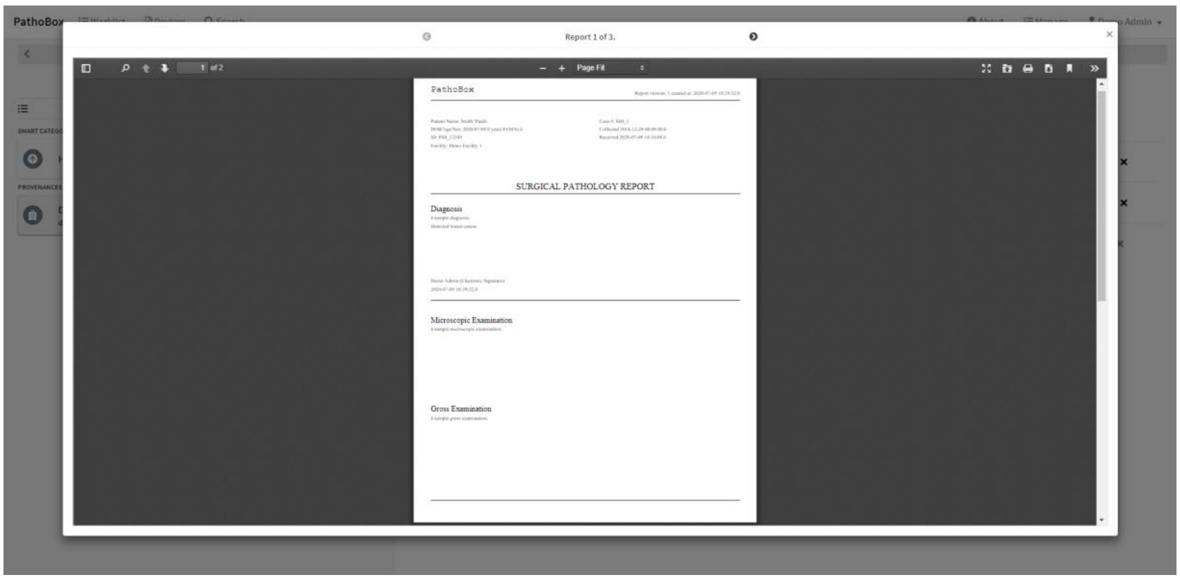
REPORT ANNOTATIONS

Title: Example title
Description: Example description

Measurements:
Area%: "10" **Count**: "90"

Fig. 41 - Screenshot of an annotation exported to a report

Fig. 42 displays the report viewer window that allows the preview of a study report that is in the inbox. It is a modal form with navigation keys to navigate between reports produced but not yet delivered in the inbox. When the user previews reports from the viewer, the same modal is displayed but with only one report in the list.



Report 1 of 3.

PathoBox

SURGICAL PATHOLOGY REPORT

Diagnosis

Microscopic Examination

Gross Examination

Fig. 42 - Screenshot of the report viewer window

5.2. Performance Assessment

The most critical process in the Pathobox is data retrieval for feeding the visual navigation over the whole slide image pyramid, including the display of associated annotations. The performance of the access and visualization process was evaluated using the developer tools provided by the browser. It was tested the loading time of a slide and the performance of the viewer when a lot of annotations were drawn on the image.

The application was deployed on a remote server to simulate a real usage environment. With an average internet speed of 35Mbit/s, the images in the viewer took on average 1-2 seconds to load. As the user zooms in on the image, tiles of higher resolution are fetched but since fewer of them are requested, loading times stay relatively the same, taking 1-2 seconds to render the new resolution. Starting at the smallest resolution possible and zooming in to the highest resolution possible does not change the loading times.

Assuming the same conditions, the loading times of a study with annotations do not represent a noticeable change in loading times. This is to be expected as the size of the annotations is very small when compared to the size of the images.

The usability of the viewer was assessed by how responsive and fluid it feels when panning and zooming the image. A performance drop was noticed when a lot of annotations were drawn on the image. This is to be expected since the viewer is updating the annotations one by one when the magnification level changes. Some occasional stutters appear when the user pans to an area of the viewer densely populated with annotations. This is to be expected since the viewer needs to update the canvas and all its graphics overlays when changes to the viewport happen.

6. Conclusion

This section presents the main contributions of this work, the results achieved, the problems found during development and finally it discusses future work with the platform. The section ends with some final considerations.

6.1. Contributions

The purpose of this dissertation was to research and develop a standard solution to answer the market needs for a web-based reporting platform for the very demanding clinical area of digital pathology. The result was a cloud-ready solution designated as PathoBox. The proposed system allows the upload of case studies to a web platform, where they can be assigned to pathologists for reporting. It was developed a web viewer of digital slides with annotation tools and basic image analysis tools that allows pathologists to generate a report following the standards of digital pathology reporting, in a semi-automatic way. The expandability and modifiability of the system were addressed with modular design patterns used in both the client and server applications.

The doctors have access to an inbox page with pending requests where they can filter the studies by request/provenance/priority and can manage the study reports. A search page is also available that allows doctors to search for studies from the different archives connected to the platform.

The tile-based image viewer can be used for visualization of high-resolution digital images of pathology that can aggregate several gigabytes of data, even with an internet connection with moderate bandwidth. The performance of the viewer was tested, and the results showed that loading times of the images were relatively quick with a small impact on the user experience.

The application makes use of an RBAC system to protect resources from being accessed by unauthorized entities in the platform. The access privileges are defined through user roles that have effect within one organization. These roles can mix to create multiple privilege profiles.

The UI was designed with efficiency in mind, to improve the reporting workflow. Most actions are one click away from the user. The UI has a minimalist design, but it still retains a modern look. Users can customize certain elements of the UI to better fit their preferences.

Administrators of medical facilities have access to a management page where they can configure the organization roles, facilities, users, and authentication providers.

Alongside this work, in cooperation with another collaborator from Aveiro University, the collaborative pathology viewer mentioned in section 2.6 was developed. The work contributed with three conference papers, [9], [32], [33], to the scientific community. They present and discuss a framework for a collaborative whole slide image viewer as well as its advantages in professional and educational environments.

6.2. Problems Found

There were many problems that halted the development process of PathoBox, from technical limitations of the tools and outdated software to challenges found during the implementation of certain features. Some of the tools the project was dependent on were outdated. This led to an initial effort in updating or adapting those components.

The integration of image analysis was a taxing feature to implement. There was no prior knowledge about digital pathology analysis workflows. An extensive review of the literature and tools available was needed to understand the problems that pathologists have with image analysis and how to solve them. In this branch, there is no tool to solve all the problems and so is often practice for pathologists and researchers to develop their own tools and use a variety of different tools together. Besides the complexity of the topic, some tools had some unexpected issues during development, most notably, the lack of proper support for compressed WSI DICOM files.

6.3. Future Work

The present work built the foundations for a web-based pathology image viewer and future work for the platform will revolve around improving the existing work, as a result of physicians' feedback.

In the report of studies, implementation of a template system that allows medical facilities to define how they want their reports to be constructed. The current implementation followed the general guidelines for digital pathology reporting specified in [62] but this is not a mandatory format so it is expected that different medical organizations use slightly different report structures.

In the analysis of pathology images, the introduction of new methods and the integration of different tools in the viewer is an important aspect.

In the management of reports, the possibility to configure different delivery mechanisms, like automatic upload to a Dropbox folder, instead of sending the reports through email. Likewise, the configuration of new input mechanisms for file upload is a necessity for further automation of the platform.

6.4. Final Considerations

Digital pathology is a new branch in digital medicine that has yet not been widely adopted. While there is a lot of development and work performed in whole slide image viewing and analysis, an effort to combine all these tools in a centralized platform destined to the report of case studies is a novel idea with very few competitors in the area. There is a growing market interest for these types of solutions in clinical pathology. It was discussed how the proposed work can bring advantages for pathologists by speeding up the workflow and reducing operation costs. Therefore, patient care can also be improved given the shorter diagnostic times. As presented, with the help of digital pathology networks and systems, diagnosis can be made faster and more accurate.

The presentation of the product to interested clients is the next milestone of the project. The goal is to validate the solution and gather important feedback from the end user.

Bibliography

[1] T. D. Chordia, A. Vikey, A. B. Choudhary, Y. Samdariya, and D. S. Chordia, “Current status and future trends in telepathology and digital pathology,” *J. Oral Maxillofac. Pathol.*, vol. 20, no. 2, pp. 178–182, May 2016.

[2] L. Pantanowitz *et al.*, “Review of the current state of whole slide imaging in pathology,” *J. Pathol. Inform.*, vol. 2, no. 1, p. 36, 2011.

[3] G. Bueno, M. M. Fernández-Carrobles, O. Deniz, and M. García-Rojo, “New Trends of Emerging Technologies in Digital Pathology,” *Pathobiology*, vol. 83, no. 2–3, pp. 61–69, Apr. 2016.

[4] E. Patterson, M. Rayo, C. Gill, and M. Gurcan, “Barriers and facilitators to adoption of soft copy interpretation from the user perspective: Lessons learned from filmless radiology for slideless pathology,” *J. Pathol. Inform.*, vol. 2, no. 1, p. 1, 2011.

[5] A. Vodovnik, “Diagnostic time in digital pathology: A comparative study on 400 cases,” *J. Pathol. Inform.*, vol. 7, no. 1, Jan. 2016.

[6] A. Saco, J. A. Bombi, A. Garcia, J. Ramírez, and J. Ordi, “Current Status of Whole-Slide Imaging in Education,” *Pathobiology*, vol. 83, no. 2–3, pp. 79–88, Apr. 2016.

[7] M. M. Triola and W. J. Holloway, “Enhanced virtual microscopy for collaborative education,” *BMC Med. Educ.*, vol. 11, no. 1, 2011.

[8] K. Foster, “Medical education in the digital age: Digital whole slide imaging as an e-learning tool,” *J. Pathol. Inform.*, vol. 1, no. 1, p. 14, 2010.

[9] R. Lebre, R. Jesus, P. Nunes, and C. Costa, “Collaborative framework for a whole-slide image viewer,” in *Proceedings - IEEE Symposium on Computer-Based Medical Systems*, 2019, vol. 2019-June.

[10] L. Pantanowitz, J. Szymas, D. Wilbur, and Y. Yagi, “Whole slide imaging for educational purposes,” *J. Pathol. Inform.*, vol. 3, no. 1, p. 46, 2012.

[11] J. Griffin and D. Treanor, “Digital pathology in clinical use: where are we now and what is holding us back?,” *Histopathology*, vol. 70, no. 1, pp. 134–145, Jan. 2017.

[12] G. Yousef, M. Bellis, S. Metias, C. Naugler, A. Pollett, and S. Jothy, “Digital pathology: Attitudes and practices in the Canadian pathology community,” *J. Pathol. Inform.*, vol. 4, no. 1, p. 3, 2013.

[13] L. Pantanowitz, N. Farahani, and A. Parwani, “Whole slide imaging in pathology: advantages, limitations, and emerging perspectives,” *Pathol. Lab. Med. Int.*, p. 23, Jun. 2015.

[14] B. A. Beckwith, “Standards for digital pathology and whole slide imaging,” in *Digital Pathology: Historical Perspectives, Current Concepts Future Applications*, Springer International Publishing, 2016, pp. 87–97.

[15] F. Valente, L. A. B. Silva, T. M. Godinho, and C. Costa, “Anatomy of an Extensible Open Source PACS,” *J. Digit. Imaging*, vol. 29, no. 3, pp. 284–296, Jun. 2016.

[16] J. M. Silva, T. M. Godinho, D. Silva, and C. Costa, “A community-driven validation service for standard medical imaging objects,” *Comput. Stand. Interfaces*, vol. 61, pp. 121–128, Jan. 2019.

[17] W. D. Bidgood, S. C. Horii, F. W. Prior, and D. E. Van Syckle, “Understanding and Using DICOM, the Data Interchange Standard for Biomedical Imaging,” *J. Am. Med. Informatics Assoc.*, vol. 4, no. 3, pp. 199–212, May 1997.

[18] C. Viana-Ferreira, C. Costa, and J. L. Oliveira, “Dicoogle relay - A cloud communications bridge for medical imaging,” in *Proceedings - IEEE Symposium on Computer-Based Medical Systems*, 2012.

[19] P. Mildenberger, M. Eichelberg, and E. Martin, “Introduction to the DICOM standard,” *European Radiology*, vol. 12, no. 4. Springer, pp. 920–927, 01-Apr-2002.

[20] R. Singh, L. Chubb, L. Pantanowitz, and A. Parwani, “Standardization in digital pathology: Supplement 145 of the DICOM standards,” *J. Pathol. Inform.*, vol. 2, no. 1, p. 23, 2011.

[21] C. Daniel *et al.*, “Recent advances in standards for collaborative Digital Anatomic Pathology,” *Diagn. Pathol.*, vol. 6, no. SUPPL. 1, Mar. 2011.

[22] T. Marques Godinho, R. Lebre, L. B. Silva, and C. Costa, “An efficient architecture to support digital pathology in standard medical imaging repositories,” *J. Biomed. Inform.*, vol. 71, pp. 190–197, Jul. 2017.

[23] D. Haak, Y. Z. Filmwala, E. Heder, S. Jonas, P. Boor, and T. M. Deserno, “An imagej plugin for whole slide imaging,” in *Informatik aktuell*, 2014, pp. 415–420.

[24] R. Singh, L. Chubb, L. Pantanowitz, and A. Parwani, “Standardization in digital pathology: Supplement 145 of the DICOM standards,” *J. Pathol. Inform.*, vol. 2, no. 1, p. 23, 2011.

[25] G. Romero Lauro *et al.*, “Digital pathology consultations - A new era in digital imaging, challenges and practical applications,” *J. Digit. Imaging*, vol. 26, no. 4, pp. 668–677, Aug. 2013.

[26] C. Allan *et al.*, “OMERO: Flexible, model-driven data management for experimental biology,” *Nature Methods*, vol. 9, no. 3. pp. 245–253, Mar-2012.

[27] P. Bankhead *et al.*, “QuPath: Open source software for digital pathology image analysis,” *Sci. Rep.*, vol. 7, no. 1, Dec. 2017.

[28] R. Marée *et al.*, “Collaborative analysis of multi-gigapixel imaging data using Cytomine,” *Bioinformatics*, vol. 32, no. 9, pp. 1395–1401, May 2016.

[29] M. Stritt, A. K. Stalder, and E. Vezzali, “Orbit Image Analysis: An open-source whole slide image analysis tool,” *bioRxiv*, p. 731000, 2019.

[30] M. Satyanarayanan, A. Goode, B. Gilbert, J. Harkes, and D. Jukic, “OpenSlide: A vendor-neutral software foundation for digital pathology,” *J. Pathol. Inform.*, vol. 4, no. 1, p. 27, 2013.

[31] "Free whole slide image viewer – PMA.start – Universal digital microscopy software." [Online]. Available: https://free.pathomation.com/. [Accessed: 08-Jun-2020].

[32] R. Jesus, P. Nunes, R. Lebre, and C. Costa, "Role-Based Architecture for Secure Management of Telepathology Sessions," *Stud. Health Technol. Inform.*, vol. 270, pp. 663–667, Jun. 2020.

[33] P. Nunes, R. Jesus, R. Lebre, and C. Costa, "Data and sessions management in a telepathology platform," in *HEALTHINF 2020 - 13th International Conference on Health Informatics, Proceedings; Part of 13th International Joint Conference on Biomedical Engineering Systems and Technologies, BIOSTEC 2020*, 2020, pp. 455–462.

[34] K. Abouelmehdi, A. Beni-Hessane, and H. Khaloufi, "Big healthcare data: preserving security and privacy," *J. Big Data*, vol. 5, no. 1, pp. 1–18, Dec. 2018.

[35] H. Li, L. Yu, and W. He, "The Impact of GDPR on Global Technology Development," *Journal of Global Information Technology Management*, vol. 22, no. 1. Taylor and Francis Inc., pp. 1–6, 02-Jan-2019.

[36] R. Lebre, L. Bastião, and C. Costa, "Shared Medical Imaging Repositories.," *Stud. Health Technol. Inform.*, vol. 247, pp. 411–415, 2018.

[37] M. N. Gurcan, L. E. Boucheron, A. Can, A. Madabhushi, N. M. Rajpoot, and B. Yener, "Histopathological Image Analysis: A Review," *IEEE Rev. Biomed. Eng.*, vol. 2, pp. 147–171, 2009.

[38] M. Veta, J. P. W. Pluim, P. J. Van Diest, and M. A. Viergever, "Breast cancer histopathology image analysis: A review," *IEEE Transactions on Biomedical Engineering*, vol. 61, no. 5. IEEE Computer Society, pp. 1400–1411, May-2014.

[39] H. Fox, "Is H and E morphology coming to an end?," *Journal of Clinical Pathology*, vol. 53, no. 1. BMJ Publishing Group, pp. 38–40, 01-Jan-2000.

[40] M. Al-Abbadi, "Basics of cytology," *Avicenna J. Med.*, vol. 1, no. 1, p. 18, 2011.

[41] S. Beg, S. Vasenwala, N. Haider, S. Ahmad, V. Maheshwari, and M. A. Khan, "A comparison of cytological and histopathological findings and role of immunostains in the diagnosis of soft tissue tumors," *J. Cytol.*, vol. 29, no. 2, pp. 125–130, Apr. 2012.

[42] S. A. Hoda and R. S. Hoda, "Rubin's Pathology: Clinicopathologic Foundations of Medicine, 5th Edition," *JAMA*, vol. 298, no. 17, p. 2070, Nov. 2007.

[43] S. Robertson, H. Azizpour, K. Smith, and J. Hartman, "Digital image analysis in breast pathology—from image processing techniques to artificial intelligence," *Translational Research*, vol. 194. Mosby Inc., pp. 19–35, 01-Apr-2018.

[44] H. Reza Tizhoosh and L. Pantanowitz, "Artificial intelligence and digital pathology: Challenges and opportunities," *J. Pathol. Inform.*, vol. 9, no. 1, Jan. 2018.

[45] J. Schindelin, C. T. Rueden, M. C. Hiner, and K. W. Eliceiri, "The ImageJ ecosystem: An open

platform for biomedical image analysis," *Molecular Reproduction and Development*, vol. 82, no. 7–8. John Wiley and Sons Inc., pp. 518–529, 01-Jul-2015.

[46] C. A. Schneider, W. S. Rasband, and K. W. Eliceiri, "NIH Image to ImageJ: 25 years of image analysis," *Nature Methods*, vol. 9, no. 7. pp. 671–675, 28-Jul-2012.

[47] B. A. Mysona *et al.*, "Qupath automated analysis of optic nerve degeneration in brown Norway rats," *Transl. Vis. Sci. Technol.*, vol. 9, no. 3, pp. 22–22, Feb. 2020.

[48] M. R. Lamprecht, D. M. Sabatini, and A. E. Carpenter, "CellProfiler^{TM}: Free, versatile software for automated biological image analysis," *Biotechniques*, vol. 42, no. 1, pp. 71–75, Jan. 2007.

[49] A. E. Carpenter *et al.*, "CellProfiler: Image analysis software for identifying and quantifying cell phenotypes," *Genome Biol.*, vol. 7, no. 10, p. R100, Oct. 2006.

[50] D. P. Pop and A. Altar, "Designing an MVC model for rapid web application development," in *Procedia Engineering*, 2014, vol. 69, pp. 1172–1179.

[51] S. bin Uzayr, N. Cloud, T. Ambler, S. bin Uzayr, N. Cloud, and T. Ambler, "RequireJS," in *JavaScript Frameworks for Modern Web Development*, 2nd ed., Apress, 2019, pp. 91–131.

[52] E. Brown, "Web development with node and express: leveraging the JavaScript stack," 2nd ed., O'REILLY, 2019.

[53] M. Ramos, M. T. Valente, and R. Terra, "AngularJS performance: A survey study," *IEEE Software*, vol. 35, no. 2. IEEE Computer Society, pp. 72–79, 01-Mar-2018.

[54] N. Jain, P. Mangal, and D. Mehta, "AngularJS: A Modern MVC Framework in JavaScript," *J. Glob. Res. Comput. Sci.*, 2014.

[55] "OpenSeadragon." [Online]. Available: http://openseadragon.github.io/. [Accessed: 08-Jun-2020].

[56] "Fabric.js Javascript Canvas Library." [Online]. Available: http://fabricjs.com/. [Accessed: 09-Jun-2020].

[57] M. Gajewski and W. Zabierowski, "Analysis and comparison of the spring framework and play framework performance, used to create web applications in Java," in *2019 IEEE 15th International Conference on the Perspective Technologies and Methods in MEMS Design, MEMSTECH 2019 - Proceedings*, 2019, pp. 170–173.

[58] M. Ahmed, M. M. Uddin, M. S. Azad, and S. Haseeb, "MySQL Performance Analysis on a Limited Resource Server: Fedora vs. Ubuntu Linux," 2010.

[59] Z. Parker, S. Poe, and S. V. Vrbsky, "Comparing NoSQL MongoDB to an SQL DB," in *Proceedings of the Annual Southeast Conference*, 2013, p. 1.

[60] G. Matei, "Column-Oriented Databases, an Alternative for Analytical Environment," 2010.

[61] B. B. Rad, H. J. Bhatti, and M. Ahmadi, "An Introduction to Docker and Analysis of its Performance,"

2017.

[62] J. Rosai *et al.*, “Standardization of the surgical pathology report.,” *Mod. Pathol.*, vol. 5, no. 2, pp. 197–199, Mar. 1992.

[63] “Microscope Objective Specifications | Nikon’s MicroscopyU.” [Online]. Available: https://www.microscopyu.com/microscopy-basics/microscope-objective-specifications. [Accessed: 09-Jun-2020].