Rogério Gomes Lopes Moreira

Building an imaging-based research platform for experiments with Brain connectivity data

November 2019

Building an imaging-based research platform for experiments with brain connectivity data

Rogério Gomes Lopes Moreira

UMinho 2019

Universidade do Minho
Escola de Engenharia

Rogério Gomes Lopes Moreira

Building an imaging-based research platform for experiments with brain connectivity data

Master's Dissertation
Integrated Master's in Informatics Engineering

Dissertation oriented by
Victor Manuel Rodrigues Alves
Nicolás Francisco Lori

November 2019

Acknowledgements

I would like to thank my supervisor, Victor Alves for his guidance, availability, sharing and encouragement. His advices and assistant were truly important to me. I would also like to thank Nicolás Lori, my co-supervisor, who taught me so much about neuroimaging.

Finally, to my parents, my brother and Jorge for all the support and advice in my decisions. To my course colleagues, Samuel, Gustavo and Diogo who supported me greatly and were always willing to help me. To Luis, my teacher who truly changed the way I saw the world. I'm forever thankful for his insights and mind opening. And finally, to Madalena, for being with me all the time in good and bad and supporting me through my academic journey. Thank you for all your love.

DECLARATION

Name: Rogério Gomes Lopes Moreira

Dissertation Title: Building an imaging-based research platform for experiments with brain connectivity data

Mentors: Victor Manuel Rodrigues Alves, Nicolás Francisco Lori

Conclusion Year: 2019

Master Designation: Mestrado Integrado em Engenharia Informática

Master Branch: Informática Médica

Universidade do Minho, _____/_____/_____

Signature: _____

STATEMENT OF INTEGRITY

I hereby declare having conducted this academic work with integrity. I confirm that I have not used plagiarism or any form of undue use of information or falsification of results along the process leading to its elaboration.

I further declare that I have fully acknowledged the Code of Ethical Conduct of the University of Minho.

Universidade do Minho, _____/_____/_____

Signature: _____

ABSTRACT

Within the past decade, not only societies in general but also medicine and healthcare, in particular, have changed tremendously. In large part because of the rapid dissemination of computers and digital communications which lead to the appearance of new medical disciplines, such as Medical Informatics. Nowadays, one of the most prominent field in Medical Informatics is Medical Imaging, as it is implied, it is a collection of methodologies and techniques used in order to visually and spatially represent parts of the brain for diagnostic and research purposes.

In the research ecosystem, Neuroimaging is an increasing popular field, with applications in neurology and psychiatry. However, due to the difficulties to handle Neuroimaging data, since data has its own specificities, researchers have encountered problems to correctly handling this data. This can be a crucial issue specially with large volumes of Neuroimaging data and all the research materials associated.

This work aims to architect and build a research platform to correctly archive Medical Imaging data and all the associated research materials, where researchers can exchange imaging data and collaborate in Neuroimaging research projects. The platform offers a correct way to collect and store all imaging data, archiving all of patient exams with the correspondent information, making available the correspondent information to researchers in a confidential, secure and efficient way.

The two main outcomes of this work are an architecture of a platform that manages all imaging data and associated research materials, plus an open-source Python package to easily interact with that platform.

- **Keywords:** Data Sharing, Medical Imaging, Neuroinformatics, Neuroimaging, XNAT.

RESUMO

Na última década as sociedades em geral, mas a medicina e os cuidados de saúde em particular mudaram significativamente. Em grande parte, devido à rápida disseminação dos computadores e das comunicações digitais, o que permitiu o aparecimento de novas disciplinas médicas como é o caso da Informática Médica. Hoje em dia, um dos campos mais proeminentes na Informática Médica é a Imagem Médica, a coleção de metodologias e técnicas usadas para visualmente e espacialmente representar partes do cérebro para diagnóstico e investigação.

Em investigação, a Neuroimagem é cada vez mais popular, com aplicações que vão desde a neurologia até à psiquiatria. Contudo, os investigadores têm encontrado um problema em como gerir estes dados já que estes têm as suas próprias especificidades e são, normalmente, armazenados em grandes quantidades.

Este trabalho pretende arquitetar e construir uma plataforma para armazenar e gerir dados de imagem médica e todos os materiais de investigação associados onde os investigadores possam arquivar corretamente e partilhar com outros interessados. Assim como, construir ferramentas para interagir programaticamente com a plataforma, tendo como objetivo a sua integração no fluxo de trabalho atual.

- **Palavras-Chave:** Partilha de Dados, Imagiologia Médica, Neuro-informática, XNAT.

TABLE OF CONTENTS

1 Introduction 15

1.1 Context and Motivation 16

1.2 Objectives 19

1.3 Dissertation structure 19

2 State of the art 21

2.1 Medical Imaging data particular demands 22

2.1.1 Data availability 23

2.1.2 Data dimensionality and size 23

2.1.3 Data formatting 23

2.1.4 Data properties 24

2.2 Why it is important to share Medical Imaging data and research materials 24

2.3 Human Connectome Project 25

2.4 Central Neuroimaging Data Archive 25

2.5 Northwestern University Neuroimaging Data Archive (NUNDA) 26

2.6 Gift-Cloud 26

3 Platform design and architecture 28

3.1 Purposed solution 29

3.2 Platform architecture 31

3.2.1 Overview 31

3.2.2 Integration with other instances and hospital systems 31

3.2.3 Data security and access control 32

3.2.4 Workflow 32

3.3 Technologies and Concepts 33

3.3.1 Virtualization software 33

3.3.2 Data repository management system 35

3.3.3 Database management system (DBMS) 39

4 Building the research platform – XNAT@DI 40
4.1 Platform Server 42
4.2 Sharing data between instances 43
4.3 Containerization and platform deploy 44
4.4 XNAT configurations 49
4.5 XNAT plugins and pipelines 50
4.6 Organizing and uploading existing data 51
5 XNATUM – A Python API for XNAT 53
5.1 Context and motivation 54
5.1.1 Python in Neuroimaging and neuroscience 55
5.1.2 XNAT REST API 55
5.1.3 Development 56
5.2 XNATUM Implementation 58
5.3 Use-case examples 61
5.4 Using a virtual RAM drive 64
5.5 Discussion 65
6 Conclusions 69
6.1 Work contributions 71
References lxxii

Appendices lxxix

A - XNAT User Guide ..... *lxxx*

A.1 Introduction ..... *lxxx*

A.2 Site Access ..... *lxxx*

A.3 Navigation ..... *lxxxi*

A.4 Data Access ..... *lxxxii*

A.5 Manage sessions data with horos ..... *lxxxiii*

B – XNAT Data Model ..... *lxxxiv*

LIST OF FIGURES

Figure 1.1 – UMinho Neuroimaging project 17
Figure 3.1 - Platform conceptual architecture 30
Figure 3.2 – Platform conceptual architecture network 31
Figure 3.3 - Docker engine layers (image from [47]) 34
Figure 3.4 - The interface of Portainer, the Docker management tool 35
Figure 3.5 - XNAT architecture [54] 36
Figure 3.6 – Platform conceptual architecture with XNAT 37
Figure 4.1 – XNAT@DI platform architecture applied at UMINHO 42
Figure 4.2 – XNAT JAAT method explained 45
Figure 5.1 - XNATUM interactions with XNAT and the filesystem 54
Figure 5.2 - XNAT REST API syntax 55
Figure 5.3 - XNATUM GitHub repository 57
Figure 5.4 - XNATUM Pypi page 58
Figure 5.5 - XNAT REST model 59
Figure 5.6 - Daily downloads since the package is available at Pypi 67
Figure 5.7 - Daily downloads proportion between different versions of Python 67
Figure 5.8 - Daily downloads between different versions of Python 67
Figure 5.9 - Daily downloads proportion between different versions of Python 68
Figure 5.10 - Daily downloads between different system 68
Figure 5.11 - Daily downloads proportion between different system 68
Figure A.1 - UMinho XNAT platform lxxx
Figure A..2 - HOROS XNAT location configuration lxxxiii
Figure B.3 - XNAT Data Model lxxxvi

LIST OF PROGRAMS

Program 4.1 - Docker Compose configuration file 46
Program 4.2 - The symbolic links created 47
Program 4.3 - The exposed ports on the XNAT container 47
Program 4.4 - The exposed ports 47
Program 4.5 - The exposed ports on the Nginx container 48
Program 4.6 - New fields added to the XML project subjects configuration file 52
Program 4.7 - Python script to upload all sessions to XNAT@DI 52
Program 5.1 - Example of creating a connection to a XNAT instance 59
Program 5.2 - Creating a connection to a XNAT instance using the Python context operator 60
Program 5.3 - Getting all experiments of a XNAT project 60
Program 5.4 - XNATUM session upload function 60
Program 5.5 - Download project sessions through XNATUM 61
Program 5.6 - Download project sessions to directory through XNATUM 61
Program 5.7 - Download subject sessions through XNATUM 62
Program 5.8 - Download subject sessions to directory through XNATUM 62
Program 5.9 - Download specific subject session to directory through XNATUM 62
Program 5.10 - List all project subjects through XNATUM 62
Program 5.11 - Get project sessions labels through XNATUM 63
Program 5.12 - Get all information from a subject within a project through XNATUM 63
Program 5.13 - Upload a sessiont a subject through XNATUM 63

ABBREVIATIONS AND ACRONYMS

A

AI	Artificial Intelligence
API	Application Programming Interface

B

BIRN	Biomedical Informatics Research Network

C

CNDA	Central Neuroimaging Data Archive
CSV	Comma-separated values
CT	Computed Tomography

D

DICOM	Digital Imaging and Communications in Medicine
DBMS	Database Management System
DL	Deep Learning

F

fMRI	Functional magnetic resonance imaging

G

GUI	Graphical User Interface

H

HDD	Hard Disk Drive
HTTP	Hypertext Transfer Protocol
HCP	Human Connectome Project

J

JAAT	Joint Anonymization and Archive Transmission

L

Loni IDA	Loni Image Data Archive

LXC Linux Containers

M

ML Machine Learning

MRI Magnetic Resonance Imaging

MR Magnetic Resonance

MVCC Multiversion concurrency control

N

NIfTI Neuroimaging Informatics Technology Initiative

NM Nuclear Medicine

NIAC Neuroimaging Informatics and Analysis Center

NGINX Engine-X

NA-MIC National Alliance for Medical Image Computing

O

OS Operating System

P

PET Positron-Emission Tomography

PITR Point-in-Time-Recovery

R

REST Representational State Transfer

S

SFTP Secure File Transfer Protocol

SCP Service Class Provider

SPECT Single-photon emission computed tomography

SQL Structured Query Language

SSD Solid-State Drive

U

UI User Interface

UMINHO University of Minho

US United Stated

V

VM Virtual Machine

X

XDAT eXtensible Data Archive Toolkit

XML eXtensible Markup Language

XNAT eXtensible Neuroimaging Archive Toolkit

XSD XML Schema Definition

XTK X-Ray Toolkit

Y

YAML YAML Ain't Markup Language

GLOSSARY

Dataset A collection of examples. Each example contains one or more features and a label (if using supervised training).

DICOM DICOM (Digital Imaging and Communications in Medicine) is a standard for handling, storing, printing, and transmitting information in Medical Imaging. It includes a file format definition and a network communications protocol.

Magnetic Resonance Imaging (MRI) Non-invasive technique used disease detection and treatment monitoring. MRI scanners are particularly well suited to image the soft tissues of the body e.g., brain, spinal cord and nerves, muscles and ligaments.

Medical Imaging Informatics A discipline that results from the combination of medical images and biomedical informatics. Serves to enhance the improvement and knowledge of clinical care. On the one hand, Medical Imaging allows the study of a disease state *in vivo*. On the other hand, biomedical informatics is involved in the development of computer science techniques for creating and managing medical data.

Nifty A simple, minimalistic format which has been widely adopted in Neuroimaging research, allowing scientists to mix and match image processing and analysis tools developed by different teams.

XNAT An open source imaging informatics software platform designed to facilitate common management and handling of Neuroimaging and associated data.

1 INTRODUCTION

1.1 CONTEXT AND MOTIVATION

Medical informatics is a recent medical discipline when comparing with other medical disciplines that have been around for centuries [1]. Within the past decade societies in general, and medicine and healthcare in particular, have tremendously changed, in large because of the rapid dissemination of computers and digital communications [2]. This allowed new medical disciplines to appear, such as Medical Informatics, which is the intersection of computer informatics and healthcare [3]. This new discipline deals with the resources, devices, and methods required to optimize the acquisition, storage, retrieval, and use of information in healthcare systems.

One of the most prominent field in Medical Informatics is Medical Imaging, as it is implied, it is a collection of methodologies and techniques used in order to visually and spatially represent parts of the human body for diagnostic and research purposes. This field is so important nowadays that it is hard to imagine a medical diagnostic without the use of imaging tools since they allow doctors to have a clear picture of the problem. In research, Medical Imaging has gained considerable attention due to its widespread utility in a multitude of clinical applications and the considerable improvements in the techniques [4].

One of the most prominent fields in Medical Imaging is the representation/study of the brain, known as Neuroimaging. This field is a set of imaging techniques that either directly or indirectly image the structure or function of the nervous system, which includes the brain [5]. Neuroimaging comprises several different modalities such as Magnetic Resonance Imaging (MRI), Positron Emission Tomography (PET), Computed Tomography (CT) and functional MRI (fMRI) [6]. These techniques have major clinical applications in neurology, serving as a crucial part in diagnosis and, more recently, in psychiatry, with special focus on psychiatric disorders by helping to understand the interactions between the brain and the body. All of these make Neuroimaging an increasing popular field of research, especially in the last 20 years [7].

In conjunction with this, in the last few years has been a growing interest in including Machine Learning (ML) [8] and Deep Learning (DL) [9] in the analysis of Neuroimaging data.

Machine Learning addresses the question of how to make computers improve automatically though experience. It is one of most rapidly growing technical fields, joining areas like computer science, statistics and data science [10].

Deep Learning is one of the most recognized ML techniques, which allows computational models composed of multiple layers to learn representations of data with multiple levels of abstraction. These methods have dramatically improved hard computer problems like speech recognition, visual object

recognition and object detection. Deep Learning discovers a structure in large data sets by using an algorithm to indicate how a machine should change its internal parameters, used to compute the representation in each layer from the representation in the previous layer [11].

A large number of studies show that ML and DL can be used to extract new exciting information from Neuroimaging data [12]. This field of research has revealed itself to be very complex from the beginning as it requires analysis of complex and multivariate data [13]. Data with both these characteristics is precisely what leads to the development of computer techniques such as ML and DL which allow to train classifiers that decode data and extract the relevant information. This would not be possible with the so-called "traditional computing", since the volumes of information to decode are extremely large [13].

The recent progress in ML has been driven both by the development of new algorithms and by the availability of data. With this, researchers rely more and more upon large datasets that sometimes are not correctly labeled and archived, which can result in unreliable results. This is especially true in Neuroimaging since the data is difficult to handle and is most of the time sliced in multiple files.

For all of these reasons, a research platform to correctly store Medical Imaging data plus all the associated research materials is a crucial part of a larger sought ecosystem where researchers can exchange data and collaborate in Neuroimaging research projects.

One of these ecosystems was developed through the collaboration between the Algoritmi Research Center and the Life and Health Sciences Research Institute (ICVS), both at University of Minho (UMinho) [14], with the objective to map and search for patterns on the human brain (Fig 1.1).

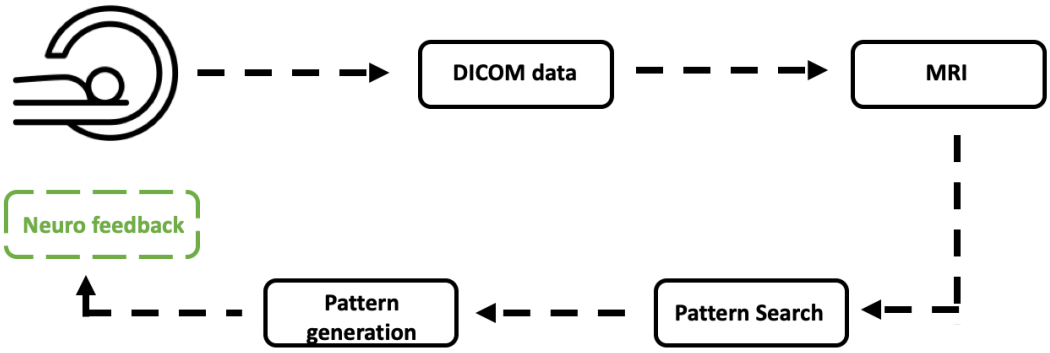

Figure 1.1 – UMinho Neuroimaging project

This work has the purpose to architect a platform to archive Neuroimaging data and build a prototype with the planned architecture to be integrated in the current system workflow of the UMinho Neuroimaging project but implemented in such a way that it can easily be transferred to other workflow systems. The developed platform has the objective to facilitate the distribution of Neuroimaging data available in both

public datasets and acquired locally at the ICVS MRI scanner. With the workflow used before, researchers moved Neuroimaging data and the associated research materials via hard-drives or platforms like Redmine [15], which is not scalable as the dataset sizes increase.

The platforms that handle Medical Imaging data and in particular neuroimages have unique characteristics, that make it different from traditional sharing systems. Also, medical research has rigid workflows that can't be changed easily. Commonly, platforms that try to change these workflows don't succeed. The system to be built must seamlessly integrate in the actual workflow, with only minor adaptations, but also be integrable in a global Neuroimaging data sharing network (Fig. 1.2).

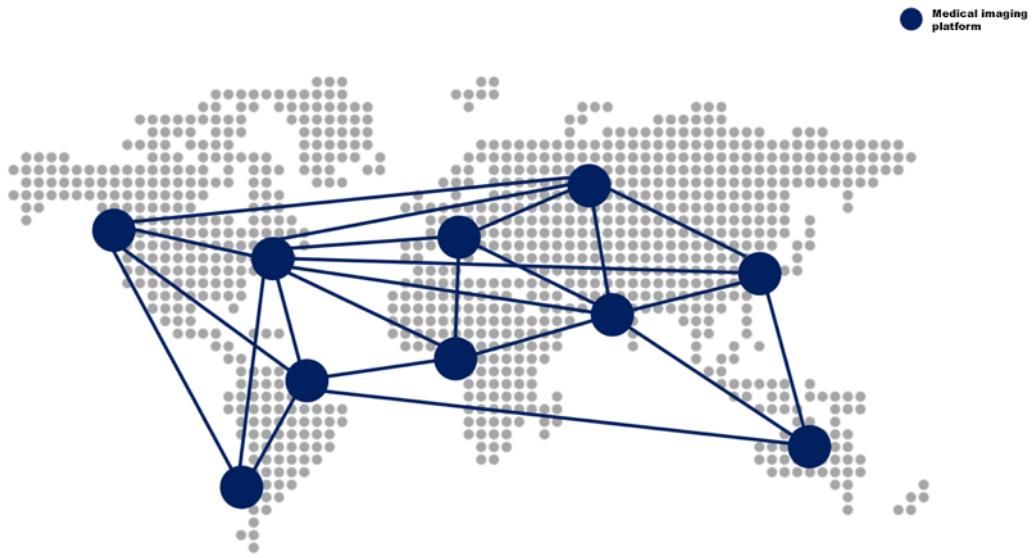

Figure 1.2 – Example of multiple Medical Imaging platforms connected to share research data

Over the last years, ICVS and Algoritmi have been collaborating in studies about Neuroimaging and nowadays all the ongoing research involves a large number of disperse researchers, large datasets and different data sources. With this increase in popularity, a new problem was encountered. How to properly store, manage and make available this data and the research materials associated?

The methods used until now duplicate the space used and don't ensure properly data archiving, plus, as research increasingly specializes in a narrower range (Fig. 1.2) of applications it becomes harder and harder to obtain sufficient data if it is dispersed across researchers and projects.

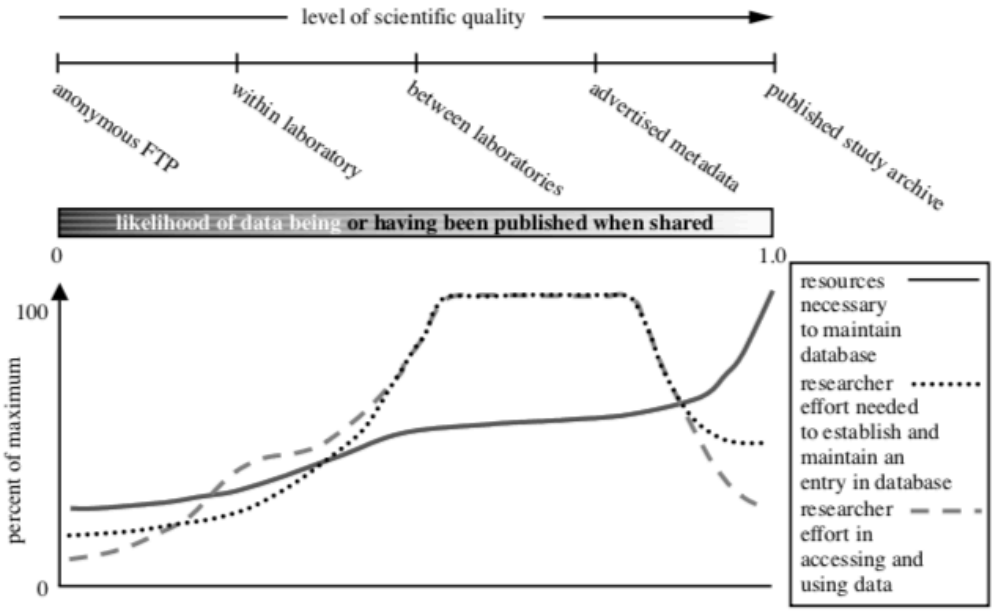

Figure 1.3 - Comparing scientific quality versus amount of shared data (image from [16])

Taking for example another research ecosystem like the one at UMinho, the Cancer Imaging Archive [17] has more than 25.000 patients/subjects, 73.000 studies and 21 million images which represents more than 11 terabytes of data, all available to researchers. Any researcher can easily select data from the global dataset with the required characteristics and use the data without the need to make a new acquisition.

1.2 OBJECTIVES

The main objective of this work is to architecture an imaging-based research platform for the implementation of experiments with Medical Imaging data and associated research materials, with a particular focus on Neuroimaging. Also, as this type of platform involves multiple iterations, a prototype of the work will be available in ICVS and Algoritmi to manage all the Neuroimaging data used there. Furthermore, as researchers need a simple way to programmatically interact with the platform, there is also the development of both APIs and libraries for the interaction with the developed platform.

1.3 DISSERTATION STRUCTURE

The two focuses of this work are the XNAT@DI platform and the XNATUM package. The dissertation is structured as follows. Section 2 reviews published systems in the field and discusses the additional

requirements that have motivated the development of the XNAT@DI project. Section 3 is an overview of the system's features, technologies and the overall architecture of the platform. Section 4 presents implementation details of the platform. Section 5 presents an overview and the implementation of the XNATUM package, including descriptions of features, performance and testing. Finally, Section 6 presents conclusions and future work.

2 STATE OF THE ART

In the last century, the industrial and the scientific world have witnessed great changes brought by the development of technology. These changes have been modernizing many fields and industries [18]. Biomedicine is one of the areas that has benefited more from these advances, which led to the development of new areas [18] such as health informatics, the application of software engineering to healthcare, Medical Imaging, bioinformatics and neuroinformatics.

At the same time, the ability to obtain digital information in the medical research world has increased in big part due to a significant implementation rate of electronic medical records, which by 2025 can reach $33000 million in market value [19].

Therefore, the need to share and maintain data resources at very different scales, from large studies with researchers distributed around the world to individual research centers, resulted in the development of data management systems specialized in Medical Imaging and Neuroimaging. Medical Imaging management systems have special requirements which led to the development of special ontologies [20], data format exchanges [21] and different approaches to management system like the Human Imaging Database (HID) [22] , the LONI IDA [23] and The Extensible Neuroimaging Archive Toolkit (XNAT) [24]. This type of studies often include data passing through a series of capture, quality control, processing, and utilization steps which also incorporate measures from a range of other research materials (e.g. genetic, clinical and neuropsychological data). All these steps must be integrated, in a secure way, with the imaging measures into a unified dataset [25]. This means that, apart from the subjects, the imaging process is intrinsically a digital electronic enterprise as image acquisition, processing, databasing, and sharing are all accomplished in the digital domain. Each step of this process, therefore, affords the opportunity to capture all the pertinent information that characterizes the steps [26].

2.1 MEDICAL IMAGING DATA PARTICULAR DEMANDS

The healthcare industry historically has generated large amounts of data, driven by record keeping, compliance, regulatory requirements and patient care [27]. While most of the data is stored in hard drives, the current trend is toward a rapid digitization of these large datasets [28]. However, Medical Imaging data and the associated research materials have particular demands that differ from other domains where this type of management platform is applied. Due both to the characteristics of the data itself and to the applications in which they are used. Furthermore, Medical Imaging and the associated research materials have strict regulations and pre-set workflows [29] which are widely used in the academic community and industry, and hence can be difficult to change.

Also, Medical Imaging data obeys to strict rules for data processing including, for example, explicit consent of the data subject [30]. All of these reasons make Medical Imaging and Neuroimaging a special field on the data management systems topic with very particular demands on the data such as: availability, dimensionality, formatting and properties.

2.1.1 DATA AVAILABILITY

Acquiring, annotating and distributing medical image data sets have higher cost than other computer visions tasks. Generating Medical Imaging in any of the modalities is costly, and annotating images requires high levels of expertise from clinicians and researchers. Additionally, due to privacy concerns, sharing data sets between institutions is logistically and legally challenging. For all of these reasons, it is extremely important to correctly archive all acquired images.

2.1.2 DATA DIMENSIONALITY AND SIZE

Data dimensionality encountered in medical image analysis and computer-assisted intervention typically ranges from 2D to 5D. Many medical images, including MRI, CT, PET and SPECT capture volumetric images. At the same time, capturing high-resolution data in multiple dimensions is often necessary to detect small but clinically important anatomy and pathology. The combination of these factors results in large data sizes for each sample, which impact computational and memory costs.

2.1.3 DATA FORMATTING

Data sets in Medical Imaging are typically stored in formats very different from those used in computer vision tasks. To support the higher-dimensional medical image data, specialized formats have been adopted (e.g. DICOM, Nifty, Analyze). These formats frequently also store metadata that is critical to image interpretation, including spatial information (e.g. anatomical orientation and voxel anisotropy), patient information (e.g. demographics and identifiers) and acquisition information (e.g. modality types and scanner parameters). These Medical Imaging specific data formats are typically not supported by existing management platforms, requiring custom infrastructure for image loading and analysis.

2.1.4 DATA PROPERTIES

The characteristic properties of medical image content pose opportunities and challenges. Medical images are obtained under controlled conditions, allowing more predictable data distributions. In many modalities, images are calibrated so that spatial relationships and image intensities map directly to physical quantities and are inherently normalized across subjects. For a given clinical workflow, image content is typically consistent, potentially enabling the characterization of plausible intensity and spatial variation for data augmentation.

2.2 WHY IT IS IMPORTANT TO SHARE MEDICAL IMAGING DATA AND RESEARCH MATERIALS

Researchers around the world have argued that more rapid scientific discoveries are possible within a culture of shared data [31], not just because some questions can only be answered with large datasets but also because of the increase of use of Machine Learning (ML) and Deep Learning (DL) techniques, which require large datasets in order to discover patterns [8]. For all these reasons, it's important to share in a meaningful and proper way Medical Imaging data. This will accelerate progress in our fundamental understanding of the world, improving publication and data quality, reducing the cost of research and increasing the return on current research investments.

Searching for example PubMed, one of the largest medic publications repositories, we can find innumerous publications about Neuroimaging (over 22.000 just fMRI-related) that were published between the early 1990s and 2011. "A conservative estimate of the data this represents amounts to 12.000 datasets with 12 subjects each and hour-long scans per subject, at a cost of 300$/hour. This corresponds to 144.000 scans hours (144TB of raw data and over 43 million US dollars in scan hour costs)" [25]. However, the proportion of such data currently available in public repositories is less than a few percent. Even when available for all research community, the authorization required to access the data may slow their re-distribution and use [25].

With all of this in mind, it is important to start sharing all Medical Imaging data produced and used at UMinho for all interested researchers which will accelerate the studies and save time and money but also, starting to think globally as a research Neuroimaging community in which all researchers can collaborate and use each other's data. Below are presented some examples of platforms that already help researchers manage Neuroimaging data.

2.3 HUMAN CONNECTOME PROJECT

As mentioned before, a revolution in noninvasive Neuroimaging methods over the pasts decades has enabled the analysis and visualization of human brain structure, function and connectivity in unprecedented detail [32]. The Human Connectome Project (HCP) is part of that. "The consortium led by Washington University, University of Minnesota and Oxford University is undertaking a systematic effort to map macroscopic human brain circuits and their relationship to behavior in a large population of healthy adults" [32]. The project was launched in July 2009 [33] with the goal to build a "network map" that will shed light on the anatomical and functional connectivity within the healthy human brain, as well as produce data to facilitate research into brain disorders such as Dyslexia, Autism or Alzheimer's [34].
The HCP has made substantial improvements in data acquisition, analysis, and sharing. In aggregate, these advances constitute a new "HCP – style" Neuroimaging paradigm, offering a modern alternative to the traditional approach that has dominated the field for more than two decades.
The ConnectomeDB, the data management platform that houses all data generated by the Human Connectome Project, clinical evaluations, behavioral data and more is based on the XNAT platform.

2.4 CENTRAL NEUROIMAGING DATA ARCHIVE

The Central Neuroimaging Data Archive (CNDA) is an informatics platform to manage and sharing neuroimaging and related data. It was built to support the Neuroimaging research group at Washington University, enabling a wide variety of researches like oncology and cardiovascular. The NeuroInformatics Research Group (NRG) operates the CDNA at the Washington University School of Medicine [24].
The CNDA was designed to facilitate common data management and productivity tasks for Neuroimaging and associated data. The platform includes features like DICOM direct upload, a REST API to upload and download data, quality control modules, web-based interface, powerful and sophisticated search, online image viewer and pipeline engine for automating image processing tasks. This platform originated XNAT, an open-source platform [35] available to everyone.
The CDNA currently hosts more than 1200 projects, from over 30.000 subjects with a total of almost 60.000 individual sessions. From these sessions, almost 78% are MR sessions, 11% are PET sessions, 6% are CT sessions and the others are distributed between US sessions, DX sessions, and NM sessions [36].

The group responsible for the CDNA develops technologies that support Neuroimaging and imaging informatics research that helps us better understand the living brain. The group contributes to projects like XNAT, the Human Connectome Project, NIAC and, of course, the CDNA.

2.5 NORTHWESTERN UNIVERSITY NEUROIMAGING DATA ARCHIVE (NUNDA)

The Northwestern University Neuroimaging Data Archive (NUNDA) is a XNAT-powered data archiving system with the objective to secure Neuroimaging data storage, centralize data management, automatize Neuroimaging data processing [37]. This platform follows a different approach from the previous examples since it works like a federated data archive, where project owners retain autonomous control over their data and define project-specific procedures. Currently, NUNDA hosts 131 projects, 4783 subjects, 7972 imaging sessions, and 79,187 scans. Imaging sessions include structural, functional and diffusion magnetic resonance imaging (MRI) scans and the projects range from schizophrenia to dementia, from cancer to cognitive neuroscience, and from humans to primates to rodents. NUNDA is also used as a multisite consortia central collection and repository.

2.6 GIFT-CLOUD

"GITF-Cloud is a data platform that simplifies and automates the process of accessing and sharing data for Medical Imaging research" [38]. This platform has been developed to meet the needs of GIFT-Surg, an international research collaboration that is developing new imaging methods for fetal surgery [39]. Gift-Cloud consists of a central server and optional gateways servers within each different institution and is built using cross-platform technologies, offering flexibility regarding the hardware and operating systems. Some of the features that this platform provide are data upload with fully automated configurable and client-side anonymized patient information, subject matching and patient list mapping while preserving subject anonymity in the research data.

This are only a few of the available examples of platforms to manage and distribute Neuroimaging data. The study of the other existent platforms allows to collect some of the most important features to have on the platform to be built:

- Facilitate common data management and productivity tasks for Neuroimaging and associated data;

- Work as a federated platform where project owners retain autonomous control over their data and define project-specific procedures, but also allow to have global projects, which can be used by any researcher;
- Have the possibility to connect optional gateways servers within each clinical institution;
- Includes features like DICOM direct upload, REST API to upload and download data, quality control modules, web-based interface, powerful and sophisticated search, online image viewer and a pipeline engine for automating image processing tasks.

3 PLATFORM DESIGN AND ARCHITECTURE

The system to be conceived, must be a secure data platform that simplifies and automates the process of accessing and sharing research materials for Medical Imaging with general applicability which can be adapted to benefit research projects across a variety of fields at any research facility.

The main goal of the research platform is to provide a flexible, clinical and researcher-friendly system for sharing Medical Imaging data and research materials across multiple departments and research centers. The system aims to satisfy the demands of patient confidentiality, data security and to bring the benefits from a web platform to imaging research; with the goal of encouraging more data sharing, both standardizing and simplifying the data transfer and anonymization processes, and the providing of centralized storage and backups which simplify the sharing of researcher-derived results. The platform will achieve all of these aims by building upon existing and well-established technologies, accessible and usable to different types of users, and with an ease-of-use appropriate for those that are more or less familiarized with this kind of data and technology.

3.1 PURPOSED SOLUTION

A general system diagram of the platform is shown in Fig.3.1. The platform consists of a central server and optional gateway servers for clinical institutions and research departments. Therefore, the already existing technologies used must be cross-platform compatible, offering complexity with regard to the required hardware and operating system. The service must provide a data upload feature with automated anonymization of patient information and which integrates with software already in use at the research labs like Horos [40] or OsiriX [41].

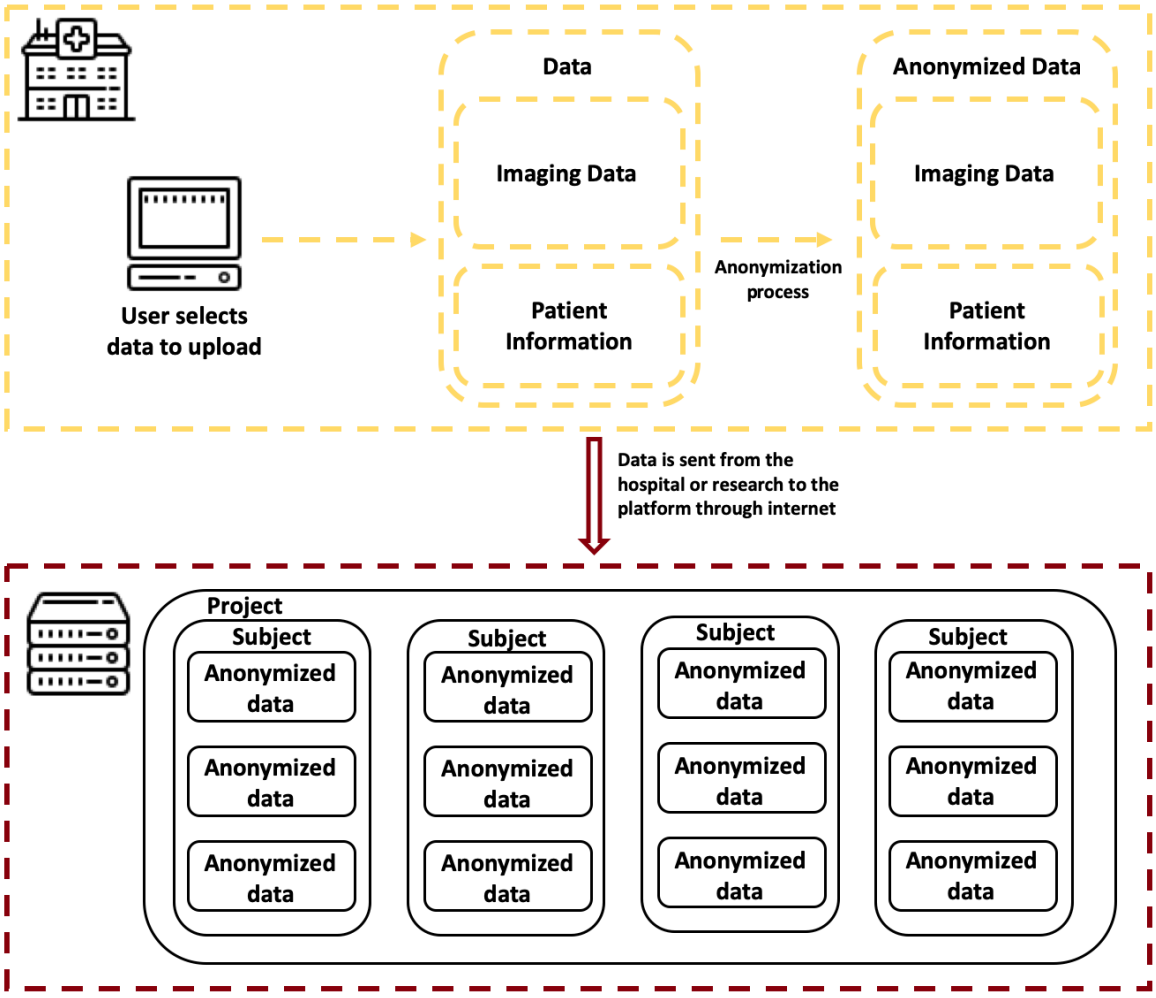

Figure 3.1 - Platform conceptual architecture

3.2 PLATFORM ARCHITECTURE

3.2.1 OVERVIEW

In consequence of the necessity for research centers to manage data there are a lot of different data management systems to facilitate collaborative research and data sharing in Neuroimaging such as Lify [42], Capterra [43], Nora [44] and XNAT [45].

The anonymized research and associated research materials will be hosted on a dedicated instance, with this, users can access and download data within the secure website from a web interface access to any browser. The platform (Fig. 3.2) may also provide a set of protocols and services to be accessed programmatically, to allow integration with third-party software and tools, in a form of a Representational State Transfer Application Programming Interface (REST API).

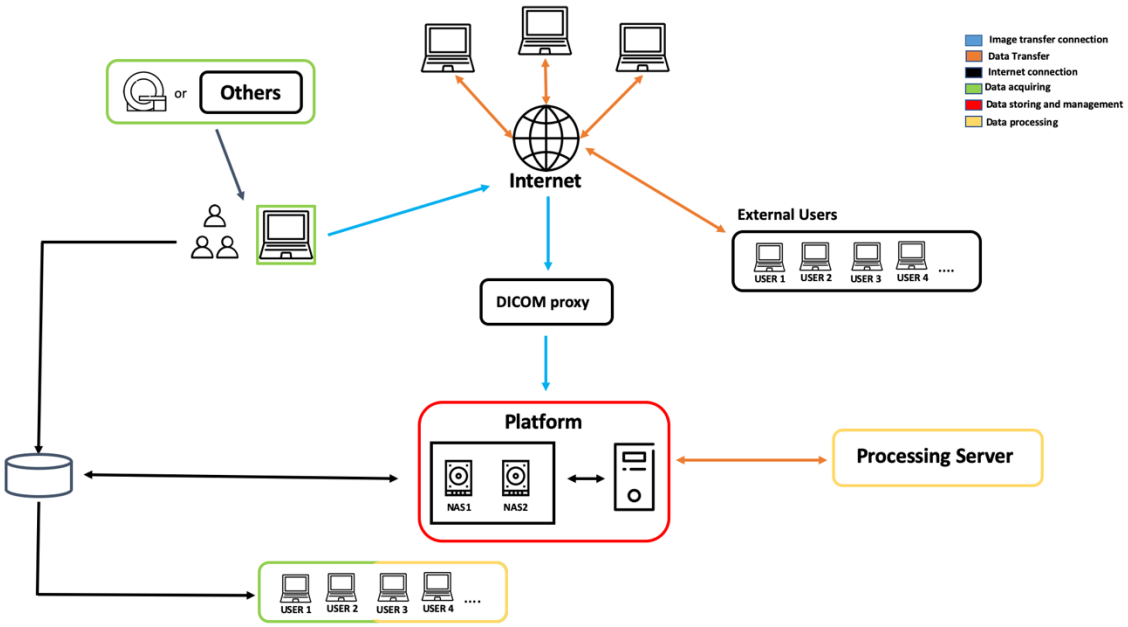

Figure 3.2 – Platform conceptual architecture network.

3.2.2 INTEGRATION WITH OTHER INSTANCES AND HOSPITAL SYSTEMS

The platform must allow the connection of optional gateways to enable each institution and research department to provide image uploading mechanisms suited to their needs. Each interested institution can provide a gateway from their infrastructure to others. There are two main ways this is done with the

current standards: syncing projects between instances, or sharing raw imaging data. The first method establishes a persistent data sync of all or selected data between instances, which allows the researcher to upload the data just once and guarantee that the data is automatically uploaded to the other instances. The second method allows, for example, to move large amounts of DICOM data from hospital facilities and research centers.

3.2.3 DATA SECURITY AND ACCESS CONTROL

All the data stored on the platform to be built is anonymized and not identifiable. All data exchange will happen over the Internet in an encrypted connection with the server certificate ensuring clients can only connect to the genuine server. The server firewalls restrict access only to trusted clients and limit the maximum attempts of failed connections, blocking consecutive failed connections.
All data access requires a personal account. A user can request a user account to the primary Investigator or to a platform administrator.

3.2.4 WORKFLOW

The datasets are either coming from a researcher or directly from a hospital with an MRI machine. There are two main ways data can be uploaded to the platform, via REST API or via the web interface. When uploading data, the most straightforward process is first to upload all project metadata and then the associated subjects and complementary information which can be both made via REST API or via CSV upload on the web interface. Then the anonymized sessions can be uploaded to their correspondent project and subject or to the pre-archive to be analyzed by a project researcher before inclusion in the corresponding project.
Researchers retrieve the anonymized data using integrated software or simply by downloading needed data via the web interface.

3.3 Technologies and Concepts

In this chapter are described some of the technologies that will be used to build the platform as well as well as a justification in why a particular technology was chosen in favor of others. The general preference was to use already well established open-source technologies existent in the market for many years and which have proven their worth in the research area.

3.3.1 Virtualization software

The chosen virtualization software was Docker, a program created to isolate processes from the system on which it is running. It was created to isolate an application and the resources required to run that application from the hardware on which it runs. Docker uses pre-built Linux Containers (LXC) which are an operating system level virtualization built-in directly on the Linux kernel, without the overhead of a full-fledged VM. XNAT application depends in a few different and distinct applications which can cause different types of errors. Docker solves this problem via containerization, ensuring compatibility across platforms and, therefore, eliminating "works on my machine" issues in research and software developing. Despite appearances, the most common Docker containers are not VMs. They have a key difference, while VMs don't share the same Linux kernel, in the Docker containers the contrary happens. Docker containers share the same Linux kernel with the hardware they are running, resulting on much less boot time and resource usage than VM's. This has made Docker particularly attractive to industry and has thus seen a steep rise in adoption of the technology. Docker will be used to containerize the platform on the server as well as simplify and guarantee the consistency of the installation and setup of the system, since some of the used third-party software used is only available for Linux-based systems.
Docker was chosen because it satisfies the practical requirements, and the accessibility of Docker Hub enables images to be quickly found and deployed. "Virtual machines like those created in Virtual Box [46] or VMware [47] provide lots of range in terms of operating systems that can be launched and allow native access to the machine through a GUI. However, though these are great features, they are not necessary for the platform." [48]

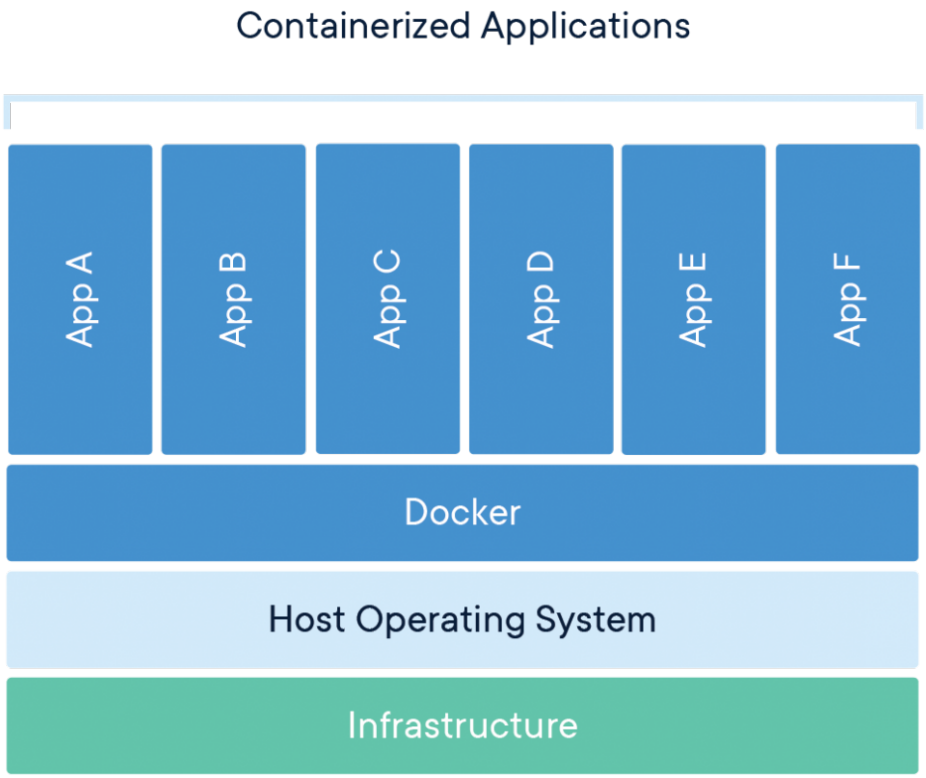

Figure 3.3 - Docker engine layers (image from [49])

3.3.1.1 DOCKER MANAGING ENVIRONMENTS

The Docker API allows a lack of options for interfacing with Docker, the containers, and images from CLIs to desktop applications and web-based management tools. These tools go from Kitematic [50], the default GUI program that ships with Docker to Shipyard [51], a web-based app that provides an interface for containers. Although, the chosen tool was Portainer [52] a UI management software which allows to manage Docker environments and simply deploy any docker container to run on the Docker engine. All Docker resources, including containers, images, volumes and networks can be managed via Portainer [52]. This solution was chosen since it is web-based, which means that can be accessible in any browser without the need to install other third-party software, is free and open-source.

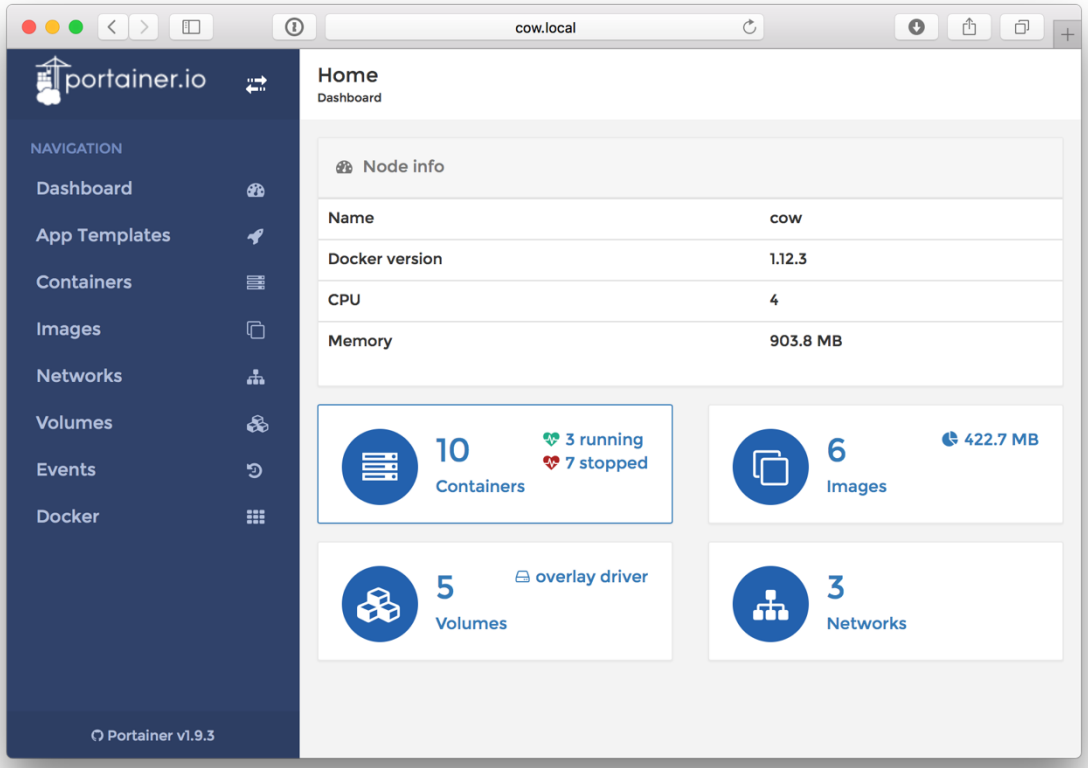

Figure 3.4 - The interface of Portainer, the Docker management tool

3.3.2 DATA REPOSITORY MANAGEMENT SYSTEM

The chosen Data Repository Management System was the eXtensible Neuroimaging Archive Toolkit (XNAT) [45], one of the oldest and more stable open-source data repositories available. It is developed by the Neuroinformatics Research Group with the purpose of addressing and facilitating data management challenges in Neuroimaging studies. It consists of an image repository to store raw and post-processed images, a database to store metadata and non-imaging measures and user interface tools for accessing, searching and exploring the data. User interface tools include a secure web application, command line tools and desktop applications for organizing and uploading local data. It supports full DICOM workflow for all common imaging modalities and an extensible system for modeling and storing clinical, behavioral, and other non-imaging data. XNAT has automated tools to capture data from multiple sources and keeps them in a secure repository and distributes the data to authorized users. XNAT relies heavily on XML and XML Schema [53]. XNAT uses XML Schema Definition (XSD) to define data types and to generate custom components, graphical and local content for its Presentation, Application and Data tiers. Also, XML is used for security, input validation and queries. Therefore, XNAT provides an ad hoc workflow for

Neuroimaging data acquisition and sanitizing, that consists of automated data acquisition directly in the scanner and/or via upload followed by a strict quality control procedure, where we can include data anonymization. However, XNAT also has some problems like inefficient metadata storage. The XNAT data-model can be resumed in three main data types: Experiments, Subjects and Projects. Imaging data from the scanners enter the workflow using mechanisms like the Digital Imaging and Communications in Medicine (DICOM), Secure File Transfer Protocol (SFTP), or portable hard media. Non-imaging data such as clinical assessments, subject demographics and genetic measures are passed via web-based forms, csv uploads or XML (eXtensible Markup Language) [54]. XNAT is used by institutions and data sharing projects around the world including the Biomedical Informatics Research Network (BIRN), the National Alliance for Medical Imaging Computing (NA-MIC) and the Informatics for Integrating Biology & the Bedside (I2B2) [55].

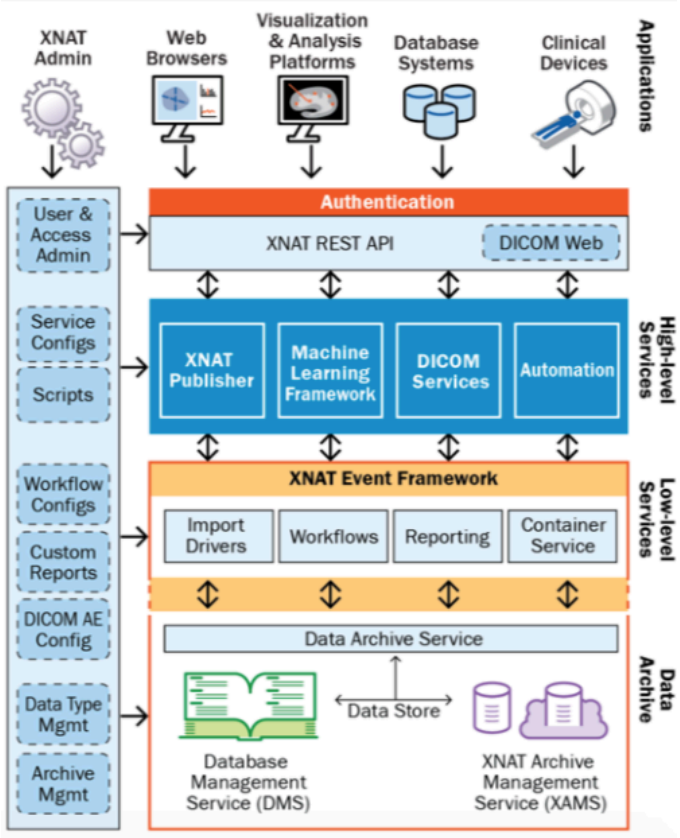

Figure 3.5 - XNAT architecture [56]

A data model is an abstract model that organizes certain elements and standardizes how they can relate to each other. It explicitly determines the structure of data and for that reason it is sometimes referred to as a data structure. XNAT server is a platform built for imaging research purposes, his functions can

archive, import, process and securely distribute the data. The core data-model used in XNAT previews the use of XNAT server in various projects. There are three core data-types in the XNAT data model: Projects, Subjects and Experiments.

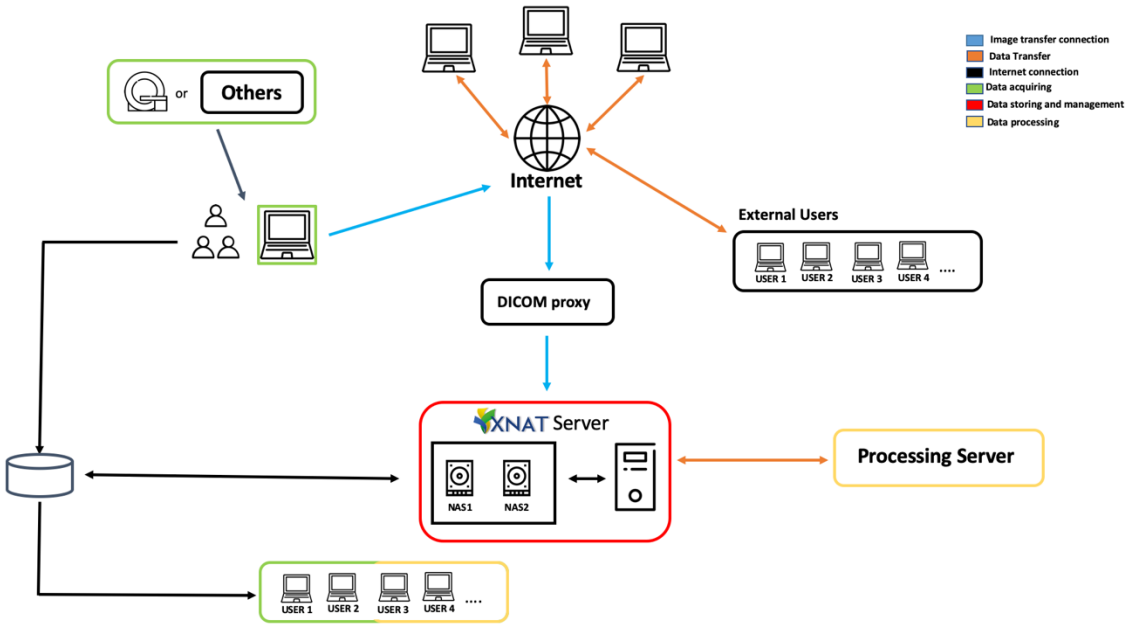

Figure 3.6 – Platform conceptual architecture with XNAT

Projects

A project is used to define a collection of a certain data stored in the server. The project is used to define a secure structure of data. Users can have certain permissions for data within a certain project. They can be Owners, Members or Collaborators. This permission defines who can read, save or delete data. For example, in a specific research study we can give permission for the principal researcher to save, read and delete data but for an internal researcher only to read the data.

Subjects

A subject is anyone who participates in a study. Imaging studies has traditionally focused on human studies, however, the subject could be non-human. It exists in a context of a project and it's "owned" by the project where it was created on. Also, subjects can be shared with other projects to capture longitudinal data from various studies.

Experiments

An experiment is an event by which imaging data or non-imaging data is acquired. It cannot exist outside the context of a project and can be shared into other projects.

Subject Assessments

A subject assessment is a specific kind of experiment which is intrinsically associated with a subject. It contains a foreign key to the subject. Subject Assessments are a common extension point in XNAT. For example, if a user is trying to model new data, it is usually an extension of subject assessments. By default, XNAT provides only one extension of subject assessment which is Imaging Session. It is possible to create new subject assessments and therefore create a model of a variety of subject tests and forms.

Imaging Sessions

Imaging sessions are specific type of Subject Assessment. It is used to capture the data acquired in the normal course of Imaging studies. It adds a collection of image scans, reconstructions and image assessments, to capture the data as scanned, preprocessed and processed.

MR Session / PET Session / CT Session

There are different types of Imaging Sessions. They correlate to the types defined by the DICOM standard. Imaging Sessions introduce a collection of sub-elements which are used to capture imaging data.

Scans

Scans are related with the different Imaging sessions type. They are used to model individual scan series from a single scanning session. A single Session usually contains multiple scans of different types. They also specify the files which make up that scanning series (including RAW data).

Reconstructions

A single imaging session can contain multiple image reconstructions. These are typically the results of a pre-processing stream.

Image Assessments

A single imaging session can contain multiple imaging assessments. They are typically the results of a processing stream and can contain both generated images and statistics. Image Assessments are in general the result of processing results in an XSD like freesurfer or fsl.

3.3.3 DATABASE MANAGEMENT SYSTEM (DBMS)

The chosen Database management system was PostgreSQL [57], an open-source object-relational database management system that extends the SQL language combining features to safely store and scale data. PostgreSQL has earned a strong reputation for its architecture, reliability, data integrity, robust feature set, extensibility and the consistently innovative solutions. Some of the highlights are custom data types, data integrity, scalability, concurrency, performance, security and extensibility. The main characteristics that make PostgreSQL different from other SQL databases are the Multi-Version Concurrency Control (MVCC), the Point in Time Recovery (PITR) and tablespaces replication. The system to be conceived will use PostgreSQL to store user accounts, settings and all project specific information like the subject information. The Medical Imaging files are not stored in the database.

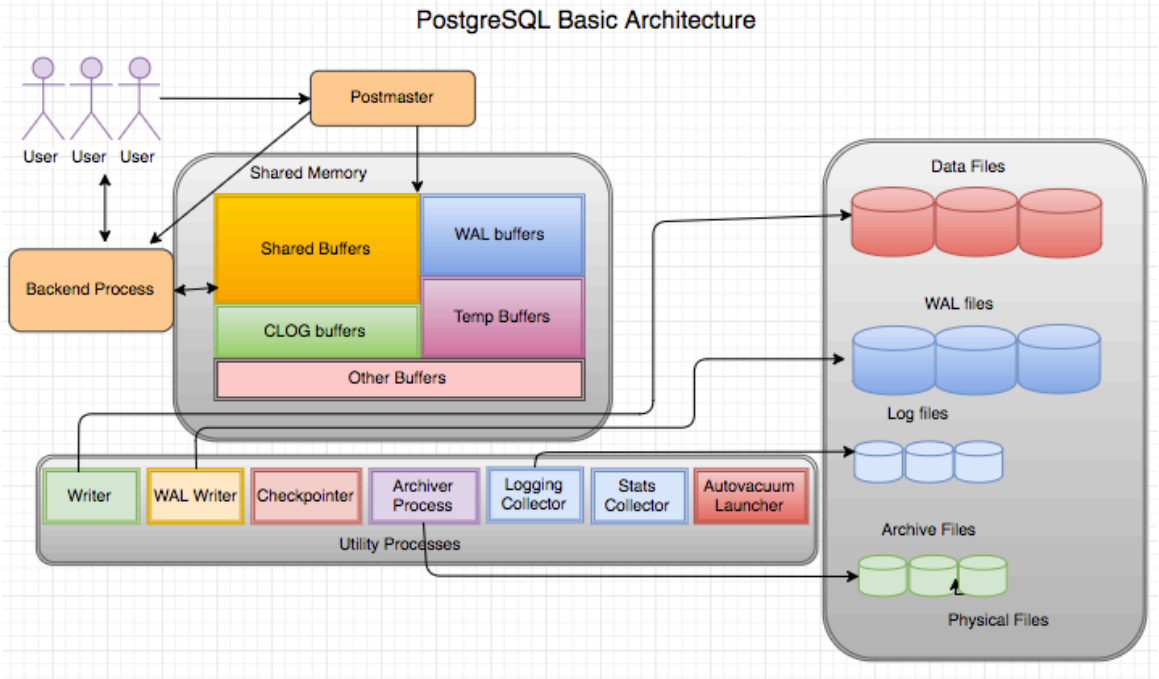

Figure 3.7 - PostgreSQL basic architecture (image from [58])

4 Building the research platform – XNAT@DI

The overall architecture and platform design, from now one called XNAT@DI were presented on the previous chapter. In this chapter the technical implementation details will be described. These details include the server-side implementation using Docker environments, XNAT customization to the platform special needs and the upload of already existing data. The platform is a prototype from what was planned on the previous chapter (Fig 4.1) with special focus on high processing environments, Machine Learning (ML) and Deep Learning (DL). The platform server is placed on the Algoritmi Research Center facilities which will serve not only research studies ongoing there but also studies in ICVS or Braga's main hospital. As the platform introduces a new step on the researchers workflow of Medical Imaging data and associated research materials, the platform tests couldn't be done in a laboratory controlled environment, instead, the platform deployment was done from day one with successive iterations, tests and corrections being done in a step-by-step basis to make sure that all requirements were fulfilled.

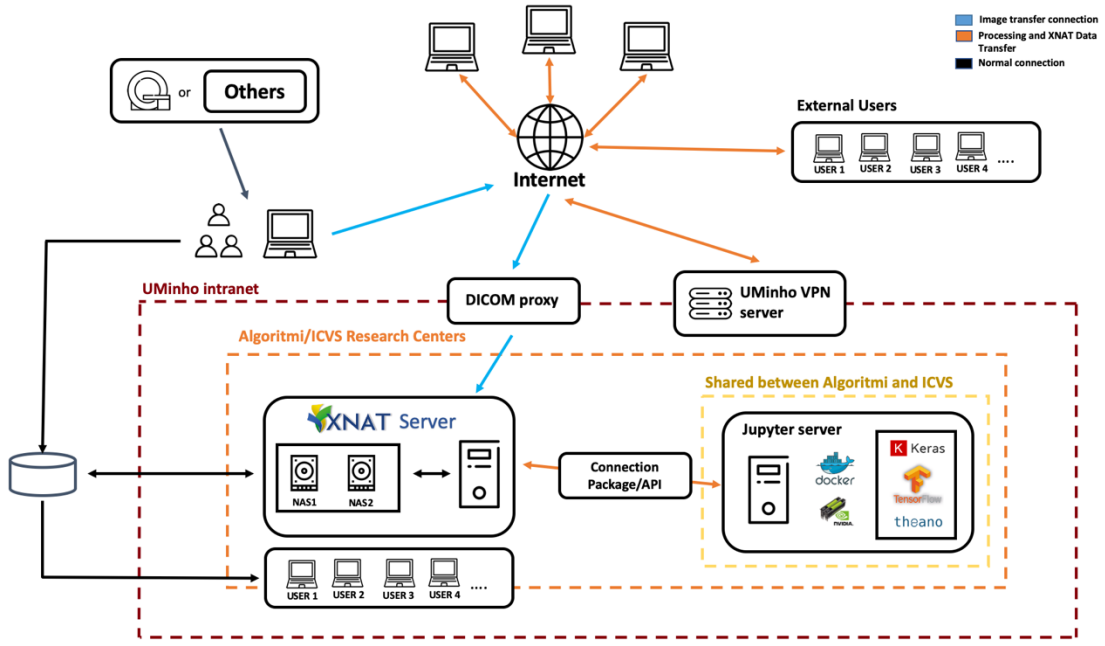

Figure 4.1 – XNAT@DI platform architecture applied at UMinho

The developed platform covers a lot of features and functionalities, that can be easily not understood by unexperienced users. A document was developed as user guide, in appendices to this dissertation, where every most-used functionality is explained with some examples on how to do things so as to quickly and intuitively start using XNAT@DI.

4.1 PLATFORM SERVER

The starting point was a server that was already used in Algoritmi where researchers collaborate through different Docker containers and Jupyter notebooks. Because this server was already in service, we could start right away developing the platform. The research platform is hosted on a dedicated virtual machine running on Ubuntu 16 under Apache Tomcat and uses Oracle Java. The web server is managed through NGINX and the database is managed using PostgreSQL. All of these services have a dedicated virtual machine managed through Portainer [52]. The anonymized research data is hosted on a dedicated XNAT instance running on version 1.7.5. On top of this, some server-side implementations were necessary, and they needed to take in consideration several main points:

Capacity

Data storage can present a limitation in this type of project since Medical Imaging takes a lot of disk space to store. In XNAT the storage is managed through PostgreSQL database on an extensible hard disk. With the current design, it is expected to host more than 10.000 imaging sessions without any disk space problems. The operating limits of the system may increase exponentially based on XNAT usage in a lot of projects, however, this can be easily increased by adding new hard drives to the server.

Scalability

The system allows the addition of unlimited new local gateways with no limits on the number of users that can access them. The total capacity of the system is limited by the server design, which is adequate for our purposes. However, if individual installations require large volumes of data upload or download a redundant server design may be more appropriate.

Performance

The main points related with performance take into account that the two most common actions are data upload and data download. These two actions can have a huge impact on the server since it is necessary a persistent connection. XNAT@DI prioritizes low impact on network over speed and latency. As the server is on *eduroam* (Education Roaming) network data upload is bandwidth limited.

Fault Tolerance

To prevent data loss, files physically stored on the server are regularly backed up to secondary disks. If the server connection is not available or data upload fails, uploading is re-attempted multiple times with an exponentially increasing delay [59].

Taking into account all of these discussed topics and what was discussed previously, the server configuration where the platform is hosted has:

- OS: Ubuntu 14.04 LTS (64 bit) as the operative system
- CPU: Intel Xeon E5-1650
- RAM: 64Gb
- Secondary Memory: 2 Disks of 2TB and 1 disk of 512Gb
- GPU: NVIDIA P6000
    - Cuda Parallel-Processing Colors 3840
    - GPU 24 GB GDDR5X
    - FP32 Performance 12 TFLOPS

4.2 SHARING DATA BETWEEN INSTANCES

Sharing data between instances is critical in this kind of platform. It's very common to have multiple departments to work on multiple projects that can sometimes share data between them. In XNAT there are two main ways to connect two instances: using the XNAT Project Sync plugin or the Joint Anonymization and Archive Transmission (JAAT).

The XNAT Project Sync plugin is more suitable to share data between research institutions. This method establishes a persistent data sync of all or selected data between XNAT instances, which allows the data-

owning researcher to upload the data just once and guarantees that the data is automatically uploaded to the other instances.

The JAAT method was developed to allow the migration of large amounts of DICOM data. This method is highly suitable for large amounts of data. For example, to transfer sessions between the hospital where the MRI machine is and the XNAT server.

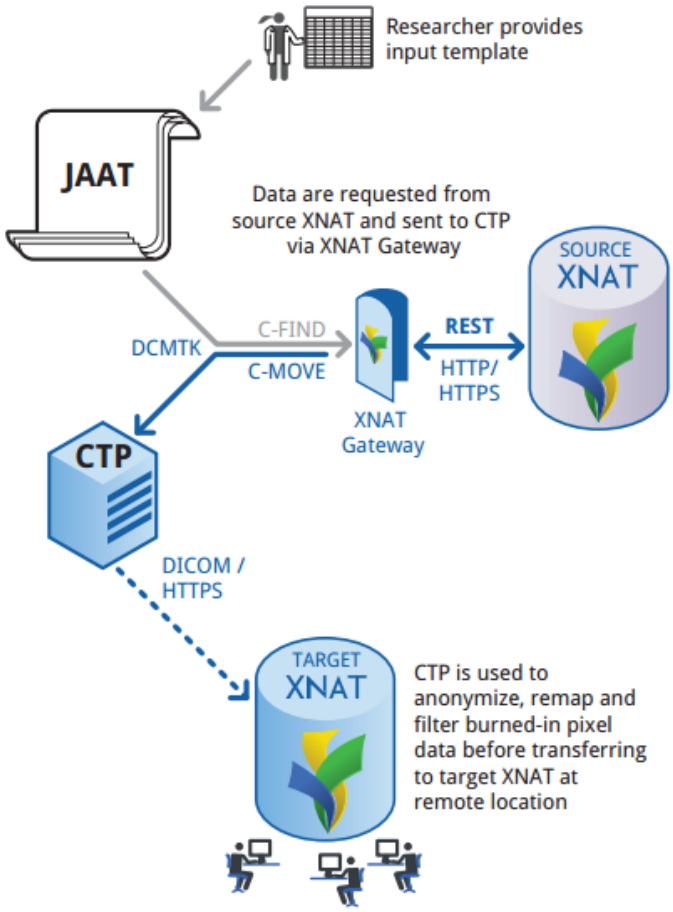

Figure 4.2 - XNAT JAAT method explained

4.3 CONTAINERIZATION AND PLATFORM DEPLOY

As mentioned on the previous chapters all services running on the server are on top of Docker containers. A Docker container is a process/service that runs directly on the server, slightly different than a regular process because the Docker daemon along with the Linux kernel ensures it runs in total isolation, which allows to have multiple services running at the same time on a single computer, without interfering witch

each other [59]. There were already multiple containers deployed on the server, all managed with Portainer [52]. XNAT would be only “one more service running”, although to run it with the needed features (specially redundancy and scalability) some changes had to be made. As discussed on the previous chapters it is necessary to run multiple services at the same time, including Tomcat, Postgres, NGINX and, obliviously, the main XNAT container. To do this, we need a way to manage and deploy at the same time all these containers. This can be made with Docker Compose [60], which allows to manage theservice versions, the connections between containers, and even which ports to expose in each container. Compose is a tool built-in directly on the Docker engine to define and run multiple-container environments in a single network. To use the tool, an YAML [61] file with the containers and services specification was created. The YAML file below bootstraps the deployment of three containers:

```
version: '1'
services:
  xnat-web:
    build:
      context: ./xnat
      args:
        XNAT_VER: '1.7.4.1'
        SMTP_ENABLED: 'false'
        SMTP_HOSTNAME: fake.fake
        SMTP_PORT:
        AMTP_AUTH:
        SMTP_USERNAME:
        SMTP_PASSWORD:
        XNAT_DATASOURCE_DRIVER: 'org.postgresql.Driver'
        XNAT_DATASOURCE_URL: 'jdbc:postgresql://xnat-db/xnat'
        XNAT_DATASOURCE_USERNAME: 'xnat'
        XNAT_DATASOURCE_PASSWORD: 'xnat'
        XNAT_HIBERNATE_DIALECT: 'org.hibernate.dialect.PostgreSQL9Dialect'
        TOMCAT_XNAT_FOLDER: XNAT@DI
        XNAT_ROOT: /data/xnat
        XNAT_HOME: /data/xnat/home
    ports:
      - "8081:8080"
      - "8000:8000"
      - "8104:8104"
    expose:
      - "8080"
      - "8104"
    volumes:
      - ./xnat-data/home/logs:/data/xnat/home/logs
      - ./xnat-data/home/plugins:/data/xnat/home/plugins
      - ./xnat-data/archive:/data/xnat/archive
      - ./xnat-data/cache:/data/xnat/cache
      - ./xnat-data/prearchive:/data/xnat/prearchive
      - ./xnat-data/ftp:/data/xnat/ftp
      - ./xnat-data/build:/data/xnat/build
      - ./xnat-data/pipeline:/data/xnat/pipeline
      - /var/run/docker.sock:/var/run/docker.sock
    depends_on:
      - xnat-db
    environment:
      - CATALINA_OPTS=-Xms128m -Xmx1024m -Dxnat.home=/data/xnat/home -
agentlib:jdwp=transport=dt_socket,server=y,suspend=n,address=8000
      - XNAT_HOME=/data/xnat/home

  xnat-db:
    build: ./postgres
    expose:
      - "5432"
    volumes:
      - ./postgres-data:/var/lib/postgresql/data

  xnat-nginx:
    build: ./nginx
    ports:
      - "80:80"
    expose:
      - "80"
    links:
      - xnat-web
```

Program 4.1 - Docker Compose configuration file

Tomcat + XNAT

This container uses Tomcat and XNAT web application. Tomcat is an open source implementation of the Java Servlet, JavaServer Pages, Java Expression Language and Java WebSocket technologies [62] and is used as proxy to the XNAT container. The deployment of these applications also includes setting system resources and setting the service user and permissions so that the Tomcat process can read and write from the XNAT archive and working folders. As mentioned before, the server has a main disk, which is a Solid-State Disk, and additional Hard Disk Drives (HDD). Taking in consideration that:

1. SSDs are much faster on read and write operations but HDDs are cheaper;
2. The application files are accessed much more than the Medical Imaging files.

All the operative systems (OS) and Docker containers run on top of the SSD, but files are stored in the HDDs disks. This ensures that the platform is fast, and the storage is cheap to maintain. To do this some symbolic links from inside the container to the HDD were created:

```
1.        - ./xnat-data/home/logs:/data/xnat/home/logs
2.        - ./xnat-data/home/plugins:/data/xnat/home/plugins
3.        - ./xnat-data/archive:/data/xnat/archive
4.        - ./xnat-data/cache:/data/xnat/cache
5.        - ./xnat-data/prearchive:/data/xnat/prearchive
6.        - ./xnat-data/ftp:/data/xnat/ftp
7.        - ./xnat-data/build:/data/xnat/build
8.        - ./xnat-data/pipeline:/data/xnat/pipeline
9.        - /var/run/docker.sock:/var/run/docker.sock
```

Program 4.2 - The symbolic links created

This ensures that the XNAT container writes all data on the secondary disks, but all the necessary folders remain in the container. The HDD disks containing the folders are backed up to other HDD disks attached to the server. Also, as discussed on the previous chapters, it was necessary to create a DICOM SCP

```
1.      ports:
2.        - "8081:8080"
3.        - "8000:8000"
4.        - "8104:8104"
5.      expose:
```

Program 4.3 - The exposed ports on the XNAT container

receiver [63]. For this, it was necessary to expose a new port in which the receiver will get all the Medical Imaging data.

PostgreSQL

PostgreSQL [57] is an open-source object-relational database system that extends the SQL language combining features to safely store and scale data. PostgreSQL has earned a strong reputation for its architecture, reliability, data integrity, robust feature set, extensibility and the consistently innovative solutions. Some of the highlights are custom data types, data integrity, scalability, concurrency, performance, security and extensibility. XNAT uses the PostgreSQL database to store all of its persistent data that's not stored in files on the local storage device like subject information or user accounts.

NGINX

NGINX is a web server that uses a non-threaded, event-driven architecture. It also does other important things, like load balancing, HTTP caching or reverse proxying. A NGINX proxy configuration requires a server block which configures connections, and a location block, which points to the specific Tomcat instance wanted for the web to connect to. The following configuration sets up a listener on port 80 (http). This is then associated with the Tomcat running on port 8080.

```
1.      ports:
2.        - "80:80"
3.      expose:
4.        - "80"
```

Program 4.5 - The exposed ports on the Nginx container

Finally, when starting to run the multi-container environment with *docker-compose up*, several directories are created to save the persistent data:

- **postgres-data** - contains the XNAT database;
- **xnat-data/archive** - contains the XNAT archive;
- **xnat-data/build** - contains the XNAT build space;
- **xnat-data/home/logs** - contains the XNAT logs;
- **xnat-data/home/plugins** - Initially contains nothing. Serves to customize XNAT with plugins.

4.4 XNAT CONFIGURATIONS

After building and installing the XNAT multi-container environment some customizations had to be made directly on the administration XNAT web interface. These changes include:

- **Add the DICOM Store Service Provider** – Register a new DICOM Store Service Provider to XNAT configurations on the previous exposed port (8104).
- **Change the Session Idle Check Interval to check every 10 minutes -** This controls how often the system checks to see if there are incoming DICOM sessions in the pre-archive. This value is specified in milliseconds.
- **Enable Site-wide Anonymization script –** An auto anonymization script was added based on what is in use at Washington University School of Medicine and Howard Hughes Medical Institute.

4.5 XNAT PLUGINS AND PIPELINES

XNAT platform has the ability to use third-parties services to add custom functionalities or modules. This is done either through pipelines which can be automatically set to run when, for example, new data is uploaded or through plugins which add extra functionalities (generally user-invoked). Building the plugins and pipelines requires running a *gradlew* [64] script inside the Docker container in which all the compiled classes and static content is built into a single output file located in the *build/libs* folder.

After building the plugin or pipeline, deploying it can be done in three steps:

1. Stop Tomcat container;
2. Copy JAR file and build to plugins folder in XNAT files system;
3. Start Tomcat container again.

Some of the installed plugins and pipelines are:

- **Clinical Data Types** [65] - A collection of additional typical clinical data types;
- **IQ Assessment** [66] – A new IQ datatype;
- **Radiological Assessment** [66] – A new Radiological datatype, which includes edit and report pages;
- **mricron** [67] - DICOM and PARREC flavored pipelines for auto imaging data conversion to NIFTI format;
- **Quality Assessment Protocol** [68] - QA analysis on functional/structural MRI data;
- **mricrogl** [69] - New generation of DICOM-to-NIFTI format conversion pipeline (using dcm2niix);
- **DTI-preprocessing** [70] - Compute preprocessing corrections on MRI DTI scans;
- **MRI bias field correction** [71] - Correcting intensity non-uniformity (i.e. bias fields);
- **MRI anatomical defacer** [72] - Automated facial traits removal (defacing) of anatomical scan data;
- **Image Viewer** [73]– Neuroimage viewing module built with XTK [74] and Google Closure [75] (Fig. 4.3).

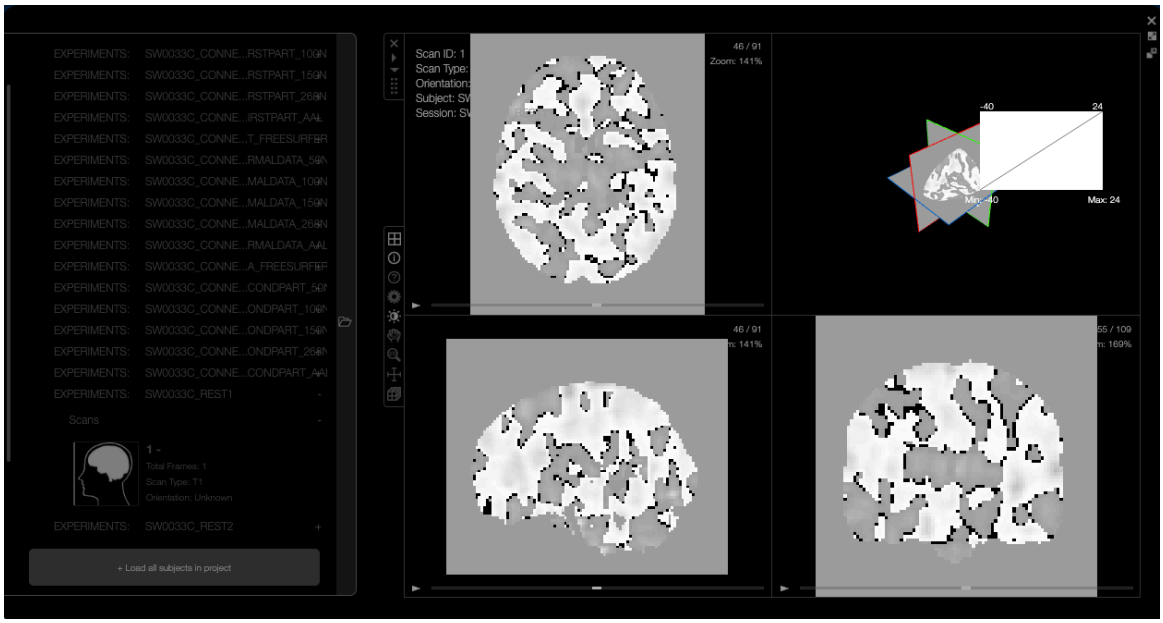

Figure 4.3 - Image Viewer built-in the XNAT@DI platform

4.6 ORGANIZING AND UPLOADING EXISTING DATA

One of the pre-defined requirements was that existent data must be uploaded to the platform. As research and data collecting had already been occurring for many years at ICVS and Algoritmi, there was the need to find a way to correctly make this data available through XNAT@DI.

The way researchers typically share data before the here-described platform was available involved having an CSV file with subject information like age, gender and then organize the scans in folders with the correspondent subject ID. Thus, from this we concluded that to upload this data to XNAT@DI, the workflow would have to have two steps:

1. Upload subject data to the platform from the CSV file;
2. Upload and labeling all the sessions scans and images.

However, some of this CSV files are custom to each project since they have custom variables oftentimes related with the research field of the project and not included on the XNAT generic types. Hence, for each project, a custom variable set with specific data types was created. To include new variables, an XML file has to be modified with the specific project configuration.

```
<FieldMapping data-type="xnat:subjectData" create-
date="Tue Jan 05 03:35:32 PDT 2019" title="BrainConnect" ID="1294343251432">
    <field>xnat:subjectData/ID</field>
    <field>xnat:subjectData/fatherAge</field>
    <field>xnat:subjectData/motherAge</field>
</FieldMapping>
```

Program 4.6 - New fields added to the XML project subjects configuration file

After this, the CSV could be uploaded though the platform built-in uploader. After creating the subjects, all the session scans need to be uploaded to the specific subject. This was done through a Python script that maps all the CSV content and uploads the corresponding images with the correct label that identifies the scan.

```
with open('data.csv', 'r') as _filehandler:
    csv_file_reader = csv.DictReader(_filehandler)
    for row in csv_file_reader:
        subject = row['Codigo_SW']
        print subject
        experiment = subject + "_CORRELATION"
        scan = "1"
        filepath = "/AllData/" + \
           subject + \
            "_fnc_vol_stime_mcf_bet_mni_coreg_denoised_wScrubbing.nii.gz"
        file = project.subject(subject).experiment(experiment).scan(
            scan).resource("CORRELATION").file(os.path.basename(filepath))
        print file
        if not(file.exists()) and os.path.isfile(filepath):
            file.put(filepath, format=" ", content=" ", tags=" ")
            project.subject(subject).experiment(
                experiment).scan(scan).attrs.set("type", "CORRELATION")
```

Program 4.7 - Python script to upload all sessions to XNAT@DI

5 XNATUM – A PYTHON API FOR XNAT

As imaging platforms grow in size and complexity, the time researchers spend managing the data increases. As a result, researchers try to find tools to automatize the process of data management and processing tasks, usually by using high-level programming languages.

As XNAT is highly used on this area [76] a structured and programmatic way to access data is more than necessary. In this chapter is introduced XNATUM, a Python package developed in the context of this work that interacts directly with any XNAT instance providing users direct access to all imaging data stored with an efficient and easy to use API, both as a backend library to XNAT clients or as an alternative frontend from the command line.

5.1 CONTEXT AND MOTIVATION

Researchers are producing and storing Neuroimaging and related data at an increasing rate, XNAT@DI is no exception. The need to manage and share this data at radically different scales (e.g. Human Connectome Project vs. XNAT@DI) led to the development of Neuroimaging management systems [23]. These efforts to deal with this type of data led to the appearance of ontologies for neuroscience [77], the development of databases and data management systems including the Human Imaging Database (HID) [22], the LONI Image Data Archive (LONI IDA) [23] and XNAT. However, these systems don't offer simple ways to programmatically interact with the stored data. For this reason, a lot of tools for managing the Neuroimaging and associated data have been developed to use in addition to these systems. XNAT allows to use the web graphical user interface to manually upload and download data, which is intuitive on a small scale but becomes more and more impractical and error prone on a larger one.

All these reasons explain why it is more than necessary to find a way to directly interact programmatically with the platform, since relying on previously scripted APIs helps maintain consistency of access data and appropriate versioning.

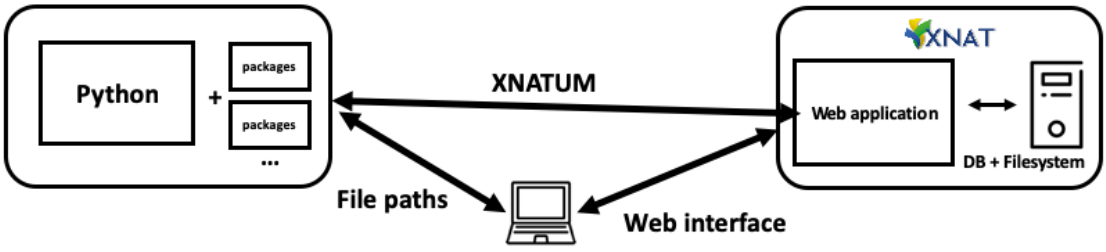

Figure 5.1 - XNATUM interactions with XNAT and the filesystem

5.1.1 PYTHON IN NEUROIMAGING AND NEUROSCIENCE

The chosen language to code the here-described package/library is Python since it not only enjoys growing success in the Neuroimaging and neuroscience communities but also it is commonly used in Machine Learning (ML) which is being more and more applied in this area, including at the Algoritmi and ICVS research centers. In "Python in neuroscience" [78] this growth is justified with the existence of support for object-oriented, functional and procedural programming while having a simple and consistent syntax and very efficient open-source scientific packages such as SciPy [79] and NumPy [80]. Hence, its flexibility and concise syntax speeds, a lot, the process of prototyping new algorithms and scripts.
Python already defines a standard database interface, which is Python DB-API [81], thus XNATUM can act as an interface to the database, wrapping up that directly with the REST API.

5.1.2 XNAT REST API

The Representational State Transfer (REST) was first described in "Architectural Styles and the Design of Network-based Software Architectures" [82] by Roy Thomas Fielding. REST is a generalization of the architectural principals of the World Wide Web and is used to develop web services. The architecture describes a set of resources, which can be entities or collections, with standardized Uniform Resource Identifiers (URIs).
The interaction with the resources relies on common HTTP operations, GET, PUT, POST and DELETE, mapped to specific semantics. This means that resources map to a set of views to represent the data on the server independently from the way it is stored. XNAT uses these principles combined with generic syntax, which consists in a sequence of component parts to describe the communication protocol, the resource location and any additional information (some examples of endpoints can be seen on Table 3.1). RESTful architecture organizes resources in a hierarchy. Basically, URLs paths are constructed using a fixed set of keywords with parent-child relations.

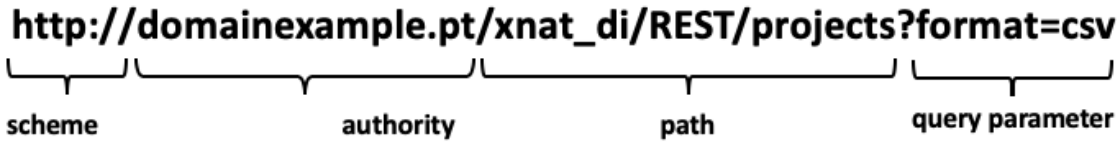

Figure 5.2 - XNAT REST API syntax

In XNAT keywords are paired with an ID to point to a specific resource. Also, the collections of resources returns a list of identifiers. Since HTTP does not include any operations to filter the results, XNAT uses the query parameters component of URLs to perform additional operations in order to enable selecting and filtering the outputs, as well as choosing the output format.
The XNAT REST API is separated in two parts: the hierarchical structure previously described and the search engine. The search engine is responsible for the navigation through the database and the files downloads.

Table 5.1 - Example of REST API endpoints

Description	Method	URL
Upload scan	POST	/data/archive/projects/PROJECT_ID/subjects/SUBJECT_ID/experiments/EXPT_ID/scans/SCAN_ID/files
Upload project	POST	/data/archive/projects/PROJECT_ID/files
Upload subject	POST	/data/archive/projects/PROJECT_ID/subjects/SUBJECT_ID/files
List of projects	GET	/data/archive/projects/

5.1.3 DEVELOPMENT

There are already a countless number of packages [83] to interact with XNAT from Python, however a big part of them are either not currently maintained, have incompatibility with the last versions of Python or are too complicated for unexperienced users. XNATUM is a Python package/library that unifies the access for data providing on XNAT and which pretends to address all of these pain points:

- Simple to maintain in order to receive regular updates;
- Offer usage examples to new users;
- Simple API and methods to interact with.

The development was also done in Python since it has recently gained a strong momentum in the Neuroimaging and neuroscience communities. Combined with powerful scientific libraries, it now provides a credible alternative to other high-level platform independent interpreted language like MATLAB [84]. It offers a very large set of software utilities such as XML parsing, database and web interface modules; as well as conversion modules (e.g. convert DICOM files to Nifti format) which can be used alongside

XNATUM. All the package code is open-source and available online[1] under the MIT license, as well as its documentation and examples. The package can easily be installed using a Python package manager like Pypi[2].

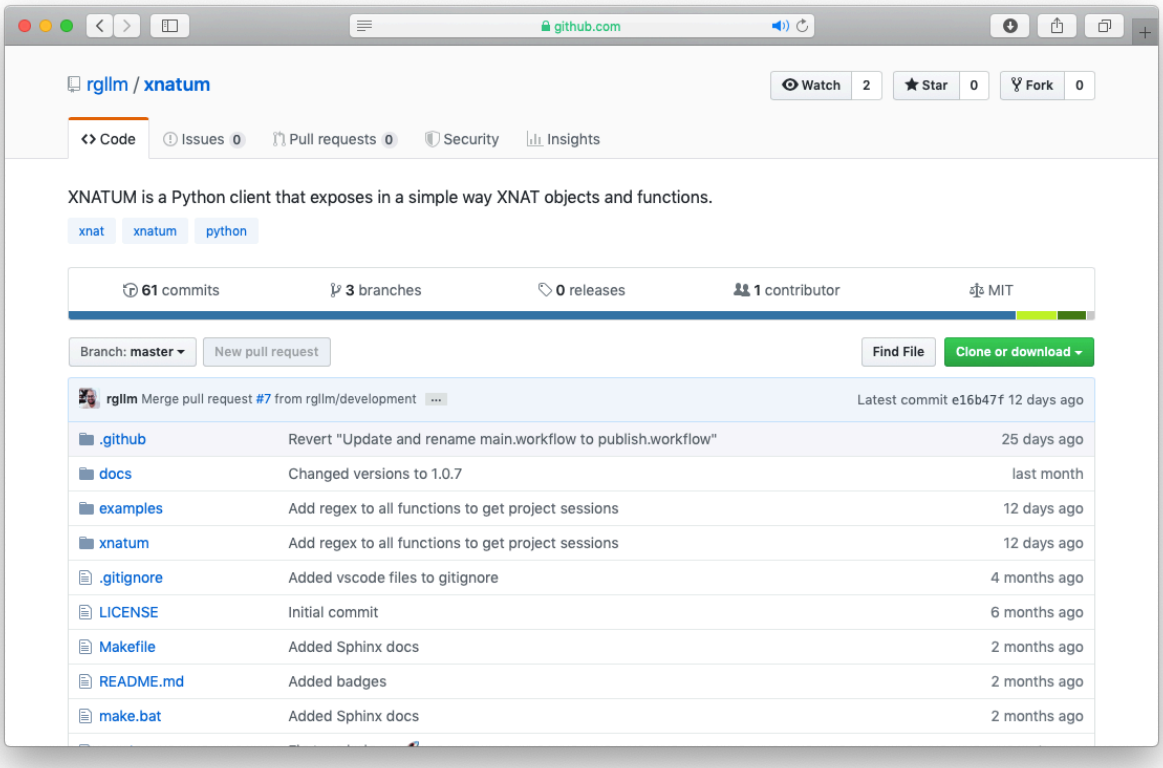

Figure 5.3 - XNATUM GitHub repository

[1] XNATUM GitHub repository - http://github.com/rgllm/xnatum
[2] XNATUM Pypi page - http://pypi.org/project/xnatum/

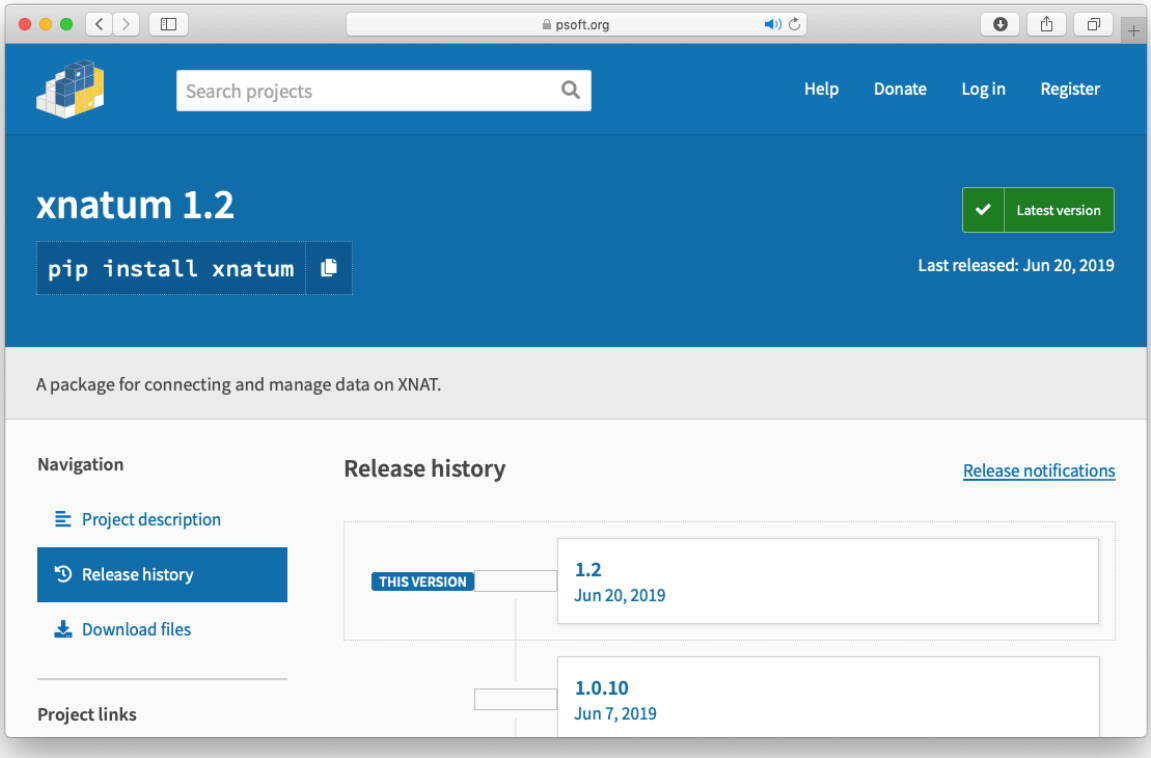

Figure 5.4 - XNATUM Pypi page

5.2 XNATUM IMPLEMENTATION

XNATUM is implemented on top of the XNAT REST API to enable easy communication between the XNAT instance and the Python language. In this subchapter, the general design and implementation of the library are described.

XNATUM combines different components to interact with the XNAT instance. The core relies on the XNATPy [45] package and in the requests Python module, which are in charge of issuing calls to the XNAT server. The structure of the REST calls itself is described in one of the previous subchapters. XNATUM maps the REST API to Python objects and methods, modelling the REST API HTTP responses to parsed Python objects.

The XNAT REST API is composed of a set of resources disposed hierarchically, which corresponds to the REST structure itself. The REST structure gives access to files (e.g. images and information), as well as metadata and it reflects the XNAT data model, which was presented on a previous chapter. The URLs describe the directories paths on a file system and effectively they can be viewed as such.

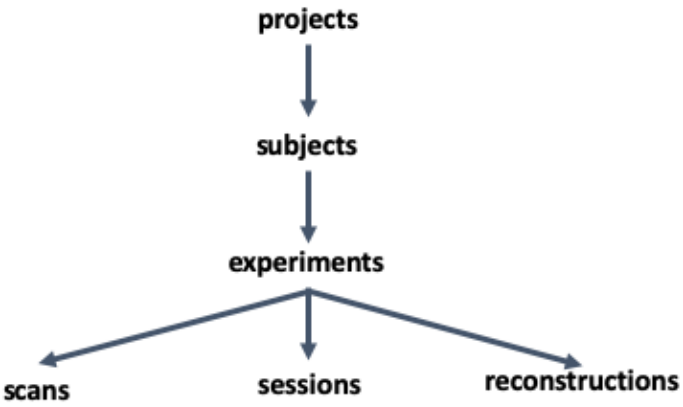

Figure 5.5 - XNAT REST model

For example, with the REST API it is possible to look for all female subjects that are 23 years of age within a project or have a specific answer to an assessment. Using a merge of the XNATPy package, which implies a lot of knowledge in Python and APIs to use, with the Python request module, XNATUM is able to retrieve all the assessments from a subject in a single statement whereas the REST API from XNAT would require multiple HTTP calls.

XNATUM uses Python classes to create an initial server configuration, which persists to be used in multiple server operations. The connection holds the login information, the server information and a session. It will also send a ping every 14 minutes to keep the connection alive, even if no operation is being executed.

```
1.  import xnatum
2.
3.  session = xnat.connect('http://mivbox.di.uminho.pt/XNAT@DI/', user='xnatum', pass
    word='secret')
```

Program 5.1 - Example of creating a connection to a XNAT instance

XNATUM also allows self-closing sessions that can also be used using the Python context operator, to remove the possibility of unforeseen errors. As soon as the session doesn't exist anymore, for example because an error/exception was thrown away, the session is automatically disconnected.

```
1.  import xnatum
2.
3.  with xnat.connect('http://my.xnat.server') as session:
4.      print(session.projects)
```

Program 5.2 - Creating a connection to a XNAT instance using the Python context operator

When a session is established and maintained within the scope, it is fairly easy to explore the data on the XNAT server. The data structure of XNAT is mimicked as Python objects. The connection gives access to a listing of all projects, subjects, and experiments on the server. All levels on the XNAT server: projects, subjects, experiments, scans, resources, files can be browsed.

Although, there are situations on which there is the need for looping over data. To do this, XNATListing objects can be used as Python dictionaries.

```
1.  project = self.session.projects[lproject]
2.
3.  for subject in project.subjects.values():
4.      for experiment in subject.experiments.values():
5.          allexperiments.append(experiment)
6.  return allexperiments
```

Program 5.3 - Getting all experiments of a XNAT project

If there is the need to download data, this can be done using a helper function to either download it to a target directory or to the current directory. This will create a data structure similar to that of XNAT on the local disk.

To add new data into the XNAT server, it is possible using the XNAT REST import service with an HTTP POST request. It allows the upload of a zip file containing new experiments and XNAT will automatically store it in the correct place. Although, it is dangerous to add data straight into the archive due to the lack

```
1.  try:
2.      self.__prearc_session = self.session.services.import_(
3.          zipfname,
4.          overwrite="append",
5.          project=project,
6.          subject=subject,
7.          experiment=subject + "_" + session,
8.          destination=destination,
9.          trigger_pipelines=False,
10.     )
11. except:
12.     print("Unexpected error during XNAT import:")
13.     print(sys.exc_info())
```

Program 5.4 - XNATUM session upload function

of reviewing. So, the suggested workflow for the XNAT@DI platform is to use the destination argument as 'prearchive' in the function to send sessions to the server. This will make the sessions being uploaded to the project, but also makes the pre-archive be reviewed by the primary researcher before being available to research purposes.

5.3 USE-CASE EXAMPLES

XNATUM is a powerful yet easy to use Python package. As the initial goal was to create a new package in which researchers could rely one, even those not familiar with Python programming. For this reason, some working examples were created to serve as the entry point to use the package.

Download project sessions

This example uses the *download_project_sessions* function to download all images within a project to the directory where the Python script is being executed.

```
1.  xnat = Xnat(args.server, args.username, args.password)
2.  xnat_sessions = xnat.download_project_sessions(args.project)
```

Program 5.5 - Download project sessions through XNATUM

Download project sessions to directory

This example uses the *download_project_sessions_to_directory* function to download all images within a project to the dirpath argument folder. If the folder does not exist XNATUM creates the folder.

```
1.  xnat = Xnat(args.server, args.username, args.password)
2.  xnat_sessions = xnat.download_project_sessions_to_directory(args.project, dirpath
    )
```

Program 5.6 - Download project sessions to directory through XNATUM

Download subject sessions

This example uses the *download_subject_sessions* function to download all images within a project from a specific subject to the directory where the Python script is being executed.

```
1.  xnat = Xnat(args.server, args.username, args.password)
2.  xnat_sessions = xnat.download_subject_sessions(args.project, args.subject)
```

Program 5.7 - Download subject sessions through XNATUM

Download subject sessions to directory

This example uses the *download_subject_sessions_to_directory* function to download all images within a project from a specific subject to the dirpath argument folder. If the folder does not exist XNATUM creates the folder.

```
1.  xnat = Xnat(args.server, args.username, args.password)
2.  xnat_sessions = xnat.download_subject_sessions_to_directory(args.project, args.su
    bject, dirpath)
```

Program 5.8 - Download subject sessions to directory through XNATUM

Download specific subject session to directory

This example uses the *download_single_subject_session_to_directory* function to download all images within a project to the dirpath argument folder. If the folder does not exist XNATUM creates the folder.

```
1.  xnat = Xnat(args.server, args.username, args.password)
2.  xnat_sessions = xnat.download_single_subject_session_to_directory(args.project, a
    rgs.subject, args.session', dirpath)
```

Program 5.9 - Download specific subject session to directory through XNATUM

Get list of subjects

This example uses the *get_list_subjects* function to list of subject that exist on a project.

```
1.  xnat = Xnat(args.server, args.username, args.password)
2.  print(xnat.get_list_subjects(args.project))
```

Program 5.10 - List all project subjects through XNATUM

Get project sessions

This example uses the *get_project_sessions* function to list all sessions labels within a project, identifying the correspondent subject.

```
1.  xnat = Xnat(args.server, args.username, args.password)
2.  xnat_sessions = xnat.get_project_sessions(args.project)
```

Get subject info

Program 5.11 - Get project sessions labels through XNATUM

```
1.  xnat = Xnat(args.server, args.username, args.password)
2.  print(xnat.get_subject_info(args.project, args.subject))
```

Program 5.12 - Get all information from a subject within a project through XNATUM

This example uses the *get_subject_info* function to get all subject metadata within a project.

All the examples described here, and more are available in the XNATUM GitHub repository[3] .

Upload session

This example uses the *import_resource* function to upload a session to a project from a subject. It starts to create the session item structure, sorts it by the format and then upload de resources files to the XNAT server.

```
1.  # Creating connection and load experiment object
2.  xnat = Xnat(args.server, args.username, args.password)
3.  item = xnat.session.projects[args.project].subjects[args.subject].experiments[0]
4.
5.  # Sending file
6.  files = sorted( glob.glob( '{}/sessions/{}_*.*'.format( args.input, args.subject
    ) ) )
7.  xnat.import_resource(item, 'FILE', files)
8.  print('File imported.')
```

Program 5.13 - Upload a sessiont a subject through XNATUM

[3] XNATUM examples folder on the GitHub repository - https://github.com/rgllm/xnatum/tree/master/examples

5.4 USING A VIRTUAL RAM DRIVE

Talking specifically about the workflow at Algoritmi and ICVS, XNAT@DI platform has a unique workflow and setup. Researchers work and develop scripts in Jupyter notebooks hosted on the same server in which XNAT is running, the only difference is that they are running on different containers.
This allows true collaboration between all researchers and the XNAT@DI. The goal here is to allow to not replicate Medical Imaging data across different Jupyter notebooks and containers but to host medical images in XNAT and use those images directly in all Jupyter notebooks.
The first attempt to do this relied on the fact that XNAT REST API exposes an API which includes the entire DICOM file, the Medical Imaging raw format, with various DICOM data elements.

A DICOM data element, or attribute, is composed of the following parts [63] :

- a tag that identifies the attribute, usually in the format (XXXX, XXXX) with hexadecimal numbers, and may be divided further into DICOM Group Number and DICOM Element Number;
- a DICOM Value Representation (VR) that describes the data type and format of the attribute value.

For example, the tag (0018,8151) corresponds to the X-Ray Tube Current in μA and the tag (0010,0010) corresponds to the Patient's Name. There is also a tag for the specific scan image. The Nifty file format, a file format used after the medical image is processed (can be compared to JPEG) and works in the same way as DICOM; with different tags and attributes as all images are anonymized and combined by default. Taking this into account, the plan was to read and use all the images from the DICOM and Nifty files by selecting the correspondent tag field from the XNAT REST API. As explained before, the API creates a JSON endpoint for each session inside a "project and subject" file and these endpoints contain all DICOM tags and attributes. However, XNAT truncates the attributes from the DICOM dump available on the API to under 64 characters. So, it's only possible to read the first 64 characters of the "Pixel Data" attribute and not the complete images.
Thus, the next approach was to use RAM Drive which is a block of random-access memory that software can use as if the memory were a disk drive. Python has an API to generate temporary files and directories that uses this OS built-in functionality. The module creates temporary files and directories and provides automatic cleanup which can be used as context managers. However, as this is a specific approach for our platform and XNATUM is a global package that can be used in any XNAT installation, this is not

included in the core functionality. Therefore, this is added as a working example on the XNATUM repository making use of the functionality previously described to download images by indicating the destination folder. The example uses the "tempfile.mkdtemp" [85] Python functionality that creates temporary directory in the most secure manner possible with no race conditions in the directory creation. The directory is readable, writable, searchable only by the creating user ID and is available as long as the user wants it to be during the program execution. This Python functionality, when executed, returns the temporary directory path and therefore the unique thing to do on XNATUM is to indicate this path to the correspondent function so as to download the sessions by project, subject or session label. When the program is executed, all the sessions are downloaded directly from XNATUM to the temporary directory and can be used after that in the research script.

5.5 DISCUSSION

Historically, Neuroimaging researchers have used *ad hoc* procedures for maintaining their analysis and data. Over the last few years, there have been several attempts to build systems that manage Neuroimaging data. Moreover, the ability to programmatically access Neuroimaging data is becoming increasingly important to perform batch analysis and administration tasks, as these databases increase in size.

Apart from this, one of the most widely used Neuroimaging management systems is XNAT, which includes many useful and powerful features including, as discussed before, a REST API. XNATUM provides a bridge between the XNAT instance used by the researcher to archive their data and the analysis tool. Combined with Python, it can be used as an alternative both to the XNAT web frontend and to a vast set of Python tools available in the neuroscience domain. The package focuses on ease of use, combining RESTful services and XNATPy with clear semantics and helper features.

Since one of the main focus of the neuroscience research at UMinho is using Machine Learning (ML) and Deep Learning (DL) in the neuroscience/medicine imaging research field, XNATUM combines a series of features to help researchers focuses in the research, abstracting from the data to use. For example, there is a feature on XNATUM to simply obtain and download all sessions within a XNAT project and automatically label each session as train data or test data.

The version 1.0 of XNATUM for Python is available publicly since March 28 of 2019 and it is currently mainly used by researchers that use the XNAT@DI platform. As any software in their early steps, there are a couple of things that can be improved and added to the package. These improvements and new

features are discussed in the next chapter. The primary goal in creating this package was for researchers to have a simple way to manage data in the XNAT@DI platform, and that was accomplished since there are already researchers testing it and/or using it in their experiments. The images bellow list some of the stats from the Pypi repository.

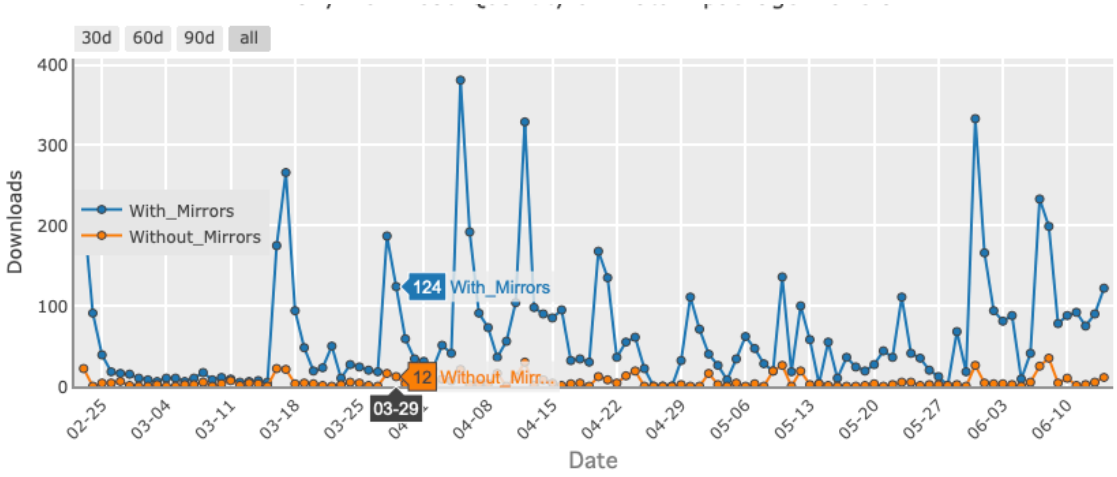

Figure 5.6 - Daily downloads since the package is available at Pypi

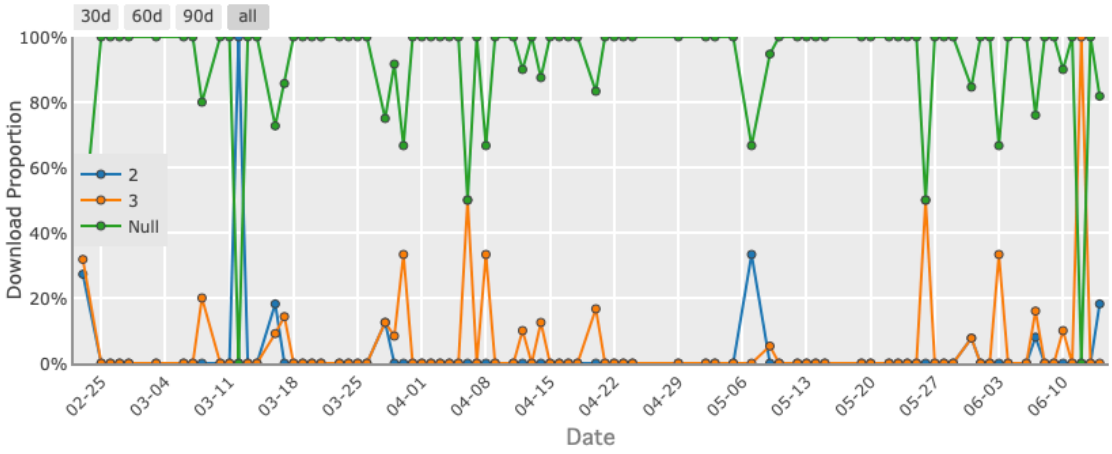

Figure 5.7 - Daily downloads proportion between different versions of Python

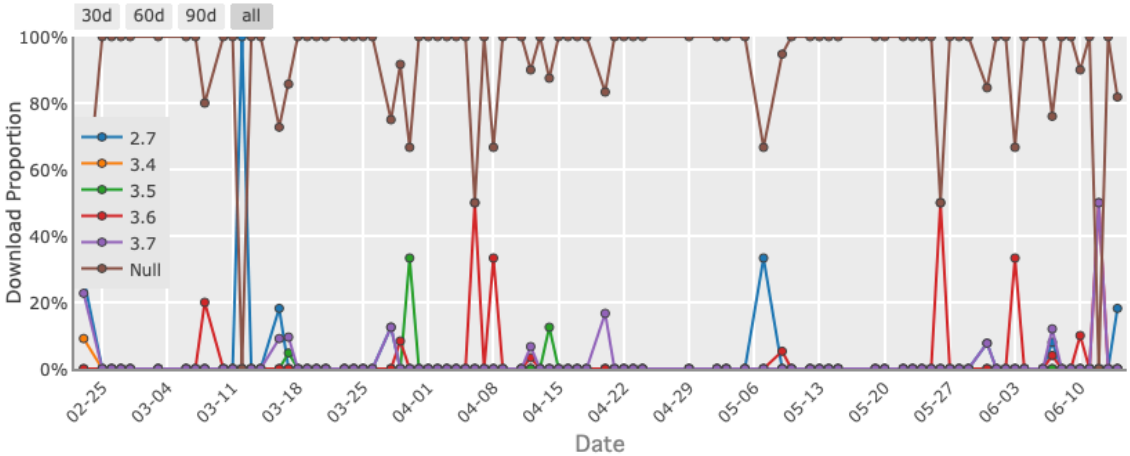

Figure 5.8 - Daily downloads between different versions of Python

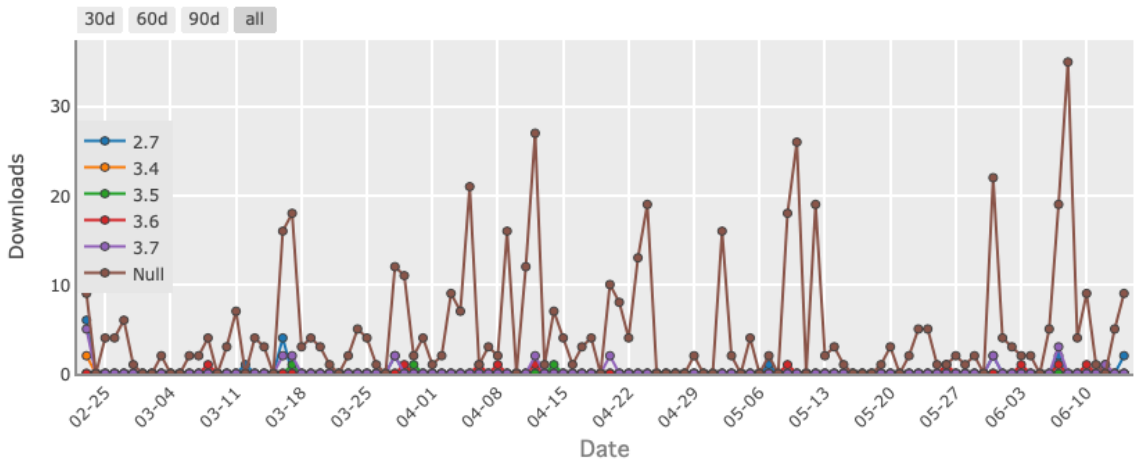

Figure 5.9 - Daily downloads proportion between different versions of Python

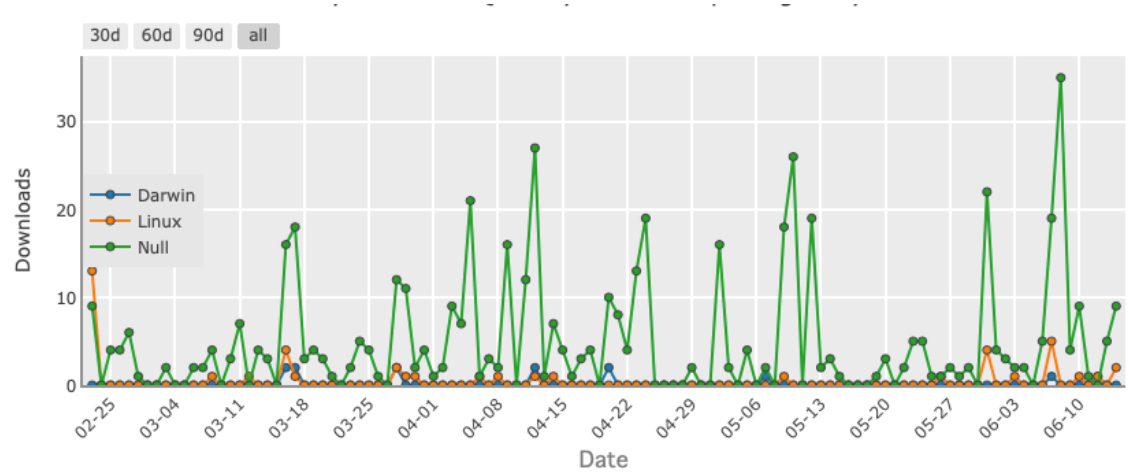

Figure 5.10 - Daily downloads between different system

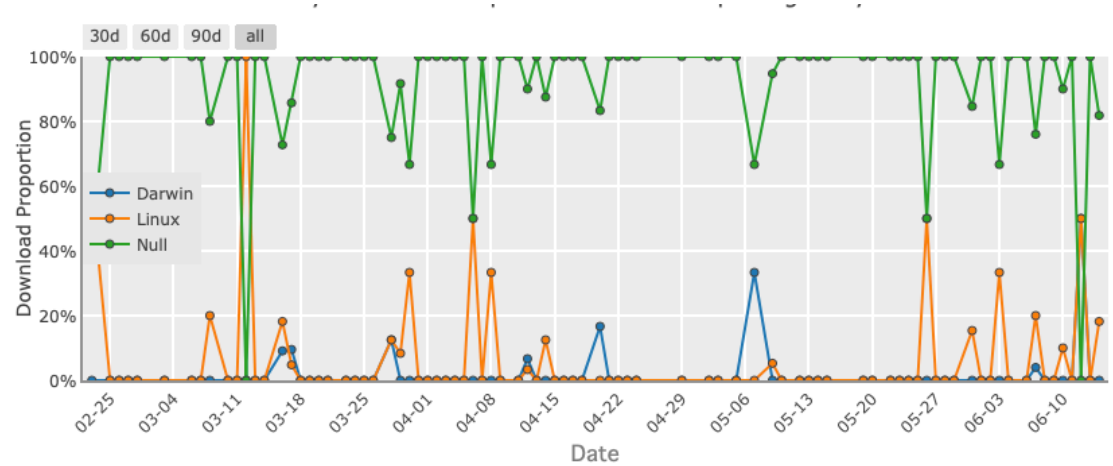

Figure 5.11 - Daily downloads proportion between different system

6 CONCLUSIONS

This work started from a simple and explicit goal of facilitating the sharing of Medical Imaging studies, with the hope of further advancing the progress being currently made in this area at Algoritmi and ICVS. However, several challenges still need to be overcome before the XNAT@DI reaches its goals. Nevertheless, once these challenges are surpassed there is enormous potential for this XNAT@DI platform, for example, as an official record for the scientific work in the area of Medical Imaging research at UMinho. More than that, since a record of all data acquired will be publicly available to the UMinho scientific community, diverse researchers without connection between them can benefit from each other's data, which is something that was not possible before and implied new Medical Imaging acquisitions. Plus, this will facilitate the greater involvement of scientists from a diverse range of backgrounds, bringing to this research area their expertise.

As in any software platform there are a couple of things that can be improved and added to XNAT@DI, one of the most important being the direct connection between the XNAT@DI platform with the city main hospital for direct access to the MRI sessions captured there. This will allow having more recent data accessible to researchers. There is also a project to add a new XNAT instance at the UMinho ICSV and connect both XNAT instances to share data between them. Furthermore, there is also a need for adding new pipelines to XNAT@DI, for example, for normalizing the size of the brain imaging sessions uploaded to the platform before they become available to researchers, hence guaranteeing the research is done in brain imaging sessions with the exact same size.

The other part of this work involved the creation of the XNATUM, a Python package that helps download and upload data to any XNAT instance with a simple and intuitive API. The project is open-source and available at GitHub, accepting contributions from any person who wants to collaborate on the project. The package can be installed from the Pypi repository and counts with more than 200 downloads. However, there are a couple of things that can be improved in this topic, such as the package documentation. Currently, all functions available are documented but are not available without looking at the code. This can be improved with a central website with both documentations and examples. This in conjunction with adding a set of unit and integration tests to facilitate new contributions to the package, while making sure new code does not break any existing features, will make a really big difference now that the package is used by a higher number of users and accepts external contributions.

6.1 WORK CONTRIBUTIONS

- XNAT@DI platform;
- XNAT@DI promotion and researchers training/workshop;
- XNATUM Python package;
- XNATUM demo examples.

References

[1] R. Haux, "Medical informatics: Past, present, future," *Int. J. Med. Inform.*, vol. 79, no. 9, pp. 599–610, 2010.

[2] H. Muller, X. Gao, and S. Luo, "Medical imaging and medical informatics," *Comput. Methods Programs Biomed.*, vol. 92, no. 3, pp. 225–304, Dec. 2008.

[3] F. Sullivan, "What is health informatics?," *J. Heal. Serv. Res. Policy*, vol. 6, no. 4, pp. 251–254, 2001.

[4] H. P. Meinzer, M. Thorn, M. Vetter, P. Hassenpflug, M. Hastenteufel, and I. Wolf, "Medical imaging: Examples of clinical applications," *ISPRS J. Photogramm. Remote Sens.*, vol. 56, no. 5–6, pp. 311–325, Aug. 2002.

[5] T. D. Wager and E. E. Smith, "Neuroimaging studies of working memory: A meta-analysis," *Cogn. Affect. Behav. Neurosci.*, vol. 3, no. 4, pp. 255–274, Dec. 2003.

[6] T. E. Conturo, N. F. Lori, T. S. Cull, E. Akbudak, A. Z. Snyder, J. S. Shimony, R. C. McKinstry, H. Burton, and M. E. Raichle, "Tracking neuronal fiber pathways in the living human brain," *Proc. Natl. Acad. Sci. U. S. A.*, vol. 96, no. 18, pp. 10422–10427, Aug. 1999.

[7] K. Kamnitsas, C. Ledig, V. F. J. Newcombe, J. P. Simpson, A. D. Kane, D. K. Menon, D. Rueckert, and B. Glocker, "Efficient multi-scale 3D CNN with fully connected CRF for accurate brain lesion segmentation," *Med. Image Anal.*, vol. 36, pp. 61–78, Feb. 2017.

[8] J. G. Lee, S. Jun, Y. W. Cho, H. Lee, G. B. Kim, J. B. Seo, and N. Kim, "Deep learning in medical imaging: General overview," *Korean J. Radiol.*, vol. 18, no. 4, pp. 570–584, 2017.

[9] H. Greenspan, B. Van Ginneken, and R. M. Summers, "Guest Editorial Deep Learning in Medical Imaging: Overview and Future Promise of an Exciting New Technique," *IEEE Transactions on Medical Imaging*, vol. 35, no. 5. pp. 1153–1159, May-2016.

[10] M. I. Jordan and T. M. Mitchell, "Machine learning: Trends, perspectives, and prospects," *Science (80-.).*, vol. 349, no. 6245, pp. 255–60, Jul. 2015.

[11] A. Hamidinekoo, E. Denton, A. Rampun, K. Honnor, and R. Zwiggelaar, "Deep learning in mammography and breast histology, an overview and future trends," *Med. Image Anal.*, vol. 47, pp. 45–67, Jul. 2018.

[12] J.-D. Haynes and G. Rees, "Decoding mental states from brain activity in humans," *Nat. Rev. Neurosci.*, vol. 7, no. 7, pp. 523–534, Jul. 2006.

[13] F. Pereira, T. Mitchell, and M. Botvinick, "Machine learning classifiers and fMRI: a tutorial overview.," *Neuroimage*, vol. 45, no. 1 Suppl, pp. S199–S209, Mar. 2009.

[14] R. Magalhães, P. Marques, T. Veloso, J. M. Soares, N. Sousa, and V. Alves, "Construction of functional brain connectivity networks," in *Advances in Intelligent Systems and Computing*, 2015, vol. 373, pp. 303–310.

[15] "Redmine." [Online]. Available: http://www.redmine.org/. [Accessed: 09-Jul-2019].

[16] J. D. Van Horn, J. S. Grethe, P. Kostelec, J. B. Woodward, J. A. Aslam, D. Rus, D. Rockmore, and M. S. Gazzaniga, "The Functional Magnetic Resonance Imaging Data Center (fMRIDC): The challenges and rewards of large-scale databasing of neuroimaging studies," *Philos. Trans. R. Soc. B Biol. Sci.*, vol. 356, no. 1412, pp. 1323–1339, 2001.

[17] F. Nat, "The Cancer Imaging Archive (TCIA) - A growing archive of medical images of cancer," *The Cancer Imaging Archive (TCIA)*, 2017. [Online]. Available: https://www.cancerimagingarchive.net/. [Accessed: 01-Jun-2019].

[18] D. D. Feng, *Biomedical Information Technology*. ELSEVIER ACADEMIC PRESS, 2008.

[19] R. Miotto, F. Wang, S. Wang, X. Jiang, and J. T. Dudley, "Deep learning for healthcare: Review, opportunities and challenges," *Brief. Bioinform.*, vol. 19, no. 6, pp. 1236–1246, 2017.

[20] S. D. Larson, S. M. Maynard, F. Imam, and M. E. Martone, "NeuroLex.org-A semantic wiki for neuroinformatics based on the NIF standard ontology," in *CEUR Workshop Proceedings*, 2009, vol. 559.

[21] S. Gadde, N. Aucoin, J. S. Grethe, D. B. Keator, D. S. Marcus, and S. Pieper, "XCEDE: An extensible schema for biomedical data," *Neuroinformatics*, vol. 10, no. 1, pp. 19–32, Jan. 2012.

[22] D. B. Keator, J. S. Grethe, D. Marcus, B. Ozyurt, S. Gadde, S. Murphy, S. Pieper, P. D. Greve, R. Notestine, H. J. Bockholt, and P. Papadopoulos, "A national human neuroimaging collaboratory enabled by the biomedical informatics research network (BIRN)," *IEEE Trans. Inf. Technol. Biomed.*, vol. 12, no. 2, pp. 162–172, Mar. 2008.

[23] J. D. Van Horn and A. W. Toga, "Is it time to re-prioritize neuroimaging databases and digital repositories?," *Neuroimage*, vol. 47, no. 4, pp. 1720–1734, Oct. 2009.

[24] D. S. Marcus, T. R. Olsen, M. Ramaratnam, and R. L. Buckner, "The extensible neuroimaging archive toolkit: An informatics platform for managing, exploring, and sharing neuroimaging data," *Neuroinformatics*, vol. 5, no. 1, pp. 11–33, Mar. 2007.

[25] J.-B. Poline, J. L. Breeze, S. Ghosh, K. Gorgolewski, Y. O. Halchenko, M. Hanke, C. Haselgrove, K. G. Helmer, D. B. Keator, D. S. Marcus, R. A. Poldrack, Y. Schwartz, J. Ashburner, and D. N. Kennedy, "Data sharing in neuroimaging research," *Front. Neuroinform.*, vol. 6, no. April, pp. 1–13, 2012.

[26] W. J. Bug, G. A. Ascoli, J. S. Grethe, A. Gupta, C. Fennema-Notestine, A. R. Laird, S. D. Larson, D. Rubin, G. M. Shepherd, J. A. Turner, and M. E. Martone, "The NIFSTD and BIRNLex vocabularies: Building comprehensive ontologies for neuroscience," *Neuroinformatics*, vol. 6, no. 3, pp. 175–194, Sep. 2008.

[27] W. Raghupathi Viju Raghupathi, *An Overview of Health Analytics*, vol. 04. 2013.

[28] W. Raghupathi and V. Raghupathi, "Big data analytics in healthcare: promise and potential," *Heal. Inf. Sci. Syst.*, vol. 2, no. 1, pp. 1–10, 2014.

[29] S. T. C. Wong, D. Tjandra, H. Wang, and W. Shen, "Workflow-enabled distributed component-based information architecture for digital medical imaging enterprises," *IEEE Trans. Inf. Technol. Biomed.*, vol. 7, no. 3, pp. 171–183, 2003.

[30] E. Society, "The new EU General Data Protection Regulation: what the radiologist should know," *Insights Imaging*, vol. 8, no. 3, pp. 295–299, 2017.

[31] R. A. Poldrack, "NIH Public Access The future of fMRI in Cognitive Neuroscience," vol. 62, no. 2, pp. 1216–1220, 2014.

[32] A. Moser, K. Range, and D. M. York, "NIH Public Access," *Bone*, vol. 23, no. 1, pp. 1–7, 2008.

[33] National Institutes of Health (NIH), "NIH Launches the Human Connectome Project to Unravel the Brain's Connections," 2009. [Online]. Available: https://www.nih.gov/news-events/news-releases/nih-launches-human-connectome-project-unravel-brains-connections. [Accessed: 01-Jan-2019].

[34] L. Geddes, "Human brain mapped in unprecedented detail," *Nature*, Jul. 2016.

[35] D. S. Marcus, K. A. Archie, T. R. Olsen, and M. Ramaratnam, "The open-source neuroimaging research enterprise," *J. Digit. Imaging*, vol. 20, no. SUPPL. 1, pp. 130–138, 2007.

[36] J. Gurneya, T. Olsenb, J. Flavina, M. Ramaratnamc, K. Archied, J. Ransforde, R. Herricka, L. Wallacea, J. Clinea, W. Hortona, and D. S. Marcusa, "The Washington University Central Neuroimaging Data Archive," *HHS Public Access*, vol. 91, no. 2, pp. 165–171, 2015.

[37] K. Alpert, A. Kogan, T. Parrish, D. Marcus, and L. Wang, "The Northwestern University Neuroimaging Data Archive (NUNDA)," *Neuroimage*, vol. 124, pp. 1131–1136, 2016.

[38] T. Doel, D. I. Shakir, R. Pratt, M. Aertsen, J. Moggridge, E. Bellon, A. L. David, J. Deprest, T. Vercauteren, and S. Ourselin, "GIFT-Cloud: A data sharing and collaboration platform for medical imaging research.," *Comput. Methods Programs Biomed.*, vol. 139, pp. 181–190, Feb. 2017.

[39] D. T. Fetzer and O. C. West, "The HIPAA Privacy Rule and Protected Health Information. Implications in Research Involving DICOM Image Databases," *Acad. Radiol.*, vol. 15, no. 3, pp.

390–395, Mar. 2008.

[40] Horos, “Horos Project 2015,” 2015. [Online]. Available: https://horosproject.org/. [Accessed: 29-May-2019].

[41] “OsiriX DICOM Viewer | The world famous medical imaging viewer,” *OsiriX*, 2019. [Online]. Available: https://www.osirix-viewer.com/. [Accessed: 31-May-2019].

[42] “Medical imaging software: Platform for modern radiology.”

[43] “Best Medical Imaging Software | 2019 Reviews of the Most Popular Tools & Systems.” [Online]. Available: https://www.capterra.com/medical-imaging-software/. [Accessed: 13-Jul-2019].

[44] “Nora Medical Imaging Platform.” 2017.

[45] “XNAT.” [Online]. Available: https://www.xnat.org/. [Accessed: 24-May-2019].

[46] Oracle, “Oracle VM VirtualBox,” pp. 1–362, 2012.

[47] VMware, “VMware,” 2019. [Online]. Available: https://www.vmware.com/. [Accessed: 27-Jul-2019].

[48] G. Kiar, K. J. Gorgolewski, D. Kleissas, W. G. Roncal, B. Litt, B. Wandell, R. A. Poldrack, M. Wiener, R. J. Vogelstein, R. Burns, and J. T. Vogelstein, “Science in the cloud (SIC): A use case in MRI connectomics,” *Gigascience*, vol. 6, no. 5, pp. 1–10, May 2017.

[49] Docker Inc., “What is a Container?: A standardized unit of software,” *Docker.Com*, 2019. [Online]. Available: https://www.docker.com/resources/what-container. [Accessed: 30-Jun-2019].

[50] “Kitematic | Docker Documentation.” [Online]. Available: https://docs.docker.com/kitematic/. [Accessed: 02-Jul-2019].

[51] “shipyard project.” [Online]. Available: http://shipyard-project.com/. [Accessed: 02-Jul-2019].

[52] Portainer, “Portainer Management, Docker User Interface, Container Software - Auckland, Singapore, San Francisco | Emerging Technology Partners,” 2019. [Online]. Available: https://www.portainer.io/. [Accessed: 31-May-2019].

[53] M. Böhnlein and A. U. Vom Ende, “XML - Extensible Markup Language,” *Wirtschaftsinformatik*, vol. 41, no. 3, pp. 274–276, 1999.

[54] F. Falahati, E. Westman, and A. Simmons, “Multivariate data analysis and machine learning in Alzheimer’s disease with a focus on structural magnetic resonance imaging,” *J. Alzheimer’s Dis.*, vol. 41, no. 3, pp. 685–708, 2014.

[55] Y. Huo, J. Blaber, S. M. Damon, B. D. Boyd, S. Bao, P. Parvathaneni, C. B. Noguera, S. Chaganti, V. Nath, J. M. Greer, I. Lyu, W. R. French, A. T. Newton, B. P. Rogers, and B. A. Landman,

"Towards Portable Large-Scale Image Processing with High-Performance Computing," *J. Digit. Imaging*, vol. 31, no. 3, pp. 304–314, 2018.

[56] D. Marcus, "XNAT Clinical Analytics Platform (XCAP): An Informatics Platform for Machine Learning Applications in Clinical Research."

[57] "PostgreSQL: The world's most advanced open source database," 2013. [Online]. Available: https://www.postgresql.org/. [Accessed: 01-Jun-2019].

[58] B. Shaik, "Simple PostgreSQL Blog: PostgreSQL Architecture," 2013. [Online]. Available: http://bajis-postgres.blogspot.com/2013/10/postgresql-architecture.html. [Accessed: 30-Jun-2019].

[59] T. Bui, "Analysis of Docker Security," 2015.

[60] Docker Inc., "Overview of Docker Compose," *docs.docker.com*, pp. 1–6, 2015.

[61] C. Evans, "The Official {YAML} Web Site," *{YAML:} {YAML} Ain't Markup Language*, 2011. [Online]. Available: http://yaml.org/. [Accessed: 06-Jun-2019].

[62] The Apache Software Foundation, "Apache Tomcat," 2011. [Online]. Available: http://tomcat.apache.org/. [Accessed: 06-Jun-2019].

[63] Mildenberger Peter, M. Eichelberg, and E. Martin, "Introduction to the DICOM standard," p. 8, 2001.

[64] "The Gradle Wrapper," 2018. [Online]. Available: https://docs.gradle.org/current/userguide/gradle_wrapper.html. [Accessed: 06-Jun-2019].

[65] "Clinical Data Types - XNAT MarketplaceXNAT Marketplace." [Online]. Available: https://marketplace.xnat.org/plugin/clinical-data-types/. [Accessed: 06-Jun-2019].

[66] "IQ Assessment - XNAT MarketplaceXNAT Marketplace." [Online]. Available: https://marketplace.xnat.org/plugin/radiological-assessment/. [Accessed: 06-Jun-2019].

[67] "mricron XNAT Pipeline." [Online]. Available: https://github.com/jhuguetn/xnat-pipelines/tree/master/mricron.

[68] "QAP." [Online]. Available: https://github.com/jhuguetn/xnat-pipelines/blob/master/QAP/README.md.

[69] "mricrogl." [Online]. Available: https://github.com/jhuguetn/xnat-pipelines/tree/master/mricrogl.

[70] "DTI-preprocessing." [Online]. Available: https://github.com/jhuguetn/xnat-pipelines/tree/master/dti_preprocessing.

[71] "MRI bias field correction." [Online]. Available: https://github.com/jhuguetn/xnat-

pipelines/tree/master/bias_correction.

[72] "MRI anatomical defacer." [Online]. Available: https://github.com/jhuguetn/xnat-pipelines/tree/master/mri_anat_deface.

[73] "xnatdev / xnat-image-viewer-plugin." [Online]. Available: https://bitbucket.org/xnatdev/xnat-image-viewer-plugin/src/master/. [Accessed: 07-Jun-2019].

[74] "XTK." [Online]. Available: https://github.com/xtk/X#readme.

[75] Google, "Closure Tools - Google Developers," 2013. [Online]. Available: https://developers.google.com/closure/. [Accessed: 07-Jun-2019].

[76] NRG Lab, "Who Uses XNAT?" [Online]. Available: https://www.xnat.org/about/xnat-implementations.php. [Accessed: 11-Jun-2019].

[77] S. D. Larson and M. E. Martone, "Ontologies for neuroscience: What are they and what are they good for?," *Front. Neurosci.*, vol. 3, no. MAY, pp. 60–67, 2009.

[78] E. Muller, J. A. Bednar, M. Diesmann, M. Gewaltig, M. Hines, and A. P. Davison, *Python in Neuroscience*. 2015.

[79] SciPy developers, "SciPy.org — SciPy.org," 2019. [Online]. Available: https://www.scipy.org/. [Accessed: 12-Jun-2019].

[80] A. J. Nathan and A. Scobell, "How China sees America," *Foreign Affairs*, vol. 91, no. 5. p. All, 2012.

[81] "PEP 249 – Python Database API Specification v2.0 | Python.org." [Online]. Available: https://www.python.org/dev/peps/pep-0249/. [Accessed: 12-Jun-2019].

[82] R. T. Fielding, "Architectural Styles and the Design of Network-based Software Architectures," University of California, 2000.

[83] "Pypi XNAT search." [Online]. Available: https://pypi.org/search/?q=xnat.

[84] MathWorks, "MATLAB - MathWorks - MATLAB & Simulink," *Matlab*, 2019. [Online]. Available: https://www.mathworks.com/products/matlab.html. [Accessed: 13-Jun-2019].

[85] Python.org, "tempfile: Generate temporary files and directories," 2019. [Online]. Available: https://docs.python.org/2/library/tempfile.html. [Accessed: 02-Jul-2019].

APPENDICES

A - XNAT USER GUIDE

A.1 INTRODUCTION

XNAT@DI is an instance of the popular Neuroimaging archive system XNAT (xnat.org) and archives all imaging data acquired at the ICVS and Hospital of Braga and used in the Department of Informatics.

This guide covers the most commonly performed operations with the XNAT@DI web portal. For more detailed information on how to use the portal please visit the XNAT wiki, or if you can't find the answer for your question, please email a XNAT administrator.

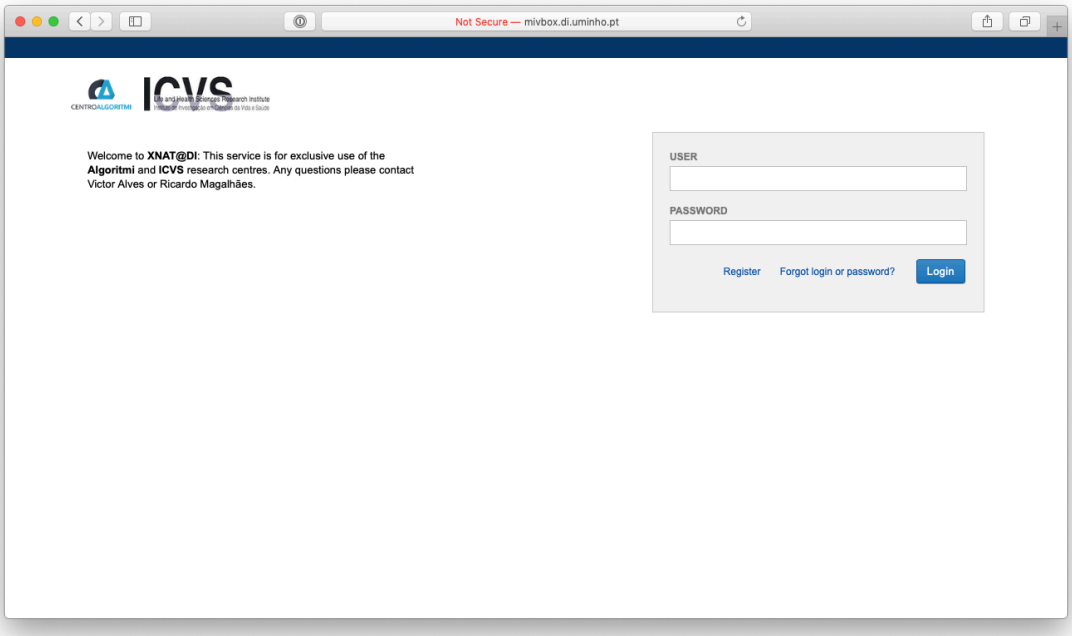

Figure A.1 - UMinho XNAT platform

A.2 SITE ACCESS

Registration

All UMinho staff and students can register to use XNAT@DI. For this, they need to email Prof. Victor Alves at valves@di.uminho.pt asking for an account in the platform. Staff and students may use their institutional email. After this it is necessary to be granted access to a project. Non-UMinho users should also email Prof. Victor Alves with their preferred username and contact details, including the contact

details of an investigator for the project(s) they wish to be granted access to in the comments sections. XNAT@DI administrations will enable the account after confirming with the relevant project investigator(s).

Login

To access to the portal, access the address the platform address or copy/paste this address into a web browser address bar. Next insert your login details in the login box. If any problem occurs, try to delete your cookies. If the problem persists, please contact Prof. Victor Alves. Login information is displayed in the top right-hand corner of all pages on the site above the general search bar. To logout simply click the "logout" link. Login sessions will expire after 15 mins of inactivity, with the remaining time displayed next to the username of the logged-in user. If you wish to renew your session click the "renew" link also in the top right-hand side of the screen, and the 15 min counter will start again.

Project Membership

Access to imaging data is controlled via project membership. Members are added to a project by administrators and/or project investigator.

To ask the administrators to add a new member to a project, the investigator must email valves@di.uminho.pt with the name and email address of the person to be added and the project name that the access is to be given to.

If you are a project owner, you can use the "Access" tab of the project page. Access is granted by on the Access tab of the project page, new members are added by selecting the available users from the list.

Members can be removed from a project by a project owner or MBI administrators by clicking the "Remove" button next to their names.

A.3 NAVIGATION

XNAT@DI ID Conventions

All projects acquired and make available at XNAT@DI must contain a name. Generally, this name is the projects name.

Home Screen

The home screen provides several convenient methods to navigate the imaging data that is accessible to the user. A list of recently accessed projects by the user will appear in the bottom left corner of the screen and a list of recently uploaded scanning sessions on the right. Clicking on these entries will navigate the user to the corresponding project/session. To narrow down the list of projects, subjects or sessions, search criteria can be entered into the search box in the middle of the page. To return to the home screen from any page, select the XNAT logo in the top left corner.

Command Ribbon

The command ribbon is accessible from all pages within the site. Clicking on the “Browse” opens a hierarchical menu that provides links to all projects, subjects and imaging sessions that are accessible to the user.

A.4 DATA ACCESS

It is the researcher's responsibility to verify the quality of the imaging data uploaded to XNAT. Generally, the data uploaded to the XNAT instance goes to the "Pre-archive". The research must review and archive the data himself, giving a project to it.

Snapshots and DICOM Headers

After an imaging session is uploaded to XNAT, image snapshots are generated for recognized image types and can be displayed along with other basic meta-data by toggling the +/- button next to the scan name on the session page. Detailed metadata can be accessed from the DICOM headers via the link “View DICOM Headers”.

Image Viewer

XNATUM provides a convenient built-in viewer for visual inspection of imaging data, which can be accessed by the “view images” link on the “Actions” menu on the session page.

A.5 MANAGE SESSIONS DATA WITH HOROS

To manage sessions through Horos or other DICOM medical image viewer you just to have to add XNAT as a new image location on the correspondent DICOM port.

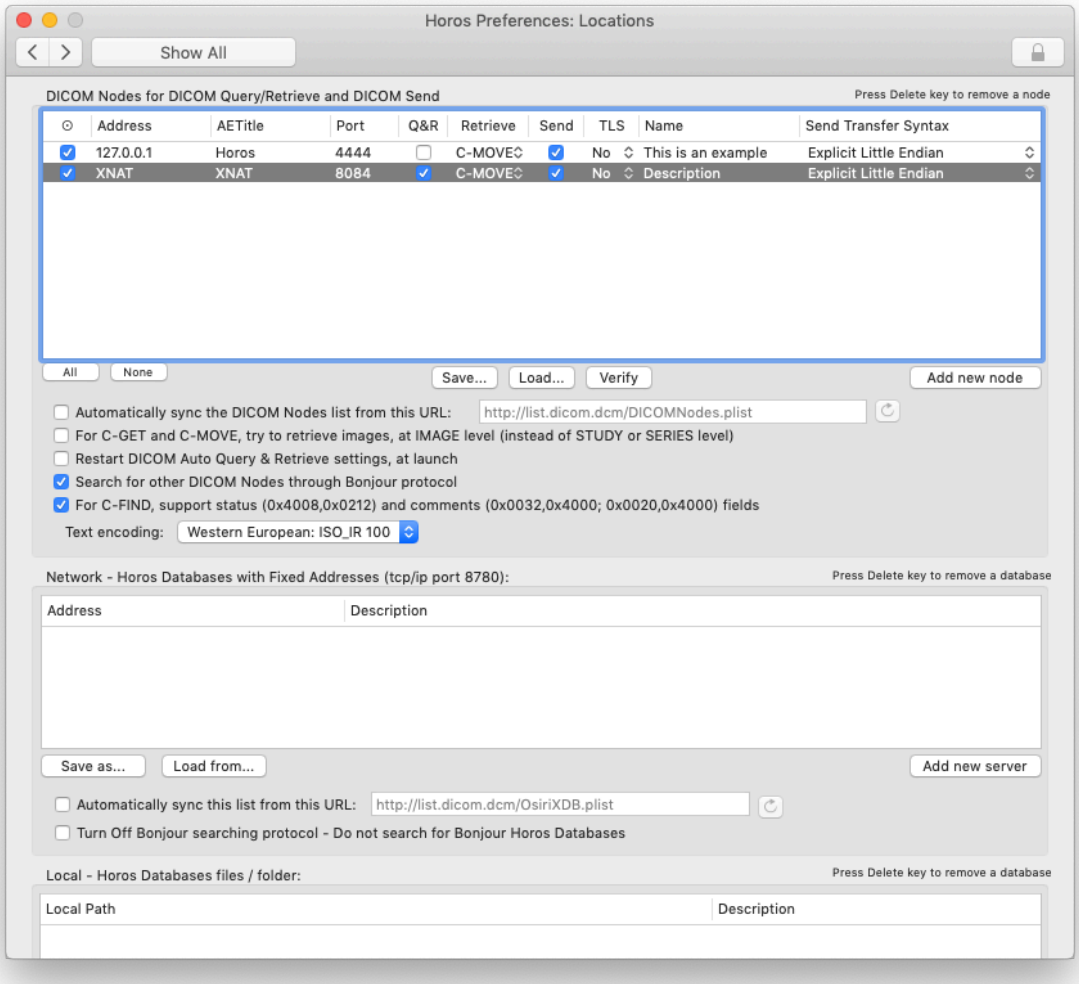

Figure A..2 - HOROS XNAT location configuration

B – XNAT DATA MODEL

A data model is an abstract model that organizes certain elements and standardizes how they can relate to each other. It explicitly determines the structure of the data and for that reason it is sometimes referred to as a data structure.

XNAT server is a platform built for imaging research purposes. Its functions can archive, import, process and securely distribute the data. The core data-model used in XNAT previews the use of the XNAT server in various projects. There are three core data-types in XNAT data model: Projects, Subjects and Experiments. These three data-types relate with each other.

Projects

A project is used to define a collection of a certain data stored in the server. The project is used to define a secure structure of data. Users can have certain permissions for data within a certain project. They can be Owners, Members or Collaborators. This permission defines who can read, save or delete data. For example, in a specific research study we can give permission for the principal researcher to save, read and delete data but for an internal researcher only to read the data.

Subjects

A subject is anyone who participates in a study. Imaging studies has traditionally focused on human studies, however, the subject could be non-human. It exists in a context of a project and it's "owned" by the project where it was created on. Also, subjects can be shared with other projects to capture longitudinal data from various studies.

Experiments

An experiment is an event by which imaging data or non-imaging data is acquired. It cannot exist outside the context of a project and can be shared into other projects.

Subject Assessments

A subject assessment is a specific kind of experiment which is intrinsically associated with a subject. It contains a foreign key to the subject. Subject Assessments are a common extension point in XNAT. For example, if a user is trying to model new data, it is usually an extension of subject assessments. By default, XNAT provides only one extension of subject assessment which is its Imaging Session. New subject assessment tools need to be created to model a variety of subject tests, forms, etc.

Imaging Sessions

Imaging sessions are specific type of Subject Assessment. It is used to capture the data acquired in the normal course of Imaging studies. It adds a collection of image scans, reconstructions and image assessments, to capture the data as scanned, preprocessed and processed.

MR Session / PET Session / CT Session

There are different types of Imaging Sessions. They correlate to the types defined by the DICOM standard. Imaging Sessions introduce a collection of sub-elements which are used to capture imaging data.

Scans

Scans are related with the different Imaging sessions type. They are used to model individual scan series from a single scanning session. A single Session usually contains multiple scans of different types. They also specify the files which make up that scanning series (including RAW data).

Reconstructions

A single imaging session can contain multiple image reconstructions. These are typically the results of a pre-processing stream.

Image Assessments

A single imaging session can contain multiple imaging assessments. They are typically the results of a processing stream and can contain both generated images and statistics. Image Assessments are in general the result of processing results in an XSD like freesurfer or fsl.

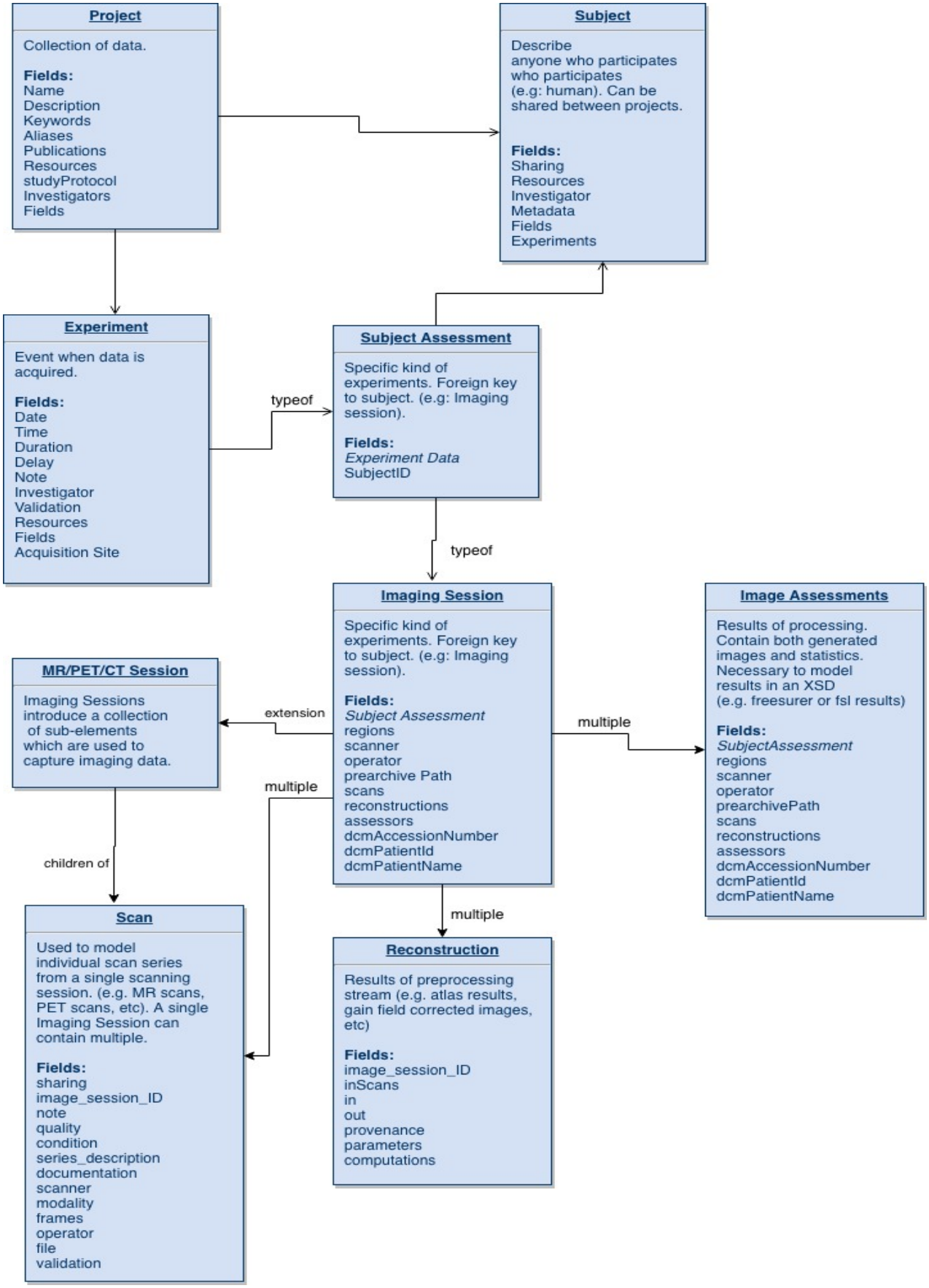

Figure B.3 - XNAT Data Model