Dissertation

Master in Computer Engineering – Mobile Computing

Strategies for Digital Preservation of Information

Carlos André Lopes Santos Rosa

Leiria, *September 2017*

This page was intentionally left blank

Dissertation

Master in Computer Engineering – Mobile Computing

Strategies for Digital Preservation of Information

Carlos André Lopes Santos Rosa

Master thesis carried out under the supervision of Professor Patrício Rodrigues Domingues, Professor at School of Technology and Management of the Polytechnic Institute of Leiria and co-supervision of Professor Olga Marina Freitas Craveiro, Professor at the School of Technology and Management of the Polytechnic Institute of Leiria.

This page was intentionally left blank

Acknowledgements

I would like to start by thanking Marco Cova, Hugo Larcher and Professor Catarina Reis for the opportunity of making this Master. Their motivation to embrace this challenge and support throughout the development of this work were very important to me. Thank you.

I would like to thank Professor Olga Craveiro and Professor Patrício Domingues for all the guidance, patience, insight and work dedicated to me and to this thesis. Their participation in this work was undoubtedly of paramount importance to the results achieved. Thank you.

I would also like to thank my wife Ana Vilhena for all the support and comprehension given during this endeavor. Without her precious help, it would have been extremely difficult to accomplish this work. My thank goes to all my family as well. Thank you for helping me achieve this important goal in my life.

Finally, I would like to thank the Escola Superior de Tecnologia e Gestão of the Polytechnic Institute of Leiria and Professor Carlos Grilo for the help and opportunity of making this master. Thank you.

This page was intentionally left blank

Resumo *(in Portuguese)*

Com a entrada na era digital, a informação digital produzida verificou um crescimento exponencial. Segundo Seamus Ross, a informação digital é um produto cultural [1]. A crescente dependência na informação digital está a mudar a forma como a nossa cultura é registada. O suporte de armazenamento físico, a estrutura lógica da informação e a sua interpretação já não estão obrigatoriamente relacionados. A Internet proporcionou um ambiente prospero ao desenvolvimento de novas comunidades assim como ao volume de informação por elas produzido.

Também o software e o hardware evoluíram e com eles a capacidade de produzir informação mais detalhada, de melhor qualidade, mas também mais exigente em termos de armazenamento. Um bom exemplo do mesmo é o conteúdo multimédia, áudio e vídeo, mas muitos outros haverão. Esta nova realidade despertou a consciência para a necessidade de preservar toda esta informação para as gerações vindouras. Contrariamente aos seus semelhantes analógicos, os formatos digitais requerem um esforço diferente para serem conservados visto estarem expostos a fragilidades como a deterioração do suporte onde estão armazenados ou a obsolescência de formatos. Várias medidas devem ser tomadas para garantir o seu acesso a longo prazo. Para satisfazer esta necessidade os repositórios de dados digitais evoluíram, possuindo agora funcionalidades capazes de implementar diferentes estratégias para a preservação digital de dados.

Esta tese apresenta uma análise do presente estado da arte do software disponibilizado sob licença de código aberto de repositório para a preservação digital de dados identificando as cinco soluções mais relevantes. Dessas soluções, escolhemos a solução com melhores características técnicas e com uma maior comunidade de utilizadores, o RODA, para a qual propomos e implementamos melhorias a uma estratégia de preservação de dados existente: a federação. Esta melhoria consiste na construção de um mecanismo de interoperabilidade capaz de permitir a interação do RODA com outros sistemas.

Esta melhoria é feita através da implementação de um protótipo constituído por um servidor CMIS dentro do repositório, que comunica com aplicações cliente através do protocolo CMIS. A este protótipo damos o nome de RODA OpenCMIS Server, ou RODA-OCS. O protótipo permite ao RODA expor publicamente conteúdos que nele se encontrem guardados num ambiente controlado, de uma forma autenticada e segura através de uma

integração do mesmo com o sistema de permissões nativo do RODA. Este protótipo não só implementa um mecanismo de navegação de ficheiros como também um mecanismo de pesquisa capaz de encontrar conteúdos baseados nos seus metadados, tanto técnicos, como descritivos.

Por fim, apresentamos uma demonstração do funcionamento do protótipo. Mediante a utilização de um conjunto de dados concebido para testar as funcionalidades implementadas pelo protótipo, apresentamos na prática o seu funcionamento e analisamos os resultados obtidos.

Palavras-chave: dados digitais, preservação, repositórios, código aberto

This page was intentionally left blank

Abstract

With the advent of the digital era, the digital information produced has grown exponentially. According to Seamus Ross, digital information is a cultural product [1]. The growing dependency on digital information is changing the way our culture is recorded. There is no longer a strict relation between the logical structure of information, the physical storage support and its interpretation. Internet provided the right environment not only for the thriving of new communities but also for the growth of the information produced by them.

Software and hardware have also evolved. Along with them came new capabilities of producing more accurate and space demanding information. One good example is multimedia content, audio and video, but there are many more. This new reality rose an awareness for the need to preserve all this information for future generations to come. Unlike their analogue peers, digital formats require a different effort to maintain as they are exposed to such threats as the deterioration of the medium they are stored in or formats obsolescence. A number of actions must be taken to ensure their long-term access. To address this need, digital repositories have evolved, accommodating now sets of features capable of implementing different strategies for the digital preservation of information.

This thesis presents an analysis of the current state of the art open source repository software for the digital preservation of information identifying the five most relevant solutions. From those solutions, we picked the most feature rich and broader user community software, RODA, to which we propose and implement further improvements to an existing preservation strategy: federation. These improvements consist in building into the system an interoperability mechanism capable of allowing RODA to interact with other systems.

This improvement is made by implementing a prototype composed by a CMIS server inside the repository, which communicates with client applications through the implementation of the CMIS protocol. We name this prototype RODA OpenCMIS Server, or in short, RODA-OCS. The prototype allows RODA to expose contents stored inside it publicly, under a controlled environment, in an authenticated and secure way. This goal is achieved by integrating our prototype with RODA native permissions system. RODA-OCS not only implements a file browser mechanism for content navigation but also a query engine

capable of searching and retrieving contents based on their metadata, either technical or descriptive.

Finally, we present a demonstration of the functioning of RODA-OCS. Through the use of a dataset conceived to test the functionalities implemented by our prototype, we showcase and further analyze the obtained results.

Keywords: digital data, preservation, repositories, open source

This page was intentionally left blank

List of figures

This page was intentionally left blank

List of tables

This page was intentionally left blank

List of acronyms

ACL – Access Control List

AES – Advanced Encryption Standard

AGLS – Australian Government Locator Service

AGPL – Affero General Public License

AIP – Archival Information Package

ANSI – American National Standards Institute

API – Application Programmable Interface

BSD – Berkeley Software Distribution

CCSDS – Consultative Committee for Space Data System

CERN – Conseil Européen pour la Recherche Nucléaire

CIFS – Common Internet File System

CMIS – Content Management Interoperability System

CPA – Commission on Preservation and Access

CQL – Context Query Language

CSS – Cascading Style Sheets

DESY – Deutsches Elektronen-Synchrotron

DIDL – Digital Item Declaration Language

DIP – Dissemination Information Package

DROID – Digital Record Object IDentification

EAD – Encoded Archival Description

ECM – Enterprise Content Management

EPFL – École polytechnique fédérale de Lausanne

FNAL – Fermi National Accelerator Laboratory

FTP – File Transfer Protocol

GNU – GNU Not UNIX

GPL – General Public License

HTML – HyperText Meta Language

HTTP – HyperText Transfer Protocol

IIPC – International Internet Preservation Consortium

IoT – Internet of Things

ISO – International Standardisation Organization

JAAS – Java Authentication and Authorization

JDBC – Java DataBase Connectivity

JSON – JavaScript Object Notation

LAMP – Linux Apache MySQL PHP

LDAP – Lightweight Directory Access Protocol

LGPL – Lesser General Public License

LOCKSS – Lots of Copies Keep Stuff Safe

MARC – Machine Readable Cataloguing

MARC21 – Machine Readable Cataloguing 21

MARCXML – Machine Readable Cataloguing in eXtensible Markup Language

METS – Metadata Encoding Transmission Standard

MIME – Multipurpose Internet Mail Extensions

MIT - Massachusetts Institute of Technology

MIX – Metadata for Images in XML

MODS – Metadata Object Description Schema

NGO – Non-Governmental Organizations

NZGLS – New Zealand Government Locator Service

OAI-ORE – Open Archives Initiative Object Reuse and Exchange

OAI-PMH – Open Archives Initiative Protocol for Metadata Harvesting

OAIS – Open Archival Information System

OCLC – Online Computer Library Centre

OS – Operating System

PHP – Pre-Hypertext Processor

PREMIS – PREservation Metadata Implementation Strategies

RDF – Resource Description Framework

REST – Representational State Transfer

RFC – Request For Comments

RLG – Research Libraries Group

RODA – Repository of Authentic Digital Objects

RODA-OCS - RODA Open CMIS Server

RSS – Really simple Syndication

SIP – Submission Information Package

SLAC – Stanford Linear Accelerator Center

SMB – Server Message Block

SOAP – Simple Object Access Protocol

SPARQL – Simple Protocol and RDF Query Language

SQL – Structured Query Language

SRU – Search / Retrieval via URL

SWORD – Simple Web-service Offering Repository Deposit

TCK – Test Compatibility Kit

UNESCO – United Nations Educational, Scientific and Cultural Organization

URL – Unique Resource Link

USA – United States of America

UTF – Unicode Text Format

UUID – Universal Unique IDentifier

WAR – Web application ARchive

WARC – Web ARChive

WEBDAV – Web-based Distributed Authoring and Versioning

WSDL – Web Service Definition Language

XACML – eXtensible Access Control Markup Language

XML – eXtensible Markup Language

This page was intentionally left blank

Table of Contents

1. Introduction

Information preservation can simply be defined as the set of processes to store, index and access information [2]. In recent years, the creation of digital content has grown exponentially. Gantz and Reinsel report that the so called digital universe will grow from 2005 to 2020 by a factor of 300, from 130 exabytes to 40,000 exabytes [3]. They also predict that the whole set of data will double roughly every two years from 2012 to 2020. Digital video is a good example of the current data deluge: the demand for increasing resolutions and higher frame rates, despite all improvements in compression, have substantially increased the size of video files. Smartphones, with all their data sensors, namely photo and video recording capabilities, are also major contributors to the current massive production of data [4]. The Internet of Things (IoT) is poised to generate increasing amount of data, even if IoT middleware can help by reducing the volume of data to store and preserve [5]. The sheer volume of digital information to preserve is immense and will continue to grow over the years. In fact, major trends like Big Data have fostered the perception of digital data as valuable assets, strengthening the need for digital data preservation and henceforth for proper digital repositories [6]. This way, the field of digital information preservation has to address a huge challenge.

The panorama of information preservation has substantially changed with the advent of the digital era. In fact, when paper was the main medium globally adopted for storing knowledge and information, libraries were the obvious and natural places responsible for guarding, protecting and maintaining all information stored in printed formats, such as, books, papers or other. It was not until 1960's that archivists and librarians felt concerned about the preservation of electronic records. This fostered the emergence of the Machine-Readable Records branch, formed at the National Archives in the USA [7].

In the early 1990's, existing materials started being ported from printed formats to their digital equivalents, being "re-born" digital. This rose an awareness for the need to address the problem of the deterioration of digital-only information. Indeed, if exposed to mild humidity conditions and kept in a moderate environment, paper is relatively easy to preserve, or at least has a lifetime measured in decades and thus preservation can be organized accordingly. In addition, techniques such as microfilming allowed for affordable and durable preservation of paper-based documents [8]. The same does not apply to digital formats. Even if the base of digital formats is simple binary 0 and 1, digital encompasses a rich set of various resources such as text, audio/video, images, computer programs, just to name a few. Besides the main data,

additional information regarding the resources – format, software environment, operating systems, etc. – is required to properly preserve and access digital data. Digital formats are very fragile and even on controlled environments, there must be an active management to assure their good shape and longevity [7] [8].

The paper paradigm shift to the digital reality clearly reflected itself on other knowledge institutions, not only libraries and archives. Schools, universities and other institutions also found themselves in a situation where using paper as the main support for storing information was no longer the best choice, either because of storage space issues, preservation issues or simply because of the advance of technology. It no longer made sense to keep using outdated and less flexible means to keep information. However, if on one side there was already an awareness about the need to preserve digital information, on the other side, data repositories that followed or implemented those standards were scarce or inexistent. The logic step for these institutions was to develop their own solutions and implementations of digital preservation repositories. Their own premises and academic communities were the perfect audience to test and perfect them. Much of the open source software has its root from an academic reality [9].

Today, we are facing yet another challenge: the Internet. In a world where the demand for being always connected is higher than ever before, the global network and its omnipresence make it the obvious choice to store and disseminate information and knowledge. With an exponential growth observed during the 1990's, the volume of information available on the Internet expanded as well. However, unlike printed formats, its ephemeral existence and highly volatile availability were shortly noticed. Internet is indeed the perfect environment to publish information that in many cases is not available elsewhere.

The awareness and need to ensure the preservation and long-term access to this information gave birth to web archiving [10], an important subset of digital information preservation. Indeed, consisting in the collection of information available in the World Wide Web for future access, the process of harvesting that information is challenging due to its heterogeneous nature. For this purpose, the WARC standard [11] was created in 2009 by the International Internet Preservation Consortium (IIPC) [12]. Standing for "Web ARChive", WARC specifies a method for combining multiple digital resources and associated metadata into an aggregate archival file. These files are then accessed by web crawling software when harvesting information from websites [13]. The resulting WARC files are passible of being ingested and stored on digital repositories for preservation. However, according to a recent survey [13], very few institutions are effectively downloading the WARC files generated by

the web crawlers and storing them in local preservation systems or repositories. In spite of not being a common practice among institutions, there is a recognition for the need to perform preservation of web content.

Digital information is fragile and faces threats like technological obsolescence and the deterioration of digital storage media. As an answer to the need of protecting it, various digital preservation repository solutions surfaced in the market, both commercial and open source. Repositories aim to address threats to digital information by implementing preservation strategies to ensure long term access. Information redundancy, formats migration or the encapsulation of contextual metadata along with information are just a few examples.

However, any of these strategies and repository capabilities can still be explored and further improved. One example is interoperability. The possibility of communication and interchange of preserved information between repositories is a road less traveled and the body of work of this thesis. From the open source repositories currently available, we picked RODA as the host to develop and integrate RODA-OCS (RODA Open CMIS Server), an interoperability server prototype. The prototype allows RODA to share its contents publicly in a secure and read-only way, without risking any information adulteration.

1.1. Motivation

With the constant growth of digital information produced by companies, institutions and individuals, the necessity for its preservation is now a given fact for everyone, not only for corporations or governments. For digital information to be interpreted by humans, it depends on a computer system capable of executing methods and functions of a given software to render it. This technological dependency represents one of the big challenges for digital preservation but it is in the very nature of technology to evolve at a much faster pace than any other subject in nowadays societies [14].

Nowadays, repository systems are designed with these needs in mind. A number of measures have been taken to allow repository software to deal with the reality of digital preservation, standards have been created and implemented. However, new standards are arising and the existing ones can still be perfected.

After making a research about the current digital repository software available and studying their characteristics [9], we concluded that repositories lacked or scarcely

implemented features that allowed them to interact with other repositories and share stored information. We also concluded that, to the best of our knowledge, none of the solutions surveyed implemented the CMIS (Content Management Interoperability System) protocol. This posed as the main motivation to implement a CMIS server in one of the studied solutions.

1.2. Main Goals and Contributions

In this thesis, we intended to identify the current main standards for digital preservation of information. To accomplish this task, we started by surveying the existing open source repository software in the market. We analyzed each solution and identified the digital preservation strategies implemented by them. This study also identified the most relevant repository solutions, from which we intended to choose the most relevant and dote it with a new preservation strategy or enhance an existing one.

The main contributions of this work were:

i) Survey of the currently available open source repository software solutions to date [9] where we established a comparison of their features and identified the most relevant solutions to the field.

ii) Choice of the most accepted and feature rich repository software, RODA, for the improvement of an existing preservation strategy, interoperability, through the implementation of an interoperability server prototype, RODA-OCS.

iii) Proposal and implementation of RODA-OCS, a functional prototype of a CMIS server.

1.2.1. Publications and software

This work resulted in the publication of a scientific article:

- "**Open Source Software for Digital Preservation Repositories: A Survey**", in the *International Journal of Computer Science & Engineering Survey* (IJCSES) Vol.8, No.3, June 2017, pages 21-39, by Carlos André Rosa, Olga Craveiro and Patrício Domingues [9].

- Another main contribution is the developed software, namely the prototype RODA-OCS: "**MEI-CM Digital Preservation Repositories**", in Bitbucket.org, by Carlos André Rosa [15].

1.3. Outline

This document is organized in a sequential order. The concepts or discussions presented in each chapter may rely on concepts, ideas or explanations presented in the previous chapters.

In Chapter 2 we started by introducing the Digital Preservation concept. We presented the existing main strategies for the field followed by the introduction of the concept of metadata. We proceeded by identifying the main characteristics of digital preservation enabled repositories followed by a comparison of the most recent open source software available during our research, to the best of our knowledge.

In Chapter 3 we presented the interoperability between repositories. The chapter started by presenting our proposed system architecture for the implementation of an interoperability feature in a repository software. We proceeded by presenting RODA, the chosen open source software to host the new functionality. Then, we explained the protocol to be implemented, the server component and CMIS Workbench, a client application created and made available by the Apache Chemistry project, used to help throughout the implementation process.

In Chapter 4 we detailed the implementation of the RODA-OCS prototype. We explained RODA repository internal modules storage and permissions, with which we interacted, followed by RODA-OCS server prototype implementation class hierarchy. We also gave the details of the server bootstrap and explained how the file browser and query engine modules work.

Finally, Chapter 5 presented a demonstration of the implemented system and the results obtained followed by Chapter 6 that presented this thesis conclusions, summarized the developed work and indicated a list of developments that can be made to further improve and expand the RODA-OCS prototype.

2. Digital Preservation

In this chapter, we describe the terminology and the main concepts relevant to the digital preservation of information, considered to be the most important for the development of this thesis.

Digital preservation is a set of activities or efforts made to ensure the long-term access to valuable digital information for future generations. These efforts involve managing and maintaining digital objects as original and as authentic as possible since their life spans are considerably shorter than the traditional printed objects, more prone to stand the test of time and last longer with minimum signs of degradation. A digital object duration relies on the quality and durability of the support or medium and format it is stored in. Those formats and the software which produce them tend to become obsolete and be replaced by newer ones. The same happens with media, which besides being replaced by newer formats and supports, also suffers from degradation. Even when properly taken care of, media supports are rather delicate objects given their construction (small parts either electrical, optical or magnetic) and are prone to become damaged easily [16].

As so, with the advance of technology and computer systems, the digital materials produced, either "born-digital" or digitized, are threatened to become unusable or obsolete. The digital preservation must be planned, the digital materials permanent value must be evaluated and strategies for its preservation must be set [16] [17]. The digital materials must be then stored in repositories capable of retaining, not only the data or records, but also context information relative to them. This additional context information is called metadata. Metadata plays a very important role in the process of preservation, with the lack of it causing digital materials to lose their meaning, even over a very short period of time [18].

2.1. Main strategies

The digital preservation of information should be thought since the moment of its creation. From taking actions as simple as making backups in cheaper storage devices to more elaborate strategies as keeping it in repositories, systems should be prepared to deal with much larger volumes of objects and with much better preservation and managing capabilities [14]. The main strategies for the preservation of digital information being implemented in the present

are: i) replication or redundancy, ii) auditing, iii) refreshment, iv) emulation, v) migration or conversion, vi) encapsulation and vii) federation. These strategies represent different efforts, set of actions or methodologies followed to ensure digital information remains usable over time.

Replication/Redundancy – The most common strategy for data preservation is redundancy. Once a digital material is created, it can be copied or replicated to other storage supports without limitations, creating several backups of the content. Anyway, although this seems a good strategy at first, it is not a good one over a long period of time. This strategy does not prevent the possible failure of all supports at the same time or their inevitable obsolescence. If the digital materials stored in those supports become illegible, either because the supports were damaged, or because there is no longer any system or device capable of reproducing them, the data is lost [17].

Auditing – Any digital preservation repository system working with large volumes of data poses several challenges to its management. One of them is the preservation of the integrity of the data stored as well as the integrity of the system itself. As the amount of data grows, the task of auditing the system may not become an easy task for a system administrator to perform manually. This creates the need for an automated auditing system within the repository. Such a system must be capable of running periodic checks on storage integrity, data integrity, sending alerts on any failure event and ultimately taking measures to repair the system itself so that no irreparable or permanent data loss occurs [19].

Refreshment – Before the support or medium where a digital material is stored in becomes damaged or outdated, the information is copied to a more updated support with the goal of preventing it from being lost over time. This procedure is not considered digital preservation by itself, but integrates the process of the digital preservation. It is vital for any strategy of digital preservation of information that all supports are checked on a regular basis to avoid data loss [17].

Emulation – This strategy consists in creating software capable of replicating the original conditions needed to reproduce digital materials on a newer or more modern system. There is a concern on recreating the original experience and context the digital material was first conceived and not only access or reproduce the content. The original files are involved in this process, which is a good approach if there is an historical value in the materials. The downside of this strategy is that the emulation software needs to be maintained and updated as the computer systems and hardware evolve. There is also the chance of the digital materials not

being correctly reproduced by the emulation software. Without the original software, computer system and conditions there is no effective way to judge it [17] [20].

Migration / Conversion – The migration strategy consists on converting a file or digital material into a new format. If the new format holds or implements all the characteristics of the previous formats, it is possible for the digital objects to retain their original features. However, if the new format renders obsolete some of their ancestor characteristics, then information might get lost and resulting digital objects may not be an exact reproduction of the original content. This strategy focuses on preserving the content over the format of the information. Since there is a migration to a newer format the safest choice is to keep the original information as well which not always happens. Keeping the original information or digital objects allows for a new migration or conversion process in the future. This is particularly helpful if the previous conversion process was not accurate enough and the resulting or converted digital materials were not a faithful reproduction of the original ones. The new migration process may be an improved version and the resulting migrated objects may be a more perfect reproduction of the original ones [17] [20].

Encapsulation – A digital object by itself may be important and relevant for preservation, but if placed into a certain context, its original context, it definitely is of greater value. This contextual information can be specified, expressed and saved in a text format, either by the person who created the object, the application or both. To this information about the information itself we call metadata. The strategy of encapsulation consists in keeping as much information about digital objects as possible since the moment of its creation. Metadata can include details on the most varied aspects of the file, may it be a description of the file or details about the format of the file itself. There is also technical metadata, information that can be extracted from the file by specialized tools and software or included in the provided metadata as well. This metadata gives meaning and authenticity to the file and helps recreating its original conditions in the future, either if it is for migration, creating an emulator or simply making sure the digital object was not modified throughout the preservation process. Metadata can be encapsulated in a digital object internally or externally, by including a text file alongside the original object on a compressed file, for example. By choosing to include a digital object metadata externally there is also the advantage of being able to recover that same metadata even if the original object is damaged [17] [20].

Federation – A good approach to solve the problem of data loss and accessibility is to decentralize it and distribute copies of information to several locations, preferably in an

automated way. This is what the federation of data represents to digital preservation systems: a distributed network ecosystem where the intervenient preservation repositories talk to each other and share information automatically, replicating information among themselves every time a new peer or package is added to the network. This is the most modern approach with only few systems capable of implementing. Nevertheless, advances are being made in this field, such as defining standards for the communication among different repository software that wish to provide this feature or capability. However, there are still issues to be solved, namely: synchronization between systems, access rights to the digital materials and confidentiality [19].

None of the previous strategies is able to give a robust and definite answer to the digital information preservation challenge alone. Consequently, any system aiming to ensure long-term access to its contents must consider implementing a combination of any of the strategies mentioned above. Only by taking different approaches, a system will be able to tolerate a catastrophe, shall it happen.

2.2. Metadata

Metadata is the information about the information itself, or better, it is the data which provides information about various aspects related to the existing data. Metadata plays a very important role as it allows an easier way to work with data, characterizing and summarizing basic information contained in it [21]. There are four types of metadata: i) descriptive, ii) technical, iii) structural and iv) preservation.

Descriptive - The descriptive type is the metadata defined by the producer of the information. This information is used to describe a resource and give it context. It can include common elements, such as the identification of the author, the title of the material or the main keywords that can be used in the description of the resource. However, and since it is user defined, it can be anything, as long as it refers to the resource. Descriptive metadata plays an important role in the authenticity enforcement and guarantee of the digital materials when allied to the encapsulation preservation strategy, as stated before. It provides materials with its original context for future access and rendering.

Technical – We identify technical metadata as the information related to the technical aspects of a digital material, such as a file type, the creation date or the person who accessed the resource for the last time, just to mention a few. Technical metadata is often extracted with

tools designed for that purpose, although it can also be defined by the producer of the information.

Structural – Structural metadata is the information that defines the structure of the information and how digital materials relate to each other. It identifies versions, containers, compound objects, siblings, hierarchies and any other possible relations established between different materials. Structural metadata is especially important in the context of the digital preservation of information as it helps to identify the necessary conditions to render a set of files or digital materials in the future.

Preservation – The resulting information about the preservation actions taken over information. It can vary from a record of the operations performed over the material, like the calculation of checksums for fixity checking, to the verification of file formats obsolescence. The preservation metadata can also include information about the handling of the digital material and also work as an audit trail, identifying all the intervenients (either systems users or the systems themselves) in the preservation process.

The relevance of metadata in the digital preservation field led to the creation of metadata standards. The creation of these standards was a joint effort between the various standards communities, institutions and countries. Much work has been performed by entities like the American National Standards Institute (ANSI) or International Standardization Organization (ISO) just to name a few. EAD, Dublin Core and PREMIS are the standards used in our work for the descriptive and preservation metadata types. However, there are other standards developed to work with the remaining types of metadata.

EAD [22] **–** The acronym stands for Encoded Archival Description, a standard developed for encoding archival paper collections and maintained by the Technical Subcommittee for Encoded Archival Description and the American Library of Congress. This standard data dictionary consists of an XML element set and schema capable of storing all the information related to the paper collections. Given the fact that different document collections share many common metadata attributes, the EAD standard has proved its flexibility to be used to describe document collections outside of the scope it was originally developed for. Besides, many of its attributes can be mapped to other standards attributes, thus widening its adoption and increasing its popularity. This standard is used to store descriptive metadata about the digital materials.

Dublin Core [23] **–** A standard developed in 1995 at the invitational workshop in Dublin, Ohio in the Unites States of America, hence the first word of its name, “Dublin”. The other

word, “Core”, origins from the generic coverage intention of the standard. Dublin Core data dictionary consists in a fifteen element XML set and a validation schema. The elements are broad, allowing this standard to describe a wide range of resources. This standard is used to store descriptive metadata about the digital materials.

PREMIS [24] – This standard name is an acronym for **PRE**servation **M**etadata **I**mplementation **S**trategies. PREMIS consists of data dictionary expressed in XML, an XML validation schema and supporting documentation created with the purpose of supporting the preservation of digital materials and ensure their long-term usability. This standard was developed by the American Library of Congress and is maintained by its Editorial Committee.

2.3. Digital Preservation Enabled Repository

Digital repositories were born from the emergence and wide-spread use of digital materials. With the growth of this trend came the necessity of creating means of storing, browsing and retrieving digital information. At first, this was the main challenge, to be able to organize ever-growing digital collections, soon to be found that these were also exposed to the effects of time.

There are some questions a digital repository should be able to answer in order to be considered enabled for preservation. These questions involve the authenticity of the preserved digital materials, its integrity and accessibility over long periods of time, as well as, its usability [25].

The following list enumerates the main identified features a digital repository should have to be considered preservation-enabled in the context of this thesis.

Digital Preservation Enabled Repositories Main Features

A digital preservation-enabled repository should comply with the reference model for the OAIS (Open Archival Information System) standard, implementing all units of its functional model and digital objects metadata support and management system. It should support four types of metadata: i) descriptive, ii) technical, iii) structural and iv) preservation metadata. It should also implement authentication and authorization mechanisms to allow materials

ownership, action tracking and enforce user accountability. Another important feature is to allow the definition of strategies for data preservation planning and action.

Any digital preservation repository should comply with open standards: i) preservation, ii) metadata and iii) interoperability, ensure the integrity, authenticity and usability of digital objects over time by executing periodic checksum calculation and verification over existing files, as well as, existing file formats obsolescence checks, support interoperability with other repository systems through the implementation of interoperability standards such as Open Archives Initiative Protocol for Metadata Harvesting (OAI-PMH), Simple Web-service Offering Repository Deposit (SWORD) and Search / Retrieval via URL (SRU).

Other desirable features would be the provision for browsing files within each AIP (Archival Information Package) of the repository and showing each file details, the implementation of a log system capable of tracking all operations over digital objects and users, the implementation a tagging system capable of characterizing digital objects for context and searching purposes beyond their metadata information.

Lastly, it should be able to ensure access and dissemination of information over long periods of time and possess a technical / hardware infrastructure adequate to the continuous maintenance and security of the stored digital objects.

2.4. Open Source Software for Digital Preservation Repositories

In this Section, we presented a wide-scale comparisons of leading open source software solutions that can appropriately store and preserve digital information. We highlighted the main features of each system, the licensing model, the main preservation capabilities and which standards and protocols are supported. Furthermore, we provided an in-depth analysis of five of the most relevant open source solutions for digital information preservation [9].

2.4.1. Digital Information Preservation: models, standards and protocols

We analyzed OAIS and some other main standards, as well as, some protocols that are relevant for digital information preservation.

The OAIS Reference Model [9]

The emergence of digital information preservation took a while. In 1994, a task force was created from the joint effort of two groups, the Commission on Preservation and Access (CPA) and the Research Libraries Group (RLG), both comprised of archivists and publishers. This task force studied the needed actions for ensuring long-term preservation and continued access to digital materials. Later, the Consultative Committee for Space Data Systems (CCSDS) was asked to define rules and methodologies for long term archival/storage of digital data generated from space missions. The result of this effort was the Open Archival Information System (OAIS) reference model. OAIS is the first reference model on digital data preservation [26]. It became a standard for digital information preservation in January 2002 as ISO-STD 14721. In 2012, an updated version of this model was published (ISO 14721:2012) [27]. This model focuses on providing a structure along with a lexicon of well-defined concepts and frameworks for any archive or system to be built with the purpose of preserving information and making it available for long-term use by a designated community or target group.

To be OAIS-aware, an information preservation solution needs to provide functionality to deal with ingestion, preservation and dissemination of archived digital materials. For this purpose, the OAIS reference model defines that at least the following six high-level services need to be provided by the archival and preservation solutions. They are: i) ingest; ii) archival storage; iii) data management; iv) preservation planning; v) access and vi) administration [27]. On top of that, OAIS defines an environment with three main roles: management, producer and consumer. Management is in charge of the system, while a producer is the entity that aims to preserve data in the archive preservation solution. Finally, consumers are the individuals/organizations that can access the preservation system to retrieve information.

Regarding the content to archive and preserve, the OAIS model is centered on the information package. The information package comprises the object to preserve, the needed metadata for long time preservation, the access permissions and how the whole data should be interpreted when accessed. Specifically, the OAIS defines three distinct information packages: i) Submission Information Package (SIP); ii) Archival Information Package (AIP) and iii) Dissemination Information Package (DIP) [27]. The SIP represents the source information which is inserted into the archival system by the producer entity. The AIP is the information that is actually archived, complemented with the metadata needed for a proper preservation and future accesses. The DIP represents the information provided to a consumer request. Its format and content may adapt to the profile of the consumer. For instance, a content archived under a

given encoding format, e.g. UTF16, may be delivered to consumers in another encoding format, such as, UTF8.

Figure 1 presents the main services and the functional entities of the OAIS reference model. The rectangle-shaped boxes represent the high-level services that need to be provided by an OAIS-oriented preservation solution. As can be seen, a SIP is first processed by the ingestion module. The ingestion procedure of a SIP yields an AIP to be kept in the archival storage and a set of metadata that feeds the data management service. The AIP is the crux of the information preservation system. It comprises the original information to preserve, as well as, the data needed to interpret the information. OAIS recommends four types of metadata: i) descriptive (provided by the user), ii) technical (extracted by specific tools), iii) preservation (data from operations carried out during the preservation process, e.g. checking of file checksums), and iv) structural (defines relationships between files) [27].

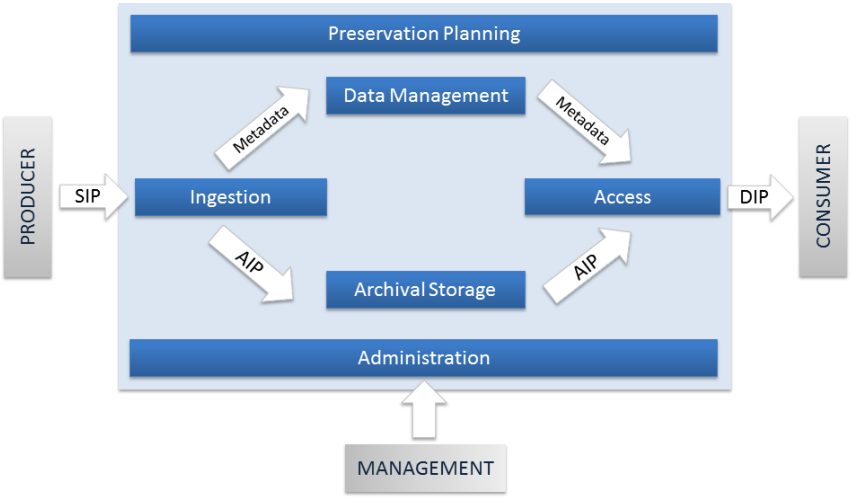


Figure 1 - Open Archival Information System (OAIS) Reference Model (adapted from [26]).

When an access request is received by the archival and preservation system, the access service interacts with the data management and the archival storage services to produce the DIP as requested by the consumer. Preservation planning, shown at the top of Figure 1, is a service transversal to the whole system. It represents the preservation strategy, dealing with external issues such as changes in technology (e.g., obsolescence of a given type of storage) or adjustment in the interaction with producers and consumers. Finally, administration is another transversal service. As the name implies, it deals with administrative issues. Specifically, it

coordinates to fulfil the needs of the other five main services, besides monitoring the performance and managing the maintenance needs of the whole system.

On the matter of interoperability, the OAIS model defines three main categories: i) cooperating; ii) federated; and iii) shared functional areas [27]. Cooperating repositories provide for at least some compatibility at the SIP and DIP level. For instance, a DIP of one repository can be ingested, and thus can act as a SIP into collaborating repositories. Federated repositories aim to provide for integrated services, with a request for data (DIP) possibly filled by two or more distributed repositories. Finally, repositories can share resources needed to support functional activities such as ingestion, storage or data management, to name just a few.

Main Standards and Protocols [9]

Two main protocols for interoperability are the Open Archives Initiative Protocol for Metadata Harvesting (OAI-PMH) [28] and the Search/Retrieval via URL (SRU) [29]. OAI-PMH was created by the Open Archives Initiative for repository interoperability. It consists of six verbs or services invoked over HTTP. The verbs/services are: i) GetRecord; ii) Identify; iii) ListIdentifiers; iv) ListMetadataFormats; v) ListRecords; vi) ListSets [28]. The repositories can act as data providers, exposing structured data through the protocol or as service consumers making requests through the protocol to harvest metadata from the providers. SRU is a XML-based protocol to allow search queries over the internet. It uses the Contextual Query Language (CQL) standard [29], a syntax for representing queries to retrieve data from the repository and exposes them in a structured form through XML.

Metadata standards define the main characteristics needed to describe digital objects, such as, videos, sounds, images, texts and web sites. The main standards are Dublin Core [30], PREservation Metadata Implementation Strategies (PREMIS) [31], Machine Readable Cataloguing (MARC) [32], Encoded Archival Description (EAD) [33], Metadata Encoding and Transmission Standard (METS) [34] and the Metadata Object Description Schema (MODS) [34].

The Dublin Core standard was created in 1995 and is maintained by the Dublin Core Metadata Initiative. It comprises 15 properties with metadata vocabularies and technical specifications, which can describe a wide range of resources [30]. PREMIS, MARC, EAD and METS are all XML-based standards. PREMIS, developed by the Online Computer Library Center (OCLC) and Research Libraries Group (RLG), consists of a data dictionary, an XML schema and supporting documentation. MARC was developed by the American Library of

Congress for cataloguing digital objects stored in a repository. METS is a part of MARC for encoding descriptive, administrative and structural metadata about digital objects within a repository. MARC21 is the most recent version, while MARCXML is an extension of MARC21 with additional features for sharing and networked access of bibliographical information [35]. MODS is another MARC21 compatible XML set for descriptive metadata. EAD is a descriptive XML-based standard aimed at describing the hierarchy structure of archival data. It bears some similarity with MARC, although EAD focuses on archives, while MARC is oriented towards bibliographical materials [33].

Others, more specific, standards and protocols considered in this thesis are the Digital Item Declaration Language (DIDL) [36] for video content, the Metadata for Images in XML (MIX) [37] for still images, the Technical Metadata for Text (TextMD) [38] for text, the Advanced Encryption Standard (AES) [39] for encryption, the DOCument Mediated Delivery (DocMD) [40], AudioMD for audio [41], and VideoMD for video [41]. It is important to note that regarding metadata standards, there is no consensus about which ones are the most important. As such, this study presents the ones implemented by each reviewed system.

2.4.2. Comparison of Digital Information Preservation Software

We compared eleven digital information preservation software solutions. The selection criteria included several aspects relevant to identify the most modern, flexible and reliable systems available to date. Next, we present the selection criteria.

The study is restricted to preservation software which are available under an open source license. The relevance to the field of the software solution is another important criterion. The relevance is estimated either by the size of the users community or the size of the contributors community or both. For this study, we selected the systems that have the broader communities supporting them. Complementing the community size, is the dynamicity of the project. Although it is not an exclusion factor for this study, most of surveyed projects are currently active, having released at least one version in the last six months at the time of this writing. There are two exceptions: Archimède [42] and DAITSS [43]. Both were included due to their importance to the digital preservation field and target audience. Another important request for inclusion in this study is the adherence to state of the art digital preservation and metadata

standards. The assessed systems implement the most relevant digital preservation and metadata standards to the field, namely, OAIS, OAI-PMH, SRU, Dublin Core, MARC, PREMIS, MARC and METS.

Open Source Digital Preservation Software Solutions [9]

Table 1 identifies the eleven software solutions surveyed in this study. It lists each solution author and the classification given by authors for their software product. The table also identifies the studied version of the software, released year and the open source license under which it is available. The software solutions are presented in ascended alphabetical order of the project name.

Table 1 - Main characteristics of surveyed software solutions.

Software	Authors classification	Latest Version/ Release Year	Licensing	Author(s)
Archimède [42]	Institutional Repository	2.0 2006	GPLv2	Bibliothèque de l'Université Laval, France
Archivematica [44]	Digital Preservation System	1.6 2016	AGPLv3	Artefactual, Canada
DAITSS [43]	Digital Preservation Repository System	2.4.20 2014	GPLv3	Florida Center for Library Automation, Florida, USA
DSpace [45]	Institutional Repository	6.0 2016	BSD 3-Clause [46]	DuraSpace, Oregon, USA
EPrints [47]	Institutional Repository	3.4 2016	GPLv3	University of Southampton, UK
Fedora [48]	Digital Repository	4.6.0 2016	Apache 2	Fedora Community; DuraSpace Oregon, USA
Greenstone [49]	Digital Repository	3.0.7 2016	GPLv2	New Zealand Digital Library Project University of Waikato New Zealand
Invenio [50]	Digital Library / Document Repository	3.0 2016	GPLv2	CERN; DESY; EPFL; FNAL; SLAC
LOCKSS [51]	Library-led Digital Preservation System	1.70.2-1 2016	BSD	Stanford University California, USA
RODA [52]	Digital Repository	1.2.0 2016	LGPLv3	Keep Solutions; University of Minho, Portugal
Xena [53]	Digital Preservation Software	6.1.0 2016	GPLv3	The National Archives of Australia

Main Features [9]

We identify and describe the main features of the surveyed systems. The main identified features are as follows: i) digital preservation strategies; ii) authorization/authentication; iii) search capabilities; iv) previews; v) reporting capabilities; vi) support for multilingual; and vii) dynamism of the community of developers.

Digital preservation strategies focus on the strategies made available by each system to ensure long-term access, integrity and authenticity of stored data. Authorization and authentication features assess the existence of access control mechanisms and the ability to track users action over data. Advanced search focuses on the capability of the software solution to allow for searches over stored data, letting users specify filters and properties of data. Operating System (OS) Support gives the information about the three main desktop operating systems: Linux, Windows and MacOS. Previews assess the ability of the preservation software to generate previews, thumbnails and other small excerpts of stored digital objects, saving users from the need to download full packages just to peek at their contents. Reporting evaluates the capability to produce simple/detailed reports with the preserved information. Multilingual assesses support for interaction in the user native idiom. This feature is relevant for data dissemination purposes. Moreover, even if the information itself may solely exist in one language, it is still important that the repository software can support multiple idioms. Finally, community of developers focuses on whether there is a vast and dynamic set of developers involved with the software. This is especially important for open source software, as it may define the difference between failure and success. Table 2 lists the main features of the surveyed solutions. Besides the features presented in the table, all the surveyed solutions support authentication/authorization and allow for advanced search.

Table 2 - Main features of the surveyed solutions.

Software	Digital Preservation Strategies	OS Support	Previews/Reporting/Multilingual/ Community of Developers
Archimède [42]	Encapsulation	Linux, Windows, MacOS	- / yes / yes / -
Archivematica [44]	Migration, Emulation	Linux (Ubuntu)	- / - / - / -
DAITSS [43]	Refreshment Migration	Linux, Windows, MacOS	- / - / - / -
DSpace [45]	Encapsulation Federation	Linux, Windows, MacOS	- / yes / yes / yes
EPrints [47]	Encapsulation	Linux, Windows, MacOS	yes / yes / yes / -

Software	Digital Preservation Strategies	OS Support	Previews/Reporting/Multilingual/ Community of Developers
Fedora [48]	Encapsulation Federation	Linux, Windows, MacOS	- / - / - / yes
Greenstone [49]	Encapsulation	Linux, Windows, MacOS	yes / yes / yes / -
Invenio [50]	Encapsulation	Linux	yes / yes / yes / yes
LOCKSS [51]	Encapsulation Federation, Migration	Linux	- / - / - / -
RODA [52]	Migration	Linux, Windows, MacOS	yes / yes / yes / yes
Xena [53]	Migration	Linux, Windows, MacOS	- / - / - / -

Preservation Standards / Metadata Standards Support [9]

This Section identifies the preservation and metadata standards supported by each software. The preservation standards are OAIS [27], OAI-PMH [28] (versions 1 and 2) and SRU [29]. The identified metadata standards are Dublin Core [30], MARC / MARC21 / MARCXML [54, 35], EAD [33], PREMIS [55], METS [56], MODS [57], DIDL [58], MIX [37], AES [39], DocMD [40] and TextMD [59]. Table 3 shows the support for these standards given by each software. Specifically, in Table 3, a check sign means that the standard is supported. The inexistence of a check sign means that no indication of the standard support was found during this study. Nonetheless, we cannot assert that any of the standards are unsupported, as many of the systems studied have flexible architectures, therefore supporting 3rd parties plugins, capable of enabling those standards support. Note that Xena does not support any of the covered standards. Additionally, since only DAITSS natively supports the MIX, AES, TextMD, DocMD, audio and video formats, these standards are not included in Table 3 to preserve space. For the same reason, the column MARC includes the standards MARC, MARC21 and MARCXML.

Table 3 - Standards and protocols supported by each of the surveyed software solutions.

Software	OAI	OAI-PMH v2	SRU	Dublin Core	MARC	PREMIS	METS	MODS	DIDL
Archimède [42]	✔	✔		✔					
Archivematica [44]	✔			✔		✔	✔		
DAITSS [43]	✔					✔	✔		
DSpace [45]	✔	✔	✔	✔	✔			✔	
EPrints [47]	✔		✔	✔			✔	✔	✔

Software	OAI	OAI-PMH v2	SRU	Dublin Core	MARC	PREMIS	METS	MODS	DIDL
Fedora [48]	✔		✔	✔		✔	✔		
Greenstone [49]	✔		✔	✔	✔		✔		
Invenio [50]	✔		✔	✔	✔				
LOCKSS [51]				✔					
RODA [52]	✔	✔		✔		✔	✔		
Xena [53]									

Table 4 - Supported file formats (part 1).

Software	Image	Audio	Video
Archimède [42]			
Archivematica [44]	BMP, GIF, JPG, JP2, PCT, PNG, PSD, TIFF, TGA, RAW	AC3, AIFF, MP3, WAV, WMA	AVI, FLV, MOV, MPEG-1, MPEG-2, MPEG-4, SWF, WMV
DAITSS [43]	✔	✔	✔
DSpace [45]	✔	✔	✔
EPrints [47]	JPG, JPEG, PNG, GIF, TIFF, AIFF	MP3	MPEG
Fedora [48]	✔	✔	✔
Greenstone [49]			
Invenio [50]	✔	✔	✔
LOCKSS [51]	✔	✔	✔
RODA [52]	TIFF, TIF, JPG, JPEG, PNG, BMP, GIF, ICO, XPM, TGA	WAV, MP3, MP4, FLAC, AIF, AIFF, OGG, WMA	MPG, MPEG, VOB, MPV2, MP2V, MP4, AVI, WMV, MOV, QT
Xena [53]	BMP, GIF, PSD, PCX, RAS, XBM, XPM, PNG, TIFF	MP3, WAV, AIFF, OGG, FLAC	-

Supported File Formats [9]

We enumerate the file formats recognized by each system for data ingestion. Each system is capable of ingesting files of any type and storing them. However, only recognized file types allow the systems to perform operations such as migration, normalization or other preservation operations specific to each file type on the ingestion phase. Table 4 and Table 5 group file formats in eight sets: Image, Audio, Video (Table 4) and Text, Applications, Vector, Email, and Other (Table 5). These tables identify the file types recognized by each of the surveyed solutions. A check sign means that the file type is supported. On the contrary, the inexistence of a check sign means that no indication of support for the file formats was found during this study.

Table 5 - Supported file formats (part 2).

Software	Text	Applications	Vector	Email	Other
Archimède [42]	XML, HTML, PDF, RTF, TXT	MS Word, MS PowerPoint, OpenOffice			JavaBeans
Archivematica [44]	TXT, PDF, RTF	DOCX, XLSX, PPTX, PPT, XLS, DOC, WPD	AI, EPS, SVG	PST, MailDir	X3F, 3FR, ARW, CR2, CRW, DCR, DNG, ERF, KDC, MRW, NEF, ORF, PEF, RAF
DAITSS [43]	✔	✔			
DSpace [45]	✔	✔			
EPrints [47]	PDF, HTML, ASCII, CSV, XML	MS Word			ZIP, GZ
Fedora [48]	✔				
Greenstone [49]	TXT, PDF, HTML	MS Word, MS PowerPoint, MS Excel			
Invenio [50]	✔	✔			
LOCKSS [51]	✔	✔			
RODA [52]	PDF, XML	DOC, DOCX, ODT, RTF, TXT, PPT, PPTX, ODP, XLS, XLSX, ODS	AI, CDR, DWG	EML, MSG	BIN
Xena [53]	PDF, SQL, HTML, TXT	MS Office, MS Project MPP, Open Office, WordPerfect		PST	MBX, ZIP, GZIP, TAR, GZ, JAR, WAR

2.4.3. Most Relevant Open Source Digital Preservation Solutions

Five of the surveyed software solutions stand out as the most relevant ones for institutions looking to implement digital repositories. These solutions are: RODA, DSpace, Fedora, Greenstone and EPrints. These solutions are feature rich and have a broad community of users. They are, in most cases, the first option for digital library management and long-term preservation. All of these solutions implement the OAIS reference model with the exception of Greenstone. Still, Greenstone is included in this chapter due to its wide use by UNESCO countries [60].

We review each of the five projects, explaining their main purposes, providing some historical background and highlighting their main features. We also briefly reference the technologies used by each system. The information was collected on the projects official

websites, scientific publications and on official documentations, such as promotional leaflets, brochures, etc.

RODA – Repository of Authentic Digital Objects [9]

The Repository of Authentic Digital Objects (RODA) [61] is a digital repository licensed under the open source LGPLv3 license. It follows and provides functionality of the OAIS model. It is developed by Keep Solutions, a Portuguese company, in cooperation with University of Minho, and its research community. RODA targets not only academic institutions that wish to build their own digital repository, but also museums, libraries or any other institution with similar needs [62].

The built-in preservation strategy of RODA is migration. It features all the steps this strategy encompasses, i.e., normalization, conversion, replication and preservation. It also supports other strategies, such as emulation or encapsulation through its extendibility and configuration capabilities. RODA supports several main standards and is capable of ingesting information, normalize objects for data preservation and allow to browse the repository. It also provides advanced search over the entire repository contents, previews of stored digital objects for text based objects, images, audio or video files and downloading the preserved information [52].

RODA has the following main features [52]:

- It enforces authenticity through PREMIS preservation metadata.
- It provides for access control and permission management, with flexible configuration and tracking of user actions.
- It conforms to open standards, supporting OAIS, METS, EAD, PREMIS and MIX.
- It is vendor independent, being able to use the most convenient hardware and operating system. This is also important for digital preservation purposes as it does not restrict content to any proprietary format or technology.
- It is scalable through a service-oriented architecture that supports load balancing with several servers.
- It has embedded preservation actions such as format conversions, normalization steps during ingest, checksum verifications, reporting actions, notification events and emails. All of this is controlled by a task scheduler. This allows for specific actions to be triggered by a set of administrator defined rules.

- It supports multiple formats, allowing for normalization and migration of formats on the ingested information.
- It has extensibility capabilities and provides support for 3rd party systems integration through the exposure of functionality via web services. This allows other systems to easily communicate with RODA and let them add more functionality to the system.
- It has advanced ingest workflow, as stated in the technical requirements document [52]. RODA supports the ingest of new digital material, as well as, associated metadata in four distinct ways: i) online submission (self-archiving); ii) offline submission using a client application called "RODA-in" (offline self-archiving); iii) batch import by depositing SIPs via FTP or SMB/CIFS; and iv) integration with 3rd party document management software via invocation of SOAP Services or client API.
- It supports multiple idioms, an important feature for a digital preservation repository aimed for broad audience.

RODA is built on top of a plethora of technologies. The main ones are Java (programming language and implementation), Google Web Toolkit (user interface), OpenLDAP [63] (Authentication), Fedora Linux (Data Services), ImageMagick [64], OpenOffice, GhostScript [65], JOD Converter [66], MEncoder [67], SoundConverter [68] and gStreamer [69] (migration and conversion), JHove/JHove2 [70] and DROID (Digital Record Object Identification) for automatic validation and characterization [71].

DSpace [9]

DSpace is a repository software built with data preservation in mind [72], and licensed under the BSD open source license. It enables easy and open access to all types of digital content including text, images, video and data sets. DSpace recognizes and manages a large number of file format and MIME types, such as the most common formats PDF, Word, JPEG, MPEG and TIFF files. Although out-of-the-box DSpace only recognizes common file formats, other formats can be managed through a simple file format registry. This way, it is possible to register any unrecognized format, so that it can be identified in the future [45].

The web applications provide interfaces for administration, deposit, ingest, search, and access to assets stored and maintained on a file system or on similar storage system. This way, it is highly customizable and configurable through a web-based interface [73]. The system provides for two main preservation strategies, encapsulation and federation. The metadata, including access and configuration information, is stored in a relational database. Under the

federation strategy, DSpace acts as a peer repository in a decentralized network of repositories. DSpace is cross platform, supporting Linux, MacOS and Windows.

The benefits from a large community of developers and contributors who keep evolving and improving its features make DSpace one of the most used solution for libraries, educational institutions, governments, non-profit and even commercial organizations. Originally, the project, developed by MIT Libraries and Hewlett-Packard (HP) Labs, had its first release in 2002. The community is currently under the control of DuraSpace, an independent not-for-profit organization formed in 2009 by merging Fedora Commons and DSpace. Since then, DuraSpace invests in open technologies that promote durable, persistent access to digital data. It collaborates with academic, scientific, cultural, and technology communities by supporting projects and creating services to help the preservation of the collective digital heritage [45].

The main features of DSpace are:

- Full stack web application consisting of a database, storage manager and a front end.
- Built-in authentication/authorization system that can be integrated with 3rd party authentication mechanisms.
- Permissions system with access control over assets.
- Configurable database engine (PostgreSQL or Oracle)
- Configurable file storage, either local file system or cloud-based service.
- Specific data model architecture with configurable workflows.
- Configurable metadata schemas through the mapping of specification of new fields over the default Dublin Core structure.
- Configurable browse and search, as well as, full text search capabilities.
- Open standards compatibility.
- Data integrity check and reinforcement.
- Full import/export of the repository feature for disaster recovery.
- Multilingual support.

DSpace provides for various standards, namely: OAIS, OAI-PMH, Dublin Core, OAI-ORE (Open Archives Initiative Object Reuse and Exchange), SWORD (Simple Web-service Offering Repository Deposit), WebDAV (Web-based Distributed Authoring and Versioning), OpenSearch, OpenURL, RSS (Really Simple Syndication), and Atom.

The main technologies used by DSpace are Java (programming language and implementation), Angular 2 (user interface), LDAP and Shibboleth [74] (3rd party authentication), and the database engines PostgreSQL and Oracle.

Fedora [9]

Fedora is a repository software suite that provides management and dissemination of digital content. It is licensed under the Apache 2 open source license. It targets digital libraries and archives. Fedora features in the list of the most widely used repository software. It has an established user base of academic institutions, universities, libraries and government agencies. The software is conceived for both data access and preservation. Fedora is able to provide specialized access to very large and complex digital collections of historic and cultural materials, as well as, scientific data [48].

The project was born in 1997 at the Cornell University in Ithaca, New York under the name of Flexible Extensible Digital Object Repository Architecture. It later adopted the Fedora acronym as its official designation after being referred by that name in a scientific article [75]. Although the project was clearly born before the Fedora Linux distribution by Red Hat, some legal issues were raised about the software designation. However, both parties agreed to maintain the Fedora name associated to their projects, as long as there was a clear association with the digital repositories systems in one case and the open source computer operating system in the other. The Fedora Repository is supported by a large community of developers, led by the Fedora Leadership Group and is under the stewardship of DuraSpace not-for-profit organization [48].

Fedora has a robust architecture that enables it to handle collections with millions of objects [48]. It ensures the longevity and durability of data by storing all information in files without any software dependency and allowing the rebuilding of the complete repository at any time. It also implements semantic web capabilities by resorting to the SPARQL query language to query repositories [76]. It supports the definition of complex relations between the digital objects stored. In the latest release, federation capabilities were also added, allowing the software to act as a peer repository in a distributed network of digital preservation repositories [77].

The main features of the digital preservation software Fedora are:

- Standard based services via RESTful APIs in accordance with current web standards [78].

- Authentication, authorization and access control through integration with 3rd party standards compliant authentication frameworks.
- Pluggable security authorization modules: role-based, XACML or Web Access Control.
- Storing, preserving and access capabilities to any type of file of any size.
- Advanced storage options for files and metadata with customizable file systems and databases.
- High scalability architecture, able to handle millions of files and metadata records.
- Semantic web application with native linked data support through the SPARQL for RDF protocol.
- Extensibility through plug-in modules capable of providing OAI-PMH dissemination or SWORD deposit.
- Advanced search, indexing and discovery through 3rd party applications.
- Preservation services such as fixity checking, audit trail, versioning, backup and restore.
- Batch operations capabilities over a single repository to achieve better consistency and performance.
- Adherence to open formats.
- Easy deployment through a WAR file (Web application ARchive).

The major standards supported by Fedora are OAIS, OAI-PMH, Dublin Core, METS, and PREMIS. Regarding technologies, Fedora resorts to Java (programming language and implementation), LDAP and Shibboleth [74] (3rd party authentication), and MySQL and PostgreSQL as database engines.

Greenstone [9]

Greenstone is a software suite for building and distributing digital library collections, and is licensed under GNU GPLv2. It is aimed for educational institutions, universities, libraries, public service institutions and UNESCO partner communities which wish to build their own digital libraries, especially in developing countries. Greenstone provides a way of collecting and organizing digital collections, publish them on the web or act as a standalone application and store the information in any storage medium, either hard drives or any removable media. In spite of being a digital repository software, Greenstone does not follow the OAIS reference model or implement explicitly any data preservation strategy. Despite not being a digital preservation repository per se, Greenstone is focused in this study, since it implements some

key features for data dissemination. This software is also relevant to the digital repository target audiences [49].

Greenstone is produced by the New Zealand Digital Library Project at the University of Waikato. It is developed and distributed in cooperation with UNESCO and the Human Info NGO in Belgium. Greenstone can be run as a web server, with full search capabilities and metadata-driven digital resources. Alternatively, it can be run on a non-networked environment as a standalone application, being installed on a computer or operating from removable media. The software also has interoperability capabilities with other systems through the implementation of contemporary standards like OAI-PMH or METS for metadata. Due to its support of these protocols, Greenstone is capable of interchanging information with systems like DSpace. This allows it to export/import from DSpace any collection available within these formats [49].

The main features of Greenstone are as follows:

- Two distinct versions of the software for installation: a standalone application and a web server.
- Server version for the Android platform with digital library operating self-contained on an Android device. This might be particularly interesting for anyone who wishes to make a library available without having to assemble and configure a conventional web server.
- Authentication/authorization service through JAAS (Java Authentication and Authorization Service).
- Interoperability capabilities through the OAI-PHM and METS standards for data interchange with other digital repository systems.
- Two separate user interfaces: A reader web-based interface and a librarian Java-based interface.
- Built-in metadata management.
- Built-in advanced search with customization possibilities.
- Librarian interface that can be configured to manage remote Greenstone installations.
- Multilingual support.

Greenstone supports the standards OAI-PMH, METS, Dublin Core (qualified and unqualified), and Bibliographic records as specified by RFC 1807 [79]. It also supports AGLS and NZGLS. AGLS (Australian Government Locator Service) is an extension to the Dublin Core [80], to improve the visibility, manageability and interoperability of governmental online

services, while NZGLS (New Zealand Government Locator Service) is based on AGLS. It is a metadata standard implemented and maintained by the Archives of New Zealand, with the goal to classify and categorize New Zealand government agency information and services [81].

Technology-wise, Greenstone relies mostly on Java for implementation and user interface. The authentication and authorization is performed through JAAS (Java Authentication and Authorization Service).

EPrints [9]

EPrints [47] is a software package for building open access repositories, licensed under GNU GPLv3. EPrints is primarily used for institutional repositories and scientific journals as it provides open access, i.e., immediate online access to the full text of research articles within the repository. Its flexible configuration and web-based nature allow it to be also used as a repository for images, research data or audio archives. EPrints provides a set of ingest, preservation, dissemination and reporting services for institutions open access needs [82].

EPrints was created in 2000 as a result of the 1999 Santa Fé meeting, where the discussion for the creation of a communication protocol for digital repositories interoperability gave birth to the OAI-PMH protocol [82]. EPrints provides a stable yet flexible infrastructure on which institutions have been building their open access digital repositories. Examples includes governmental departments, universities, hospitals and non-profit organizations [83]. Through EPrints Services – a not-for-profit commercial services organization – academic and research institutions can benefit from training, as well as, aid on the development and hosting of repositories. The project has been developed at the University of Southampton, School of Electronics and Computer Science. It encompasses developers, librarians and users.

The software provides a web-based application, as well as, a command-line interface based on the LAMP architecture. However, instead of the PHP language that is standard with LAMP, EPrints resorts to the PERL programming language. It uses a plugin architecture for importing and exporting data, creating representation of objects appropriate for indexation of search engine and for user interface widgets. The plugins are developed in the PERL language.

EPrints provides for the following main features [47]:

- A full stack web application consisting of a database, storage manager and a customizable front-end web interface.
- An authentication and authorization system for documents and published materials.

- Host, management, share and collaboration on teaching and learning materials all in one place.
- Lightweight metadata collection with METS and MODS export plugins.
- Support the ingestion of practically any type of file.
- Tag mechanism and collection-based methods to classify digital materials.
- Advanced search with autocomplete features.
- Multiple idiom support.

The following standards are available within EPrints: OAIS, OAI-PMH, SWORD, Dublin Core, METS, MODS (Metadata Object Description Schema) and DIDL (Digital Item Declaration Language) [36]. Regarding software, EPrints is based on PERL (programming language and implementation), HTML/CSS (user interface) and uses MySQL as its backend database server.

Other Digital Repositories Solutions [9]

The digital repository solutions Archimède [84], Archivematica [84], DAITSS, Invenio [50], LOCKSS [51] and Xena [53] are also valid choices to deal with the digital preservations needs of an institution. However, only LOCKSS and Invenio have a good level of acceptance by their target audiences. This may be due to a lack of features or to a lack of standards compliance of the other solutions. Some solutions address the challenge of preserving data from a standalone application approach. This is the case for Xena, a solution that may be suitable in some cases, but does not seem reliable in the long run. Other solutions, like LOCKSS, are trying to break ground by implementing new preservation strategies, federation, which may also be a drawback for users looking for a system with given proofs and reliability. Another limitation in most of these software solutions is that they were built as digital libraries management software, lacking features of general purpose digital repositories.

Besides the aforementioned software, there are still another six systems that were summarily analyzed for this study: OpenBiblio [85], IntraLibrary [86], SobekCM [87], Opus [88], MyCoRe [89] and Koha [85]. None of them features in the software comparison as all of them fall out of the scope of this survey either for lack of features, lack of information or for not being complete digital preservation repository solutions. Nonetheless, even though they do not qualify for a deeper analysis, we found important to reference them for completeness of this study.

3. Interoperability Between Repositories

As stated before, the possibility of communication and interchange of preserved information between repositories, besides being one of the most recent practices in the digital preservation field, is a very desirable feature in any repository software. After analyzing the latest open source software for digital preservation repositories, to the best of our knowledge, we reached the conclusion that interoperability features could be further developed in most of the solutions studied.

From the various repository software analyzed, we chose RODA to host the implementation of an interoperability mechanism. Our choice for RODA relied, both on the adequate technical characteristics of the software, and the wide users community. We proceeded by researching the available interoperability protocols. The CMIS protocol, supported by the Apache Chemistry project [90], posed as our preferred choice for being a mature protocol, now in its version 1.1 and also for having a library written in java, the language used to develop RODA.

In this chapter, we start by presenting an overview of the architecture for RODA-OCS prototype implementation in Section 3.1. In Section 3.2, we explain the usage of RODA, the chosen digital preservation repository, and the features to be implemented. We then introduce the OpenCMIS server in Section 3.3, and its role in the system. In the last section, we present the CMIS Workbench Testing Tool, its main features and the purposes for which it was used during the system implementation.

3.1. Architecture

To implement interoperability between digital preservation repositories we selected **RODA** as the open source software to be used. Figure 2 identifies the RODA-OCS prototype architecture.

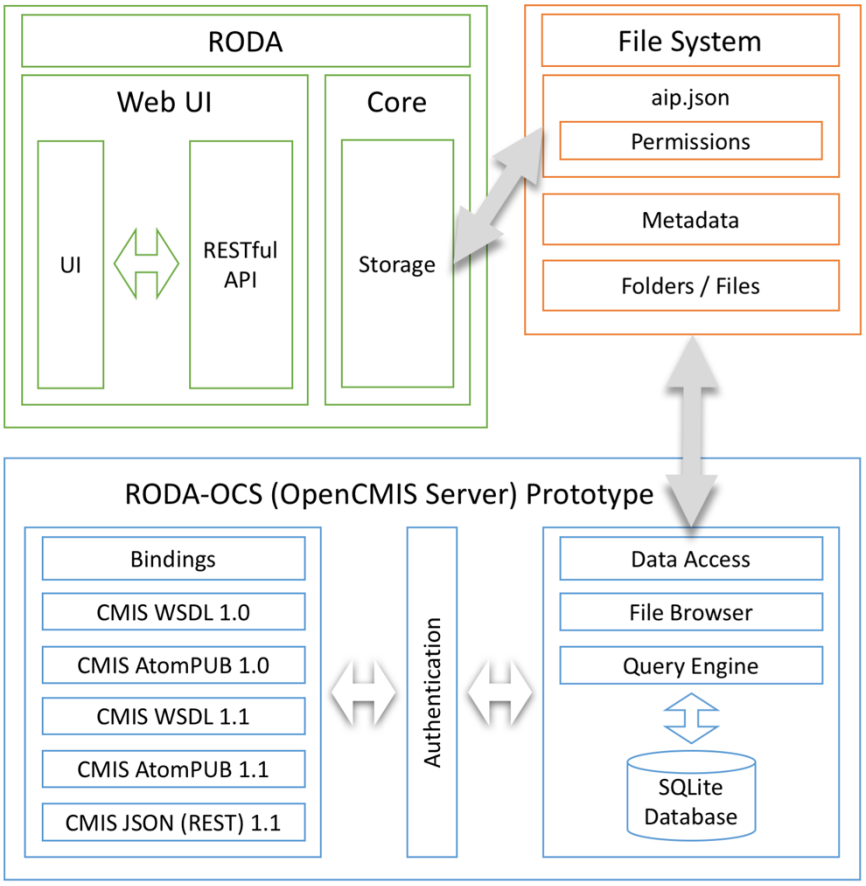


Figure 2 – Digital Preservation Repository Interoperability Architecture

We identified three main components in the solution: i) RODA, ii) File System and iii) RODA-OCS (OpenCMIS Server) Prototype. Although each component possesses more functionality, we represented in this diagram only the ones relevant for this thesis.

RODA – The repository software provides a **web based user interface** based in Java servlet technology[1] which communicates with the server backend through the implementation of a **RESTful API** [91]. This RESTful API is responsible to establish a bridge between any action performed by the user in the user interface and the repository core functionality. In RODA core we have a **storage** module. This module will handle all the interaction with the

[1] Java Servlets implementation: https://bitbucket.org/meicmdigitalpreservation/java-servlet25-example

repository configured storage mean, in this case, the **File System**. The storage module is also responsible for creating, accessing and managing the contents of an AIP. The AIP will hold important information for our system, such as a list of **permissions** over the AIP objects (**folders and files**) and the location of metadata files, if there are any, referring to the AIP objects. This information is stored in the repository in an "**aip.json**" file, a JSON-based format. There is only one "aip.json" file to represent each package [52].

File System – The file system is the storage medium where the repository contents are kept. RODA allows different choices and configurations for the storage of a repository, both local and remote. The one used in this implementation is the local file system meaning all the data stored in the repository will be written to the server local hard drive. Since RODA is implemented in Java, the Java Virtual Machine provides us an abstraction layer when accessing the file system. The Java Virtual Machine acts as a middleware between Java code and the different operating systems hiding each system specificities. This not only enables us to write multiplatform code but also to handle files and folders the same way over different operating systems. As stated before, the file system stores the repository AIPs. Each AIP contains all the data, the "Submission Information Package", the files, folders and descriptive metadata uploaded to the repository by the user along with other metadata associated to the package which is generated during the process [52].

RODA-OCS (OpenCMIS Server) Prototype – The RODA OpenCMIS Server prototype is the system component responsible for accessing our RODA repository contents and expose them through the implementation of the Content Management Interoperability Services protocol, or CMIS, for short. The server component presents various modules: i) bindings, ii) authentication and iii) data access. Each component implements the protocol specifications [91].

Bindings: This module implements the server interfaces proposed by the specification, either in the 1.0 version or the 1.1 revision. The version 1.0 bindings are WSDL (Web Service Definition Language) / SOAP (Simple Object Access Protocol) and AtomPUB. Version 1.1 also implements these two interfaces along with the REST (Representational State Transfer) / JSON (JavaScript Object Notation) interface, later added to the specification to provide a more modern and lightweight approach to the protocol.

Authentication: In order to connect to the server and access its contents, the CMIS protocol specifies a login mechanism for authenticated access. This mechanism is based

on a configuration file holding the different users credentials and access levels. The protocol suggests two levels of access to the repository: i) read only and ii) read and write access. Since the contents being exposed through the server belong to the information packages provided by each user, the access level configured is always "read" access. This ensures no contents are changed when interacting with the CMIS server.

Data Access: Once a user is authenticated to the CMIS server, the **Data Access** module provides a list of contents available through its **File Browser** submodule. The file browser submodule is responsible for reading the permissions defined in the "aip.json" file, exposing only the allowed folders and files.

There is also the **Query Engine** submodule, a module which allows any client application to perform searches over the CMIS repository contents. A **SQLite** [93] **database** was used to accomplish this task, since the protocol specifies the compatibility with the SQL-92 SELECT specification, namely the SELECT, FROM, WHERE, GROUP BY, HAVING and ORDER BY clauses. The SQLite is used to register all the files and folders stored in the server, the folder structure and their associated metadata. The query engine will then use the information stored in the SQLite database to perform queries over the repository contents. This technique allows full compatibility with the SQL-92 specification by taking advantage of the existing implementation in the SQLite database and provides a very fast and efficient search mechanism over data.

3.2. RODA – Repository of Authentic Digital Objects

RODA is an OAIS compliant repository. Figure 3 identifies the modules defined by the reference model, its interactions and information exchanged between them.

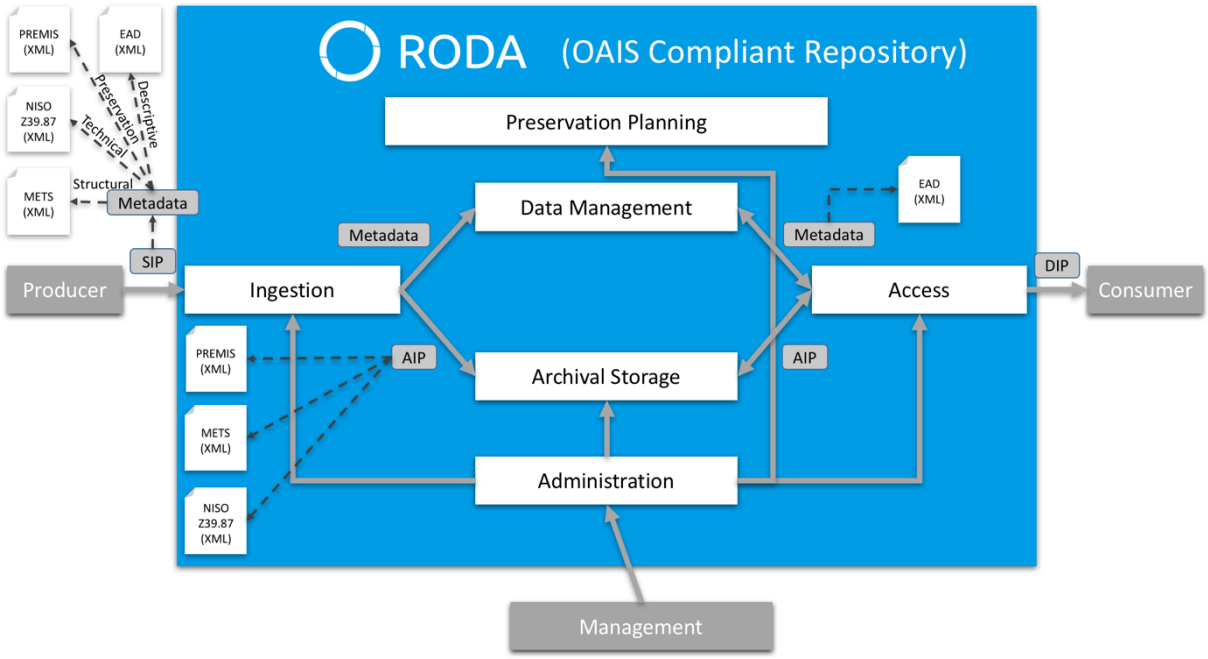


Figure 3 – RODA Repository Architecture (adapted from [92])

There are three possible user roles when interacting with the repository: i) Producer, ii) Consumers and iii) Managers.

Producers - The producers are the users or client systems that provide information to be preserved. They are responsible for loading information into the repository [93].

Consumers - The users that request the information stored in the repository. Upon request, the digitally preserved information is delivered to the consumer. The DIP, or Distribution Information Package, consists of a package with the original data, the descriptive metadata that gives it context, the metadata resulting from the preservation actions performed over the original data and the resulting processed data, if there is any.

Managers - The users who perform the administrative tasks over the repository. The administrative tasks include user management, SIPs approval, risks planning among others.

RODA implements all the functional modules specified by the OAIS reference model. The repository is divided in six modules: i) Ingestion, ii) Data Management, iii) Archival Storage, iv) Access, v) Preservation Planning and vi) Administration [92].

Ingestion - The Ingestion module is the repository entry point. This is the module responsible for receiving a SIP, check for the information inside it and separate the data from the metadata. The data is analyzed, compatible file formats are recognised, technical metadata is extracted and any preservation actions configured are performed. The data is then sent to the Archival Storage module. The extracted technical metadata along with any descriptive metadata file encapsulated within the SIP and preservation metadata generated are submitted to the Data Management module.

Data Management - The Data Management module manages all the metadata about the repository stored SIPs. This module processes the generated and extracted metadata during the ingestion process. It allows the creation and association of new descriptive metadata to an existing SIP. It also allows the edition of any SIPs associated metadata.

Archival Storage - This module interacts with the repository configured storage. RODA allows two types of storage: i) remote storage, resourcing to a Fedora repository and ii) local storage, where data is kept in a folder in the local file system. The stored information inside the repository is called an AIP. It is this module is responsible to perform all the read and write operations over the AIP files, as well as register those alterations in the AIP "aip.json" file.

Access - The Access module is responsible for preparing the preserved data for delivery to the consumers every time a DIP is requested. The DIP generation consists in gathering all the AIP data, both the original data and the data generated from any preservation action, encapsulate it with all the existing metadata, both descriptive and preservation metadata, create a package and deliver it to the user.

Preservation Planning – This module holds a set of rules defined by the digital archivist to be applied over the repository data on different events. The events span from the ingestion of a SIP, periodic fixity checks to the detection of storage failure among others that can be created. It is through these rules that the repository knows what measures must be taken to respond to an event and ensure data integrity.

Administration – The Administration module is responsible for managing all the repository configurations, user management, repository permissions and groups, the Ingestion

process options, automatic and manual steps, the Archival Storage to be used, either local or remote, the Preservation Planning rules and the DIP creation process by the Access module.

The interoperability features implemented in RODA-OCS prototype focused on retrieving data stored in the repository and expose it through the implementation of the CMIS protocol, explained in detail in Chapter 0.

3.3. CMIS – Content Management Interoperability Services - Protocol

The CMIS or Content Management Interoperability Services protocol is developed by the OASIS, a non-profit consortium, with the aim of standardizing the interoperability amongst Enterprise Content Management systems through the use of modern web services interfaces. The protocol allows any document system or repository to share its contents with other systems in vendor-neutral formats over the Internet by resourcing to Web Services and Web 2.0 interfaces. The protocol defines two distinct system roles in its specification, a client and a server, meaning a system can act as a supplier of information, as a consumer, or both [94]. This fact is of particular interest for the digital preservation field as it allows a repository to be able to serve its contents to other repositories but also to harvest contents from other repositories later storing them for redistribution. As mentioned before, to a decentralized network of repositories architecture, with content sharing capabilities between them, we call **Federation**.

The protocol defines a domain model, an abstraction layer and binding mechanisms which allow applications to manipulate content stored in the repository. The domain model defines four types of objects: i) documents, ii) folders, iii) relationships and iv) policies. The document object represents a file within the repository, as the folder object represent folders. The relationship objects are designed to represent relations between objects allowing objects to be contextualized and implement ad-hoc searching mechanisms without resorting to the use of databases. The policy objects are responsible for controlling the access to document and folder objects and the implementation of Access Control Lists [91].

The protocol specifies three possible ways of binding to a CMIS enabled repository: i) web services, ii) AtomPub and iii) REST. The Web Services interface is a WSDL / web service binding implemented under the specification of SOAP. The AtomPub binding is implemented following the Atom Publishing protocol specification. The third binding option is the

implementation of a RESTful architecture where the communication is made through JSON, also referred to as *Browser binding* by the OASIS Technical Committee [97] in its CMIS version 1.1 specification.

The version 1.0 specification of the CMIS protocol was approved on May 1, 2010 but due to limitations was updated to its current version, version 1.1 on December 12, 2012. The Web Services and AtomPub bindings were implemented since version 1.0, but it was not until version 1.1 release of the protocol that the REST binding was made available.

There were several reasons for choosing the REST protocol for the implementation made in this thesis. The first reason lies in the fact that this protocol is an open standard, its implementation is open source which is suitable given the academic nature of this work. Also, the Apache Chemistry project [90] also communicates through the REST protocol, besides being a well-documented and mature implementation, which motivated this choice. The Apache Chemistry project provides an open source implementation of the CMIS protocol in various programming languages such as Python, PHP, .NET, Objective-C or JavaScript besides the aforementioned Java language. The final reason was the protocol characteristics and specification. The implementation aimed to provide a digital preservation repository with new features that were relevant to the digital preservation field, as stated before.

3.4. OpenCMIS Server

Being **Federation** one of the most recent strategies in the digital preservation field [19], it is still rare to find repository software with these capabilities. As such, given the contribution of the CMIS protocol in addressing this issue along with the lack of RODA interoperability features, we felt important to give a contribute in this sense. One possible way to accomplish this endeavour was by implementing an OpenCMIS Server.

The Apache Chemistry OpenCMIS Server Framework provides a means to build a custom CMIS server. The project provides developers with “OpenCMIS Server Development Guide”, a tutorial consisting of a PDF document and the source code for a fully working example of a FileBridge server. The example is written in the Java programming language [95] by Jay Brown, ECM Architect at IBM and Florian Müller, ECM Architect at SAP [96].

In the guide source code, we can find the implementation of a CMIS server based on the Apache Chemistry OpenCMIS Server Framework. The server in the example is fully compliant

with the CMIS protocol in its versions 1.0 and 1.1 specifications, implementing the three binding methods mentioned before, Web Services (WSDL / SOAP), AtomPub and Browser (REST / JSON). The server also implements an authentication mechanism, a very simple example of a query engine, allowing a client application to perform searches over the repository objects and other mechanisms such as server extensions which fall out of the scope of this thesis [96].

The supplied server code[2] provided the basis for the implementation made in RODA doting it with the capabilities of exposing its contents in an authenticated and structured way. In this sense, we provided the repository with a communication interface capable of sharing the server contents with other systems or client applications, enabling it with an interoperability mechanism. Furthermore, we opted by using the CMIS protocol given its open source nature and market acceptance [91]. Lastly, since the chosen open source digital repository technology used was Java, Apache Chemistry OpenCMIS also fulfilled the requisite of the server programming language and technology to use.

3.5. CMIS Workbench

The CMIS Workbench application is a CMIS protocol desktop client supplied by the Apache Chemistry project to help developers implement and test their server implementations. The tool works not only as a repository browser, but also as an interactive testbed for the OpenCMIS client API [97].

While none of the CMIS Workbench features presented in this Chapter were implemented by us, we present and detail the usage of the ones relevant for the development of RODA-OCS prototype. This application allowed us to test RODA-OCS prototype authentication and authorization system, the file browser and folder navigation, the retrieval of stored objects details and metadata and the query engine. The CMIS Workbench application also allowed us to test our prototype compatibility with the CMIS protocol specification during its development through its Test Compatibility Kit.

The application allows us to login to CMIS repositories. The login window presents three possible methods to login to a CMIS repository: i) Basic, ii) Expert and iii) Discover. Figure 4

[2] OpenCMIS Server Implementation: https://bitbucket.org/meicmdigitalpreservation/opencmis-server-example

illustrates the **Basic** login method. We are able to connect to a repository through the three available binding methods specified by the protocol: AtomPub, Web Services (WSDL/SOAP) and Browser (REST/JSON). We need to provide the URL to the repository, choose the binding method, giving a username and a password. The other parameters, although also part of the specification, can be left with their default values. After providing the necessary fields we can load the repositories by clicking the button "Load Repositories". From the list below the button we choose one of the available repositories to connect to and click the "Login" button, as shown by Figure 4.

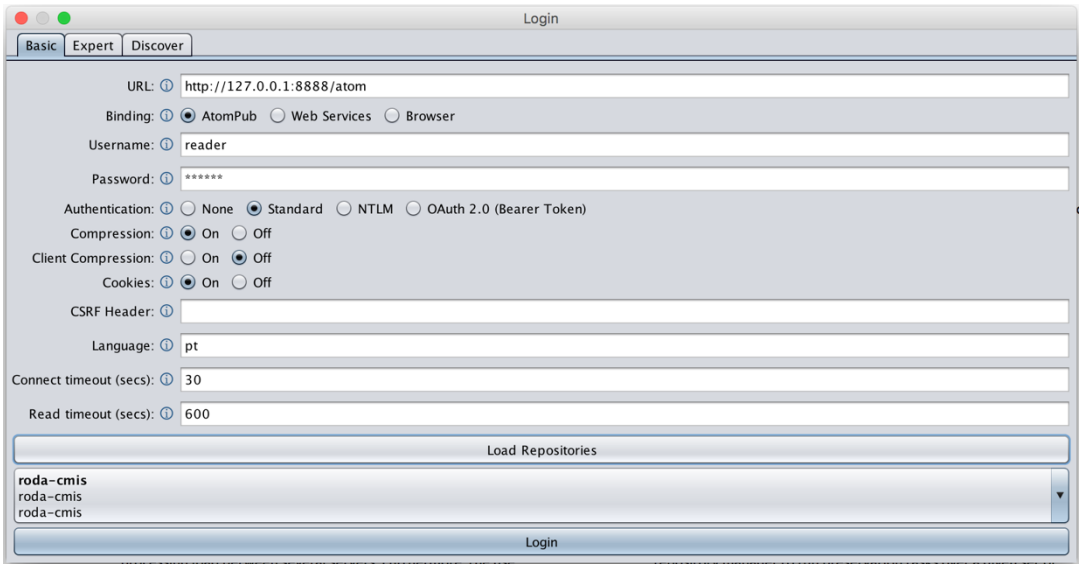


Figure 4 – Repository Login window – Basic authentication

Figure 5 represents the **Expert** login method to a repository. By defining the values for the listed variables, we are able to load the available repositories from a given URL. After providing the correct values for the variables, the connection procedure is the same as specified in Figure 4. This login method may be particularly interesting for users who wish to save the correct login information in a text file for further use.

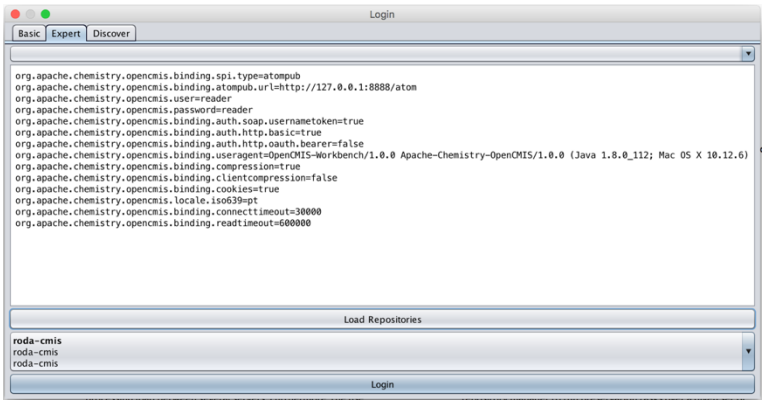


Figure 5 – Repository Login window – Expert authentication

In Figure 6 we present the last login method, the **Discover** method. By providing an URL, the application will try to infer the needed information to connect to a repository. Once the information is retrieved, the user chooses an entry from the list to connect to. The connection procedure is also the same as specified in Figure 4.

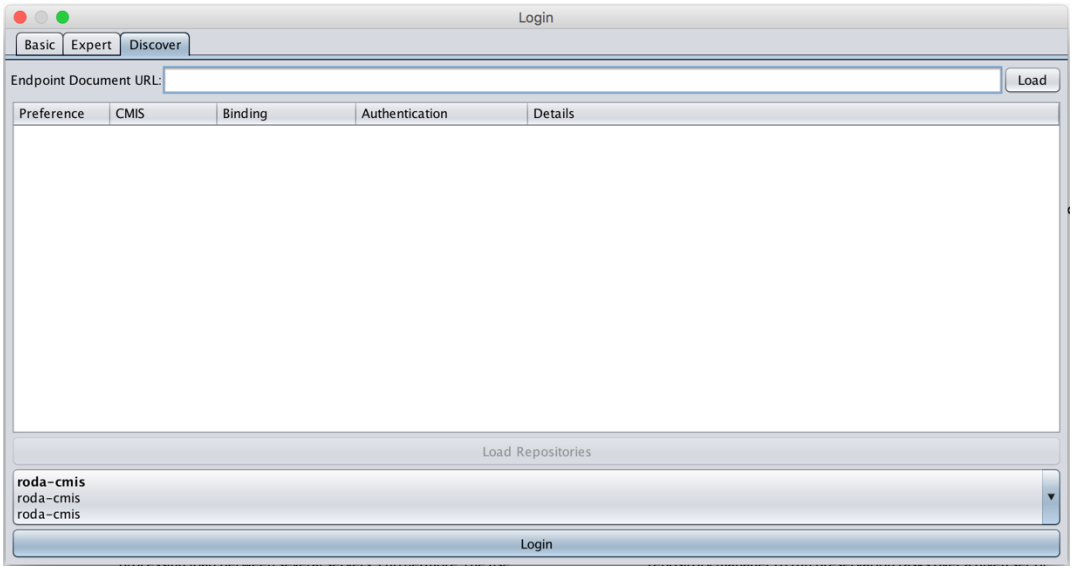


Figure 6 – Repository Login window – Discover authentication

After a successfully login into a repository, the application shows us the browser window. Figure 7 illustrates the main window. The application presents us a main menu with buttons to the main operations a user can perform over a repository. Below the main menu, the application is divided into two panels. The left panel is the repository browser where a user can navigate the repository file tree. On the right, the application presents a details window where the user can see the selected object details. The details are divided in tabs each one compiling the various aspects about the selected object. The tabs are: i) Object (details), ii) Actions, iii) Properties, iv) Relationships, v) Renditions, vi) ACL (Access Control Lists), vii) Policies, viii) Versions, ix) Type and x) Extensions.

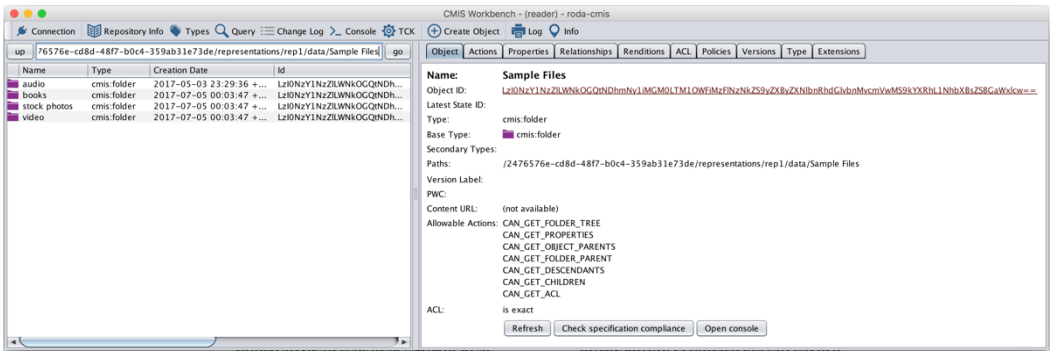


Figure 7 – Application main window – Repository files browser

Figure 8 illustrates the “Type” tab with the information about the selected object in the left panel. The “Type” tab contains a list of all the metadata fields associated with a given type. Each field is passible of being filled with value about the object. The list of metadata fields and respective values can then be consulted in the “Properties” tab. The metadata about each repository object plays a vital role. It is through the several objects metadata that repository searches are performed. Therefore, if no metadata is associated with the objects, performing a meaningful search over a very big repository (with many objects) may be a challenging task, if possible at all.

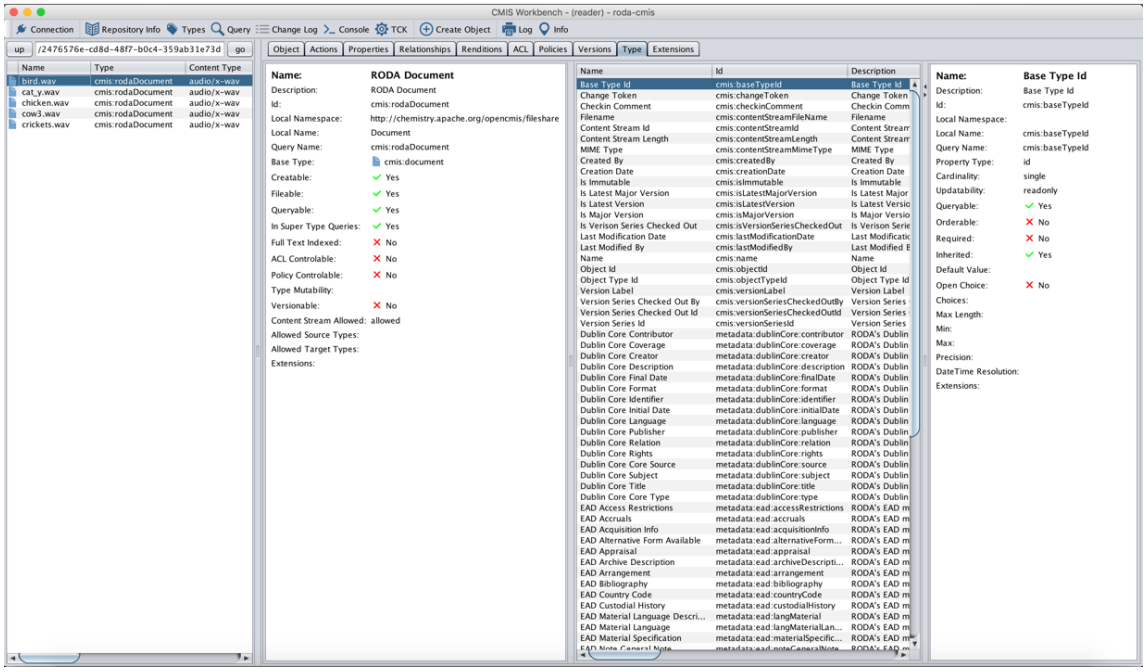
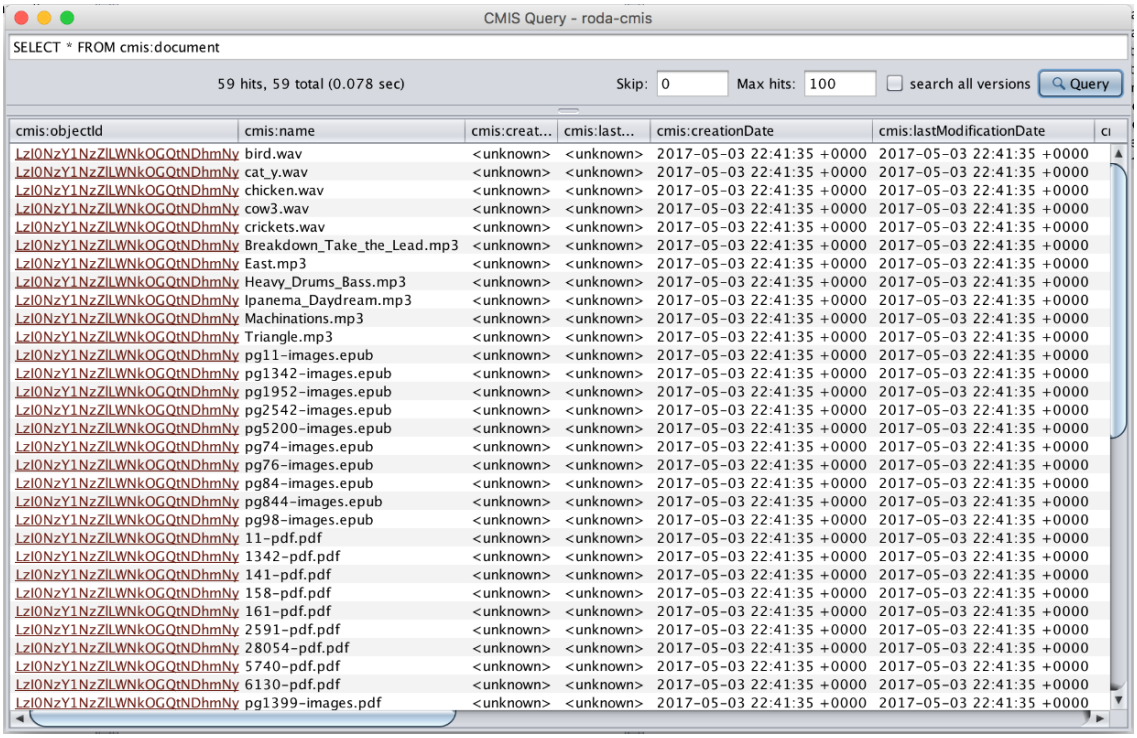


Figure 8 – RODA Document object type properties

In Figure 9 we have the repository query window. The textbox at the top of the window allows us to specify an SQL query to be run against the repository objects. The repository supports the SELECT subset of the SQL-92 specification [98]. Once the query is ran, through the click on the "Query" button, the results are presented in the list below. We can then sort the columns in an ascending or descending order by clicking on the column headers. By clicking on each object id (the "cmis:objectId" column) the object information is loaded in the main window.



cmis:objectId	cmis:name	cmis:creat...	cmis:last...	cmis:creationDate	cmis:lastModificationDate	ci
LzI0NzY1NzZlLWNkOGQtNDhmNy	bird.wav	<unknown>	<unknown>	2017-05-03 22:41:35 +0000	2017-05-03 22:41:35 +0000	
LzI0NzY1NzZlLWNkOGQtNDhmNy	cat_y.wav	<unknown>	<unknown>	2017-05-03 22:41:35 +0000	2017-05-03 22:41:35 +0000	
LzI0NzY1NzZlLWNkOGQtNDhmNy	chicken.wav	<unknown>	<unknown>	2017-05-03 22:41:35 +0000	2017-05-03 22:41:35 +0000	
LzI0NzY1NzZlLWNkOGQtNDhmNy	cow3.wav	<unknown>	<unknown>	2017-05-03 22:41:35 +0000	2017-05-03 22:41:35 +0000	
LzI0NzY1NzZlLWNkOGQtNDhmNy	crickets.wav	<unknown>	<unknown>	2017-05-03 22:41:35 +0000	2017-05-03 22:41:35 +0000	
LzI0NzY1NzZlLWNkOGQtNDhmNy	Breakdown_Take_the_Lead.mp3	<unknown>	<unknown>	2017-05-03 22:41:35 +0000	2017-05-03 22:41:35 +0000	
LzI0NzY1NzZlLWNkOGQtNDhmNy	East.mp3	<unknown>	<unknown>	2017-05-03 22:41:35 +0000	2017-05-03 22:41:35 +0000	
LzI0NzY1NzZlLWNkOGQtNDhmNy	Heavy_Drums_Bass.mp3	<unknown>	<unknown>	2017-05-03 22:41:35 +0000	2017-05-03 22:41:35 +0000	
LzI0NzY1NzZlLWNkOGQtNDhmNy	Ipanema_Daydream.mp3	<unknown>	<unknown>	2017-05-03 22:41:35 +0000	2017-05-03 22:41:35 +0000	
LzI0NzY1NzZlLWNkOGQtNDhmNy	Machinations.mp3	<unknown>	<unknown>	2017-05-03 22:41:35 +0000	2017-05-03 22:41:35 +0000	
LzI0NzY1NzZlLWNkOGQtNDhmNy	Triangle.mp3	<unknown>	<unknown>	2017-05-03 22:41:35 +0000	2017-05-03 22:41:35 +0000	
LzI0NzY1NzZlLWNkOGQtNDhmNy	pg11-images.epub	<unknown>	<unknown>	2017-05-03 22:41:35 +0000	2017-05-03 22:41:35 +0000	
LzI0NzY1NzZlLWNkOGQtNDhmNy	pg1342-images.epub	<unknown>	<unknown>	2017-05-03 22:41:35 +0000	2017-05-03 22:41:35 +0000	
LzI0NzY1NzZlLWNkOGQtNDhmNy	pg1952-images.epub	<unknown>	<unknown>	2017-05-03 22:41:35 +0000	2017-05-03 22:41:35 +0000	
LzI0NzY1NzZlLWNkOGQtNDhmNy	pg2542-images.epub	<unknown>	<unknown>	2017-05-03 22:41:35 +0000	2017-05-03 22:41:35 +0000	
LzI0NzY1NzZlLWNkOGQtNDhmNy	pg5200-images.epub	<unknown>	<unknown>	2017-05-03 22:41:35 +0000	2017-05-03 22:41:35 +0000	
LzI0NzY1NzZlLWNkOGQtNDhmNy	pg74-images.epub	<unknown>	<unknown>	2017-05-03 22:41:35 +0000	2017-05-03 22:41:35 +0000	
LzI0NzY1NzZlLWNkOGQtNDhmNy	pg76-images.epub	<unknown>	<unknown>	2017-05-03 22:41:35 +0000	2017-05-03 22:41:35 +0000	
LzI0NzY1NzZlLWNkOGQtNDhmNy	pg84-images.epub	<unknown>	<unknown>	2017-05-03 22:41:35 +0000	2017-05-03 22:41:35 +0000	
LzI0NzY1NzZlLWNkOGQtNDhmNy	pg844-images.epub	<unknown>	<unknown>	2017-05-03 22:41:35 +0000	2017-05-03 22:41:35 +0000	
LzI0NzY1NzZlLWNkOGQtNDhmNy	pg98-images.epub	<unknown>	<unknown>	2017-05-03 22:41:35 +0000	2017-05-03 22:41:35 +0000	
LzI0NzY1NzZlLWNkOGQtNDhmNy	11-pdf.pdf	<unknown>	<unknown>	2017-05-03 22:41:35 +0000	2017-05-03 22:41:35 +0000	
LzI0NzY1NzZlLWNkOGQtNDhmNy	1342-pdf.pdf	<unknown>	<unknown>	2017-05-03 22:41:35 +0000	2017-05-03 22:41:35 +0000	
LzI0NzY1NzZlLWNkOGQtNDhmNy	141-pdf.pdf	<unknown>	<unknown>	2017-05-03 22:41:35 +0000	2017-05-03 22:41:35 +0000	
LzI0NzY1NzZlLWNkOGQtNDhmNy	158-pdf.pdf	<unknown>	<unknown>	2017-05-03 22:41:35 +0000	2017-05-03 22:41:35 +0000	
LzI0NzY1NzZlLWNkOGQtNDhmNy	161-pdf.pdf	<unknown>	<unknown>	2017-05-03 22:41:35 +0000	2017-05-03 22:41:35 +0000	
LzI0NzY1NzZlLWNkOGQtNDhmNy	2591-pdf.pdf	<unknown>	<unknown>	2017-05-03 22:41:35 +0000	2017-05-03 22:41:35 +0000	
LzI0NzY1NzZlLWNkOGQtNDhmNy	28054-pdf.pdf	<unknown>	<unknown>	2017-05-03 22:41:35 +0000	2017-05-03 22:41:35 +0000	
LzI0NzY1NzZlLWNkOGQtNDhmNy	5740-pdf.pdf	<unknown>	<unknown>	2017-05-03 22:41:35 +0000	2017-05-03 22:41:35 +0000	
LzI0NzY1NzZlLWNkOGQtNDhmNy	6130-pdf.pdf	<unknown>	<unknown>	2017-05-03 22:41:35 +0000	2017-05-03 22:41:35 +0000	
LzI0NzY1NzZlLWNkOGQtNDhmNy	pg1399-images.pdf	<unknown>	<unknown>	2017-05-03 22:41:35 +0000	2017-05-03 22:41:35 +0000	

Figure 9 – Repository query window

Figure 10 illustrates the TCK, or Test Compatibility Kit window, the application API Client test suite. This functionality allows us to perform a test on the API full compatibility with the CMIS protocol. By default, all the features are selected for testing, but we can refine our test battery by checking only the features we would like to test. This allows us to test specific features in the developing stage of a server API. Once we have selected the features we wish we can perform the test by clicking the "Run TCK" button.

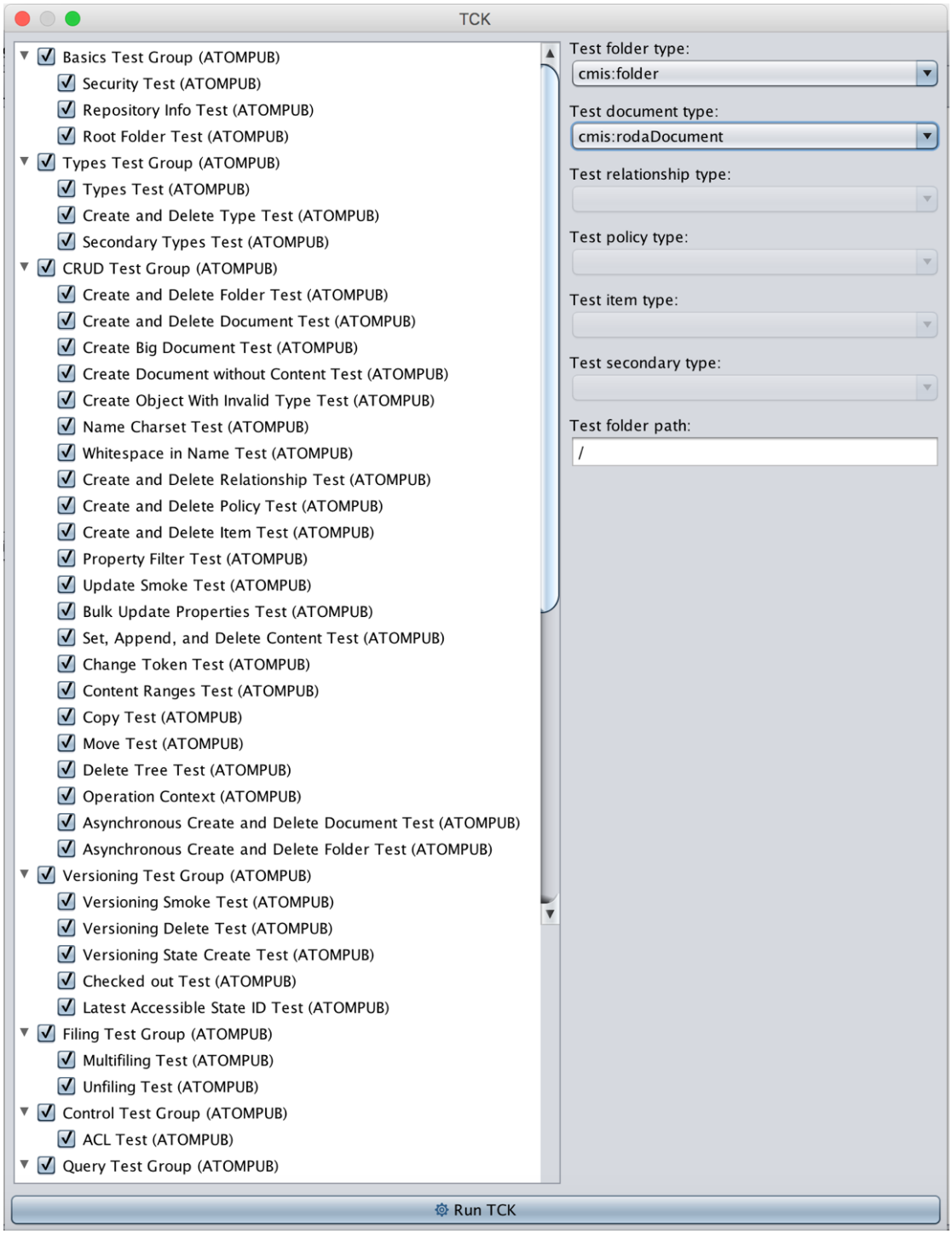


Figure 10 – API Client test window

Once the test battery is launched, a progress bar is shown by the application. Figure 11 illustrates the test report window with the results of the selected features tests. There are three possible color codes for the results: i) green, ii) yellow and iii) red. The tests marked in green were ran successfully, the ones marked in yellow had a minor issue or unexpected response during execution. The tests marked in yellow show an arrow behind the feature test. By clicking the arrow, the user provided with an explanation of the issue. The tests marked in red are the ones that failed and they did not complete the operation. The feature text is replaced by the error occurred.

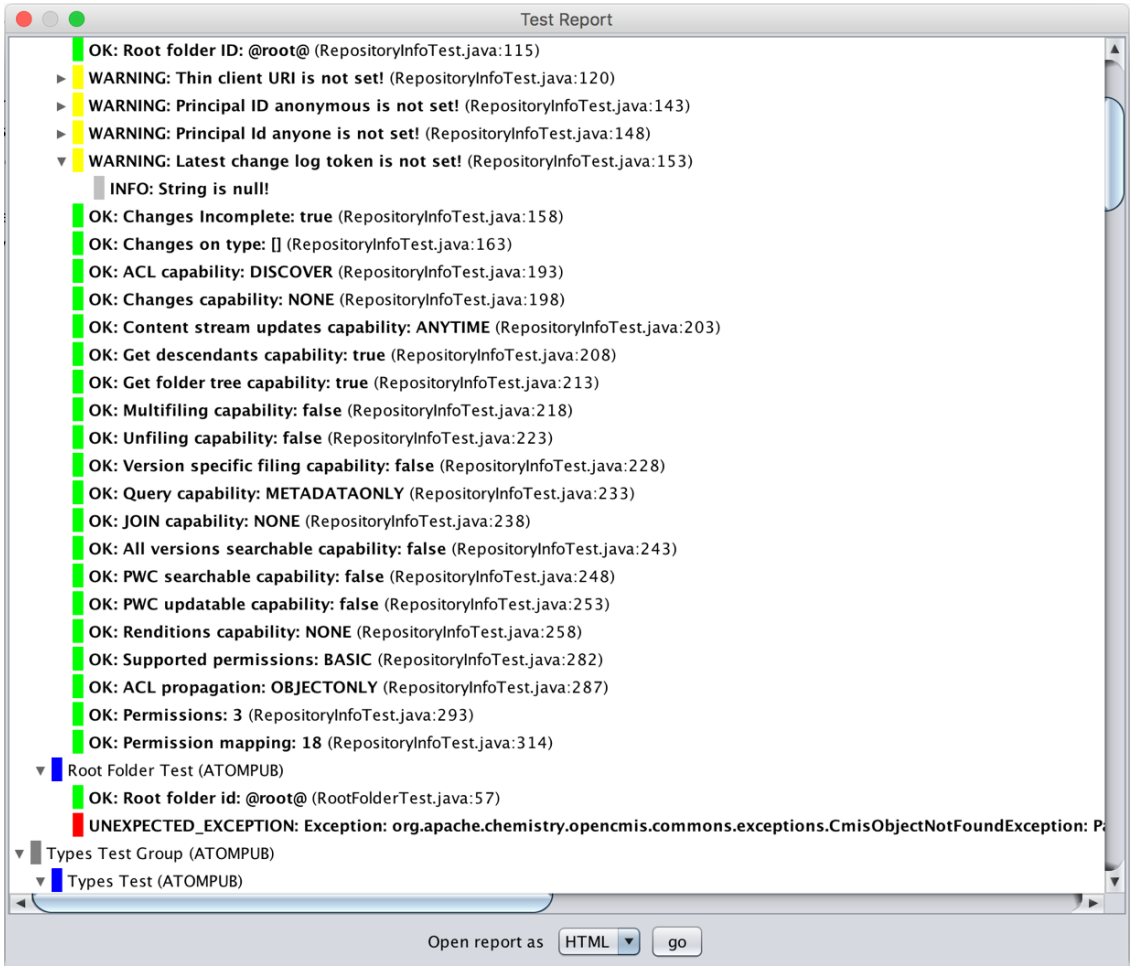


Figure 11 – API Client test result

4. Implementation

This chapter details the implementation made in RODA to dote it with further interoperability capabilities. We developed RODA-OCS, an OpenCMIS server prototype, inside RODA. This prototype was integrated with RODA local file storage and native permissions system. Through RODA permissions system users can define which contents they wish to make available to RODA-OCS and be shared publicly. RODA-OCS own permissions system then provides the environment for those contents to be shared in an authenticated and secure way. Any system or client application accessing RODA-OCS prototype is capable of navigating the server contents through the implemented file browser mechanism. The prototype also implemented a query feature which allows users to perform queries over server the shared contents. The queries are performed over the files and folders metadata.

We considered the integration of such prototype inside RODA important as it brought the repository communication capabilities through the CMIS protocol, that to the best of our knowledge, at the time of our study were inexistent. Furthermore, the CMIS protocol is mature and integrates various other repository software and client applications [92]. RODA-OCS prototype gave RODA the possibility to interact with them as well.

In Section 4.1, we start by giving an overview of the folders structure kept in RODA local storage. Then, in Section 4.2, we explain the implemented solution for the permissions system. In Section 4.3 we explain the OpenCMIS server class hierarchy for the implementation and in Section 4.4 we detail the server start-up and involved processes. We then proceed to Section 4.5 where we present the repository file browser feature and its implementation. Finally, in Section 4.6, we present the server query engine capabilities and implementation.

4.1. Storage

The process begins with the load of information into the repository. The producer logs in to the system and uploads a SIP through the Ingestion module. The SIP files are processed by the ingestion module and if the ingestion process is completed successfully the SIP is prepared to be stored inside the repository.

As stated before, RODA allows two possible configurations for storage: local and remote. We chose to use the local storage configuration for our implementation. This was due to the

fact that by using local storage we have full control over the files and folders, as well as, full access to the whole RODA storage structure. If we opted by the remote storage configuration we would not have access to the storage folder structure. The lack of access would prevent us to perform an in-depth analysis of how contents are stored inside RODA and ultimately not let us accomplish our implementation. We could also configure a remote storage server locally and study it but we chose not to do it due to time limitations.

After the ingestion process, the uploaded SIP is separated into two types of data: i) files and folders and ii) related metadata and becomes an AIP. The metadata is sent to the Data Management module, while the files and folders, are sent to the Archival Storage module. Both modules persist the information in the local hard drive configured for storage inside a hidden folder ".roda". This folder is where all the information needed for the repository to work is stored. Figure 12 illustrates the folder structure.

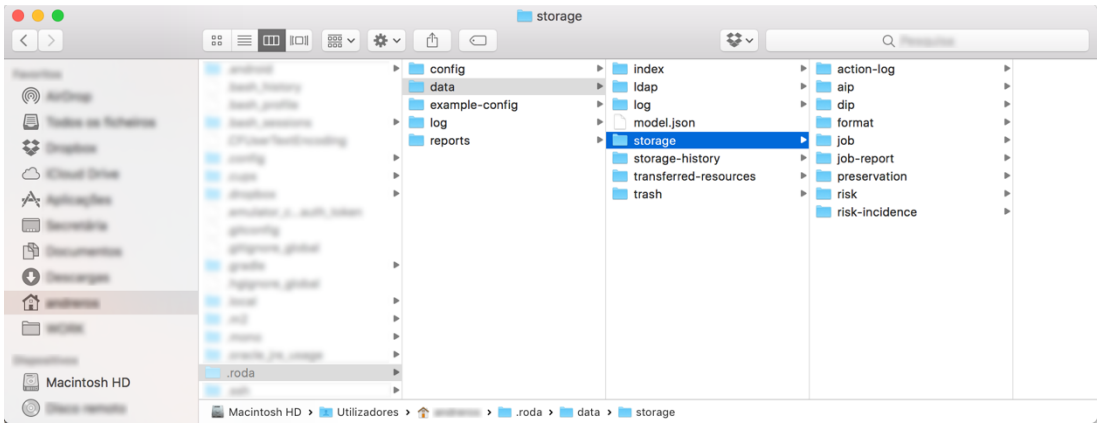


Figure 12 – RODA local storage folder

Inside RODA storage folder, we can find the following folders:

config – RODA configuration folder.

data – The folder where all the repository data is stored.

example-config – RODA default configuration to be loaded when there are no values previously saved in the **config** folder.

log – RODA application logs folder.

reports – The target folder for generated reports.

The folder where the AIPs are stored is located inside **data**. Inside this folder, we can find the following subfolders:

index – Lucene indexes folder.

ldap – LDAP service data folder.

log – General operations logs over data folder.

storage – The storage folder where AIPs will be located.

storage-history – Storage actions history folder for operation rollback, if needed.

transferred-resources – Transferred assets folder.

trash – Trash folder for deleted data.

Finally, inside the storage folder we have the following subfolders:

action-log – Logs over actions performed inside the application folder.

aip – Archival Information Packages folder.

dip – Distribution Information Packages folder.

format – Recognised formats folder.

job – Jobs process information folder.

job-report – Jobs reports folder.

preservation – Information about preservation agents and operations folder.

risk – Information about known risk and risk identification folder.

risk-incidence – Risk incidence reports folder.

The AIPs kept inside the folder **aip**, are stored under the E-ARK format [93]. An AIP structure is as follows:

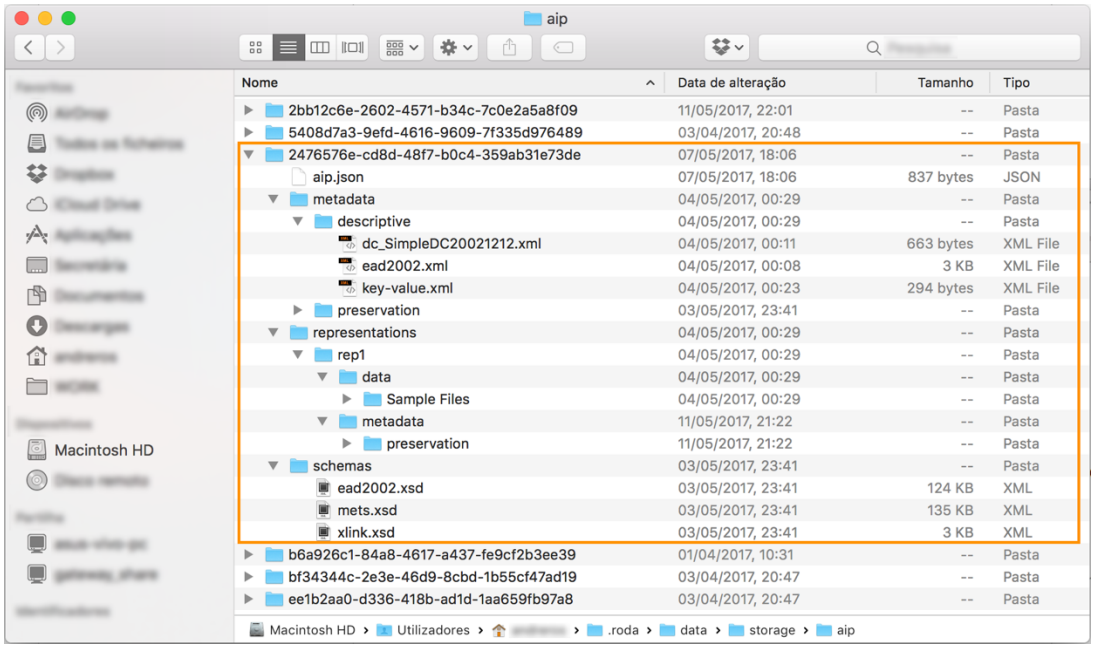


Figure 13 – AIP folder structure

The E-ARK Information Package specification defines the following folder structure [93] [99]:

```
base directory/
     -- metadata/
          -- descriptive/
          -- preservation/
          -- other/
     -- representations/
          -- <representation 1>/
          -- <representation 2>/
     -- documentation
     -- schemas
```

Where each representation folder presents the following structure:

```
<representation name>/
     -- data/
          -- [files]/
     -- metadata/
          -- descriptive/
          -- preservation/
          -- other/
```

The base directory of each AIP is a generated Universally Unique Identifier, or UUID for short. An UUID is calculated based in a system clock and time and produces an output string in the form **8-4-4-4-12** where each number represents a string of hexadecimal digits with the length indicated by the number, i.e. *123e4567-e89b-12d3-a456-426655440000* [102]. The following files and folders are inside the base directory:

aip.json – The AIP definition file. This file contains all the information about the AIP, such as its UUID, permissions, metadata files locations, representations locations, ingestion job IDs, relationships information and creation date.

metadata – The AIP metadata folder where descriptive, preservation and other possible metadata files are stored.

representations – The various AIP representations folder.

documentation – The AIP documentation folder.

schemas – The folder where metadata XML validation schemas for the utilized formats are stored.

Each representation folders are:

representation name – The folder where the AIP representation files are kept.

data – The representation payload files.

metadata – The metadata files related to this specific representation, either descriptive, preservation or other associated metadata.

4.2. Permissions

Both RODA and the OpenCMIS server have their own permissions systems. However, and although each permissions system works by itself, there is a need for these systems to interact with each other. Since RODA is the architecture component holding the contents to be exposed publicly, we used its group permissions feature to create a "cmis" group. This group contains all RODA users allowed to expose AIPs information through the OpenCMIS Server. In this implementation, and for demonstration purposes, we use the existing "admin" user to be part of the "cmis" group, as illustrated on Figure 14 and Figure 15.

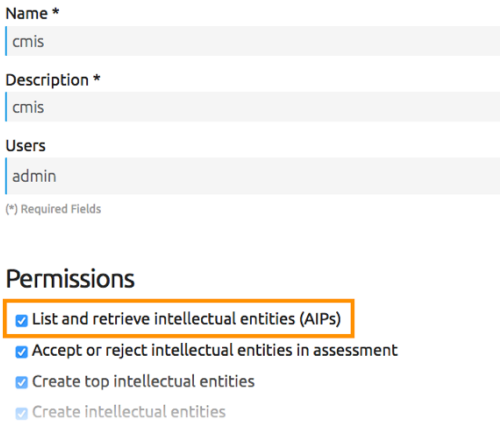


Figure 14 – CMIS users group

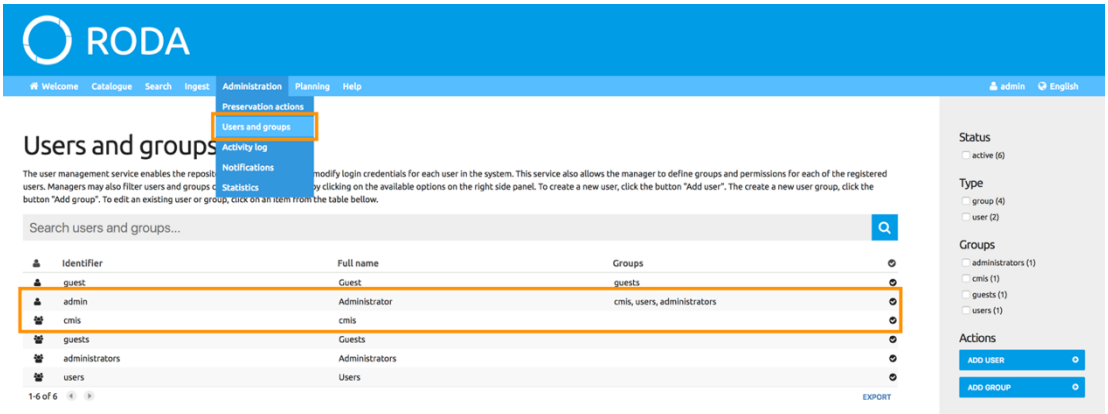


Figure 15 – RODA Administration – Users and Groups

Every AIP to be exposed via OpenCMIS Server can be configured by giving it the right permissions inside RODA Archival package permissions area, as presented by Figure 16.

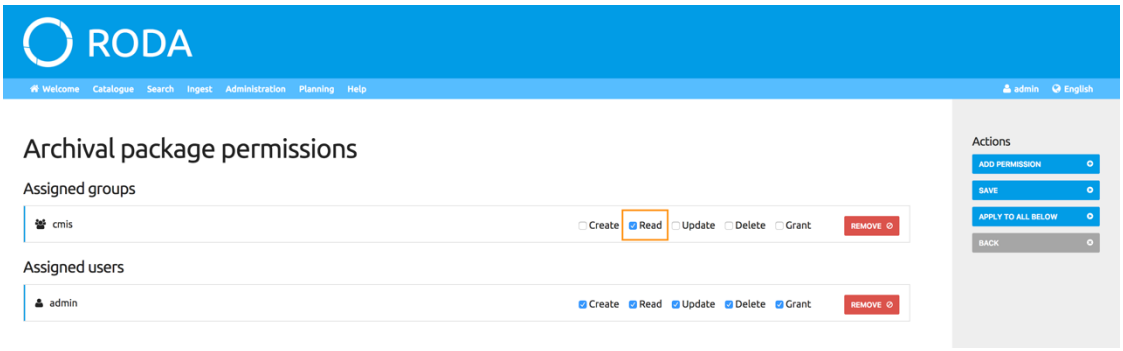


Figure 16 – Archival Information Package permissions configuration in RODA

The configured permissions will then be saved inside the "aip.json" file located in the AIP folder in the repository local storage. Figure 17 shows the permissions section inside the "aip.json" file.

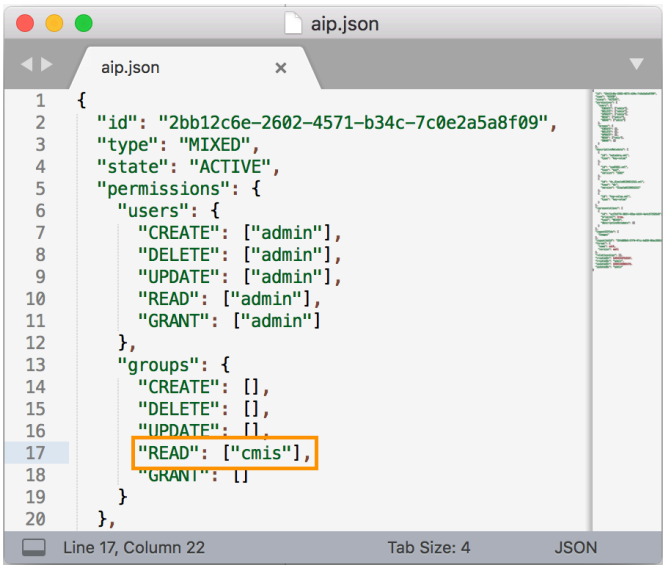


Figure 17 – AIP.json file group permissions

These configurations will then be read by the OpenCMIS Server every time someone tries to access the repository contents. Only files and folders belonging to AIPs configured with the "cmis" group are listed and shown to the CMIS clients.

On the OpenCMIS server side, there is also a mechanism for controlling users actions over data. The server possesses three levels of access to the exposed contents which are: i) read only, ii) write and iii) full access. Note that if not configured correctly it will be possible for a CMIS client to perform unwanted changes over the repository contents. To address this problem our OpenCMIS server is configured with "read only" permissions. This ensures that any client

connecting to the server will only be able to see the repository contents marked for exposure without being able to change them in any way.

4.3. OpenCMIS Server Class Hierarchy

The OpenCMIS server implementation recurs to the Apache Chemistry library to implement the CMIS protocol. In Figure 18 we detail the implemented class hierarchy for the server. This implementation used the code supplied in the "OpenCMIS Server Development Guide" [96], presented in Section 3.3, as its starting point. All classes belonging to our implementation are prefixed with the "FileBridge" term, except for the class "ContentRangeInputStream".

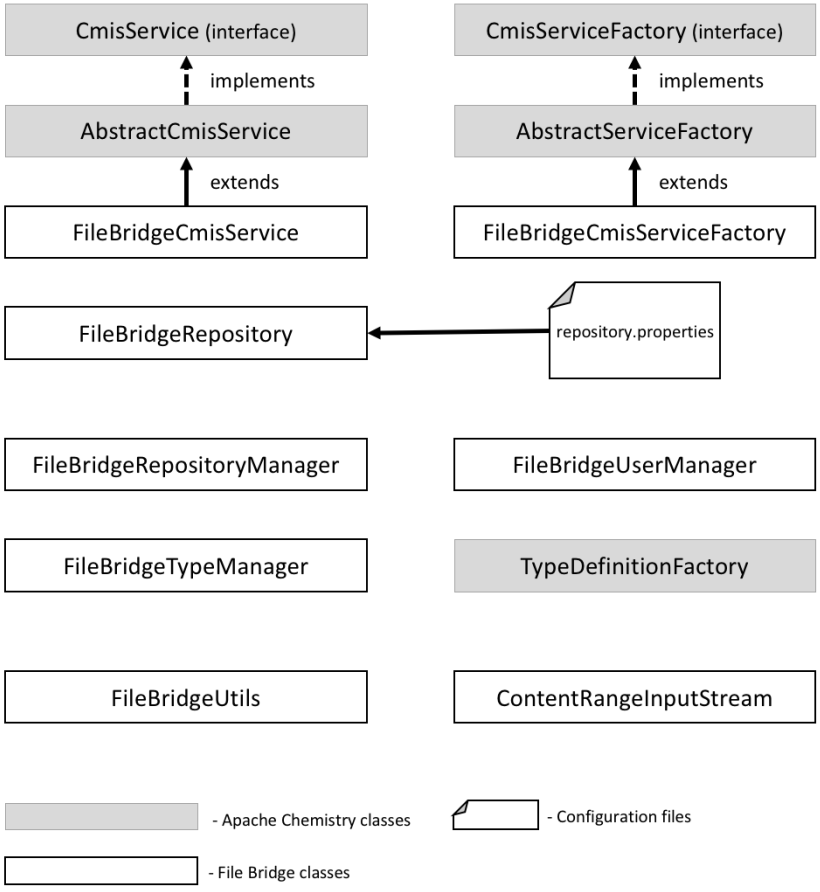


Figure 18 – OpenCMIS Server class hierarchy

FileBridgeCmisServiceFactory – This class extends from the **AbstractServiceFactory** class, which implements the **CmisServiceFactory** interface, and is responsible for producing

FileBridgeCmisService object instances. Apache Chemistry implements a factory pattern here because each connection to a CMIS server is handled by one instance of the FileBridgeCmisService.

FileBridgeCmisService – This is the class that answers to the client requests. It is wrapped in a CmisServiceWrapper class for convenience. The CmisServiceWrapper class checks for compliant requests and throws exceptions accordingly. This class extends from the **AbstractCmisService** class which implements the **CmisService** interface. The AbstractCmisService class provides an implementation to most of the CmisService interface methods reducing the required methods to be implemented to only six methods.

FileBridgeRepository – This class establishes a connection between the client requests and the repository contents. There is one instance of this class for each repository configured. The repositories configuration is stored in a **repository.properties** file, along with authentication credentials and access level permissions. This is the class where most of our server logic is implemented. Here, we map the CMIS operations over the files to the underlying operating system file system operations.

FileBridgeRepositoryManager – This class manages the existing instances of FileBridgeRepository in the server.

FileBridgeUserManager – This class manages the users configured for the various repositories in the repository.properties file, as well as their Access Control Lists (ACLs) and their read and write operations.

FileBridgeTypeManager – This class is responsible for managing all the type definitions for all the configured repositories. The CMIS protocol defines two base types: i) documents and ii) folders. Each of these types has a set of properties that are intended to work as a base for new types to be defined. Anyone wishing to implement more specific types of objects (i.e. an image, a PDF or a spreadsheet document) must use these base types parents of their custom types. The Apache Chemistry library provides a class to help with this process, the **TypeDefinitionFactory**. Through this class, we can create new already configured custom types where we will only have to define their new characteristics.

FileBridgeUtils – This is a helper class that gathers static common methods to be used across the server implementation.

ContentRangeInputStream – This is also a helper class responsible for reading a file into a stream and retrieve an excerpt of the read document, or the whole document.

4.4. OpenCMIS Server Bootstrap

When the OpenCMIS server is started, several operations are performed in order to prepare all the needed structures for the server to work.

The server starts by executing the **init()** method in the **FileBridgeCmisServiceFactory**. This method creates one instance of the **FileBridgeRepositoryManager**, one instance of the **FileBridgeUserManager** and one instance of the **FileBridgeTypeManager**. The three instances are then loaded with the repositories information configured in the **repository.properties** file. For each repository configured, a new **FileBridgeRepository** instance is created.

Upon creation, the new instance of the FileBridgeRepository will access RODA storage folder where the AIPs are kept, iterate through them and search for read permissions inside each AIPs "aip.json" file. As stated before, if the "cmis" permission is found, the contents are read and their details stored inside a SQLite database. The read is made recursively in order to reach each AIPs data directory maximum depth, hence retrieving its full contents (files and folders).

We opted to use a SQLite database[3] in our implementation[4] both for performance reasons and for standard compliance. SQLite is a mature and wide spread lightweight database engine that allows us to perform indexed searches over data with very low response times. Besides, SQLite is compliant with the SQL-92 SELECT language subset, required by the CMIS specification, which allowed us to use its query parser instead of implementing one from the beginning.

During this process, the "aip.json" "**descriptiveMetadata**" section will also be read to see if there are any metadata files. If so, the metadata corresponding to each AIP is loaded both to the database and into the server memory. This load is made by a metadata parser implemented to detect the descriptive metadata formats used by RODA. On detection, the parser extracts all the possible metadata and prepares structures to be loaded into the server memory[5]. The load into memory is due to the fact that such information must be included in the object to which it belongs when a list of objects is sent to the client.

[3] SQLite database: https://www.sqlite.org/

[4] SQLite JDBC database implementation: https://bitbucket.org/meicmdigitalpreservation/java-sqlite-jdbc-example

[5] Metadata Parser implementation: https://bitbucket.org/meicmdigitalpreservation/roda-metadata-parser

RODA supports three metadata formats to represent descriptive metadata: i) EAD, ii) Dublin Core and iii) Key-Value. The first two, EAD [22] and Dublin Core [23], are standards, the third one is not. Our implementation recognizes and supports these three formats, being able to extract and associate them to each object inside the AIP. However, there is one structural limitation in RODA implementation: each AIP only supports one metadata file of each format. This means that if we have various files, regardless of their format, inside the AIPs data folder, the existing descriptive metadata will be the same to every file. Our OpenCMIS server implementation follows the same principle.

4.5. File Browser

The file browser is a server mechanism that retrieves a list of objects, either files or folders, available in the repository, to the client applications.

A client application logs in to our server, chooses the repository (as illustrated earlier in Figure 4) and the method **getChildren()** in the **FileBridgeRepository** class is invoked. The first call to the method retrieves the folder configured as root folder in the **repository.properties** configuration file, since this the starting point for any client to start navigating the repository contents.

Similar to the server bootstrap process, RODA folder “aip” contents are iterated, each “aip.json” file is read for permissions and if the “cmis” permissions are found, the aip contents are included in the list of objects to be retrieved to the client.

Since our root folder is configured to be the “aip” folder, the first time we list the repository contents, we get a list of all the contents found in the “data” folder of each AIP. As an example, if we have three AIPs with three files each in their root “data” folder, we will obtain a list of nine files as we access the root folder of our repository.

Once we start navigating the repository folders tree, the server automatically restricts the access to a part of the folder structure path, bypassing it. The restricted part is the path between the repository root folder and each AIP root folder. This means that any path between “/” and “/../data/” will not be accessible for navigation. We apply this restriction due to the E-ARK folders structure not being designed to be navigated outside the “data” folder (all the AIP contents are inside “data”). Figure 7 in Section 3.5 shows a repository root folder when accessed from the CMIS Workbench client tool.

In the process of obtaining the list of objects to be listed for each AIP, the descriptive metadata extracted for each object in the server bootstrap is read from memory, encapsulated with the object and sent to the client. Therefore, if a CMIS client application has the capabilities of showing the various properties of a CMIS object, this information is available. Figure 19 shows a listed object properties panel, with all its properties and respective values.

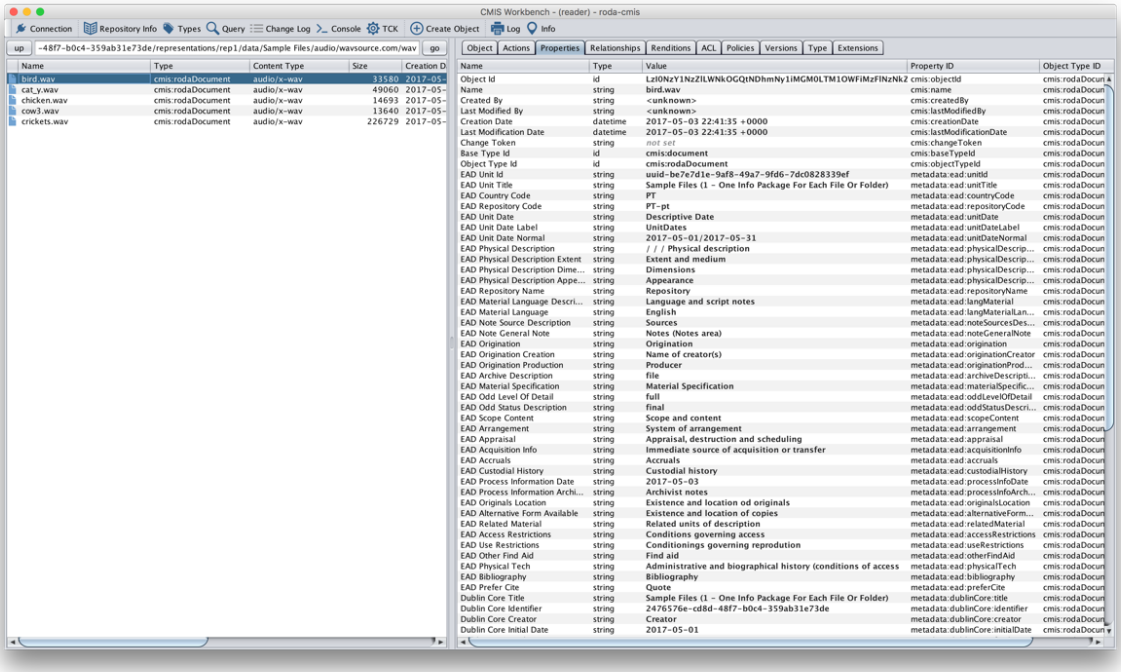


Figure 19 - RODA Document object properties in CMIS Workbench

As stated in Section 4.3 in the FileBridgeTypeManager paragraph, the CMIS protocol specifies two base types of documents: i) cmis:document and ii) cmis:folder. These two base types are intended to be extended and work as the parent entities for a hierarchy of new types. As such, our server implementation contemplates a new type of document, the **cmis:rodaDocument**. Figure 20 demonstrates the implemented object types hierarchy.

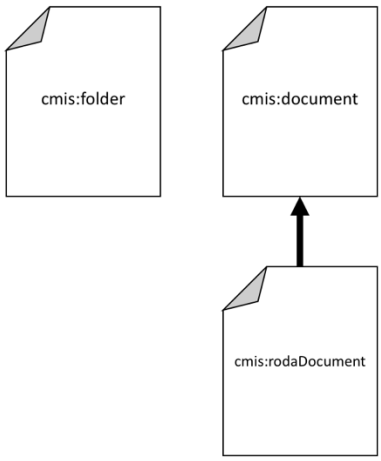


Figure 20 – OpenCMIS Server object types hierarchy

The cmis:rodaDocument extends from the cmis:document object type. It encapsulates all the properties of its parent along with all the possibly extractable values of the EAD, Dublin Core and Key-value metadata files. Figure 19 also illustrates these properties.

4.6. Query Engine

The query engine is the server mechanism which allows client applications to perform searches over the full contents of the repository.

When a client application performs a search, the **query()** method in the **FileBridgeRepository** class is invoked. This method receives a query statement that must be expressed in the SQL-92 syntax. The CMIS specification indicates the following rules:

i) The field names are property names, which will be handled as column names.
ii) The FROM clause accepts only object type names, which will be handled as tables.
iii) The FROM clause only accepts one table per query.

These rules will produce a clause with like this:

```
SELECT cmis:name FROM cmis:document WHERE cmis:name = 'some name'
```

Once a query is submitted to the server, the incoming statement is analyzed, the type of query is identified and the parameters are extracted. The query is then converted to a syntax the SQLite database engine recognizes, table and column names are escaped and navigational functions converted to their equivalent expressions if there is any.

The SQLite database implemented possesses only two tables: i) cmis:rodaDocument and ii) cmis:folder. Each table contains one column for every property defined for the object type. The cmis:folder table has all the cmis:folder base type properties and the cmis:rodaDocument table has all the cmis:document base type object properties as well as all the extra properties defined for the cmis:rodaDocument type object which are the properties that hold the metadata extracted from the EAD, Dublin Core and Key-value files.

When a query is performed, the server runs the previously adjusted query against the SQLite database. The database always retrieves a list of paths to the matching objects as the result. Those paths are then read by the server in order to prepare the objects to be retrieved to the client application, following the same process explained in Section 4.5.

Navigational functions IN_FOLDER and IN_TREE

The CMIS specification defines two navigational functions to be implemented by a CMIS compliant query engine. The navigational functions are: i) IN_FOLDER and ii) IN_TREE. Both functions receive two parameters: the folder ID and the object type to be retrieved.

Our server implements both navigational functions, but only accept the first parameter which is the folder ID. The second parameter, the object type to be retrieved is not implemented, mainly because of time reasons and also because the lack of this feature does not compromise further development and the results achieved. The IN_FOLDER navigational function restricts the results to the folder defined by the folder ID while the IN_TREE navigational function restricts the results to the folder defined by the folder ID as well as it subfolders down to its maximum depth.

This feature was implemented based in each table (either cmis:folder or cmis:rodaDocument) cmis:path field. Although neither the cmis:document nor the cmis:rodaDocument have the cmis:path field in their objects specification this field was included to support the implementation of these navigational functions. The results are obtained by evaluating the exact path of an object in the IN_FOLDER function case and by using an additional wildcard in the evaluation of the objects path in the case of the IN_TREE function.

5. System Demonstration

In this chapter, we demonstrate the functioning of RODA-OCS server prototype. For the demonstration of the prototype we created a dataset with several SIPs. Each SIP contained descriptive metadata files expressed in the three formats accepted by RODA: EAD, Dublin Core and Key-Value. The SIPs were uploaded to RODA and once successfully ingested their contents sharing permissions to RODA-OCS were configured through RODA native permissions system. The server prototype was then bootstrapped. The configured contents were registered in RODA-OCS database and the services started. For the presentation, we used the CMIS Workbench as a testing tool. We performed a connection to RODA-OCS, navigated the available folder structure and performed various queries over the repository objects.

In Section 5.2 we describe the dataset used to perform the tests to the implemented system. In Section 5.2 we show the repository contents navigation through the file browser while in Section 5.3 we proceed by showing the query engine mechanism and the features implemented.

5.1. Demonstration Dataset

To test our server implementation, we created a dataset with a total of 69 files and 21 folders. These files were organized and distributed by 12 different SIPs and uploaded to RODA. One SIP contained 56 files and 16 folders distributed by a folder tree with three levels of depth. Another SIP was a bundle of 3 files and the remaining SIPs contained 1 file each. The chosen file types included currently used formats, supported by multimedia applications. The formats span four different types of multimedia: audio, books, photos and video. The included formats were: wav and mp3 for audio, epub, pdf and txt for books, jpeg for photos and mov for video. All files included were freely available in the internet as sample files and were free to use in any context, not infringing any copyright laws.

We chose this configuration for the demonstration dataset due to the fact that for every SIP three metadata files had to be created and filled manually. Each file contained descriptive metadata in one of the formats supported by RODA: EAD, Dublin Core and Key-value. The demanding and time-consuming task of creating and populating by hand 60 fields of metadata,

distributed by the three files for each SIP, prevented us from creating a larger dataset and test the server for load.

5.2. File Browser

This Section tested the implementation of the functionality described in Section 4.5. However, before accessing the file browser from the CMIS Workbench tool, we ensured the server bootstrap, described in Section 4.4, took place successfully. For the following tests to take place, the OpenCMIS server had to be started and the SIP contents and metadata inside RODA had to be correctly configured to be read and loaded into the server.

The first access to the repository after the successful authentication retrieved the list of objects located at the root path level configured for the repository. Figure 21 illustrates the navigation from the initial parent folder to its deepest subfolder. The full path followed was **"/Sample Files/audio/wavsource.com/wav/**".

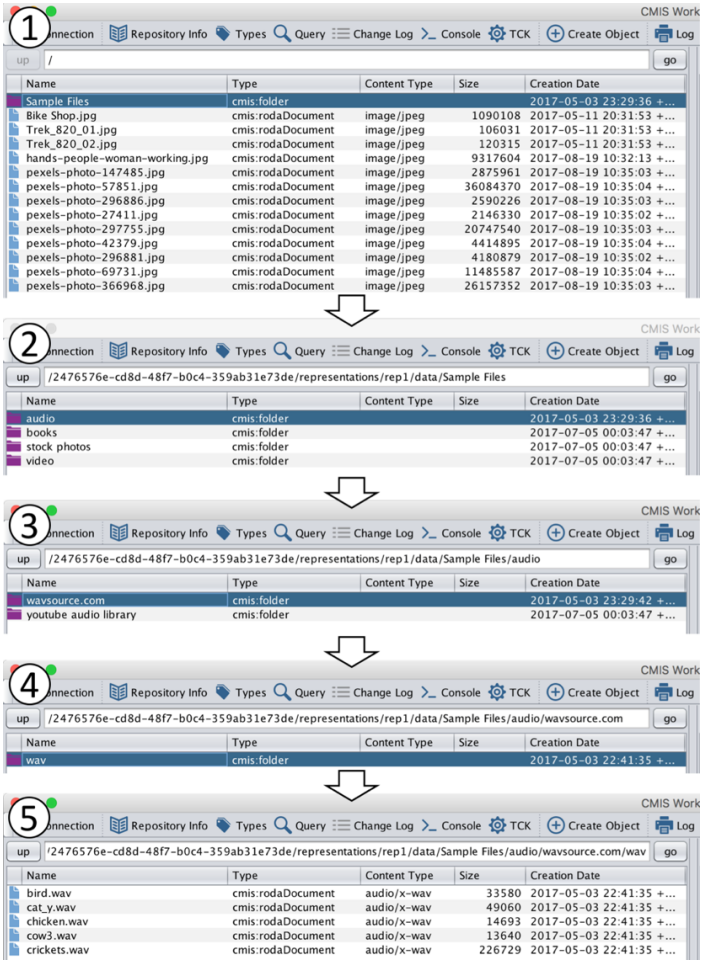


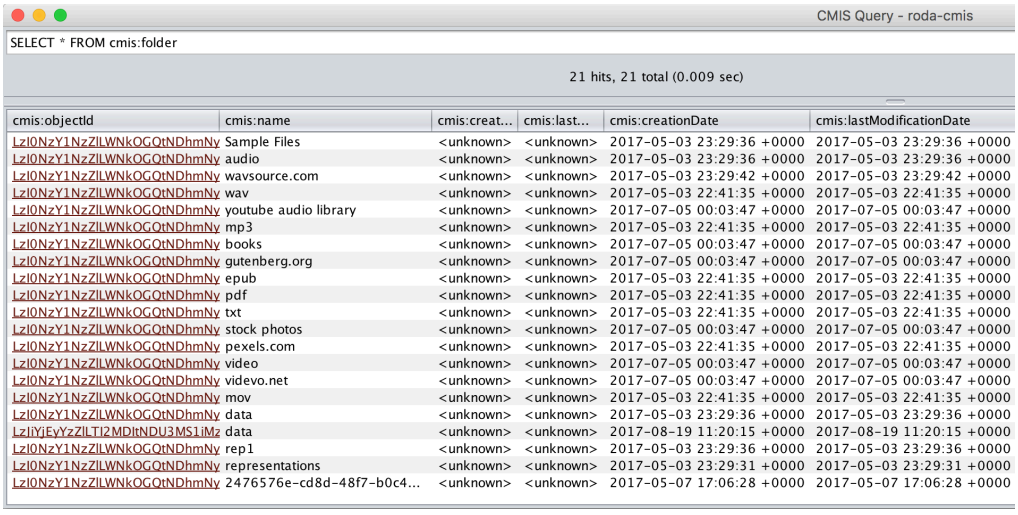
Figure 21 – File browser navigation from root folder to leaf folder

The root folder, represented in Figure 21 with the number 1, gathered all AIPs files at their "**data**" folder level. When we entered one of the listed folders from the root level, as represented in number 2 in the figure, we accessed their AIP "data" folder and navigated inside it. This resulted in the intermediate path "**2476576e-cd8d-b0c4-359ab31e73de/representations/rep1/data/**" being presented between the "/" (root path) and the "**Sample Files/audio/wavsource.com/wav/**". Navigation inside that intermediate path is forbidden, so, when navigating back to the root folder any folder belonging to the path "**2476576e-cd8d-b0c4-359ab31e73de/representations/rep1/data/**" did not retrieve any files or folders. As a result, when navigating this path, the file browser window was empty, not showing any file or folder. Number 3, 4 and 5 in Figure 21 illustrate the navigation to the "**wav**" folder, located at the deepest level of the file tree.

5.3. Query Engine

This Section demonstrates the implementation of the functionality described in Section 4.6. For the query engine to be demonstrated we assumed what we stated in Section 5.2, that the OpenCMIS server was successfully bootstrapped and the SIPs information in RODA were correctly loaded.

The query engine performs searches over the contents of the repository through the execution of SQL-92 SELECT statements. The CMIS Workbench tool possesses a window specially designed to submit queries to our OpenCMIS sever. Figure 22 shows the execution of the "**SELECT * FROM cmis:folder**" statement.

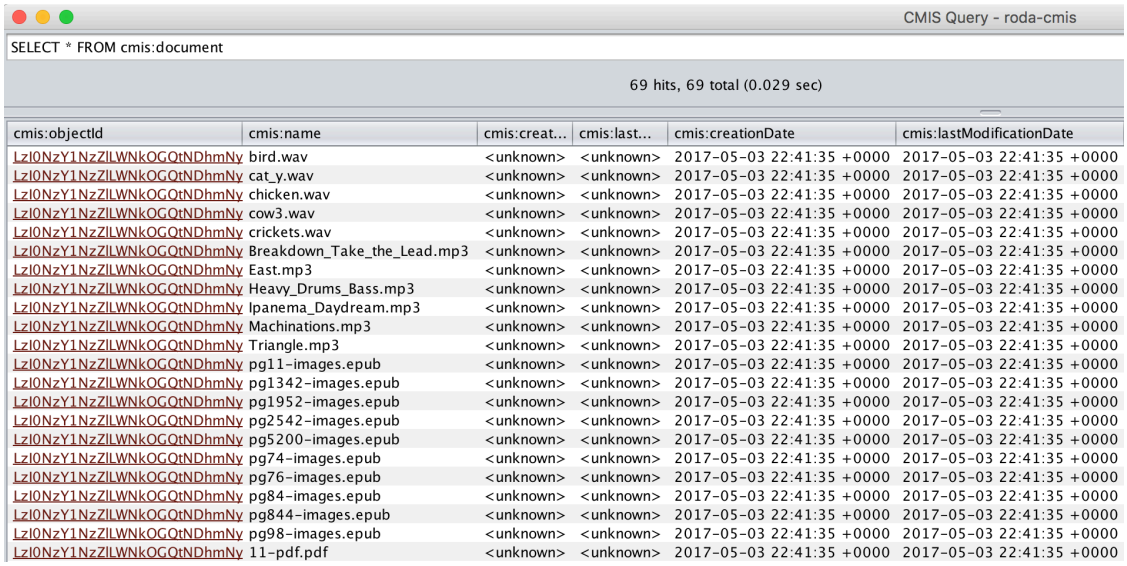


cmis:objectId	cmis:name	cmis:creat...	cmis:last...	cmis:creationDate	cmis:lastModificationDate
LzI0NzY1NzZlLWNkOGQtNDhmNy	Sample Files	<unknown>	<unknown>	2017-05-03 23:29:36 +0000	2017-05-03 23:29:36 +0000
LzI0NzY1NzZlLWNkOGQtNDhmNy	audio	<unknown>	<unknown>	2017-05-03 23:29:36 +0000	2017-05-03 23:29:36 +0000
LzI0NzY1NzZlLWNkOGQtNDhmNy	wavsource.com	<unknown>	<unknown>	2017-05-03 23:29:42 +0000	2017-05-03 23:29:42 +0000
LzI0NzY1NzZlLWNkOGQtNDhmNy	wav	<unknown>	<unknown>	2017-05-03 22:41:35 +0000	2017-05-03 22:41:35 +0000
LzI0NzY1NzZlLWNkOGQtNDhmNy	youtube audio library	<unknown>	<unknown>	2017-07-05 00:03:47 +0000	2017-07-05 00:03:47 +0000
LzI0NzY1NzZlLWNkOGQtNDhmNy	mp3	<unknown>	<unknown>	2017-05-03 22:41:35 +0000	2017-05-03 22:41:35 +0000
LzI0NzY1NzZlLWNkOGQtNDhmNy	books	<unknown>	<unknown>	2017-07-05 00:03:47 +0000	2017-07-05 00:03:47 +0000
LzI0NzY1NzZlLWNkOGQtNDhmNy	gutenberg.org	<unknown>	<unknown>	2017-07-05 00:03:47 +0000	2017-07-05 00:03:47 +0000
LzI0NzY1NzZlLWNkOGQtNDhmNy	epub	<unknown>	<unknown>	2017-05-03 22:41:35 +0000	2017-05-03 22:41:35 +0000
LzI0NzY1NzZlLWNkOGQtNDhmNy	pdf	<unknown>	<unknown>	2017-05-03 22:41:35 +0000	2017-05-03 22:41:35 +0000
LzI0NzY1NzZlLWNkOGQtNDhmNy	txt	<unknown>	<unknown>	2017-05-03 22:41:35 +0000	2017-05-03 22:41:35 +0000
LzI0NzY1NzZlLWNkOGQtNDhmNy	stock photos	<unknown>	<unknown>	2017-07-05 00:03:47 +0000	2017-07-05 00:03:47 +0000
LzI0NzY1NzZlLWNkOGQtNDhmNy	pexels.com	<unknown>	<unknown>	2017-05-03 22:41:35 +0000	2017-05-03 22:41:35 +0000
LzI0NzY1NzZlLWNkOGQtNDhmNy	video	<unknown>	<unknown>	2017-07-05 00:03:47 +0000	2017-07-05 00:03:47 +0000
LzI0NzY1NzZlLWNkOGQtNDhmNy	videvo.net	<unknown>	<unknown>	2017-07-05 00:03:47 +0000	2017-07-05 00:03:47 +0000
LzI0NzY1NzZlLWNkOGQtNDhmNy	mov	<unknown>	<unknown>	2017-05-03 22:41:35 +0000	2017-05-03 22:41:35 +0000
LzI0NzY1NzZlLWNkOGQtNDhmNy	data	<unknown>	<unknown>	2017-05-03 23:29:36 +0000	2017-05-03 23:29:36 +0000
LzIiYjEyYzZlLTI2MDItNDU3MS1iMz	data	<unknown>	<unknown>	2017-08-19 11:20:15 +0000	2017-08-19 11:20:15 +0000
LzI0NzY1NzZlLWNkOGQtNDhmNy	rep1	<unknown>	<unknown>	2017-05-03 23:29:36 +0000	2017-05-03 23:29:36 +0000
LzI0NzY1NzZlLWNkOGQtNDhmNy	representations	<unknown>	<unknown>	2017-05-03 23:29:31 +0000	2017-05-03 23:29:31 +0000
LzI0NzY1NzZlLWNkOGQtNDhmNy	2476576e-cd8d-48f7-b0c4...	<unknown>	<unknown>	2017-05-07 17:06:28 +0000	2017-05-07 17:06:28 +0000

Figure 22 – Query engine cmis:folder select

The execution of the statement retrieved the list of all the folders available in the repository, each with its specific attributes, represented in the table columns. Each object retrieved represented one unique and full path to one folder inside the repository. There were not any repeated results in the objects list retrieved. The use of the term "**cmis:folder**" in the "**FROM**" clause indicated the server we were searching for folders instead of files, as defined by the CMIS specification [100].

Figure 23 illustrates the execution of the "**SELECT * FROM cmis:document**" statement.


CMIS Query - roda-cmis

SELECT * FROM cmis:document

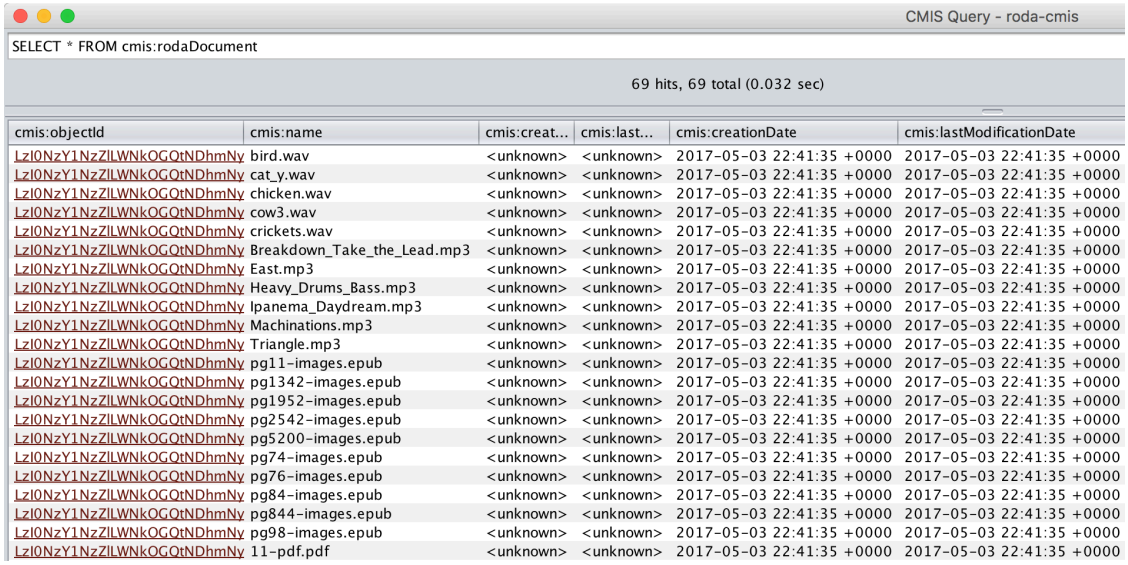
69 hits, 69 total (0.029 sec)

cmis:objectId	cmis:name	cmis:creat...	cmis:last...	cmis:creationDate	cmis:lastModificationDate
LzI0NzY1NzZlLWNkOGQtNDhmNy	bird.wav	<unknown>	<unknown>	2017-05-03 22:41:35 +0000	2017-05-03 22:41:35 +0000
LzI0NzY1NzZlLWNkOGQtNDhmNy	cat_y.wav	<unknown>	<unknown>	2017-05-03 22:41:35 +0000	2017-05-03 22:41:35 +0000
LzI0NzY1NzZlLWNkOGQtNDhmNy	chicken.wav	<unknown>	<unknown>	2017-05-03 22:41:35 +0000	2017-05-03 22:41:35 +0000
LzI0NzY1NzZlLWNkOGQtNDhmNy	cow3.wav	<unknown>	<unknown>	2017-05-03 22:41:35 +0000	2017-05-03 22:41:35 +0000
LzI0NzY1NzZlLWNkOGQtNDhmNy	crickets.wav	<unknown>	<unknown>	2017-05-03 22:41:35 +0000	2017-05-03 22:41:35 +0000
LzI0NzY1NzZlLWNkOGQtNDhmNy	Breakdown_Take_the_Lead.mp3	<unknown>	<unknown>	2017-05-03 22:41:35 +0000	2017-05-03 22:41:35 +0000
LzI0NzY1NzZlLWNkOGQtNDhmNy	East.mp3	<unknown>	<unknown>	2017-05-03 22:41:35 +0000	2017-05-03 22:41:35 +0000
LzI0NzY1NzZlLWNkOGQtNDhmNy	Heavy_Drums_Bass.mp3	<unknown>	<unknown>	2017-05-03 22:41:35 +0000	2017-05-03 22:41:35 +0000
LzI0NzY1NzZlLWNkOGQtNDhmNy	Ipanema_Daydream.mp3	<unknown>	<unknown>	2017-05-03 22:41:35 +0000	2017-05-03 22:41:35 +0000
LzI0NzY1NzZlLWNkOGQtNDhmNy	Machinations.mp3	<unknown>	<unknown>	2017-05-03 22:41:35 +0000	2017-05-03 22:41:35 +0000
LzI0NzY1NzZlLWNkOGQtNDhmNy	Triangle.mp3	<unknown>	<unknown>	2017-05-03 22:41:35 +0000	2017-05-03 22:41:35 +0000
LzI0NzY1NzZlLWNkOGQtNDhmNy	pg11-images.epub	<unknown>	<unknown>	2017-05-03 22:41:35 +0000	2017-05-03 22:41:35 +0000
LzI0NzY1NzZlLWNkOGQtNDhmNy	pg1342-images.epub	<unknown>	<unknown>	2017-05-03 22:41:35 +0000	2017-05-03 22:41:35 +0000
LzI0NzY1NzZlLWNkOGQtNDhmNy	pg1952-images.epub	<unknown>	<unknown>	2017-05-03 22:41:35 +0000	2017-05-03 22:41:35 +0000
LzI0NzY1NzZlLWNkOGQtNDhmNy	pg2542-images.epub	<unknown>	<unknown>	2017-05-03 22:41:35 +0000	2017-05-03 22:41:35 +0000
LzI0NzY1NzZlLWNkOGQtNDhmNy	pg5200-images.epub	<unknown>	<unknown>	2017-05-03 22:41:35 +0000	2017-05-03 22:41:35 +0000
LzI0NzY1NzZlLWNkOGQtNDhmNy	pg74-images.epub	<unknown>	<unknown>	2017-05-03 22:41:35 +0000	2017-05-03 22:41:35 +0000
LzI0NzY1NzZlLWNkOGQtNDhmNy	pg76-images.epub	<unknown>	<unknown>	2017-05-03 22:41:35 +0000	2017-05-03 22:41:35 +0000
LzI0NzY1NzZlLWNkOGQtNDhmNy	pg84-images.epub	<unknown>	<unknown>	2017-05-03 22:41:35 +0000	2017-05-03 22:41:35 +0000
LzI0NzY1NzZlLWNkOGQtNDhmNy	pg844-images.epub	<unknown>	<unknown>	2017-05-03 22:41:35 +0000	2017-05-03 22:41:35 +0000
LzI0NzY1NzZlLWNkOGQtNDhmNy	pg98-images.epub	<unknown>	<unknown>	2017-05-03 22:41:35 +0000	2017-05-03 22:41:35 +0000
LzI0NzY1NzZlLWNkOGQtNDhmNy	11-pdf.pdf	<unknown>	<unknown>	2017-05-03 22:41:35 +0000	2017-05-03 22:41:35 +0000

Figure 23 – Query engine cmis:document select

Like the example shown in Figure 22, the execution of the statement now retrieved a list of all the files available in the repository. Each object represented one unique file in one unique path. Even though there might have been repeated files in the repository, the path or location to each one of them was unique. Each table column represented one attribute of the "**cmis:document**" object type.

Figure 24 illustrates the execution of the "**SELECT * FROM cmis:rodaDocument**" statement.



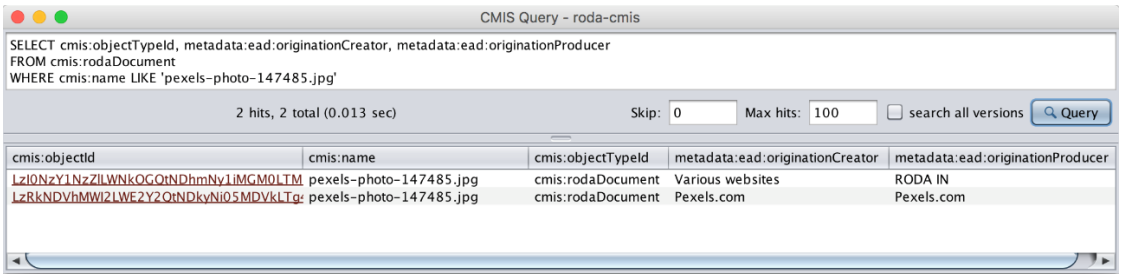
cmis:objectId	cmis:name	cmis:creat...	cmis:last...	cmis:creationDate	cmis:lastModificationDate
LzI0NzY1NzZlLWNkOGQtNDhmNy	bird.wav	<unknown>	<unknown>	2017-05-03 22:41:35 +0000	2017-05-03 22:41:35 +0000
LzI0NzY1NzZlLWNkOGQtNDhmNy	cat_y.wav	<unknown>	<unknown>	2017-05-03 22:41:35 +0000	2017-05-03 22:41:35 +0000
LzI0NzY1NzZlLWNkOGQtNDhmNy	chicken.wav	<unknown>	<unknown>	2017-05-03 22:41:35 +0000	2017-05-03 22:41:35 +0000
LzI0NzY1NzZlLWNkOGQtNDhmNy	cow3.wav	<unknown>	<unknown>	2017-05-03 22:41:35 +0000	2017-05-03 22:41:35 +0000
LzI0NzY1NzZlLWNkOGQtNDhmNy	crickets.wav	<unknown>	<unknown>	2017-05-03 22:41:35 +0000	2017-05-03 22:41:35 +0000
LzI0NzY1NzZlLWNkOGQtNDhmNy	Breakdown_Take_the_Lead.mp3	<unknown>	<unknown>	2017-05-03 22:41:35 +0000	2017-05-03 22:41:35 +0000
LzI0NzY1NzZlLWNkOGQtNDhmNy	East.mp3	<unknown>	<unknown>	2017-05-03 22:41:35 +0000	2017-05-03 22:41:35 +0000
LzI0NzY1NzZlLWNkOGQtNDhmNy	Heavy_Drums_Bass.mp3	<unknown>	<unknown>	2017-05-03 22:41:35 +0000	2017-05-03 22:41:35 +0000
LzI0NzY1NzZlLWNkOGQtNDhmNy	Ipanema_Daydream.mp3	<unknown>	<unknown>	2017-05-03 22:41:35 +0000	2017-05-03 22:41:35 +0000
LzI0NzY1NzZlLWNkOGQtNDhmNy	Machinations.mp3	<unknown>	<unknown>	2017-05-03 22:41:35 +0000	2017-05-03 22:41:35 +0000
LzI0NzY1NzZlLWNkOGQtNDhmNy	Triangle.mp3	<unknown>	<unknown>	2017-05-03 22:41:35 +0000	2017-05-03 22:41:35 +0000
LzI0NzY1NzZlLWNkOGQtNDhmNy	pg11-images.epub	<unknown>	<unknown>	2017-05-03 22:41:35 +0000	2017-05-03 22:41:35 +0000
LzI0NzY1NzZlLWNkOGQtNDhmNy	pg1342-images.epub	<unknown>	<unknown>	2017-05-03 22:41:35 +0000	2017-05-03 22:41:35 +0000
LzI0NzY1NzZlLWNkOGQtNDhmNy	pg1952-images.epub	<unknown>	<unknown>	2017-05-03 22:41:35 +0000	2017-05-03 22:41:35 +0000
LzI0NzY1NzZlLWNkOGQtNDhmNy	pg2542-images.epub	<unknown>	<unknown>	2017-05-03 22:41:35 +0000	2017-05-03 22:41:35 +0000
LzI0NzY1NzZlLWNkOGQtNDhmNy	pg5200-images.epub	<unknown>	<unknown>	2017-05-03 22:41:35 +0000	2017-05-03 22:41:35 +0000
LzI0NzY1NzZlLWNkOGQtNDhmNy	pg74-images.epub	<unknown>	<unknown>	2017-05-03 22:41:35 +0000	2017-05-03 22:41:35 +0000
LzI0NzY1NzZlLWNkOGQtNDhmNy	pg76-images.epub	<unknown>	<unknown>	2017-05-03 22:41:35 +0000	2017-05-03 22:41:35 +0000
LzI0NzY1NzZlLWNkOGQtNDhmNy	pg84-images.epub	<unknown>	<unknown>	2017-05-03 22:41:35 +0000	2017-05-03 22:41:35 +0000
LzI0NzY1NzZlLWNkOGQtNDhmNy	pg844-images.epub	<unknown>	<unknown>	2017-05-03 22:41:35 +0000	2017-05-03 22:41:35 +0000
LzI0NzY1NzZlLWNkOGQtNDhmNy	pg98-images.epub	<unknown>	<unknown>	2017-05-03 22:41:35 +0000	2017-05-03 22:41:35 +0000
LzI0NzY1NzZlLWNkOGQtNDhmNy	11-pdf.pdf	<unknown>	<unknown>	2017-05-03 22:41:35 +0000	2017-05-03 22:41:35 +0000

Figure 24 – Query engine cmis:rodaDocument select

All the objects retrieved were from the "**cmis:rodaDocument**" type. Since the "**cmis:rodaDocument**" object type extends "**cmis:document**", all the retrieved objects not only had their own attributes, but also the attributes inherited from their parent type. The "**metadata:ead:unitId**" and "**metadata:ead:unitTitle**" in Figure 24 are examples of attributes belonging only to the "cmis:rodaDocument" object type.

Search Over Metadata

One important feature of our repository is the possibility to perform searches over the contents metadata. Figure 25 shows the execution of a query over the repository where we started by selecting the file "**pexels-photo-147485.jpg**". The executed query was "**SELECT cmis:objectTypeId, metadata:ead:originationCreator, metadata:ead:originationProducer FROM cmis:rodaDocument WHERE cmis:name LIKE 'pexels-photo-147485.jpg'**"



cmis:objectId	cmis:name	cmis:objectTypeId	metadata:ead:originationCreator	metadata:ead:originationProducer
LzI0NzY1NzZlLWNkOGQtNDhmNy1iMGM0LTM	pexels-photo-147485.jpg	cmis:rodaDocument	Various websites	RODA IN
LzRkNDVhMWI2LWE2Y2QtNDkyNi05MDVkLTg	pexels-photo-147485.jpg	cmis:rodaDocument	Pexels.com	Pexels.com

Figure 25 – Metadata search – Example 1

We selected this file since it existed in the repository inside two different folders with different metadata associated to it.

In Figure 26 we added the clause "**AND metadata:ead:originationProducer LIKE 'Pexels.com'**" to select one specific file. This allowed us to filter the result list to show only the file containing the value "**Pexels.com**" defined in its "**metadata:ead:originationProducer**" field and not both files, as shown in Figure 25.

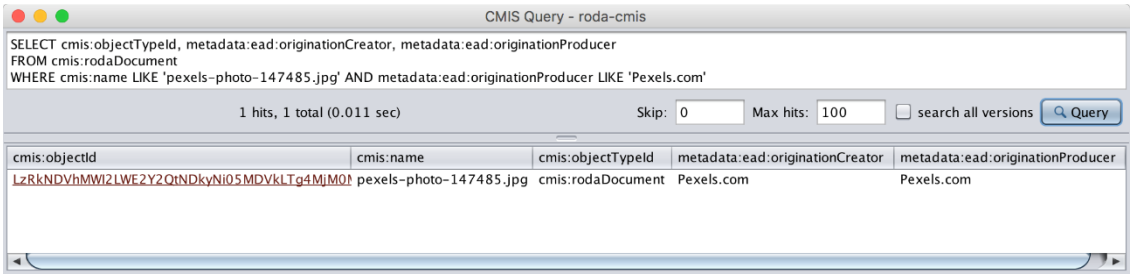


Figure 26 – Metadata search – Example 2

In Figure 27 we defined a different filter for the field "metadata:ead:originationProducer". By changing the filter to "**RODA%**", we demonstrated the query engine capability of searching over the objects metadata, not only with a defined term as a filter, but also with the wildcard character "**%**".

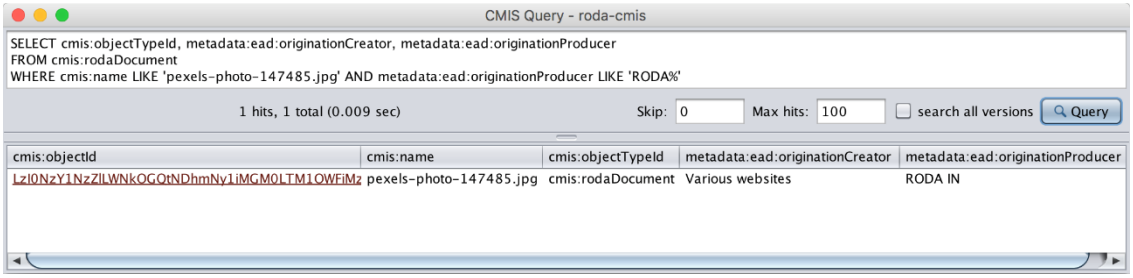


Figure 27 – Metadata search – Example 3

Navigational functions

The CMIS specification defines the implementation of the IN_FOLDER navigational function. The function received the folder ID, which was the folder "cmis:objectId", as first parameter and performed the search inside the folder corresponding to that ID.

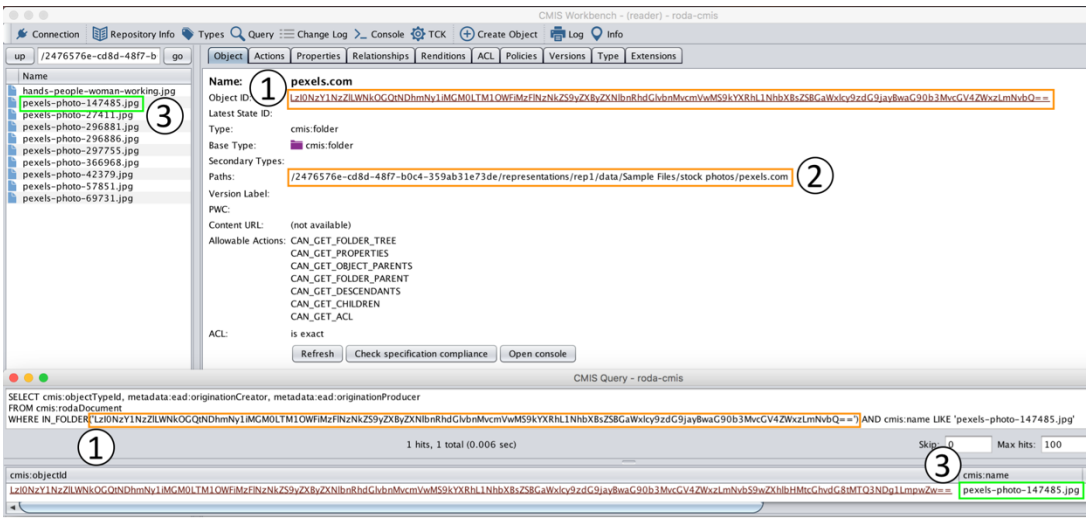


Figure 28 – IN_FOLDER navigational function

In Figure 28 we show the CMIS Workbench query window, at the bottom, and the folder where we performed a search, in the file browser window. We used the same example in Figure 25 where we selected the file "**pexels-photo-147485.jpg**". There were two files with the same name inside our repository. The folder ID, identified in Figure 28 with number 1, was passed by parameter to the function.

The list retrieved in the query window presented only one file, identified in Figure 28 by the number 3, out of the possible two existing in the repository. By clicking in the resulting file "cmis:objectId" in the list, the file browser showed us the details of the file where we confirmed the resulting file was indeed inside the chosen folder, identified by number 2.

Figure 29 shows us the implementation of another navigational function specified by the CMIS protocol, the IN_TREE. Similar to the IN_FOLDER function, the IN_TREE function is aimed to perform a search inside a given path, drilling down from a starting folder to the leaf folders inside it.

In Figure 29 we used the same example as the one we used in Figure 28 with a slight adjustment. We started our search inside the folder "**Sample Files**", identified by number 3 in Figure 29, instead of the folder "Pexels.com". Since "Pexels.com" folder itself was inside the "Sample Files" folder (belongs to its tree), by passing the IN_TREE function "Sample Files"

folder ID by parameter, identified with number 1 in Figure 29, and searching for the same "**pexels-photo-147485.jpg**", the query should yield the same result.

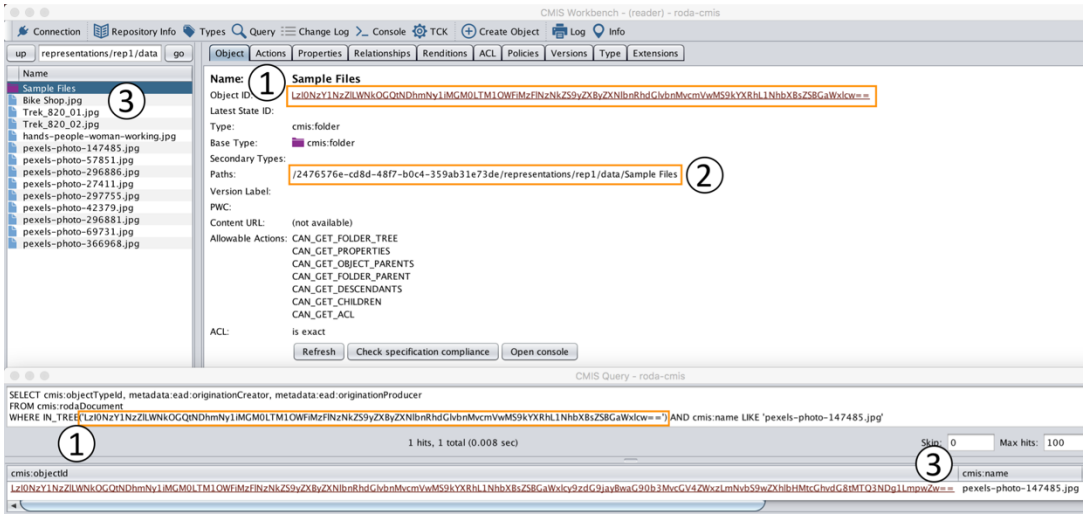


Figure 29 – IN_TREE navigational function

The performed query retrieved only one result, the "pexels-photo-147485.jpg". By clicking the file "cmis:objectId" in the result list, we were able to confirm the file was indeed contained in a folder within the tree selected for searching, identified by number 2.

6. Conclusions and Future Work

This thesis had the goal of presenting the current main strategies for the digital preservation of information. It was our goal to contribute to the field by proposing and implementing an improvement to an existing repository software. To accomplish our endeavours the most adequate situation would be choosing a repository, study its characteristics and implement the improvement. We opted for the open source repository software RODA, not only for the availability of access to the source code, but also for having the desired characteristics and being the most appropriate for an academic scenario.

From the current digital preservation strategies, the most modern and challenging one is federation. This strategy involves, among other, the existence of mechanisms for interoperability between repositories. The creation of such mechanism posed as a very appealing challenge to us, leading us to choose it as the one to implement. We chose CMIS as our communication protocol between client applications and our server, first for being an already existing and mature protocol and secondly for having its implementation available through the Apache Chemistry project in the form of a Java library, the language in which RODA is also implemented.

We implemented a CMIS v1.0 and 1.1 compliant server. Our server is not fully compliant with the specification as there are some functions that were not fully implemented, namely the navigational functions IN_FOLDER and IN_TREE from the query engine, as well as, the functions CONTAINS and SCORE which were not implemented at all. Due to time limitations, we did not develop any client application. Instead we used the Apache Chemistry CMIS Workbench client application as our testing suite.

Our implementation[6] is proof that RODA can be adapted to share information in a controlled environment. Through the RODA-OCS prototype, RODA users have the possibility to choose and share contents. Integrated with RODA build in permissions system and with read-only access, our implementation eliminates the risk of any content being altered or deleted by unauthorized users. All the server content is browsable either through the file browser functions or the query engine. The query engine is also capable of performing searches over the contents metadata.

[6] RODA CMIS Server implementation: https://bitbucket.org/meicmdigitalpreservation/roda-ocs-prototype

6.1. Future Work

As future work, there are some features to be improved and others which could be fully implemented:

i) An automated detection system for changes in the metadata. Our server is not capable of updating its contents metadata if any changes occur after the server was started. This could be accomplished through the Java events mechanism.

ii) Give the server support to use other JDBC compatible databases besides SQLite, i.e. MySQL or MariaDB.

iii) Implement the Full Text Search functions CONTAINS and SCORE in the query engine by integrating Apache Lucene [104] full-text search engine into RODA-OCS.

iv) Conclude the IN_FOLDER and IN_TREE specification implementation.

References

[1] D. S. Ross, “Changing Trains at Wigan: Digital Preservation and the Future of Scholarship,” National Preservation Office, London, 2000.

[2] S.-S. Chen, “The paradox of digital preservation,” *Computer,* vol. 34, nº 3, pp. 24-28, 2001.

[3] J. Gantz e D. Reinsel, “The digital universe in 2020: Big data, bigger digital shadows, and biggest growth in the far east,” IDC, 2012.

[4] M. Sharabayko e N. Markov, “H. 264/AVC Video Compression on Smartphones,” *Journal of Physics: Conference Series,* vol. 803, nº 1, 2017.

[5] S. Bandyopadhyay, M. Sengupta, S. Maiti e S. Dutta, “Role of middleware for internet of things: A study,” *International Journal of Computer Science and Engineering Survey,* vol. 2, nº 3, pp. 94-105, 2011.

[6] C. Lynch, “Big data: How do your data grow?,” *Nature,* vol. 455, nº 7209, pp. 28-29, 2008.

[7] P. B. Hirtle, "The history and current state of digital preservation in the United States.," *Metadata and Digital Collections: A Festschrift in Honor of Thomas P. Turner,* 2010.

[8] M. Hedstrom, “Digital preservation: a time bomb for digital libraries,” *Computers and the Humanities,* vol. 31, nº 3, pp. 189-202, 1997.

[9] O. C. P. D. Carlos André Rosa, “Open Source Software for Digital Preservation Repositories: A Survey,” *International Journal of Computer Science & Enginee ring Survey (IJCSES),* vol. 8, pp. 21-39, June 2017.

[10] J. Miranda, "Web Harvesting and Archiving," [Online]. Available: http://web.ist.utl.pt/joaocarvalhomiranda/docs/other/web_harvesting_and_archiving.pdf. [Accessed 24th November 2016].

[11] D. Gomes, J. Miranda e M. Costa, “A survey on web archiving initiatives,” em *International Conference on Theory and Practice of Digital Libraries*, 2011.

[12] International Internet Preservation Consortium, "About International Internet Preservation Consortium," International Internet Preservation Consortium, 2012. [Online]. Available: http://www.netpreserve.org/about-us. [Accessed 24th November 2016].

[13] J. Bailey and M. LaCalle, "State of the WARC: Our Digital Preservation Survey Results," Archive-It, 5th January 2016. [Online]. Available: https://archive-it.org/blog/post/state-of-the-warc-our-digital-preservation-survey-results/. [Accessed 24th November 2016].

[14] D. P. Coalition, "Digital Preservation Briefing, Digital Preservation Handbook, 2nd Edition," 2015. [Online]. Available: http://handbook.dpconline.org/docman/digital-preservation-handbook2/1552-dp-handbook-digital-preservation-briefing/file. [Accessed 1st October 2016].

[15] C. A. Rosa, “MEI-CM Digital Preservation Repositories,” 11 Sep 2017. [Online]. Available: https://bitbucket.org/meicmdigitalpreservation/. [Acedido em 11 Sep 2017].

[16] Wikpedia, "Digital preservation," Wikpedia, [Online]. Available: http://en.wikipedia.org/wiki/Digital_preservation. [Accessed 1st October 2016].

[17] L. T. Stuchell, "What is Digital Preservation?," University of Michigan, [Online]. Available: http://www.lib.umich.edu/preservation-and-conservation/digital-preservation/what-digital-preservation. [Accessed 1st October 2016].

[18] I. R. M. Trust, "Understanding Digital Records," May 2016. [Online]. Available: http://www.ica.org/sites/default/files/Digital%20Preservation%20Initatives%20Module_0.pdf. [Accessed 1st October 2016].

[19] F. Luan, M. Nygård, T. Mestl e D. o. C. a. I. Science, “A survey of digital preservation strategies,” [Online]. Available: https://research.idi.ntnu.no/longrec/papers/WDL.pdf. [Acedido em 30 October 2016].

[20] M. Ferreira, Introdução à preservação digital: conceitos, estratégias e actuais consensos., 2006.

[21] Wikipedia, "Metadata Wikipedia," Wikipedia, 17 June 2017. [Online]. Available: https://en.wikipedia.org/wiki/Metadata. [Accessed 07 July 2017].

[22] Wikipedia, "Encoded Archival Description," Wikipedia, 26 March 2017. [Online]. Available: https://en.wikipedia.org/wiki/Encoded_Archival_Description. [Accessed 8 July 2017].

[23] D. C. M. Initiative, "Dublin Core Metadata Element Set, Version 1.1," Dublin Core Metadata Initiative, 2016. [Online]. Available: http://dublincore.org/documents/dces/. [Accessed 28th November 2016].

[24] A. L. o. Congress, "Preservation Metadata Maintenance Activity," American Library of Congress, 16 Nov 2016. [Online]. Available: http://www.loc.gov/standards/premis/. [Accessed 26 Feb 2017].

[25] M. J. G. Ronald Jantz, "Architecture and Technology for Trusted Digital Repositories," *D-Lib Magazine,* vol. 11, no. Digital Preservation, p. 6, 2005.

[26] Consultative Committee Space Data System, “Reference Model for an Open Archival Information System (OAIS). Recommendation for Space Data System Practices, CCSDS 650.0-M-2,” 2012.

[27] B. Lavoie, “The Open Archival Information System (OAIS) Reference Model: Introductory Guide (DOI:dx.doi.org/10.7207/twr14-02),” Digital Preservation Coalition, GB, 2014.

[28] H. v. d. Sompel, M. L. Nelson, C. Lagoze e S. Warner, “Resource harvesting within the OAI-PMH framework,” *D-Lib Magazine,* vol. 10, nº 12, 2004.

[29] S. H. McCallum, “A look at new information retrieval protocols: SRU, opensearch/A9, CQL, and XQUERY,” em *World Library and Information Congress: 72nd IFLA General Conference and Council*, 2006.

[30] S. Weibel, J. Kunze, C. Lagoze e M. Wolf, “Dublin Core metadata for resource discovery - RFC 2413,” 1998.

[31] P. Caplan e R. S. Guenther, “Practical preservation: the PREMIS experience,” *Library Trends,* vol. 54, nº 1, pp. 111-124, 2005.

[32] G. Rebecca, “The Application/MARC Content-type - RFC 2220,” 1997.

[33] D. V. Pitti, “Encoded archival description: An introduction and overview,” *New Review of Information Networking,* vol. 5, nº 1, pp. 61-69, 1999.

[34] R. Guenther e S. McCallum, “New metadata standards for digital resources: MODS and METS,” *Bulletin of the American Society for Information Science and Technology,* vol. 29, nº 2, pp. 12-15, 2003.

[35] J. Radebaugh, “MARC21/MARCXML,” *Computers in Libraries,* vol. 27, nº 4, p. 15, 2007.

[36] I. S. Burnett, S. J. Davis e G. M. Drury, “MPEG-21 digital item declaration and Identification-principles and compression,” *IEEE Transactions on Multimedia,* vol. 7, nº 3, pp. 400-407, 2005.

[37] American Library of Congress, "Metadata for Images in XML (MIX)," American Library of Congress, 23 Nov 2015. [Online]. Available: http://www.loc.gov/standards/mix/. [Accessed 26 Feb 2017].

[38] R. Gartner, H. L'Hours e G. Young, Metadata for digital libraries: state of the art and future directions, JISC, 2008.

[39] J. Daemen e V. Rijmen, The design of Rijndael: AES - the Advanced Encryption Standard, Springer Science & Business Media, 2013.

[40] E. H. Schnell, “DocMD (DOCument Mediated Delivery),” *Journal of Hospital Librarianship,* vol. 3, nº 3, pp. 25-37, 2003.

[41] American Library of Congress, "Technical Metadata for audio and video," 2014. [Online]. Available: http://www.loc.gov/standards/amdvmd/. [Accessed 30 April 2017].

[42] Library of Laval University, "ARCHIMEDE : A canadian software solution for institutional repositories," Laval University Library, 2005. [Online]. Available: http://www.bibl.ulaval.ca/archimede/index.en.html. [Accessed 26 Feb 2017].

[43] Florida Center for Library Automation (FLCA), "DAITSS Digital Preservation Repository Software," Florida Center for Library Automation (FCLA), 2011. [Online]. Available: https://daitss.fcla.edu/. [Accessed 26 Feb 2017].

[44] P. Van Garderen, "Archivematica: Using micro-services and open-source software to deliver a comprehensive digital curation solution," em *Proceedings of the 7th International Conference on Preservation of Digital Objects*, Vienna, Austria, 2010.

[45] DURASPACE, "DSpace Repository," DURASPACE, 2016. [Online]. Available: http://www.dspace.org/. [Accessed 10th November 2016].

[46] "The 3-Clause BSD License," Open Source Initiative, [Online]. Available: https://opensource.org/licenses/BSD-3-Clause. [Accessed 25 Feb 2017].

[47] M. Beazley, "EPrints institutional repository software: A review," *Partnership: The Canadian Journal of Library and Information Practice and Research,* vol. 5, nº 2, 2010.

[48] DURASPACE, "Fedora Repository," DURASPACE, 2016. [Online]. Available: http://fedorarepository.org/. [Accessed 12th November 2016].

[49] Greenstone, "Greenstone Digital Repository Software," Greenstone, 2016. [Online]. Available: http://www.greenstone.org/. [Accessed 12th November 2016].

[50] CERN, Conseil Européen pour la Recherche Nucléaire, "Invenio Digital Library Framework," CERN, Conseil Européen pour la Recherche Nucléaire, 2016. [Online]. Available: http://invenio-software.org/. [Accessed 26 Feb 2017].

[51] V. Reich e D. S. Rosenthal, "LOCKSS (lots of copies keep stuff safe)," *New Review of Academic Librarianship,* vol. 6, nº 1, pp. 155-161, 2000.

[52] Keep Solutions, "RODA - Repository for Authentic Digital Objects: Characteristics and Technical requirements," 4th October 2013. [Online]. Available:

https://www.keep.pt/wp-content/uploads/2012/01/WP13139.1-RODA-whitepaper.pdf. [Accessed 14th November 2016].

[53] National Archives of Australia, "Xena Digital Preservation Software," National Archives of Australia, 31 Jun 2013. [Online]. Available: http://xena.sourceforge.net/. [Accessed 26 Feb 2017].

[54] American Library of Congress, "MARC Standards," American Library of Congress, 13 Dec 2016. [Online]. Available: https://www.loc.gov/marc/. [Accessed 30 April 2017].

[55] American Library of Congress, "Preservation Metadata Maintenance Activity," American Library of Congress, 16 Nov 2016. [Online]. Available: http://www.loc.gov/standards/premis/. [Accessed 26 Feb 2017].

[56] American Library of Congress, "Metadata Encoding & Transmission Standards (METS)," American Library of Congress, 9 Aug 2016. [Online]. Available: http://www.loc.gov/standards/mets/. [Accessed 26 Feb 2017].

[57] American Library of Congress, "Metadata Object Description Schema (MODS)," American Library of Congress, 1 Feb 2016. [Online]. Available: http://www.loc.gov/standards/mods/. [Accessed 26 Feb 2017].

[58] Research Library of the Los Alamos National Laboratory, "MPEG-21 Part 2: Digital Item Declaration Language (DIDL)," Research Library of the Los Alamos National Laboratory, 16 Feb 2004. [Online]. Available: http://xml.coverpages.org/mpeg21-didl.html. [Accessed 26 Feb 2017].

[59] American Library of Congress, "Technical Metadata for Text (TextMD)," American Library of Congress, [Online]. Available: https://www.loc.gov/standards/textMD/. [Accessed 27 Feb 2017].

[60] L. Sheble, I. H. Witten, R. d. Vries, R. Brown and G. Marchionini, "Greenstone User and Developer Survey 2009," Greenstone, 2009. [Online]. Available: https://greenstonesurvey.wordpress.com/greenstone-user-and-developer-survey-results/section-i-background-information/. [Accessed 27 Feb 2017].

[61] Keep Solutions, "2012 RODA Promotional Flyer," May 2012. [Online]. Available: https://www.keep.pt/wp-content/uploads/2012/05/2012-roda-promo-en.pdf. [Accessed 9th November 2016].

[62] J. Ramalho, M. Ferreira, L. Faria, R. Castro, F. Barbedo e L. Corujo, “RODA and CRiB a service-oriented digital repository,” em *Fifth International Conference on Preservation of Digital Objects*, London, UK, 2008.

[63] OpenLDAP Foundation, "OpenLDAP," OpenLDAP Foundation, 2017. [Online]. Available: https://www.openldap.org/. [Accessed 27 Feb].

[64] M. Still, The definitive guide to ImageMagick, Apress, 2006.

[65] I. Artifex Software, "GhostScript," Artifex Software, Inc., 2016. [Online]. Available: https://www.ghostscript.com/. [Accessed 27 Feb 2017].

[66] Art of Solving, Ltd., "JODConverter, the Java OpenDocument Converter," Art of Solving, Ltd., 2010. [Online]. Available: http://www.artofsolving.com/opensource/jodconverter.html. [Accessed 27 Feb 2017].

[67] "Mencoder," [Online]. Available: http://www.mplayerhq.hu. [Accessed 30 April 2017].

[68] G. Portet, "SoundConverter - GNOME Sound Convertion," 2014. [Online]. Available: http://soundconverter.org/. [Accessed 27 Feb 2017].

[69] "GStreamer: Open-source Multimedia Framework," 27 Feb 2017. [Online]. Available: https://gstreamer.freedesktop.org/. [Accessed 27 Feb 2017].

[70] S. Abrams, S. Morrissey e T. Cramer, “"What? So What": The Next-Generation JHOVE2 Architecture for Format-Aware Characterization,” *International Journal of Digital Curation,* vol. 4, nº 3, pp. 123-136, 2009.

[71] K. McHenry e P. Bajcsy, “An overview of 3D data content, file formats and viewers,” National Center for Supercomputing Applications, 2008.

[72] M. Smith, M. Bass, G. McClellan, R. Tansley, M. Barton, M. Branschofsky, D. Stuve e J. Walker, “DSpace: An Open Source Dynamic Digital Repository,” *D-Lib Magazine,* vol. 9, nº 1, 2003.

[73] DURASPACE, "DSpace Technical Specifications," 2016. [Online]. Available: http://www.dspace.org/sites/dspace.org/files/media/specsh_dspace.pdf. [Accessed 10th November 2016].

[74] S. Consortium, "Shibboleth Open-source Single Sign-On," 2017 Shibboleth Consortium. [Online]. Available: https://shibboleth.net/. [Accessed 27 Feb 2017].

[75] S. Payette e C. Lagoze, “Flexible and extensible digital object and repository architecture (FEDORA),” em *International Conference on Theory and Practice of Digital Libraries*, 1998.

[76] J. Pérez, M. Arenas e C. Gutierrez, “Semantics and Complexity of SPARQL,” em *International semantic web conference*, 2006.

[77] DURASPACE, "Fedora Technical Specifications," DURASPACE, 2016. [Online]. Available: http://fedorarepository.org/sites/fedorarepository.org/files/specsh_fedora.pdf. [Accessed 12th November 2016].

[78] L. Richardson e S. Ruby, RESTful web services, O'Reilly Media, Inc., 2008.

[79] R. Lasher and D. Cohen, "RFC 1807 - A Format for Bibliographic Records," June 1995. [Online]. Available: https://tools.ietf.org/html/rfc1807. [Accessed 27 Feb 2017].

[80] Commonwealth of Australia, "AGLS Metadata Standard," Commonwealth of Australia, 09 Jun 2015. [Online]. Available: http://www.agls.gov.au/. [Accessed 25 Feb 2017].

[81] K. Booth e J. Napier, “Linking people and information: Web site access to National Library of New Zealand information and services,” *The Electronic Library,* vol. 21, nº 3, pp. 227-233, 2003.

[82] Electronics & Computer Science, University of Southampton, "EPrints for Open Access," 2016. [Online]. Available: http://www.eprints.org/uk/index.php/openaccess/. [Accessed 13th November 2016].

[83] University of Southampton, , "ROARMAP - Registry of Open Access Repositories Mandates and Policies," [Online]. Available: http://roarmap.eprints.org/view/country/un=5Fgeoscheme.html. [Accessed 26 Feb 2017].

[84] Artefactual Systems, Inc., "Archivematica: Open-source digital preservation system," Artefactual Systems Inc., 2017. [Online]. Available: https://www.archivematica.org/en/. [Accessed 26 Feb 2017].

[85] S. Oberg, "Open Source Software," *Serials Review,* vol. 29, nº 1, pp. 36-39, 2003.

[86] Intralect, "IntraLibrary," Intralect, 20 Jan 2017. [Online]. Available: https://intrallect.com/tag/intralibrary/. [Accessed 27 Feb 2017].

[87] University of Florida Digital Collections, "SobekCM : Digital Content Management System," 2015. [Online]. Available: http://sobekrepository.org/. [Accessed 27 Feb 2017].

[88] Konrad-Zuse-Zentrum, "Opus Repository," Konrad-Zuse-Zentrum, 2012. [Online]. Available: http://www.opus-repository.org/. [Accessed 27 Feb 2017].

[89] MyCoRe, "MyCoRe, Your framework for presentation and management of digital content," MyCoRe, 10 Nov 2016. [Online]. Available: http://www.mycore.de/. [Accessed 27 Feb 2017].

[90] Apache, "Apache Chemistry," The Apache Software Foundation, 31 Jul 2017. [Online]. Available: http://chemistry.apache.org/. [Acedido em 15 Sep 2017].

[91] Wikipedia, "Representational State Transfer," Wikipedia, 10 Sep 2017. [Online]. Available: https://en.wikipedia.org/wiki/Representational_state_transfer. [Acedido em 15 Sep 2017].

[92] Wikipedia, "Content Management Interoperability Services," Wikipedia, 21 June 2017. [Online]. Available:

https://en.wikipedia.org/wiki/Content_Management_Interoperability_Services. [Accessed 8 July 2017].

[93] SQLIte, “SQLite Home Page,” 24 Aug 2017. [Online]. Available: https://www.sqlite.org/. [Acedido em 15 Sep 2017].

[94] J. F. M. F. L. C. R. Ramalho, “Relational Database Preservation through XML modelling,” Montréal, Québec, 2007.

[95] A. Pereira, “RODA-in - A generic tool for the mass creation of submission information packages,” Lisbon, Portugal, 2016.

[96] OASIS, "Content Management Interoperability Services," OASIS Nonprofit Consortium, 2017. [Online]. Available: https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=cmis. [Accessed 8 July 2017].

[97] OASIS, “Content Management Interoperability Services (CMIS) Version 1.1,” 23 May 2013. [Online]. Available: http://docs.oasis-open.org/cmis/CMIS/v1.1/os/CMIS-v1.1-os.html. [Acedido em 15 September 2017].

[98] F. M. Jay Brown, "CMISDocs Server Development Guide," GitHub, 21 October 2014. [Online]. Available: https://github.com/cmisdocs/ServerDevelopmentGuide. [Accessed 12 December 2016].

[99] F. M. Jay Brown, "OpenCMIS Server Development Guide," 25 March 2014. [Online]. Available: https://github.com/cmisdocs/ServerDevelopmentGuide/blob/master/doc/OpenCMIS%20Server%20Development%20Guide.pdf?raw=true. [Accessed 12 December 2016].

[100] Apache, “Apache Chemistry CMIS Workbench,” Apache, 5 Apr 2017. [Online]. Available: https://chemistry.apache.org/java/developing/tools/dev-tools-workbench.html. [Acedido em 5 Aug 2017].

[101] Wikipedia, “SQL-92,” Wikipedia, 8 Jul 2017. [Online]. Available: https://en.wikipedia.org/wiki/SQL-92. [Acedido em 5 Aug 2017].

[102] D. A. S. Board, “E-ARK COMMON SPECIFICATION FOR INFORMATION PACKAGES,” 16 Dec 2016. [Online]. Available: http://www.eark-project.com/resources/specificationdocs/67-e-ark-draft-common-specification-ver-017/file. [Acedido em 8 Aug 2017].

[103] Wikipedia, “Universally Unique Identifier (UUID),” 11 Sep 2017. [Online]. Available: https://en.wikipedia.org/wiki/Universally_unique_identifier. [Acedido em 15 Sep 2017].

[104] OASIS, “Content Management Interoperability Services (CMIS) Version 1.0,” 04 Nov 2011. [Online]. Available: http://docs.oasis-open.org/cmis/CMIS/v1.0/cmis-spec-v1.0.html. [Acedido em 21 Aug 2017].

[105] Apache, “Apache Lucene Full-Text Search Engine,” The Apache software Foundation, 7 Sep 2017. [Online]. Available: https://lucene.apache.org/. [Acedido em 15 Sep 2017].

[106] JISC, "Repository software survey, November 2010," JISC (non-profit organization), November 2010. [Online]. Available: http://www.rsp.ac.uk/start/software-survey/results-2010/. [Accessed 1st February 2017].

[107] A. Adewumi e N. Omoregbe, “Institutional repositories: features, architecture, design and implementation technologies,” *Journal of Computing,* vol. 8, nº 2, 2011.

[108] M. Pickton, D. Morris, S. Meece, S. Coles e S. Hitchcock, “Preserving repository content: practical tools for repository managers,” *Journal of Digital Information,* vol. 12, nº 2, 2011.