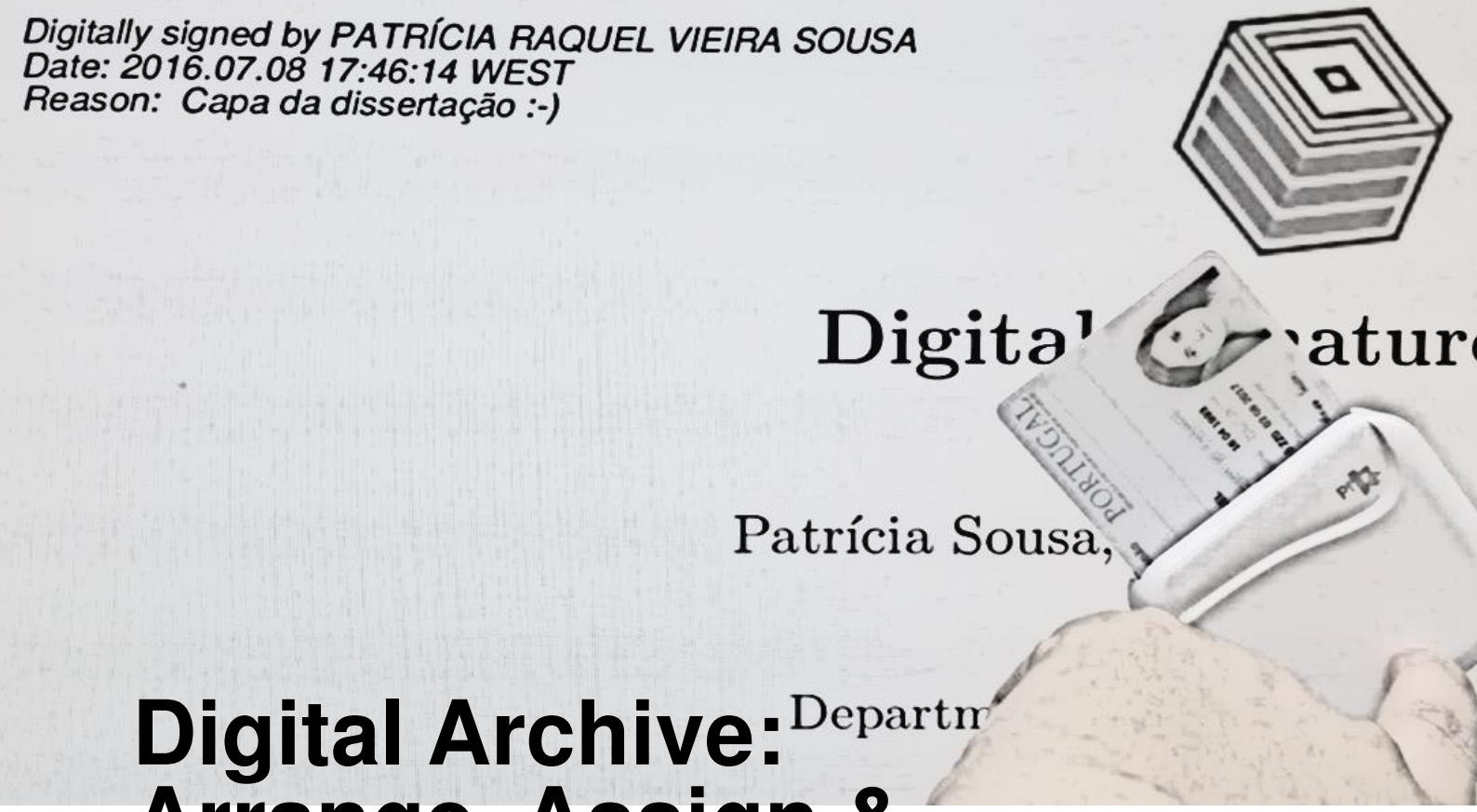
Digitally signed by PATRÍCIA RAQUEL VIEIRA SOUSA
Date: 2016.07.08 17:46:14 WEST
Reason: Capa da dissertação :-)



Digital Archive: Arrange, Assign & Sign!

Patrícia Raquel Vieira Sousa
Mestrado Integrado em Engenharia de Redes e Sistemas Informáticos
Departamento de Ciência de Computadores
2016

Orientador
Prof. Dr. Luís Filipe Coelho Antunes, Professor Auxiliar, FCUP

Coorientador
Eng. Pedro Vasconcelos Castro Lopes Faria, Consultor, HealthySyste

Todas as correções determinadas pelo júri, e só essas, foram efetuadas.

O Presidente do Júri,

Porto, _____/_____/_____

Patrícia Raquel Vieira Sousa

Digital archive: Arrange, Assign & Sign!

Departamento de Ciência de Computadores
Faculdade de Ciências da Universidade do Porto
Junho de 2016

Patrícia Raquel Vieira Sousa

Digital archive: Arrange, Assign & Sign!

Dissertação submetida à Faculdade de Ciências da
Universidade do Porto como parte dos requisitos para a obtenção do grau de
Mestre em Engenharia de Redes em Sistemas Informáticos

Orientador: Prof. Dr. Luís Filipe Coelho Antunes
Co-orientador: Eng. Pedro Vasconcelos Castro Lopes Faria

Departamento de Ciência de Computadores
Faculdade de Ciências da Universidade do Porto
Junho de 2016

To my parents and my love for all the support and patience...

Acknowledgments

I want to thank my advisor Professor Luís Antunes for the support in all this time of work, for all suggestions that improved my thesis and for this opportunity.

I also want to thank my co-advisor Pedro Faria for all dedication, support, patience and guidance that made all this work possible. Thanks for innovative ideas and the security knowledge transmitted during this thesis.

I also want to thank Professor Manuel Eduardo Correia for the interest that has always shown for this thesis work and for all the support and ideas.

For all the people of Healthy Systems, specially to Ricardo Gonçalves for the team work in the development of digital signatures application.

I want to thank my best friends for their patience, for their support and also the moments of happiness during some difficult hours.

To my parents, because without them would never be where I am and would not be what I am today.

Finally, last but not the least, for my love João Resende, for all the support not only this thesis but in all my life. For all the patience and for all unconditional love.

Resumo

Existem alguns obstáculos para evitar o papel num escritório (paperless office). Um dos obstáculos é a recolha de assinaturas, visto que, segundo um estudo, metade do papel que é impresso é para as pessoas fazerem assinturas, visto que não conhecem uma maneira ou não confiam nas maneiras que existem para efectuar assinaturas digitalmente. As assinaturas digitais têm o mesmo valor legal que as assinaturas manuscritas, desde que tenham um certificado emitido por uma entidade certificadora. Nesta dissertação, propomos uma plataforma para gestão de workflows de assinaturas digitais que integra um token seguro baseado em assinaturas digitais com um gestor documental empresarial como o *Alfresco*, onde cada utilizador pode associar um conjunto de cartões inteligentes no seu perfil para assinar documentos. Os documentos podem ser assinados com um cartão de cidadão ou outro cartão inteligente que tenha a capacidade de fazer assinaturas digitais. Implementamos um módulo no *Alfresco* que nos permite explorar muitas técnicas de workflow para simular uma tarefa real de assinaturas, como as pessoas fazem, por exemplo, quando passam um documento de papel entre vários departamentos para o papel ser assinado. Como os utilizadores podem ver o estado corrente de um documento que está a ser assinado, durante o processo inteiro de assinatura no workflow, são preservadas algumas propriedades importantes de segurança, como a confiança no sistema. Nesta dissertação descrevemos um serviço web externo de validação, que fornece uma forma dos utilizadores validarem um documento assinado. O serviço de validação mostra propriedades de segurança importantes do documento, como marcas temporais (timestamps), certificados do cartão e destaca a integridade do documento, com base nas assinaturas digitais, que foram feitas durante os workflows definidos pelo módulo do *Alfresco* desenvolvido no âmbito deste trabalho.

Abstract

There are some obstacles, towards a paperless office. One of them is the collection of signatures, since nearly half of all documents are printed for the sole purpose of collecting them. Digital signatures can have the same legal evidential validity as handwritten signatures, provided they are based on certificates issued by accredited certification authorities and the associated private keys are stored on tamper proof token security devices like smart cards. In this work, we propose a platform for secure digital signature workflow management that integrates secure token based digital signatures with the Enterprise Content Management *Alfresco*, where each user can associate a set of smart cards to his account. The documents can then be signed with the citizen card or other smart card that has digital signatures capabilities. We have implemented an *Alfresco* module that allows us to explore several workflow techniques to implement real task secure digital signatures workflows, as people for example do when they pass a paper document between various departments to be signed. Since all users can see the current state of the documents being signed during the entire signage process, important security properties like system trust are preserved. We also describe an external validation web service, that provides a way for users to validate signed documents. The validation service then shows to the user important document security properties like timestamps, certificates attributes and highlights the document integrity in face of the digital signatures that have been collected in the workflows defined by our module in *Alfresco*.

List of Acronyms

AATL Adobe Approved Trust List

AES Advanced Encryption Standard

AIIM Association for Information and Image Management

AIO All-in-One archetype

AJAX Asynchronous Javascript and XML

AMP Alfresco Module Package

AOP Aspect-Oriented Programming

API Application Programming Interface

ASP Active Server Pages

BPM Business Process Management

BPMN Business Process Model and Notation

CA Certification Authority

CAdES CMS Advanced Electronic Signatures

CAS Content Application Server

CMS Content Management Systems

CMIS Content Management Interoperability Services

CDS Certified Document Services

CRL Certificate Revocation List

DAM Digital Asset Management

DES Data Encryption Standard

DLP Discrete Logarithm Problem

DM Document Management

DMS Document Management System

DSA Digital Signature Algorithm

EAL Evaluation Assurance Level

ECM Enterprise Content Management

EDM Electronic Document Management

ElGamal ElGamal Encryption

FIPS Federal Information Processing Standard

FTL FreeMarker Template Language

HTML HyperText Markup Language

HTTP Hypertext Transfer Protocol

HTTPS Hyper Text Transfer Protocol Secure

IDE Integrated Development Environment

IT Information Technology

IIS Internet Information Services

IV Initialization Vector

JAR Java ARchive

JDK Java Development Kit

JS JavaScript

JSP JavaServer Pages

JSON JavaScript Object Notation

OCSP Online Certificate Status Protocol

MAC Message Authentication Code

MIME Multipurpose Internet Mail Extension

MMT Module Management Tool

MVC Model View Controller

NPAPI Netscape Plugin Application Programming Interface

PKCS Public Key Cryptography Standards

PAdES PDF Advanced Electronic Signature

PC Personal Computer

PDF Portable Document Format

PKI Public Key Infrastructure

PIN Personal Identification Number

PO Page Object

QR Quick Response

RAD Rapid Application Development

REST Representational State Transfer

RFC Request For Comments

RM Records Management

RSA Rivest-Shamir-Adleman Cryptosystem

SaaS Software as a Service

SOAP Simple Object Access Protocol

SDK Software Development Kit

TDD Test Driven Development

TOE Target of Evaluation

TSA Time Stamp Authority

UI User Interface

URI Uniform Resource Identifiers

URL Uniform Resource Locator

USB Universal Serial Bus

VA Validation Authority

WAR Web application ARchive

WCM Web Content Management

XAdES XML Advanced Electronic Signatures

XML eXtensible Markup Language

YUI Yahoo! User Interface

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Documents which are in printed format have been used for many years, such as books, papers, forms, contracts and any related materials [2]. Nowadays, there are a lot of reasons why people might choose to paperless environments, including reduction of the environmental harm of paper consumption and the economic cost of paper production, transfer and storage. Digital environments release people or companies from the location and physical constraints of paper and provide better support for updating, archiving, and searching of documents [3].

With the evolution of Information Technology (IT) and computer systems, the documents have been managed by computer-based document management systems. A Document Management System (DMS) can be defined as a computer system that is used to store, manage, and retrieve electronic/digital documents on a closed client/server architecture network [4].

However, DMS were interested in the file and storage/indexing/retrieval mechanisms to allow the user to classify and retrieve documents. They were initially concerned only with the file as a container. But, as market needs changed, the DMS focus shifted from file management to content management. For example, if we have a Web site, it is composed of HyperText Markup Language (HTML), eXtensible Markup Language (XML), or Active Server Pages (ASP) pages that need to be managed. For this reason, the name of the system was changed to ECM [5].

According to the authors of documents [6] [7], going paperless is condemned to failure soon. Despite of many efforts which have been done to consume less paper, companies still use large amounts of paper. There are some obstacles towards a paperless office such as: read on screen is difficult for some people especially mid aged people it was

not that easy to adapt to computer and Internet, who don't like to read on monitors and prefer to read in paper; the risk of losing data and document due to software or hardware failure; the people have fear because despite electronic storage be safer than having data on paper, some people do not trust the authenticity or security of online tools. Signatures is another obstacle towards a paperless office and according to the authors of [1], nearly half of all documents are printed for the sole purpose of adding signatures, so, we want to focus on a solution to this. Also, we want to innovate in terms of security and protection data, so, we think in one solution that has the possibility of redact the document. With this possibility, users can hide some information in a document and protect sensible and confidential data for all users that has access to the document and a restrict user list that can access to the hidden data. This way, we can protect data from undesired users that may have access to the document. Furthermore, these features are important for paperless and, no longer necessary for these tasks using printers, scanners, pens and papers. Additionally, with a signature, and even with the redaction, the papers can be damaged, and it may take more work for people involve in the task. In the case of signatures, we may need the users back to sign a document or be falsifications. In the case of the redaction, the people that can receive the documents redacted not know who changed the file or also be falsified and people can damage the documents scratching with pens.

1.1 Motivation

There are two methods of transforming a company into paperless office. One of the methods is by automating the processes that normally use paper as an essential tool. There are several technologies to make this: enterprise data automation software that is used to integrate forms and data with systems that processes them; Form Technology that is used to design various types of forms; databases device that is used to replace the function of a filing cabinet, i.e., data is made into digital form and then stored in a database with sufficient security technology and finally, digital signature allow evidence of signature in digital form. Papers are generally used as business evidences. This is required in business transactions to generate legal binding between two or more parties and workflow platforms technology that is a processes flow of an office. Normally, paper documents are used to transfer a data to other departments so that it can continue doing what is needed next (for example, one document is transferred to

[1] ALA's Legal Management: www.arx.com/files/uploads/2014/11/CoSign_ALA_Legal Management.jpg, 2014. [Online; Accessed 2015-12-17]

other department to be signed). This flow of work can be documented and transferred in digital form, using the workflow platforms.

The second method of transforming a company into paperless office is data storage transformation. In a general office, the data is normally stored and protected in a filing cabinet. This turns out to fill offices full of useless paper. Using the "Paperless Office" technology, all this data can be transformed to a digital form very easily. Some of the tools available to support this process: Scanners, book copiers, photo scanners, fax to PDF converter and more. One of the most important tools are ECM systems [8].

This two methods of transforming a company into paperless office leads us to a solution that could combine the technologies to automate processes that typically use the paper an essential tool, with a tool to store digital information, for example a ECM system, as stated above.

1.2 Proposed Solution

The work detailed in this thesis aims to provide companies a way to be able to automate their signatures processes to avoid transferring a printed data between departments. This type of transferring can result in loss of important documents or falsification of documents/signatures using printed paper. We want that companies to be able to involve several people in the automated process of signatures, safely in a ECM system. This leads companies to also benefit from a printed paper reduction and reduction of the loss of important documents due to the Internet advantages.

We will focus in integrating a digital signatures systems with an ECM system. This allows users to sign documents in a document manager, so users can also save their documents online, in digitally format. We take advantage of the workflow feature that some ECM systems have. Thus, we provide users a way to create a workflow signatures in a ECM system, so multiple users may sign the same document and can see the state of the document. We provide a secure way to users sign documents, through a smart card. The middleware to the Citizen Card was not in the dissertation goals so, we used a module that already existed in the *Healthy Systems* company. We also want to include the possibility of redaction in the signatures workflow, providing the way to users redact a document for inviting some users to sign this document that can contained confidential or sensible data that has to be hidden for confidentiality reasons. This way, provide the user the creation of a workflow with selected people

allowed to see the hidden data.

1.2.1 Objectives

The main goals of this thesis is to develop an extension for an ECM system capable of signing documents with a legally binding value for multiple users. We have the following objectives:

- **Technology Research:** Identifying the current state of the art ECM systems and digital signature workflow systems. Understanding the way the different systems interact and what can be improved with this two systems integrated.
- **Improving:** Developing and improving of a digital signature application.
- **Implementing:** Developing of an extension to the ECM system that provides a custom workflow for the users sign documents. This way, we make integration of the digital signature with an ECM system for complex and simple cases.
- **Testing:** Deploy our system implementation in a real large use case scenario (like universities or companies, that can put their users to sign documents that have to be signed by multiple users, using this system) and analyse the opinions of the users, for possible improvements of the system.

1.2.2 Features

The main features of our proposed system are as follows:

- **Workflow Management:** Our system allows the creation of sequential, parallel and group workflows. This way, the system adapts to various user needs and can simulate a real task signature, when users pass a hand-in-hand document to be signed.
- **Validation Service:** The information about the signatures of a document, is given by the validation service. The users can validate a document, any time, seeing information like timestamps, integrity, certificates and signed fields of an specific selected document.

- **Online data access:** Documents can be accessed anytime and anywhere by the users, as they are in a enterprise content management. Users are advised to have a document to sign by email and have a direct link to be able to sign without having to be looking for the document in the enterprise content management.
- **Compatibility with smart cards:** Users can sign a document with their personal Citizen Card, which facilitates the use of service, requiring only that the user has their signature PIN and smart card reader.

1.3 Contributions

During this project development, a paper was accepted and will be published in "5^{th} *International Conference on Electronic Government and the Information Systems Perspective*" to be held in *Instituto Superior de Engenharia do Porto (ISEP)*, in Porto, Portugal.

1.4 Outline

This work is divided into 5 Chapters. The current chapter presents an introduction to all the work of this dissertation.

The next chapters of the thesis are organized as it follows:

Chapter 2 presents a brief overview of the current state of the art on DMS/ECM, the digital signature workflow systems as well as a series of integrations of both, with independent systems or add-ons for *Alfresco*, a resume about *Alfresco* and redaction systems. Some security and privacy mechanisms: smart cards, digital signatures, redaction and QR-codes are detailed in Chapter 3. In Chapter 4 we present the implementation details and approaches used, specifying some *Alfresco* properties that helped in deployment. Lastly, the Chapter 5 presents some final remarks and lays the ground for future work.

Chapter 2

Related Work

In the following sections, we present an overview of a set of DMS/ECM to understand their features and to see which best fits our goal. We also present an overview of a set of digital signatures workflow systems describing their features. This information can help us to improve our system with some features that we still not have or where we can innovate. As our goal is to integrate these two systems, we also present an overview of similar existing solutions, i.e., digital signature systems (with or without workflows) integrated with an ECM, and an overview of the features/benefits. We also analyse some redaction add-ons to *Alfresco* so, we can see where we can explore this component, when we made the integration into *Alfresco*.

2.1 Document Management System

The DMS allows multiple users to store and retrieve documents on a closed client/server architecture network. Many of these systems also allow the user to manipulate documents and share documents with other system users [9].

Most systems incorporate many procedures, such as storage location, security, access control and version or revision control. One of the benefits of maintaining electronic files is the ability to keep a record of who accessed the content, whether it was downloaded, when was it accessed or viewed.

For companies, a tracking and monitoring system of documents, specially, the version or revision control, plays a very important role, as it helps in protecting the integrity and authenticity of digital information, as it provides a mechanism to the system see

the old versions. Another benefit of these systems is the group approvals, such as a contract document, for example. Without these systems, these approvals can take unexpected number of days that may lead to delay in the progress of a project. With these systems, the electronic version of the document can be shared ensuring that the correct people review the document and can create an audit trail that allows everyone in a group to see another edits while also preserving the document in its original form [10].

However, DMS were interested in the file and storage/indexing/retrieval mechanisms to allow the user to classify and retrieve documents. They were initially concerned only with the file as a container. But, as market needs changed, the DMS focus shifted from file management to content management. For example, if we have a Web site, it is composed of HTML, XML, or ASP pages that need to be managed. So, the name of the system was changed to ECM [5].

2.2 Enterprise Content Management

ECM is an integrated approach to managing an organisations information including paper documents, data, reports, web pages, and Digital Asset Management (DAM) [11]. DAM is any text item or media file that has been formatted into a binary code carrying embedded their rights of usage. A digital file without copyright is not a DAM [12].

ECM is defined by Association for Information and Image Management (AIIM) as the strategies, methods and tools used to capture, manage, store, preserve and deliver content and documents related to organisational processes. The capture of the content creates opportunities to identify document types, extracts data and validates indexed data against back-end applications. All content is in electronic form and it is a big benefit. The content can be managed to provide search capabilities; security and access controls; Records Management (RM) that ensures that access to documents is controlled and logged at all times; workflow and business processes; collaboration and analytics. Then, the content is available to be delivered to the right person, at the right time and in the right format. With this, we can ensure a smooth and efficient process. Finally, the archiving can specify how long a piece of content should be retained, where it should "live" and when it should be destroyed, ensuring user

stays compliant at all times [13]. The figure 2.1 [1] illustrate a document lifecycle management.

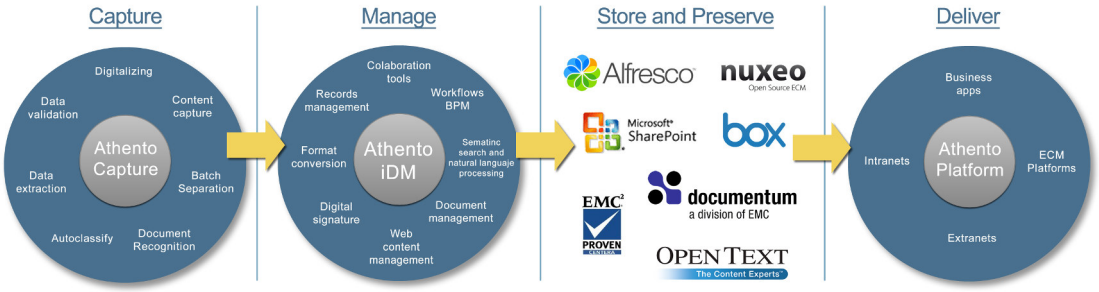


Figure 2.1: ECM document lifecycle management

One of the biggest tools of the ECM are Business Process Management (BPM) and Workflow. These tools move content through an identified business process, such as claims processing. BPM solutions are framework used to develop, deploy, monitor and optimise multiple types of process automation applications, including processes that involve both systems and people like workflows. Workflow can be seen as a task that have a initial and final state. An workflow handles approvals and prioritises the order documents are presented. The decisions of workflow are based on predefined rules developed by system owners [5].

We now proceed to describe open source DMS solutions that may fall within this category.

2.2.1 *Alfresco*

An example of an open source ECM system is *Alfresco*. This system incorporates the major applications of ECM: documents, images, Web Content Management (WCM), RM, and DAM. *Alfresco* system stands out in its services and controls that manage the content and features. The most important features of this system are the workflows, versions control, metadata management and search.

[1] Enterprise Content Management lifecycle:
`http://www.athento.com/site_media/imagenes/web/captura/ECM-components.png/`, 2014. [Online; Accessed 2015-12-1]

For a business, for example, this system has the most important features to support the content requirements of a number of business critical processes and uses. Office work, search and discover is supported by the Document Management (DM) tools. The businesses also needs workflow management capabilities that includes case management, review and approval. The creation and refinement of content and documents are supported by the collaboration applications. The scalable WCM services support the delivery and deployment of content from the enterprise to its customers. One of the most important value of this system is the capability of RM, that provides an affordable means to capture and preserve records based upon government-approved standards. The standards-based platform also provides access to applications that use these standards, such as publishing, image, and email management [1].

For a developer, the system has a benefit, the add-ons. Users can develop an *Alfresco* add-on to improve the capabilities of an *Alfresco* product. The users can make integrations with external systems, package customisations, system administration tools and community maintained language packs.

This software has a community version that is free unlike other systems. The paid version of *Alfresco* is *Alfresco One* that enterprises should consider if they require enterprise-class capabilities such as content encryption, clustering or simplified administration. That version has wide range of modules and add-ons including content encryption, RM and media management.

2.2.2 *Nuxeo*

Nuxeo is an open-source ECM designed by developers for developers [2]. This system has different options, such as the *Nuxeo Platform* that is a highly customisable and extensible content management platform for building business applications. This platform is open source and users can download it to implement, DM, DAM and case management, for example. *Nuxeo Platform* is composed of small software parts that can be assembled together to build an application: Services, that provide features to manage the users information; Components provided to implement, configure and extend the services; and Presentation framework and User Interface (UI) building blocks that provide the user interface on top of the service stack [14]. The principal feature of *Nuxeo Platform* is allow a developer to expand and configure extensions to

[2] *Nuxeo* Platform: `https://www.nuxeo.com/products/content-management-platform/`, 2014. [Online; Accessed 2015-11-25]

that platform.

For developers, *Nuxeo Studio*, despite being paid, may be very convenient when designing workflows, for example. It could save hundreds of hours in development. The authors of the white paper [3] say that users who use the services (*Studio*, *support*, *consultants*) will have extra help to be able to complete their projects.

2.2.3 *eFileCabinet*

eFileCabinet is one of most popular DMS [4]. As the name implies, this system is a virtual office(cabinet) with cabinets, drawers, folders and files. *eFileCabinet* has features like: easy scanning, capturing and management of client information, search for documents quickly and security for the business-critical files because they are in a secure database at all times [5].

This system provides a complete suite of ECM that include four plans: *eFileCabinet Desktop* is an Electronic Document Management (EDM) solution to store and manage important business documents; *eFileCabinet Online* is a hosted EDM solution; *eFileCabinet SecureDrawer* allows users to access their documents anytime, anywhere in an online client portal/file sharing service and *eFileCabinet Concentsus* is an online backup service to protect documents via a secure online remote location, so its retrievable.

These systems also have access controls, archiving and retention, collaboration, compliance management, document conversation, delivery and indexing, electronic signature, email management, full text search, print management and version control. Despite many features that often customers/companies are interested, have a major drawback and limitation for some clients/companies, the high cost of the systems and of the features. In addiction to the monthly cost, which can be rather substantial depending on how many employees the business have and the plan of the system that the business

[3] *Nuxeo Studio* Overview and Concepts:
`https://doc.nuxeo.com/display/Studio/Studio+Overview+and+Concepts`, 2015. [Online; Accessed 2015-11-26]

[4] Top Document Management Software:
`http://www.capterra.com/document-management-software/#infographi`, 2015 [Online; Accessed 2015-11-24]

[5] *eFileCabinet*: `https://try.efilecabinet.com/capterra` [Online; Accessed 2015-11-24]

choose [6].

2.2.4 *DocuWare*

DocuWare is an non-open source Professional ECM. The authors of the online documentation [14] refers to this system as state-of-the-art DMS software for Professional ECM. This system uses a centralized document pool, this means that all documents (correspondence, records and email, for example) can be stored, shared and managed, simply and securely. The documents can be managed according to their preferred categories, labels, priorities and search terms [7].

This system provides tools for efficient ECM and the major features of this system are: document lifecycle management, RM; workflow functionality; web access and universal integration features. This software has a *DocuWare On-Premise* version that is free and has *DocuWare Cloud* version which can have benefits for some enterprises like: the documents are transmitted encrypted and stored in the data centre, that provides more security; requires no installation, just a Uniform Resource Locator (URL). The client runs on all established browsers, so, people can access anytime and anywhere and just need an Internet connection; and the users only pay for what they need. The system provides flexible licensing options that gives to the client the full range of features, tailored to the number of users and documents they require [14].

This system has a limitation for developers. The users can expand the *DocuWare* feature set individually with add-on modules. But, for example, if a user want integrate DM for storage and searching in other applications, and implement extremely demanding document workflows, the user has to purchase an additional add-on licence to set the expanded feature available to all users in an organisation [8].

2.2.5 *LogicalDOC*

LogicalDOC is a platform DMS with ECM features. This system has an open source version community that has some DM features like: Full-text indexing, metadata

[6] *eFileCabinet*-Document Management Website Provides Cloud and Desktop Flexibility: `http://www.marketwired.com/press-release/efilecabinet-document-management-website-provides-cloud-and-desktop-flexibility-1662566.htm`, 2012. [Online; Accessed 2015-11-24]

[7] *DocuWare* vs. *Alfresco One*: `http://www.itqlick.com/Compare/docuware/alfresco-one`, 2016. [Online; Accessed 2016-3-27]

[8] *DocuWare*: `https://www.docuware.com`, 1998. [Online; Accessed 2015-11-25]

and templates, version control, document searching, bookmarking, multi-language support. This version has security, users, groups and task manager too.

For enterprises, the system has interested features, in the Enterprise and Business version, from users and developers which stand out: Drag and Drop from desktop, multi workspace, PDF creation, online editing, digital signature (with digital certificate) that allows multiple signatures on a document by multiple users, and professional workflow [9].

2.3 Comparison of DMS/ECM systems

To compare the different systems analysed and choose the best system to use, we decided to do a comparative table with the main features that we need in the system. Based on [1] [14] [15], we construct the following table:

Table 2.1: Comparison of DMS/ECM systems - (E-Enterprise Version, C-Community Version)

	Alfresco C	*Alfresco E*	*Nuxeo*	*DocuWare*	*eFileCabinet*
Open Source	LGPLv3	-	LGPLv2.1	-	-
Add-ons	✓	✓	✓	✓	✓
Workflows	✓	✓	✓	✓	✓
PDF Support	✓	✓	✓	✓	✓
Txt/binary support	✓	✓	✓	✓	✓
Users/Groups support	✓	✓	✓	✓	✓
Digital Signatures	-	-	-	-	-
Electronic Signatures	-	-	-	✓	✓
Record management	-	✓	✓	✓	
Redaction	-	-	-	-	-

[9] *LogicalDOC*-Product features: `https://www.logicaldoc.com/product/features.html`, 2015. [Online; Accessed 2015-11-27]

	LogicalDOC C	*LogicalDOC E*
Open Source	LGPLv2.1	-
Add-ons	✓	✓
Workflows	-	✓
PDF Support	✓	✓
Txt/binary support	✓	✓
Users/Groups support	✓	✓
Digital Signatures	-	✓
Electronic Signatures	-	-
Record management	-	✓
Redaction	-	-

We analyse some of the most popular ECM systems. We are interested in open-source systems as well as to have full control over the system and be able to create free add-ons. We also have security guarantees seeing the system code and adapt it to all our needs. We analysed some non open source systems as they could have some the features we wanted and see what we can do to innovate.

The non open source systems are *Nuxeo*, *DocuWare*, *eFileCabinet* and the enterprise versions of *Alfresco* and *LogicalDOC*. Within the non open source, we try to see those in which there exists signatures or workflows, as they are our principal focus. The enterprise version of *Alfresco* and *LogicalDOC* have workflows. *eFileCabinet* has workflows too. *LogicalDOC* enterprise version has digital signatures unlike the community version. *DocuWare* combines workflows with electronic signatures but do not have digital signature. Therefore, within the non open source, we have none combining the workflows and digital signatures options. On the open sources systems, we seek a solution that has the ability to create extensions to the developer of the ECM. After that, we look for systems Open Source that have workflows, so, *LogicalDOC community* is not an option. Among others, *Alfresco community* and *Nuxeo community* the choice was more complicated, but beyond *Alfresco* have more users, it also has much more online communities, more tutorials and help documents.

2.4 The choice - *Alfresco*

We choose *Alfresco* to make the integration of the digital signature workflow because, first of all, as we described in section 2.6, the only option alongside the *Alfresco* was the *Nuxeo*, but in comparison of both, we see that *Nuxeo* is more used in industry and companies and, perhaps for this reason, there is more documentation available for *Alfresco* and mutual aid communities.

Many ECM systems looks such as the figure 2.2. However, *Alfresco* was relatively recently created, so, it has a more modern architecture compared to other ECM systems.

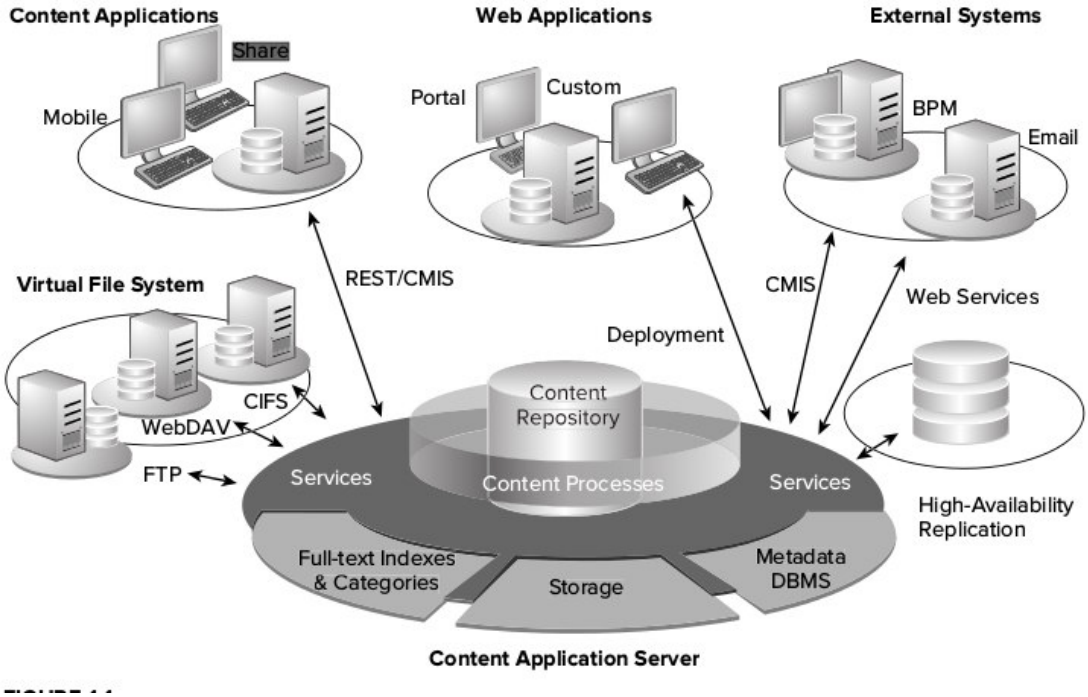


Figure 2.2: *Alfresco* system [1]

Alfresco is an ECM system and provides ECM capabilities as data services and user interfaces and applications.

At the core, there is a repository supported by a server that persists content, metadata, associations and full text indexes. *Alfresco* supports multiple programming languages and protocols, that can be used to create custom applications and solutions. Out of the box applications provides standard solutions such as DM, RM, DAM, WCM and search. It is scalable to big industry and big companies. Each service and features

of the *Alfresco* is made to scale in terms of size of data set, processing power and number of users. Therefore, the *Alfresco* architecture has a modular approach in which capabilities packaged in modules whose implementation can be changed if needed. Aspect-Oriented Programming (AOP) allow the changes and optimisation of an ECM solution.

Alfresco ECM allows the choice of the operating system, database, application server, browser and authentication system to use when deploying *Alfresco*. Developers can choose how to package *Alfresco*, for example, as a web application, a *portlet* [10] or an embedded library.

The Spring platform, at the core, provides *Alfresco* the ability to modularise features such as version, security and rules. The portion of architecture called *Web scripts*, provides *Alfresco* support scripting to simplify developing new features and new programming interfaces.

Alfresco system is highly compatible with every systems that can run Java Enterprise Edition, since *Alfresco* was created as an entirely Java application.

Alfresco applications are built upon the Content Application Server (CAS) and rely on the CAS to manage, query, access and persists content. This applications exist to provide the basic capabilities that users need to manage content. The two main applications are *Alfresco Share* and *Alfresco Repository (Explorer)*.

The Explorer is the application to manage content. This application allows the users to browse the repository, set up rules and actions and manage metadata, associations and classifications of the content. It has many capabilities for managing the repository and it is considered a system administrator tool. It is becoming obsolete with the arrival of *Alfresco* Share. However, many extensions and language packs have been built for *Alfresco Explorer*. *Alfresco Share* is the new UI built with the *Alfresco* web script technology. *Alfresco Share* provides content management capabilities with simple UI, with tools to search and browser the repository, content like thumbnails and associated metadata or preview using flash renditions of content, for example. It is organized as a set of sites that can be used as a meeting place for collaboration. *Alfresco Share* was the reason of the creation of *Alfresco* RM [1]. *Alfresco* Share provides a rich web-based collaboration environment for managing documents, *wiki* content, discussions

[10] *Portlet* is an independent visual component that can be used to provide information within a Web page.

and blogs [11].

In the *Alfresco Share* page, users have an *Alfresco* toolbar, that has: Home, opens the dashboard of the users; My Files that opens the My Files area, where users can store the personal content; Shared Files, where users can share files with other users without adding it to a site; Sites, that has options to open a recent/favourite site, create a site and view all the sites that user is member or Site Finder page where users can search for sites; Tasks, where users can see the workflows that they create and can manage tasks; People, where users can search for other users; Repository that opens the *Alfresco* repository which shows all the content stored in *Alfresco*; User Menu, where users can update the own status, can access to the own profile, change password, open the user help and log out; and Search that is used to find files, sites and people.

We made some tests in some ECM systems, and we were interested in many of *Alfresco* features in terms of usability and interface, even if some were already also present in other systems. The features that we highlight is the possibility to use a dashboard with notifications to see the new tasks, new documents and new activities, without requiring the user to navigate all menus to see the new changes. Another interesting feature is a notification in each document that warns the user that exists a workflow active, in the correspondent document, that is not yet completed. *Alfresco* has the possibility of some document actions (buttons with features), that interest us to make actions in the document and display information/options about the file. We can see the properties of the file, if there are workflows active in this file and a version history. With the version history, we can see all the older versions of the file and who modified the file.

2.4.1 Web Scripts and Surf Framework

The user interfaces capabilities are provided by application components using *Alfresco* web tier and Surf [12]. The Web Scripts are important for this, allowing the creation of a REST-based Application Programming Interface (API), providing a fastest/easiest

[11] *Alfresco* architecture - `https://docs.alfresco.com/community5.0/concepts/alfresco-arch-about.html` [Online; Accessed 2016-4-14]

[12] Surf is a toolkit for develop web applications and web sites. Surf is developed as a faster way to develop content applications using scripting and Representational State Transfer (REST) architecture.

way to extend *Alfresco* standard services. The Web Script Framework is designed aiming to make easy and quick as possible the creation of new web scripts in order to construct a RESTful interface to the *Alfresco* CAS. The REST web architecture is based on Hypertext Transfer Protocol (HTTP) requests and responses, Uniform Resource Identifiers (URI)s, and document types. Web scripts let developers implement their own RESTful API with minimal tooling required or Java knowledge. Web scripts are implemented using lightweight scripting languages such as JavaScript (JS) and FreeMarker Template Language (FTL), so their development is not restricted to only those who know the Java language. This approach to developing an *Alfresco* API means that web scripts offer many advantages over existing technologies, such as Simple Object Access Protocol (SOAP), including ease and speed of development and flexibility in API design.

To develop a web script, developers need to know the following programming languages: XML, JS and FTL technologies. XML for the description of the web script - it is URI, HTTP method, authentication and transactional needs. JS is used for the logic of the web script - here, developer may query or perform actions against content residing in the *Alfresco* content repository. The output of the JS is a model to process in the web script response. In the JS we can get, for example, the URI used to invoke web script, the currently authenticated user, the properties of this user (with a person object), and more. FTL is used for the response of the web script (such as HTML in a web page). The format of the response can be generated in HTML, XML or JavaScript Object Notation (JSON).

In the below figure 2.3, it is represented these components of the web scripts:

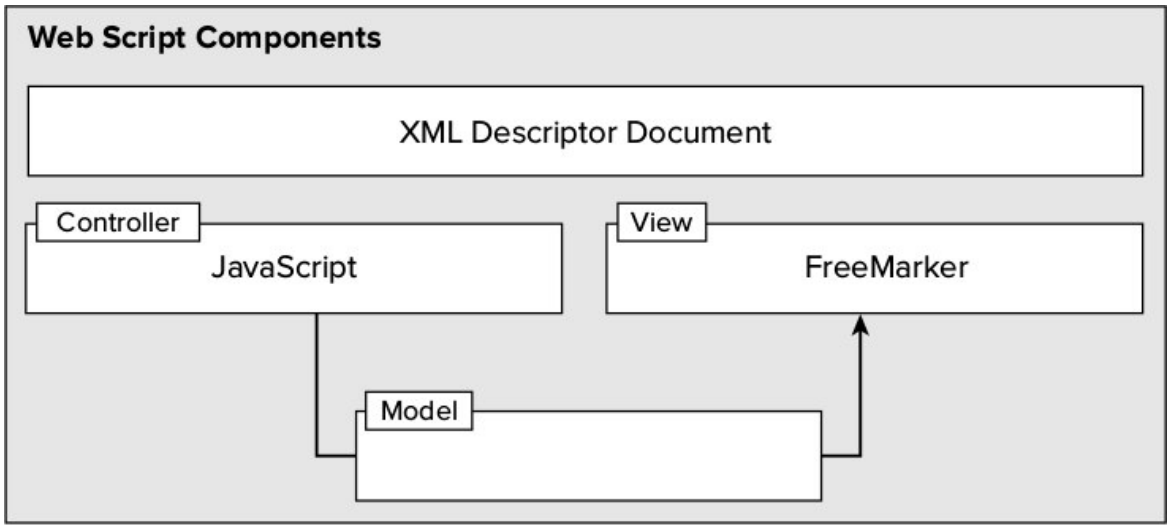


Figure 2.3: Components of a web script

This mode of development, the web scripts, has become very attractive to the *Alfresco* community, making web scripts a preferred approach to extend *Alfresco*, interact and integrate with *Alfresco*. It is very useful to allow developers to map arbitrarily the content in the repository to resources on the web. Otherwise, developers can use an out-of-the-box web scripts that already encapsulate many of the mapping. The *Alfresco* Content Management Interoperability Services (CMIS) implementation was built using web scripts [1] [13].

Sometimes, developers prefer and need to use Java to build applications or extend the *Alfresco Share*. Developers can use Surf, the web runtime framework, to extend *Alfresco Share* and construct web applications. As *Alfresco Share* was built using Surf, developers can build their extensions as a combination of Java programming and web scripts or only Java. Java is used to access or change pieces of code of *Alfresco*, CAS or *Alfresco Share* by using the *Spring* platform.

[13] Alfresco - Web scripts: `https://docs.alfresco.com/4.0/concepts/ws-architecture.html` [Online; Accessed 2016-3-30]

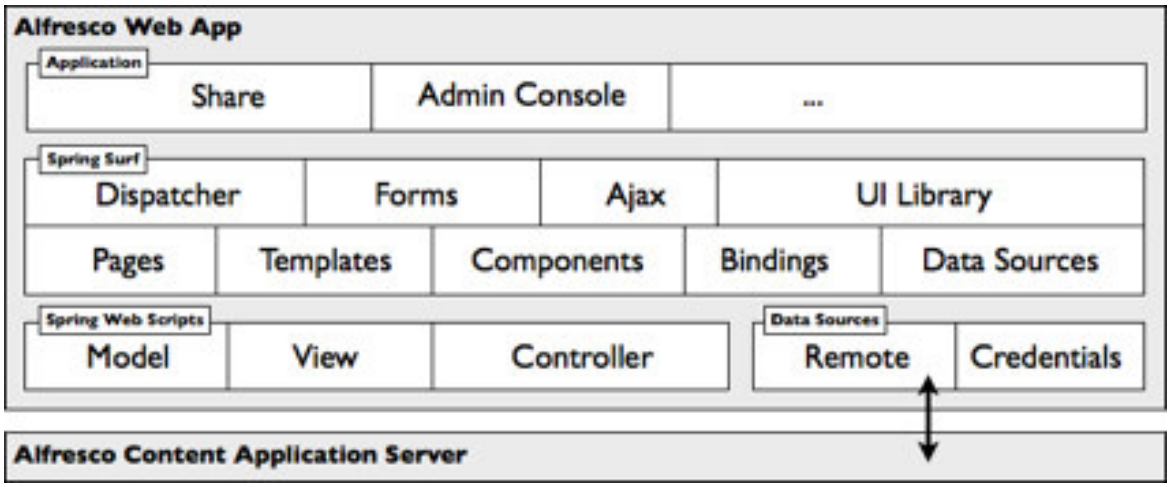


Figure 2.4: *Alfresco* Web Applications

Some features of the *Alfresco Surf*, as we can see in the figure 2.4, include a site dispatcher to easily create pages; templates for defining a page layout; the pages can be develop using FTL, JavaServer Pages (JSP), HTML or Java; the support to UI library containing reusable UI components comprising back-end application logic and front-end presentation code that can be bound into regions (or slots) within a page or template; the Asynchronous Javascript and XML (AJAX) can be used as a support for integration with the Yahoo! User Interface (YUI) library and forms can be used for rendering and collecting data. The features include also pages that users can render in multiple formats, such as PDF and format to mobile devices for example.

Alfresco 5.0 included a new UI framework constructed on *Alfresco Surf* - the *Aikau*. *Aikau* provides a modern higher-level approach to developing custom UI applications and provides an easiest way for creating pages and widgets. New pages with standard widgets can be created through *JSON* code and then, can be extended as needed using JS.

2.4.2 *Aikau* Pages

The main goal of *Aikau* is to provide a library of widgets that can be assembled into a web application for accessing an *Alfresco* repository. The *Aikau* page came to replace the Surf pages. The principal goals of the *Aikau* pages are:

- Reduce complexity of the Surf pages by removing page, template and component configurations and replace them with pages defined by an easily customisable

JSON model in the JS controller of a single web script;

- Provides a solution that allow an easy customisation and a rapid development;
- Allow the dynamic creation of pages for and within the running UI and render them without restart any servers [14];

To build an *Aikau* page, it is needed to add an XML web script descriptor, an FTL web script template, a JS web script controller with page layout or model and a widget to display the content. It is also necessary to choose which Surf Page who wants to use the basis, such as: *dp*, *hdp* or *rdp*. *hdp* for example, is used to put in the custom page the header and footer of the *Alfresco* application. In resume, it is possible to easily create pages with a defined template, for example and, create web scripts.

2.4.3 All-In-One Archetype

This archetype takes advantage of the powerful capabilities of *Alfresco* Software Development Kit (SDK) [15] to customise and run the full *Alfresco* platform embedded with all components. This is a multi-module project. To run this archetype, it does not requires additional downloads, like *Alfresco* installer and provides a good starting point for more complex and big projects where the final artifacts should be the customized *alfresco.war* and *share.war*.

The All-in-One archetype (AIO) is useful for some cases such as:

- The programmer is going to start an *Alfresco* extension project that should produce the final customized *Alfresco Web application ARchive (WAR)* and *Share WAR* artifacts.
- The project needs to access to the full regression testing suite for the *Alfresco Share UI*.
- The project needs to access to the functional testing based on the *Alfresco Share* Page Object (PO) library.

[14] Introducing *Aikau*: https://docs.alfresco.com/5.0/concepts/aikau-intro.html [Accessed; 2016-3-30]

[15] *Alfresco* SDK 2.2.0: `https://docs.alfresco.com/5.1/concepts/alfresco-sdk-intro.html` [Online; Accessed 2016-3-30]

- When the project is tested with the Rapid Application Development (RAD) features (we will describe this process in the next chapter), the programmer needs *Solr* [16] to be running.

The principal features of the AIO archetype are:

- Alfresco Module Package (AMP) packaging;
- AMP dependency management in *Maven*;
- Automatic installation of AMPs into *Alfresco WAR* and *Share WAR*;
- Easy to include extra AMPs and have them included in the WARs;
- Out-of-the-box *Alfresco* extensions such RM, Media Management, etc...;
- AMP Unit Testing support, just running *mvn test*;
- *Alfresco* Testing - the programmer does not have to write tests to protect against regression in out-of-the-box Share UI functionality, just have to use the *-Prun,regression-testing* profiles.
- Run a full *Alfresco* stack (*alfresco.war, share.war, solr4.war*) embedded in tomcat using the H2 database for demo purposes (-Prun), rapid application development and integration testing;
- Support for (remote) *Junit* and integration testing and RAD. This uses springloaded. Projects can easily be launched for this scenario using *run.sh*;
- Integrated Development Environment (IDE) Integration with *Eclipse* and *IntelliJ IDEA*. [17]

[16] *Solr* is a popular search platform for Web sites because it can index and search multiple sites and return recommendations for related content based on the search querys taxonomy. *Solr* is also a popular search platform for enterprise search because it can be used to index and search documents and email attachments.

[17] All-in-One archetype:
`https://docs.alfresco.com/5.0/concepts/alfresco-sdk-archetypes-aio.html` [Online; Accessed 2016-3-28]

2.4.4 Rapid Application Development

RAD and Test Driven Development (TDD) are big goals for the *Alfresco* SDK. The SDK is designed to support the modification of JS, FTL and Java code with hot reloading of code, via Spring Loaded. The purpose of this is the changes take effect without having to restart *Alfresco Tomcat* and without having to click the *Refresh Web Scripts* or restart anything. The hot reloading saves the time of the developing. No need to wait around *Alfresco Tomcat* restarts to see the effect of the changes of the code.

In the SDK project, programmers can change test code, re-run the test and the results are displayed immediately, allowing TDD [18].

The *Alfresco* SDK is designed to work well with *IntelliJ IDEA* and *Eclipse*. This support includes the ability to import existing SDK projects (created via the command line) into *IntelliJ IDEA* and *Eclipse*.

For the two IDEs, there are three components that work together to enable the best RAD experience:

- **Spring Loaded:** takes care of hot reloading of changes of any Java class files;
- **Refresh Repository Script:** This is a script that will POST a request to the *Alfresco repository* web application (that is, *alfresco.war*) telling it to refresh the repository (repo) web script container, so any changes to files related to web scripts will be picked up;
- **Refresh Share Script:** This is a script that will POST a request to the *Alfresco Share* web application (that is, *share.war*) telling it to refresh the *Surf web script* container, so any changes to files related to *Surf web scripts* will be picked up. This script will also clear the resource dependency caches, so JS changes, for example, are also picked up [19] [20].

[18] RAD: `https://docs.alfresco.com/5.1/concepts/alfresco-sdk-rad.html` [Online; Accessed 2016-3-30]

[19] RAD in *IntelliJ IDEA* (Hot reloading): `https://docs.alfresco.com/5.1/tasks/alfresco-sdk-rad-intellij-hot-reloading.html` [Online; Accessed 2016-3-30]

[20] RAD in *Eclipse* (Hot reloading): `https://docs.alfresco.com/5.1/tasks/alfresco-sdk-rad-eclipse-hot-reloading.html` [Online; Accessed 2016-3-30]

This tool is an important feature to the time that Alfresco takes to do restart the tomcat with the RAD functionality. With this tool, we can make changes to the code in seconds. Without the RAD it takes at least more than two minutes per compile extension + restart the tomcat. Its important too, because when we compile and restart tomcat, errors may occur, such as forgetting to compile the part of repository or share, and linger again twice as long because we have to back off the *Alfresco* and restart tomcat.

2.4.5 Workflows

One of the biggest tools of the ECM *Alfresco* is the support to the Business Process Model and Notation (BPMN) and workflows. BPMN is used to model notations for designing business processes and consists in represent the business workflow. BPMN solutions are frameworks used to develop, deploy, monitor and optimise multiple types of process automation applications, including processes that involve systems and people, like workflows.

Workflow can be seen as a task that have an initial and a final state. An workflow handles approvals and prioritises the order in which documents are presented. The decisions of workflow are based on predefined rules developed by system owners [5]. In *Alfresco*, a workflow is a sequence of connected tasks applied to a document. Each task can be performed by a person, a group, or automatically.

If a person has a document that needs reviewing and approving by an specific number of people. The sequence of tasks would be:

1. Send an email to each reviewer to alert them to review the document within a certain time, a reviewer does not need to make an exhaustive search, having at its disposal in the email, the link to download the document and link to the review page of the task;
2. Each reviewer reviews the document and approves or reject the document;
3. If enough reviewers approve, the task is completed successfully and the owner can end the workflow.

The default workflows in Alfresco, provides to the user multiple ways to automate the process for them. The five workflow definitions are:

- **Adhoc** enables a user to assign a task to a single user;
- **Group Review & Approve** enables a user to set up review and approval of content, assigning the workflow task to a group that is created with the users of *Alfresco* and by the users;
- **Parallel Review & Approve** enables a user to set up review and approval of content, assigning the workflow task to multiple users (in any order);
- **Pooled Review & Approve** enables a user to set up review and approval of content, assigning the workflow task to multiple users. One user can take ownership of the task at a time, completing it or returning it to the pool to be claimed by another user associated with the task;
- **Review & Approve** enables a user to set up review and approval of content, assigning the workflow task to a single user [21].

2.4.6 CMIS

CMIS is a modern interface to communicate with a Content Management Systems (CMS) in a standard way. The CMIS interface consists in two parts: a number of repository services for content navigation and content creation and a repository query language for content search. The standard also defines which protocols can be used to communicate with a repository and formats that should be used in requests and responses via these protocols [16]. The goal of CMIS is to access any CMS such as *Alfresco* and *Microsoft SharePoint*.

CMIS can do basic operation in *Alfresco*, such as, get repository information, list the children of the root folder, list available types and subtypes, get metadata and content and create, update and delete content. CMIS can also do advanced operations, such as, version management with check out and check in, managing permissions for documents and folders, managing relationships between objects and searching.

For example, with a simple Java CMIS program, it is possible to do these operations. It is possible to access a CMIS server with Java Client, such as, connecting and setting up a session with a repository, getting repository information, listing the children of

[21] *Alfresco* - What is a workflow?:
`https://docs.alfresco.com/4.2/concepts/wf-whatis-workflow.html` [Online; Accessed 2016-3-30]

the root or top folders, listing available types and subtypes, creating, updating and deleting content, getting the content for a document, copying and moving folders and documents, working with *Alfresco* aspects, version management with check out and check in, managing permissions for documents and folders and managing relationships between objects. With this features, for example, it is possible to put files of the home directory of the local Personal Computer (PC) in *Alfresco*, create a file in home directory with the content of a file of *Alfresco* and delete files of *Alfresco*.

There are another options of accessing a CMIS server using a scripting languages, such as, using CMIS in JS and web application pages, using CMIS in *Groovy* [17] scripts and using CMIS in *Spring Surf Web Scripts*.

2.5 Digital Signatures Workflow Systems

As we described in the section 2.2, a workflow is a task, that can be attributed to one or more users, which has an initial state and a final state. Intermediate states, the decisions of the workflow are based on predefined rules developed by system owners. Therefore, a digital signature workflow is a task that has the objective of one or multiple persons sign a document.

We now proceed to describe some of most used digital signatures workflow systems that we study.

2.5.1 *SecuredSigning*

Secured Signing is a web application Software as a Service (SaaS). This web application allows users to upload, preview and sign legally binding documents online using the secured personalized X.509 Public Key Infrastructure (PKI) digital signature technology. The system do not use electronic signature at all, only adds a graphical image of the name to the digital signature.

The service allows a registered user to receive a unique secret key for signing. The user can add a document to the system and sign it digitally. There is an option to invite a third party if required and send it to parties involved to sign, making a workflow. Anyone on the workflow is able to verify signatures on their desktop application (like

Adobe) or using the online verification service of this system [22].

2.5.2 *SigningHub*

SigningHub is a solution for document approval workflows and advanced digital signatures. This system is designed to optimise the way businesses deliver, review, approve and sign the business document. This system also adds a graphical image of the name to the digital signature.

According to the online documentation [23], this system sets the industry benchmark in terms of making the process of applying and verifying advanced digital signatures extremely easy. Users can review and sign documents from any device, location and at any time, because the document are synced across multiple devices, so, the documents are always updated with the latest version. Integrating the *SigningHub* with the Citizen Card can be easily done within the user business application web pages using the high-level API. Utilising the most advanced cryptographic security in innovative ways to minimise the complexity for users. *SigningHub* can produce the strongest level EU Qualified Signatures that are verifiable and legally enforceable for the long-term. This system has a flexible deployment and provide a complete out of the box e-Trust infrastructure, including Certification Authority (CA), real-time Validation Authority (VA) and Time Stamp Authority (TSA) servers. Existing enterprise, Internet or national level trust service providers can also be registered as trust anchors. Certified Document Services (CDS) and Adobe Approved Trust List (AATL) based signatures are fully supported for automatic trust in *Adobe Reader*.

2.5.3 DocuSign

DocuSign is an application that allows users to create a workflow of multiple users, send a document and sign legally binding documents. The process in this system is, the sender uploads a document to the system and adds a name and email of the signers and other recipients that can view the document. After this, the recipients receives the email and is given access and instructions to sign the document using an electronic or digital signature. Once signed, both the sender and recipients have access to the

[22] Secured Signing Online Form and Document Signing with PKI Digital Signature: `https://www.securedsigning.com/support/faqs`, 2015. [Online; Accessed 2015-11-28]

[23] About *SigningHub*: `https://www.signinghub.com/about-signinghub`, 2001. [Online; Accessed 2015-12-1]

signed document, and the workflow is completed to the sender [18]. This system also adds a graphical image of the name to the digital signature.

DocuSign provides innovative digital signature technology solutions. *DocuSign* act as the Certificate Authority with the *DocuSign Express Digital Signature*. Ensures instant PKI standard and X.509-compliant digital signatures for any *DocuSign* transaction. The system has card/token-based digital signatures, so, the user can sign with existing physical or software-based certificates, including smart card-based National IDs and employee badges. *DocuSign* also has a third-party cloud-based digital signatures, this way, sign with region and industry-specific certificates. This system also provides locally managed digital signatures that allows user managing digital certificates in user own data centre, behind user firewall with *CoSign Central*. Ideal for highly regulated industries, Certified for Common Criteria and Federal Information Processing Standard (FIPS) security standards for use in e-signature.

2.6 Comparison of digital signature workflow systems

To compare the different systems analysed and see features that can be added to improve what already exists in the market, we decided to do a comparative table with the some features. There are many advantages of using Open Source Software, such as, freedom, quality, customisable and cost [19]. It is important to know if this type of software have support to physical technology like Universal Serial Bus (USB) tokens or smart cards, for example. There are some type of workflow: Individual Workflow (only one person), Sequential Workflow (follows a defined order), Parallel Workflow (any order allowed) or Group Workflow (the system create groups of registered users, that can be used for multiple documents for example). The validation of all signatures, a system that validate a document with multiple users, for example, and gives the information of how many signatures lacking for the document to be signed by everyone and "accept" with a valid signature.

Table 2.2: Comparison of digital signature workflow systems

	SecuredSigning	*SigningHub*	*DocuSign*
Open Source	-	-	-
Cryptography technology	X.509	X.509	X.509
Physical technology	-	Smart Card and Mobile	Smart Card
Individual workflow	✓	✓	✓
Parallel workflow	✓	✓	✓
Sequential workflow	-	✓	✓
Group workflow	-	-	-
Validation of all signatures	✓	✓	✓

2.7 ECM systems with Digital Signatures

There are many *Alfresco* add-ons making digital signatures and there are also independent systems prepared to be integrated with *Alfresco*.

2.7.1 Independent systems

According of the authors of *CoSign* [24], many signers at security-minded businesses, governments and cloud services are already using the *CoSign* digital signature solution to automate their signature-dependent processes. This system also adds a graphical image of the name to the digital signature. *CoSign* has a lot of benefits that allows complete freedom of choice, such as, allow users to sign from any device, location and at any time; integrate digital signatures into a DM or content management or workflow automation system, including *Google Drive*, *Share Point*, *Alfresco*, *Nintex* and *K2*; and work with the content authoring applications and file types which are known for the most users, including *Word*, *Excel*, *Outlook*, PDF and PDF/A. The system also ensures the user control and security through keeping user signed documents within user enterprise domain; adapting to user existing processes instead of forcing user to adopt rigid workflows; and integrating with the user directories and enrolment methods that meet user management and authentication requirements. *CoSign* allows complying with strict industry and government regulations, providing easily verifiable

[24] About *CoSign*?: `https://www.arx.com/cbosign/why-cosign/overview/`, 2003. [Online; Accessed 2015-12-1]

proof of signer identity, signer intent and document integrity and offering signatures that can be validated by anyone at any time using applications such as *Microsoft Office* or any PDF reader like *Adobe* reader [15].

DocuSign system referred to in section 2.5.3 has a beta version for *Alfresco*. Using this system, allows to easily send a document directly to *DocuSign* for execution and signature. Once signed, the documents are automatically added back to Alfresco.

2.7.2 Digital Signatures *Alfresco* add-ons

Zylk [25] provides some add-ons for *Alfresco*, including digital signatures and validation of a signature. The digital signature add-on provides an action for signing PDF files with a Java Applet (*CryptoApplet*). The applet allows to use a local software certificate or cryptographic hardware in the signature process, and it introduces a visible signature in the PDF, being also possible to sign several selected PDFs in the document library view. The signature validation of PDF files add-on provides the information of the signatures detailed, presenting the different information about signatures like signers, signature dates and the information about certificates (timestamps, Online Certificate Status Protocol (OCSP) validation). However, this add-on no longer supports the new versions of *Alfresco community* from *4.2.x*.

Emmanuel Roux provides an add-on to digitally signed documents with certificate and eventually the signature picture(for example, the hand signature picture) [26]. This add-on allows to create/modify the signature (certificate and picture), this way, a key file is encrypted and protected with a random secret key, allowing also to view signature information with the certificate status. In the signing process, the user must mandatorily select the destination of the signed file and the key password. The user can define another parameters like sign reason, sign location, field, position of the signature and others. The user can sign multiple documents in one action. This add-on supports the new versions of *Alfresco*.

Alfresco PDF toolkit [27] allows the manipulation of PDF files, adding features to *Alfresco*. This toolkit has digital signatures functionality but only has rudimentary signing features. This toolkit do not have validation and only use software certificates.

[25] About Zylk: `http://www.zylk.net/es/web/guest`, 2012. [Online; Accessed 2015-12-3]

[26] *Alfresco* add-ons by *Emmanuel Roux*:
`https://addons.alfresco.com/addons/digital-signing` [Online; Accessed 2015-12-3]

[27] *Alfresco PDF Toolkit*: `https://addons.alfresco.com/addons/alfresco-pdf-toolkit` [Online; Accessed 2015-12-10]

The author of this toolkit, created another add-on directed to digital signatures only, called *CounterSign* [28]. *CounterSign* is a digital/electronic signature solution for *Alfresco*. *CounterSign* is a digital signature toolkit and UI extension for Alfresco, combining user key management, a rich UI and simple signatures. *CounterSign* can be extended to meet specific organisational needs. For now, this add-on supports only *Alfresco 4.x* versions and is not ready yet for the new versions unlike *Alfresco PDF toolkit* already supports the new versions of *Alfresco*.

Another project towards this topic is developed by *Sinekarta* [29], an Italian company. This project is open source and provides digital signatures, electronic storage and electronic invoicing. *Sinekarta* provides an extension for *Alfresco* to transform any document in PDF and digitally sign based on smart cards or USB tokens. The software provides the validation of signatures too.

Digitale Legale is a company that comes from a large company, skilled in DM: scan, management and archiving are the core business of *Datanet SC* [30]. *Digitale Legale* provides an extension for those who are already using *Alfresco* [31]. The add-on allow users to make storage by law with *Alfresco*. The add-on allows to make digital signatures in all documents filed with *Alfresco* with multiple signatures that can be verified with *Adobe Reader*. The use of digital signatures is required to give effect to digital archiving with *Alfresco* and make it completely replacement of the paper version.

2.7.3 Comparison of digital signature systems for *Alfresco*

In this table, we present the principal features of the independent systems to integrate on *Alfresco* and we also present the add-ons of *Alfresco*. We can compare the principal features that we need in our system. The difference of independent systems and add-ons is that the add-ons are designed to work within *Alfresco* only, however, independent systems works without *Alfresco* providing the signature functionality and can be integrated into *Alfresco*.

[28] *CounterSign*: `https://addons.alfresco.com/addons/countersign-alfresco` [Online; Accessed 2015-12-10]

[29] *Sinekarta*: `http://www.sinekarta.org/descrizione.php` [Online; Accessed 2015-12-14]

[30] *Datanet SC*: `http://www.datanetwork.it/` [Online; Accessed 2015-12-15]

[31] *Digital Legale*: `http://digitalelegale.it/` [Online; Accessed 2015-12-15]

Table 2.3: Independent digital signature systems for *Alfresco*

	CoSign	*DocuSign*
Open Source	-	-
Crypt. technology	X.509	X.509
Psychical technology	-	Smart Card
Workflow ready/independent	✓	✓
Workflow *Alfresco*	-	-
One signature	✓	✓
Multiple signatures	✓	✓
Validation	-	-

Table 2.4: Add-ons for *Alfresco*

	Zylk	*E.Roux*	*Toolkit*	*CounterSign*	*Sinekarta*	*Dig. Legale*
Open Source	✓	✓	✓	✓	✓	-
Crypt. technology	X.509	X.509	X.509	X.509	X.509	X.509
Psychical technology	✓	-	-	✓	✓	-
Workflow signatures	-	✓	-	✓	-	-
One signature	✓	✓	✓	✓	✓	✓
Multiple signatures	✓	-	-	✓	-	✓
Validation	✓	-	-	✓	✓	-

With this investigation, we can see that the most popular independent systems/add-ons have most of the features that interest us and help us to see how to add value to the market. Despite of these add-ons, our final product offer more features, such as, an improved validation service, more feedback to user and more creativity.

2.8 Redaction systems

In the comparison and analyse of the redaction systems, we want to see what type of redaction the system does (line redaction, redaction area), in other words, if the system allows to select multiple lines to redact, have the possibility to put a rectangle for example, in an area that we want to protect/hide and have the possibility to put some text above the rectangle. We want to see if the applications use any physical

technology and if exists some systems that allow redaction with digital signatures workflow. We are also interested to see if the applications support redaction of any formats. We examine not only the applications that have been made to integrate the Alfresco (or support this integration) as well as some independent redaction systems to see their features, and where we can explore the market.

2.8.1 Redaction *Alfresco* add-ons

Alfresco does not have redaction feature included. *Infograph* [32] [33] developed a software non open-source, advertising that *offers secure, viewing collaboration, redaction and publishing in a single, browser-based interface.*, however, this add-on supports only *Alfresco 4.1.x* versions and is not ready yet for the new versions. This solution is not only compatible with *Alfresco* ECM as also with ECM^2, *IBM*, *OpenText*, *SharePoint* and supports also the integration SDK, *allowing the flexibility to fit into almost any solution.*

VirtualViewer®HTML5 is an *Alfresco* Document Viewer developed by *Snowbound Software* [34], that has some features including the view of multiple documents, page manipulation and text search & extraction. The most important feature for us is the annotations and redactions. *VirtualViewer provides users with a variety annotation and redaction tools, including rubber stamps, redactions, highlighting, and sticky notes. Annotations and redactions inherit user permission levels set up in Alfresco (Manager, Collaborator, Contributor, and Consumer), ensuring confidential information is only seen by approved users. When page manipulations and annotations are finished being made, the document can be saved and committed to the repository, or even exported or distributed depending on user permissions.*

The application works in any browser in any device, such as laptops, smart phones or tablets. Is a Java solution with Java-based server components. Its not required any download of the application, *making it a trouble free solution for users as well as IT administrators.*

[32] ICG. software - `http://alfresco-demos.infograph.com/` [Online; Accessed 2016-5-3, 2016]

[33] IGC Brava Viewer - `http://www.infograph.com/sites/default/files/BravaforAlfresco.pdf` [Online; Accessed 2016-5-4]

[34] *VirtualViewer* - `http://www.snowbound.com/products/document-viewer/html5-alfresco-viewer` [Online; Accessed 2016-5-4]

Table 2.5: Redaction systems for *Alfresco*

	IGC Solutions	*VirtualViewer®HTML5*
Open Source	-	-
Physical technology	-	-
Redaction and signatures workflow	-	-
Alfresco new versions	-	✓
Select lines redaction	-	✓
Select redaction area	✓	✓
Redaction area text	-	✓
Redaction of any format	✓	✓

In this table we can see the features of both add-ons that are available in official website of add-ons [35]. *VirtualViewer* has a demonstration on-line website that we can test and see all features [36]. In the demo of the *VirtualViewer* we cannot test the application with *Alfresco* but we can see the application that is integrated in *Alfresco*. However, *ICG Solutions* does not have any demo allowed. The free trial must be accepted by the company, and as of writing this thesis, we were not granted permission and some APIs that are allowed in the website are not resolved, so, we could not test this application at all. So, we only saw features in a video demonstration and in the description of the application, thus, we cannot conclude if all features that we include in the table are present/not present with our own testing.

We conclude, however, that there is not any add-on that makes an integration with digital signatures in workflows and both does not use any physical technology. The other features are present at least in one of the two applications.

[35] *Alfresco* add-ons - `https://addons.alfresco.com/` [Online; Accessed 2016-5-4]
[36] *VirtualViewer®HTML5* - `http://html5.snowbound.com/` [Online; Accessed 2016-5-4]

Chapter 3

Security and Privacy Mechanisms

In this chapter, we present an overview of a set of security and privacy mechanisms, such as, smart cards and digital signatures. We provide a description of this mechanisms, as well as, comparisons between smart cards and software certificates, digital signatures and another types of signatures, redaction and QR-codes.

3.1 Electronic Signature vs Digital Signature

These two concepts are often confused by people in general. However, a digital signature is an electronic signature but the reverse is not the case. Electronic signature is easy to implement, because a simply typed name can serve as one. Therefore, this type of signature has many problems to maintaining integrity and security, as there is nothing to prevent one person from typing another persons name. Due to this reality, electronic signatures is an insecure way of signing documentation. Electronic signatures are vulnerable to copying and tampering, making forgery easy. There are some examples of electronic signature such as, the scanned image of the person ink signature, the signature with a digital pen, a typed name, a signature at the bottom of an email, a biometric hand-signature, a video signature or a click in an "I agree" check box. The main point is that an electronic signature is any "mark" made by the person to confirm their review/approval of the document [20].

In the case of the digital signature, this is a mathematical scheme for demonstrating the authenticity of a document. A valid digital signature gives a recipient reason to believe that the message was created by a known sender and the message was not altered

during the transport. Therefore, this sender cannot deny having sent the message, that ensures authentication, non-repudiation and integrity. Digital signatures comply laws and regulations. This helps organisations ensure signer authenticity, data integrity, and the verifiability of signed electronic documents.

Any changes made after the document has been signed invalidate the signature, thereby protecting against signature forgery and information tampering [21]. According to Portuguese law [1], electronic signatures have the same evidential validity as handwritten signatures, provided they are based on certificates issued by accredited certification entities. They are called digital signatures.

Therefore, the handwritten signature of an individual is always the same (or at least very similar) in all documents signed. This feature makes it extremely easy to fake a handwritten signature. Moreover, the digital signature of an individual is always different because the digital signature is composed of a bit pattern built with cryptographic techniques (complex mathematical procedures) and always depends on two values: The data of digital signature creation stored in the Citizen Card, for example, and the document to be signed. Thus, if one copy changes bit pattern from one document to another, the digital signature is no longer valid. In addition, to enhance security, the bit pattern is produced within the Citizen Card chip itself. This ensures that no one can clone the signature creation data stored in Citizen Card (not even the government of a country). These features make the digital signature much more secure and reliable than a handwritten signature.

Thus, the next subsections describe some sections about the digital signatures:

3.1.1 Combining Digital Signatures with Electronic Signature

Nonetheless, electronic signature can be combined with a digital signature and gain legal value. It is important, today, generate a digital signature by deriving a signature key from human biometrics. Biometrics is the science of using digital technologies to identify a human being based on the individuals unique measurable biological characteristics [22]. With an electronic biometric signature, users can see his handwritten signature in the document and this is an important feature for usability. It is important

[1] Decreto-Lei n. 290-D/99, de 2 de Agosto: `https://dre.pt/application/dir/pdf1sdip/1999/08/178A01/00020011.pdf`, 1999. [Online; Accessed 2016-3-13]

to have this complement in a signature system because users have a past connection with the signatures on the paper and are more comfortable if they can see their usual handwritten signature on the document.

3.1.2 Types of Digital Signatures

(Public Key Cryptography Standards (PKCS)#7) is a standard defined by Rivest-Shamir-Adleman Cryptosystem (RSA) describing a general syntax for data to which cryptography may be applied, such as digital signatures. PKCS#7 supports some different content types: data, signed data, enveloped data, signed-and-enveloped data, digested data, and encrypted data. Beyond PKCS#7, there are other formats to encode the cryptographic messages, that are been proposed to improve security and interoperability [23].

There are some types of digital signatures. Comparing two standards, XML Advanced Electronic Signatures (XAdES) and CMS Advanced Electronic Signatures (CAdES), that serve the purpose of digitally signing any type of data using qualified certificates. Both of the standards allow the storage of attributes such as the Multipurpose Internet Mail Extension (MIME) type of the data to be signed, signing time, for example [24]. CAdES is a set of extensions of Cryptographic Message Syntax (CMS) signed data and is built around them. CAdES enables signing of any data, including PDF; renders signature as binary data; required customisation of applications or generic signing outside the application; supports multiple signatures applied in parallel, serial by repeated signing; appearance is up to the application to provide; provides long-term validity and supports two types of signing methods: detached, the data being signed is separated from the signature and encapsulated, the data is packed within the signature structure. CAdES specification defines six profiles: CAdES is the basic form that provides authentication and integrity through elements for this; CAdES-T is the addition of the timestamp that provides non-repudiation; CAdES-C (Complete), CAdES-X (eXtended), CAdES-X-L (eXtended Long-term) and CAdES-A (Archieval). XAdES is based on CAdES but required the syntax of XML. XAdES introduces the attribute *DataObjectFormat* to describe the encoding format of the signed data. This type of signature signs at any data including PDF and binary and supports XML package or separated files; as well as the CAdES, XAdES frequently required customisation of applications or generic signing outside the application, supports multiple signatures applied in parallel, serial by repeated signing and provides long term validity. Also supports a visual signature appearance, depending on the application. The six profiles

of the XAdES are the same as those of CAdES. The different between this and the CAdES is that while CAdES renders signature as binary data, XAdES provides an XML solution [25]. PDF Advanced Electronic Signature (PAdES) is a proprietary format for digital signatures in a PDF documents where a PDF can be seen as two compartments house. The first contains the PDF document to be signed and the second contains the information required by digital signatures, like, users certificate and the encrypted digest (Digital Signature Algorithm (DSA) and RSA are supported). In PAdES, it is possible to sign more than just the document such as, time stamp obtained from a trusted server, a graphical signature, the system and the user software application. This kind of signature has some strong advantage in terms of resistance to ambiguous-presentation attacks [23]. In resume, PAdES have the signature within the PDF, in other words, has a self-contained signature; supports XML data; it has signing and verifying in PDF software; as well as XAdES, this type of signature supports a visual signature appearance in the document and provides a long term validity [25].

It is possible to make digital signatures in the documents with known programs such as: *Microsoft Word* that allows to make a digital signature to ensure the integrity of the document adding an invisible digital signature, *Libre Office* and *Adobe Acrobat Reader*, for example. It is possible to make the signature in all operating systems: *Linux*, *Windows* and *Mac*, using different software and files.

Another example of the signatures are in e-mail. In *Microsoft Outlook* it is possible to sign digitally. Another example of the software with this feature is the *Mozilla Thunderbird*.

3.1.3 Cryptography Concepts

Digital signatures use a public and private key pair that are usually purchased by a sender and issued by a CA. A key pair are mathematically related because a message encrypted with a public key can only be decrypted with a private key. For the signatures, a sender uses his private key to sign a document and the recipient uses the senders public key and the signature to confirm the authenticity of the document. The private key is received by a person and remains secret. This key is only distributed to the private key owner. The public key, can be made available for anyone and can be found by accessing a CA public database. CA is a trusted third party who verifies the identity of the person requesting the key pair and can be created through a PKI [26]. According to the authors of [27], *a PKI is a set of hardware, software,*

people, policies and procedures needed to create, manage, store, distribute and revoke digital certificates (also called public key certificate) based on public-key cryptography. PKI is an arrangement that binds public keys with respective user identities by means of a CA.

3.1.4 Security properties

The data integrity, authentication, confidentiality and non-repudiation are the main security services provided by a PKI.

- **Data integrity** provides information to the sender and receiver has the insurance that the message has not been altered during the transmission. Therefore, integrity ensures that the data signed is not modified after being signed by a person. If the person signs the document and the data has been modified, the signature becomes invalid;
- **Non-repudiation** ensures that a person signing certain data can not deny later that signed the data. Only the person that owner the smart card should sign with them, because user needs the signature Personal Identification Number (PIN) that is secret and only the owner knows. Also, if a fraudulent party access to the public key, cannot fake a valid signature. This is an important characteristic of the digital signatures. The messages, generally, include the information about the entity that sends a message, however, this information may not be accurate;
- **Authentication** - The digital signatures can be used to authenticate the source of the message. The properties of a private key can give the information about the user that is the owner of this private key, since the secret key is binding to the specific user;
- **Confidentiality** is the guarantee of the data privacy. The entities that are not allowed, cannot read the information, except the specific receiver(s).

3.1.5 Public-Key Certificate (X.509)

X.509 is the most used data format for public-key certificates, today, and it is based on the use of CAs that verify if the entity is the holder of a certain public-key by signing public-key certificates. A public key certificate is digitally signed by a issuer

of the CA to confirm that the identity or a information in the certificate belongs to the holder of the corresponding private key. A digital certificate makes a binding connection between an identity and the key pair (private/public) held by the holder of the identity [28] [29]. The actual standard is defined in Request For Comments (RFC) 5280 [30] which describes Internet X.509 PKI Certificate and Certificate Revocation List (CRL).

The standard information in an X.509 certificate includes:

- **Version** - which X.509 version applies to the certificate (which indicates what data the certificate must include);
- **Serial number** - the identity creating the certificate must assign it a serial number that distinguishes it from other certificates;
- **Algorithm information** the algorithm used by the issuer to sign the certificate;
- **Issuer distinguished name** - the name of the entity issuing the certificate (usually a certificate authority);
- **Validity period of the certificate** - start/end date and time;
- **Subject distinguished name** - the name of the identity the certificate is issued to;
- **Subject public key information** - the public key associated with the identity;
- **Extensions** (optional).

3.1.6 Digital Signature Scheme

A digital signature scheme provides a cryptographic analogue of handwritten signatures that provides much strong security guarantees. In many countries, digital signatures is a powerful tool and are accepted as legally binding. This scheme is used by a signer and a set of verifiers. The signature scheme consists of three probabilistic, polynomial-time algorithms (*Gen*, *Sign*, *Vrfy*) along with an associated message space $M{=}M_k$. The signer starts by running some randomised key-generation algorithm *Gen* to produce a pair of keys (pk,sk), where pk is the signers public key and sk is the signers private key (also called secret key). The security parameter k is implicit

in both pk and sk. For security parameter k, the signing algorithm *Sign* (possible randomised) takes as input a private key sk and a message $m \in M_k$ and takes as output a signature $\sigma \leftarrow Sign_{sk}(m)$. If $m \notin M_k$, the signature algorithm outputs $\perp$. For security parameter k, the verification algorithm *Vrfy* takes an input a public key pk, a message $m \in M_k$ and a signature σ. The output produces a bit, with $b = 1$ that means "accepted" and $b = 0$ that means "reject". This is written as b:= $Vrfy_{pk}(m, \sigma)$. If $m \notin M_k$, the verification algorithm return "reject" [31].

In summary, a digital signature is composed of a unique digital certificate for each signer; a private key which only the signer can use to sign and a public key which allows anyone to validate the signature. Signers can include, in digital signatures, for example their name, date, time stamp, their reasons for signing and also can include graphical signatures.

3.1.6.1 ElGamal Encryption

As presented in the book [32], *the ElGamal Encryption (ElGamal) cryptosystem is an asymmetric-key encryption scheme named after its inventor Taher ElGamal.* The ElGamal scheme can be used to digital signature and to cipher. The security of this method is based in the difficulty of compute a discrete logarithm - called Discrete Logarithm Problem (DLP) [2] in a finite field.

To generate a key it is necessary: a prime p and two numbers g and x, with $0 < g, x < p$.

The principal formula:

$$y = g^x \ mod \ p$$

The public key is $< y, g, p >$ and the private key is $< x >$.

Signatures with ElGamal

Let's imagine a message m. To sign the message m, it is generated a random k with $(k, p-1) = 1$ and is is calculated:

$$a = g^k \ mod \ p$$

and is used the extended Euclidean algorithm [33] to solve the equation:

$$m = (xa + kb) \ mod \ p - 1$$

[2]The DLP is the computational problem of finding $x = log_a(b)$ such that $b = a^x$

The signature is formed by the pair: $< a, b >$ and the random k keeps secret.

To verify a signature, it is used:

$$y^a a^b \ mod \ p = g^m \ mod \ p$$

A Possible Attack

For each signature it is needed a new random k value, in terms of security.

Let's imagine an intruder called *Eve* and a signer *Alice*. If *Eve* can recover the random k that *Alice* use, *Eve* can recover the private key of the *Alice*. If *Eve* gets two signed/cipher messages used the same k, even not knowing what each message contains, *Eve* can recover the x (the private key of the *Alice*).

3.1.6.2 Digital Signature Algorithm

As described in the paper [34], *DSA has been suggested and standardized by the National Institute of Standards and Technology.* It is an efficient variant of the ElGamal signature scheme described in the section 3.1.6.1, since given today security standards, an ElGamal signature is of bit length at least 2048 while DSA is of bit length 320. In the signature verification, ElGamal scheme requires three modular exponentiation with exponents of bit length at least 1024 while DSA only required two modular exponentiation with exponents of bit length 160.

Signatures generation with DSA

A signer wants to sign a document x. There are global public-key components. p is a prime number where

$$2^{L-1} < p < 2^L,$$

where $512 \leq L \leq 1024$ and L a multiple of 64, in other words, bit length of between 512 and 1024 bits in increments of 64 bits. q is a prime divisor of $(p-1)$ where

$$2^{159} < q < 2^{160},$$

in other words, bit length of 160 bits and

$$g \ = \ h^{(p-1)/q} \ mod \ p$$

The signer choose a random number $k \in 1, 2, ..., q-1$ computes

$$r = (g^k \ mod \ p) \ mod \ q,$$

and sets

$$s = k^{-1}(H(M) + xr) \ \ mod\ q,$$

where M is the message to be signed, $H(M)$ = hash of M using $SHA-1$ and x is a random or pseudo-random integer between 0 and $q-1$.

Signatures verification with DSA

To make the verification of the signature, the verifier can make:

$$w \ = \ (s')^{-1}\ mod\ q$$

$$u1 \ = \ H(M')w\ mod\ q$$

$$u2 \ = \ (r')w\ mod\ q$$

$$v \ = \ ((g^{u1}y^{y2})\ mod\ p)\ mod\ q,$$

where M', r', s' is the received versions of M, r, s.

After the calculation of these formula's, test if v is equal to r'. If this is equal, the signature verifies too.

3.1.6.3 RSA Algorithm

As described in the paper [35], this is an algorithm developed by *Rivest, Shamir e Adleman*. The original text is cipher in blocks and each block have a value, in binary, less than a given n. The block should have a size less than log_2(n) bits. Let M the original text block and C the cipher text block:

$$C = RsaPublic(M) = M^e\ (mod\ n)$$

$$M = RsaPrivate(C) = C^d\ (mod\ n) = (M^e)^d\ (mod\ n) = M^{ed}\ (mod\ n)$$

The key generation

1. Generate two random p and q primes of approximate equal size, such that, their product $n = pq$ have the required bit length;
2. Compute $n = pq$ and $\phi = (p-1)(q-1)$;
3. Choose an integer e, such that, $1 < e < \phi$, such that, $gcd(e, \phi) = 1$;

4. Compute the exponent d, such that, $1 < d < \phi$ and $ed \equiv 1(mod\ \phi)$.

5. The public-key is composed by: $k_p = e, n$ and the private key is composed by: $k_s = d, n$. The values d, p, q and ϕ are secret. [36]

Let's imagine two persons, *Alice* and *Bob*. *Alice* and *Bob* know the n value. *Alice* know the e value and only *Bob* know the d value. *Alice* encrypt a message with the public key and the *Bob* decrypt the message with the private key.

For the algorithm to work satisfactorily is needed that:

1. Be possible to choose e,d and n values, such that, $M^{ed} = M$ (mod n) for all $M < n$.

2. Be relatively easy to calculate M^e and C^d for any value of $M < n$.

3. Be impracticable to determine d given e and n.

RSA digital signature and verification

Bob wants to sign a message. First, *Bob* creates a message digest of the information to be sign. Represents this digest as an integer m, such that, $1 < m < n - 1$. *Bob* uses his private key $k_s = (n, d)$ to compute the signature $S = M^d(mod\ n)$. Finally, sends the S to the recipient, *Alice*.

Alice wants to verify the message S. *Alice* uses the *Bob*'s public key - $k_p = e, n$ to compute the formula $V = S^e(mod\ n)$. Extracts the message digest of this integer and computes the message digest of the message that has been signed. If both message digests are identical, the signature is valid [37].

Possible attacks:

- **"Brute force"** that involves trying all the keys.

- **Mathematical Attacks** - there are several approaches, all equivalent to factoring the product of two primes.

- **Attacks by timing** that depends on the decipher algorithm execution time.

3.2 Smart Cards vs. Software Certificate

Security solutions based only in software are not safe and are very vulnerable to some attacks. The reason for this lack in security is the conventional storage media use to store certificate and private key are not secure.

We can use a certificate without smart card or token but, smart cards and tokens are the most secure place to store the certificates private key. This key must be kept secure at all times and is used to create digital signature. When we place the private key in a smart card or in a token, the private key never leaves them. Typically, the certificate must be used in different computers, so, with a smart cards or tokens, we can use the certificate in more than one computer, without the need of the creation of multiple copies of the certificate and keys. This has a lower complexity and lower cost management of certificates and keys. In terms of security, the use of a physical device to store certificates provides most protection, since the cryptographic operations are independent of the operating system, thus, there is not concern of any attacks on the operating system that can put the certificates in risk.

A smart card is a device that contains an embedded integrated circuit chip. Smart cards provide some functions as encryption, mutual authentication and interact with the reader using specific protocols. A micro-controller smart card provides a secure and portable way to securely manage cryptographic keys and corresponding X.509 digital certificates, in a PKI context. Smart cards enhance the PKI security through an extra authentication level ("something you have") and with fact that cryptographic keys generated on the card never leave the card. This ability to ensure that keys cannot leave the card, provides an effective means to ensure non-repudiation. PKI smart cards can provide most main security functions in modern information systems: authentication (X.509 digital certificate), confidentiality (based on asymmetric private key), data integrity (digital signature) and non-repudiation (digital signature by asymmetric key generated and stored on the card) [38].

Furthermore, the functionality with smart cards are based in 2 factors (something you know and something you have). That is, it ensures that the user of an electronic service has the smart card and also know the PIN of the smart card.

When a user uses the smart cards to make digital signatures, it is needed the introduction of the digital signatures PIN code of the smart card. Usually, users enter the PIN code with a keyboard of the PC, therefore, with a keystroke logger [3], hackers

[3]Keystroke logging is the action of recording (logging) the keys struck on a keyboard.

can make eavesdropping of the PIN code security. In this case, it is important to use a card reader with a numeric keypad. This specialised card reader are also less vulnerable to tampering with their software or hardware and are often Evaluation Assurance Level (EAL)3 [4] certified [40].

Now we describe some examples of smart cards that can be used to make digital signatures.

3.2.1 Citizen Card

The appearance of the Citizen Cards revolutionised the way of how citizens can relate with some entities (public or private). More than one physical identification, the Citizen Card is an electronic document which allows the realisation of multiple operations without the need to face interaction.

Figure 3.1: Citizen Card example

The electronic use of the Citizen Cards allow multiple operations:

- The secure electronic authentication of citizens to public and private services;

[4] Methodically Tested and Checked - *EAL3 permits a conscientious developer to gain maximum assurance from positive security engineering at the design stage without substantial alteration of existing sound development practises. EAL3 is applicable in those circumstances where developers or users require a moderate level of independently assured security, and require a thorough investigation of the Target of Evaluation (TOE) and its development without substantial re-engineering.* [39]

- The creation of qualified electronic signatures, using the certificate available on the card, which contains the same signature value as a handwritten signature;
- The address confirmation activity (last phase of the address change process) [5].

Digital Certification allows information transiting the Internet more safely and to ensure, of course, who was the author of a transaction or a message, or even keep sensitive data protected against reading by unauthorised persons.

These are only some examples of what an user can do with a digital certificate:

- Digitally sign any computer file that is a document, e-mail program or in the case of a document having the same legal validity as a document signed by hand;
- Client authentication in Hyper Text Transfer Protocol Secure (HTTPS) server (Apache, Internet Information Services (IIS)).

The certificates contained in the card (Authentication certificate that allows authenticate citizens on a secure Web site and signature certificate that allows sign documents and files, ensuring its integrity and proving the identity of the author) are issued by the respective sub-certificate authority. The sub-certificate authorities are branches of the Common Certificate Authority for the Portuguese government.

The smart card contains a micro controller supporting the latest Java Card version that have the on-card cryptographic functions:

- RSA signature and verification;
- Qualified electronic signature (Secure Signature-creation devices "EAL 4+");
- Data Encryption Standard (DES) and TDES (Triple Data Encryption Standard);
- Message Authentication Code (MAC);
- PKCS#1 (RSA Cryptography Standard)
- PKCS#15 (Cryptographic Token Information Format Standard);

[5]Citizen Card tutorial - `https://www.cartaodecidadao.pt/documentos/Manual_Cartao_de_Cidadao_v1.26.0.pdf` [Online; Accessed 2016-4-13]

A PIN code is a numerical password consisting of 4 to 8 digits that is delivered to the cardholder and allows authenticate it with the system. The Citizen Card includes three PIN codes:

- Authentication PIN - code required for authentication on a Web site or application that uses the Citizen Card, and store information in a personal folder;
- Signing PIN - code required when a citizen want to digitally sign a document or a message;
- Address PIN - code necessary to access the address on the Citizen Card.

In terms of security, the citizen card is ready to resist to attacks such as Hardware Attacks, Timing Attacks, Simple Power Analysis and Differential Power Analysis.

In our proposed work, the idea in practical use is to make digital signatures with Citizen Card.

3.2.2 U.Porto Card

A partnership between *University of Porto* and *Banco Santander Totta*, gave rise to the U. Porto card. This card was introduced in 2008. Integrating different technologies, contact and contactless, this card provides access to a growing number of features and services within the university campus.

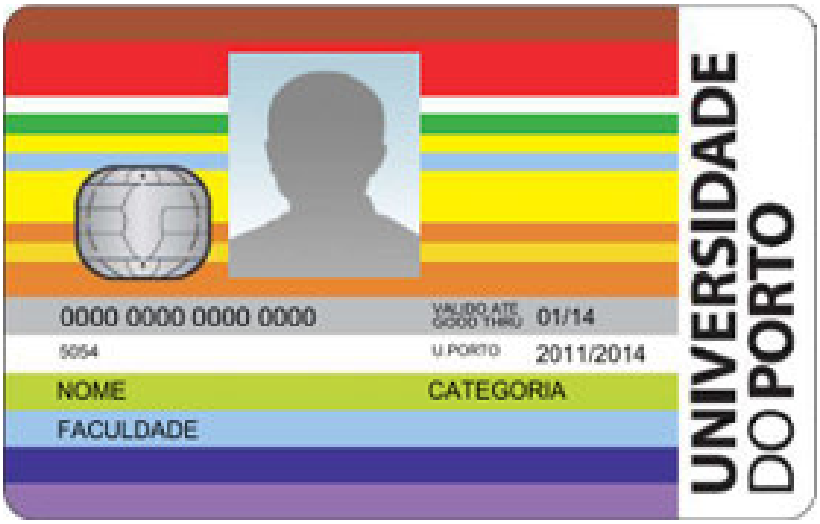


Figure 3.2: U. Porto Card example

This is a smart card that integrates an identification (provides a way to identify students and other members of U. Porto such as teachers and staff), an physical access control (that leads with the functionality of allow the access to internal services in the university for authorized persons) and third party services (access to other services provided by third parties, under special terms).

Another functionality are the digital signatures, authentication, internal payment function, attendance control and optionally ATM.

In our proposed work, the idea in practical use is to make digital signatures with U. Porto card. In our work, we have not yet developed compatibility with this card.

3.3 Redaction

The redaction is the process of hide or remove visible text or images from a document, as we can see in the figure 3.3.

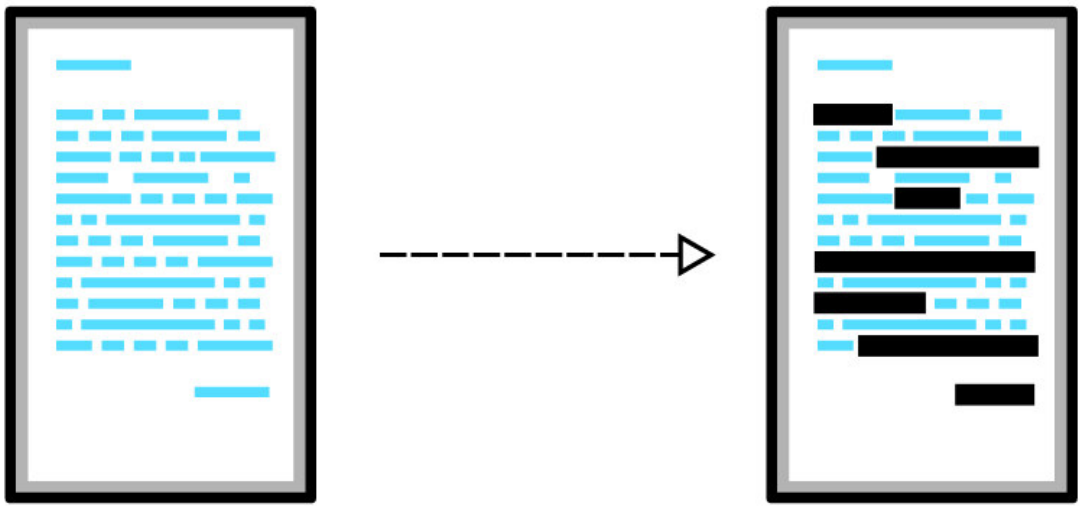


Figure 3.3: Redaction example

In the legal context, when dealing with confidential documents or privileged information, there is always a concern if the document will fall into the wrong hands. There are some cases of leakage and disclosure of sensitive data. For this reason, redaction is used to remove confidential, proprietary or privileged information from documents before submission to the court making them available for viewing out of the office.

Before the digital era, the redaction was done physically with paper. The redaction was done with the use of a black marker and then was photocopied. Often the text was still visible beneath the marker, and therefore the document had to be marked and copied many times to make the text invisible.

Another method to improve the weaknesses of the black marker, was the use of a white adhesive. It was cut a adhesive in various sizes and placed over the text. This way, is no longer possible to see the confidential text and if wanted, people can remove the white adhesive at any time. However, this method was slow and laborious, although it has improved the method of the black marker.

With the appearance of the digital era, technology allows to redact text confidential electronically in any PDF document, for instance.

Using a Mark Redaction tool, a user can select the confidential text that he wants to hide. The user can select the color of the redacted area and can also place text over the redaction. The user can select the font, size and color of the text. Usually, the redaction tools, the finalized redaction document has the appearance like the figure 3.3. At the end of the redact process, the redaction turns the document impossible for a viewer to access or uncover the redacted areas. All redaction software applications have an additional function that can search for all metadata remaining in the document and give the user the option of removing some or all of it [6].

There are many advantages to switch to electronic redaction:

- **More reliable:** In the Mark Redaction tools, generally, exists a search feature that provides to the user a way to find all the confidential information, by words or specifically phrases. The resulting redactions are more reliable because not only depend on the human eye to locate the confidential material.

- **Faster:** The electronic redaction is faster than the physical redaction to mark and remove/hide text with quick mouse strokes than to place adhesives or black ink over confidential text.

- **Cost Savings:** In addition to saving paper costs, ink toner or ribbon, there is also a time saver in the process of redaction, since workers can be freed of the task of place tape in desired locations, seek information only with eyes or paint

[6] Why You Should Switch to Electronic Redaction? `http://proparalegal.com/wp-content/uploads/2012/06/Electronic-Redaction.pdf` [Online; Accessed 2016-5-16]

the sensitive parts of the paper with black ink. The most costly labor can be saved and thus reduce the working time.

- **Appearance:** Electronic redactions look clean, neat and tidy. Paper documents redacted with markers or tape look messy and unprofessional. In addition, the roles can be damaged with ink and rip paper with tape.
- **Ecologic:** Paper, toner and electricity are saved because there is no need for scan and print of the original redacted document.

3.4 QR-Codes

QR-codes was invented by *Denso Wave* in the early 1990s. *Quick Response* refers to a fast scan. This is a two dimensional bar code (Figure 3.5). QR-codes are made through black and white pixels that allow the codification until some hundreds of characters. In comparison with the 1D dimensional bar codes (Figure 3.4), QR-codes has a more range of encoding, a larger capacity (while the bar-codes usually store up to 30 numbers, QR-codes can usually store up to 7000). Also has a high error-correction rate. QR-codes is also confidential and anti-counterfeiting in some cases [41].

Figure 3.4: Bar-code example

Figure 3.5: QR-code example

The QR-codes has some features. The high availability in mobile phones, because any camera can scan QR-codes with a QR-code reader installed in the smart phone.

Furthermore, anyone who has a smart phone knows how to use the camera and can easily learn to read the QR-codes. QRs can be printed in any printer, using common paper, so, it has a low cost. It has a damage resistance, since includes error correction data that allows to recovery until 30% of damaged tag or distorted. In terms of visibility, the code must be visible and well illuminated. In terms of security, this can be a problem of the QR-codes, since the information can be read easily by any camera-enabled device, from any direction in 360° [42].

Chapter 4

Implementation

In this chapter, we will describe the technical implementation of the proposed integration of digital signature with an ECM, in this case, *Alfresco*. We took the fact that this ECM has support for BPM and workflows to integrate digital signatures in a workflow where people could define who signs a specific document.

4.1 Digital Signatures Application

We focus on the signatures in PDF documents. We implement the signature in this type of a document because, as we can see in the section 3.1.2, this kind of signature is more resistant to attacks. One interested property is the time stamps. Timestamping is the process of securely keeping track of the creation and modification time of a document. No one, not even owner of the document, should be able to change it once has been recorded. That way, integrity is ensured. The timestamp is obtained from a trusted external server to have the guarantee that the service that we are using is not changing the timestamps [43]. This can be considered as the stamps made by a notary in a paper.

We used the Citizen Card to provide a way to users sign safely and quickly as described in section 3.2. It also facilitates the users because it is a card that everyone has. An user has only to purchase a card reader and know the PINs of the respective card. There are three PINs in Citizen Card: authentication PIN - prompted for authentication code in a Web page or application that uses the Citizen Card, and store information in the Personal Folder; PIN digital signatures - code required when

users want to digitally sign a document or message and address PIN - code necessary to access the address in the Citizen Card. People just need to know their digital signature PIN to our service. Personal information is accessed without any PIN through the X.509 certificate.

We start by implement a middleware between a Java Applet and *Alfresco* as a proof of concept. This application has the functionality to sign a PDF with a Citizen Card and extract Citizen Card information as X.509 certificate. With X.509 certificate we can obtain the personal information of the person that is the owner of the Citizen Card. In the signature, is inserted the name and date (mandatorily insert automatically when users signs a document) and users can insert a custom reason and location of the signature.

We make this integration to see which is the best ways to integrate, test workflows and explore *Alfresco*. However, once incorporated, we begin to think a way to convert this technology to a newer and safer. Browser-based applications can use Netscape Plugin Application Programming Interface (NPAPI) [1] plug-ins to communicate with native applications. Applets are not supported by some browsers. *Safari* and *Mozilla FireFox*, for example, still supports the Java Applets. However, in browsers such as *Google Chrome* and *Microsoft Edge*, the NPAPI technology is no longer supported. To make our portable and workable system in any browser, it was necessary to change technology. Furthermore, also the *Mozilla FireFox* has announced the removal of applets. The support was removed because of the security risks, such as, expose the native application functions to unwanted extensions for web applications. An attacker may exploit vulnerabilities in the extension to install malicious software on the user machine. The code that runs in NPAPI has the full permissions of the current user and is not sandboxed [2] or protected from malicious input.

We transform the Java Applet in a Java Application that runs locally in the PC. It is a Java ARchive (JAR)-Packaged Software that is to be started from the command line instead of Java Applets that runs inside a browser.

This application has more features that the other, we can extract the photo of the Citizen Card and the address (in this case, it is necessary to use the address PIN).

[1] NPAPI is an API that allows browser extensions, such as Java Applets, to be developed for web browsers.

[2] A sandbox is used to separate the programs execution. Used to run code and programs that is not trusted from untrusted third parties, users and websites. The sandboxes allow the running of the untrusted services in a controlled space with a set of strict resources, such as, restrict or disallow the network access, the inspect or the read from input devices [44]

This way, users can insert the photo in their user profile in *Alfresco*. The address is another information that can be useful in the future, for some features that we want to implement, so it is important to extract any card information through the application, so it can be adapted to integrate the *Alfresco* easily for future features.

In this application it is possible too, sign XML files. We want to increase compatibility with other types of document than PDF, and in this thesis, we start by adding already XML.

With the migration of the Java Applets to the Java Application, the interface of the signing within *Alfresco* became faster and more straightforward for the user. With Java Applets, users had to accept the execution of Java Applets in the browser and wait for Java Applets load to sign the document. However, with this application, users can sign a document and the only interaction with the interface is if the user wants to put a reason of the signature and needs mandatorily to enter the PIN code.

4.2 QR-Redactor Application

The protocol of our application of QR-redacting consists in multiple clients and a server. If we have a client A that is the person that wants to redact a document, a client B that is the person that wants to access the hide information of the document and a server. The client A makes some steps to make the redaction:

1. Cypher parts of the document with Advanced Encryption Standard (AES) (document + cryptogram + Initialization Vector (IV) + unique key);
2. Signs the cryptogram and the access list to the cryptogram;
3. Sends the cryptogram and the access list to the server;
4. Generates a QR-code (the signature of the cryptogram, the IV + the unique key and the identifier of the document).

The step (1) consists in cypher parts of the document with AES, that has some components: a document + cryptogram + IV + unique key. In this case, the cryptogram are the confidential parts of the document (the parts chosen by the client A) encrypted. Basically, if the client A has the document with the phrase *Hello, my name is Patricia* and if the client A thinks that the name *Patricia* is

confidential then the document becomes *hello, my name is ***** and the cypher is *cypher("Patricia", VI,key)*

The client A can also choose if he wants to cypher different parts of the text for different users. For example, if we have two departments: the medical and finance department, the client A wants to cypher confidential information only for medical department users and cypher another information only for finance department users. In this case, the signature of the cryptogram (step 3), by the client A, can be made in all information or only partially.

The server monitors the accesses to the cryptogram (for example, server can see when and which users access to the cryptogram) and can control these accesses to the cryptogram, which means that only the allowed persons can have access to the document hide parts of the text.

The client B makes some steps to access the document redacted (physical access):

1. Access to the cryptogram stored in the server (digital access);
2. Signature validation (to confirm the identity of the redactor, that is the client A);
3. Decode cryptogram.

The protocols are based on the following assumptions:

- Trusted server;
- Defined PKI.

4.2.1 QR-Redactor practical use

We have an application developed to make the redact of the document. It is a tool capable of protecting physical documents, leading the user through simple steps to achieve a qualified document, with all sensitive information hidden.

This application, at this time, only works with PDF document (but could be extended to other formats). The application controls the users that has access to the document sensitive data. We can select a document and insert in the application. Then, we can select which users have access to the document. After that, is displayed the document

inserted and the users can put a black box, select the size of the box, and put over the text or images in the document. Users can too, select a piece of text, and in this text, is placed a black box too. User has to use a smart card to sign, that builds and completes the redaction process. At this moment, only is allowed the Citizen Card, but this could be extended to other smart cards. After redacting the document, a new document will be created, containing a small QR-Code in the last page. This QR-Code will be the key to access the documents classified data, and its access is protected by the server, allowing the user to pick who can and cannot view the document.

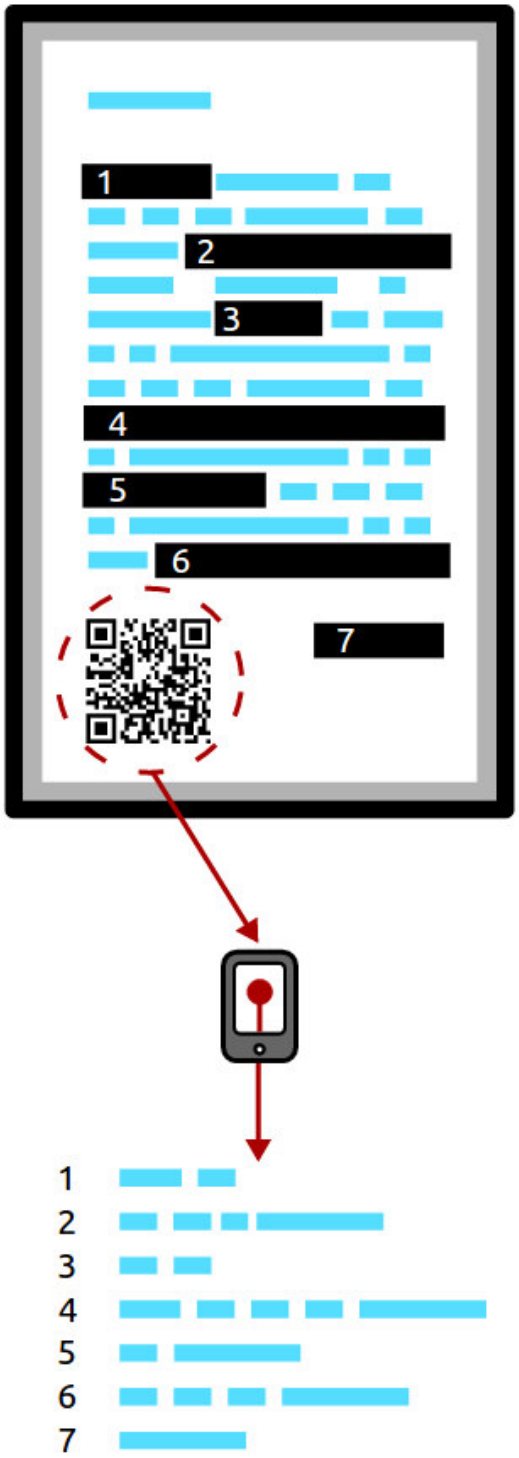


Figure 4.1: QR-Redactor *Android* Application

In the figure 4.1, we can see the process after redacting. A document is generated with a QR-code in the last page, and users that have permissions to access to the sensitive

data, with an *Android* application and a QR-code reader, can access to the data, like the figure. The redacted document has numbers in each black box, and in the smart-phone is displayed the confidential data. Users that does not have permissions, cannot access to the data.

We can see in the figure 4.2 the initial interface after the login. This application has some functionality for another applications, but, one of them, it's for QR-redactor. We have the an option *Disclose your redacted documents! Scan the QR-Code to retrieve the values that you have access to. Ensure that the environment around you is secure.* that allows to access the confidential data for the allowed users.

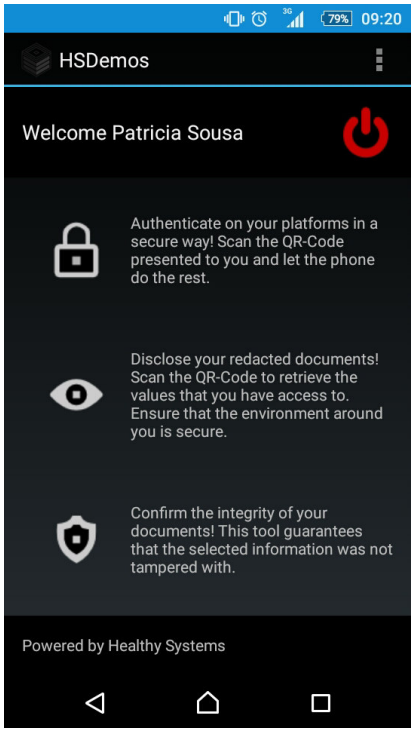


Figure 4.2: QR-Redactor *Android* Application (Initial menu)

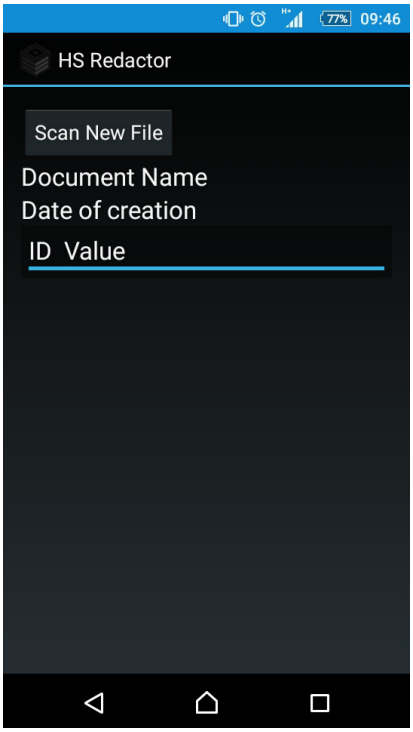


Figure 4.3: QR-Redactor *Android* Application (Scan new file menu)

In the figure 4.3, we can see the next menu that has an option *Scan new file* that opens a QR-reader to the user can reads the QR-code of the protected document. If the user is allowed to see the hidden data, is displayed a menu with the confidential data, and the numbers that corresponds to the black box of the protected document (we can see an example in the figure 4.4, the first page of the document redacted and in the figure 4.5, the last page of the document redacted. In the figure 4.6, we have an example of the hidden data for an allowed user).

[redacted]

Patrícia Sousa, Pedro Faria, Manuel E. Correia, João Resende, and Luís Antunes

Department of Computer Science, Faculty of Science, [redacted]

Abstract. There are some obstacles, towards a paperless office. One of them is the collection of signatures, since nearly half of all documents are printed for the sole purpose of collecting them. Digital signatures can have the same legal evidential validity as handwritten signatures, provided they are based on certificates issued by accredited certification authorities and the associated private keys are stored on tamper proof token security devices like smart cards. In this article, we propose a platform for secure digital signature workflow management that integrates secure token based digital signatures with the Enterprise Content Management [redacted] where each user can associate a set of smart cards to his account. The documents can then be signed with the citizen card or other smart card that has digital signatures capabilities. We have implemented an *Alfresco* module that allows us to explore several workflow techniques to implement real task secure digital signatures workflows, as people for example do when they pass a paper document between various departments to be signed. Since all users can see the current state of the documents being signed during the entire signage process, important security properties like system trust are preserved. We also describe an external validation web service, that provides a way for users to validate signed documents. The validation service then shows to the user important document security properties like timestamps, certificates attributes and highlights the document integrity in face of the digital signatures that have been collected in the workflows defined by our module in *Alfresco*.

Keywords: Digital Signatures, Workflow Management, Digital Citizen Card, Business Process Management, Alfresco

1 Introduction

Documents which are in printed format have been used for many years, such as books, papers, forms, contracts and any related materials [2]. Nowadays, there are a lot of reasons why people might choose to paperless environments, including reduction of the environmental harm of paper consumption and the economic cost of paper production, print, transfer and storage. Digital environments release people or companies of the location and physical constraints of paper and provide better support for updating, archiving, and searching of documents [3].

Figure 4.4: First page of the redacted document

Figure 4.5: Last page of the redacted document

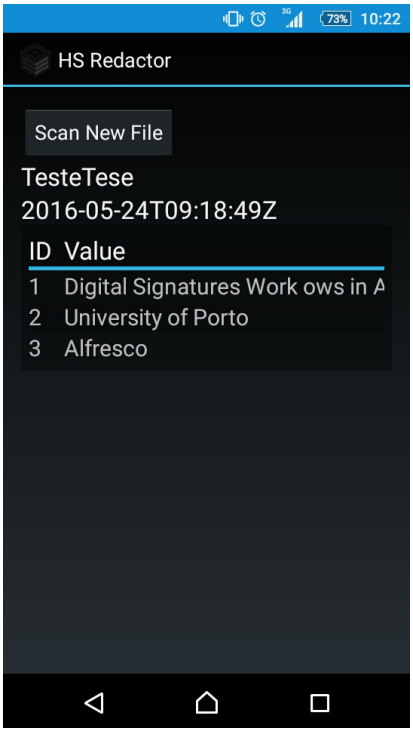


Figure 4.6: QR-Redactor *Android* Application (Hidden data menu)

The QR-safe application to the *Android* mobile phones ensures the integrity and authenticity of the text selected by the user in the process of redacting. The process combines the use of QR-Codes with a set of cryptographic primitives. All the selected data is compressed and signed by a registered user. The data is saved in QR-codes. The validation of the data can made both in offline and online environments. In resume, there are two types of QRs: the QR-Stamp that provides a way to identify the user and QR-Message that stores the data. With this tool, we can protect documents and prevent tampering, keeping the real data.

As we describe in the Related Work (section 2), there are some applications that makes redaction. There are individual applications and add-ons for *Alfresco*. However, we test the applications and we do not find an application that select people allowed to see the hidden/deleted data.

This application has the advantage of select some people that has access to the hidden data, this way, we can put the document with protected data for all users and give access only for a restricted selected people that can see the hidden data.

4.3 Alfresco

We made a middleware to integrate digital signatures with the open-source *Alfresco* ECM. As *Alfresco* has the feature of workflows, we use this feature to simulate a real signature process like in the real life. As we already described above, we can have the need of pass documents through some departments to be signed. Normally, the paper can be falsified or lost. With workflows, we can invite users to sign a document. The document is shared with all the users invited to sign the document and an alert appears in the dashboard of each user. The shared files appear in the tab *Shared Files* and also in *My Files*. The *Shared Files* is for the users that want to share a file with all users in *Alfresco*, while *My Files* is used by our workflow. When we want that a user sign a document, we share with him and the document is placed in *My Files* of the user. The alert that appears in the dashboard is placed in the left of the page, in the area *My Tasks* and in this local we can see all the active tasks that have not been treated 4.7.

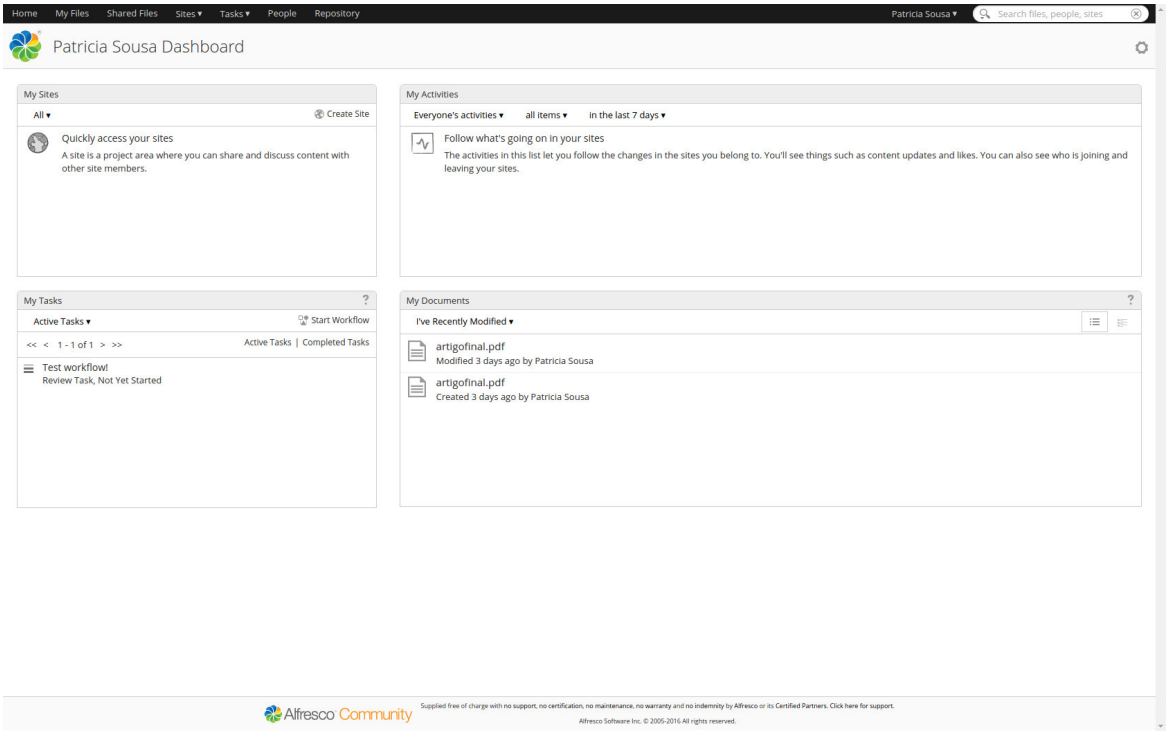


Figure 4.7: Dashboard *Alfresco*

This local is the best way for the users have a practical use of the system, because otherwise, users have to go to the to a tab Tasks, search for the active tasks and then, the next menu still has a page with information of the workflow before the signature

page. This is painful for the user because it has many menus and users want simple menus to use. For this, we used the notification on the dashboard. Also, is sent an email to all users with the task link and the download link of the document requested. The link of the task goes directly to the page to the signatures. As the signatures need a smart card, as already described in the sub-section 4.1, the user need to have their smart card "associated" to their profile. With this, we took advantage of the *Alfresco* user profile, and we add a new field to the profile to associate the smart card. In this field, we can insert the public data of the smart card and associate some information of the smart card. The information saved can be the name, number, etc. To facilitate the implementation of some smart cards, such as Citizen Cards, U.Porto Cards and Order Medical Cards, we associate and save the serial certificate of the smart cards that is extracted by the certificate X.509. With this certificate, the implementation is more easier to adapt new smart cards, since they all have the serial certificate. Another improvement in this system, is the option of verify a document. We can see if a document is valid or not, according to the integrity of the signature made.

Another important feature is the QR-redacting in *Alfresco* documents. This is a feature actually under development. We want to make a simple function that allows users to make redact putting a black box in place of text. We want to provide a way to users hide sensitive information that can be confidential for users. Users, for example, may wish to share the document, for example, without printing the document in a paper and underline with a black marker, as was traditionally done with a pen and paper. It is more work, since it was necessary to print a paper, spend this role and re-scan it, so the paper would be only a way of making redact, being completely dispensable if there was this technology that we are developing.

With this tool, users only has to have the *Alfresco* and a smart phone. As already explained in section 4.2, the application that we integrate with Alfresco, has the function of making and redact be shared only with the people desired. With a smart phone and a QR-code reader, a person can try to read with a smart phone with a QR-code reader, the QR-code generated in a document after it is done and completed the redaction and, if the person is authorized, can see the "hidden" content. Otherwise, the access to the hidden content will be denied. For this management of authorized users, we will use the mechanism of workflows in order to share content with people added to a specific workflow document. Thus, the person would redact and then would select people who wanted to make the workflow. As we are thinking about joining this also to workflows, we can join the redacting with the digital signatures. A person may want some people to sign a document but have some sensitive information that

does not want to share, in this way, the owner of workflow can make a redact the document and select people to make signatures (as in our workflow signatures) but with the addition of feature of redacting. In this way, will also be necessary to add another page with the function of add people to the workflow, but this, not only to sign but also allowed to see the hide content with the redacting.

4.3.1 *Alfresco* Modules

Alfresco modules (or add-ons) is a collection of code, configuration, scripts and media resources. The modules are typically packaged as an AMP. AMP works by modifying the *Alfresco* WAR deployment file. *A WAR file is a JAR file used to distribute a collection of files (JavaServer Pages, servlets, Java classes, XML files, tag libraries, and static web pages) that together constitute a web application.* [3]. The Module Management Tool (MMT) helps install and manage modules packaged as AMP files.

The modules are good to extend the features of Alfresco, since they can implement custom templates, custom models, web scripts or UI customisations. The *Alfresco* application typically consists of two WAR or AMP files, at least: *alfresco.war* and *share.war*. AMP files are used in more complex modules [4].

4.3.2 Web Scripts

In this case, we use Web Scripts to create a custom page to the single signatures, a page separated of the workflow, in the *Share* side. In addition to creating pages, Web Scripts play an important role in this work. In the *Alfresco Share* side, we need to have access to property *Alfresco* objects, which have only access through the repository side (server) and, this way, Web Scripts are widely used for us to get information in the client side and make this information accessible without adding too much complexity. The Web Scripts provide a page, which as already mentioned, can be JSON. In our case, we use the Web Scripts to provide Citizen Card data and the username of the user currently logged in on *Alfresco*, for example. These are two examples in this case are added to the *Person* object. We also use the Web Scripts

[3]Installing the *Alfresco* WARs:
`https://docs.alfresco.com/5.0/tasks/alf-war-install.html` [Online; Accessed 2016-3-28]

[4]Modules:
`https://docs.alfresco.com/4.2/concepts/dev-extensions-modules-intro.html` [Online; Accessed 2016-3-28]

to obtain the ticket authentication to the web service. A good advantage of web scripts is that they can be accessed externally, in this case, the external validation service that obtains the authentication ticket to fetch document information to be validate in *Alfresco*. The Web Scripts are common accessed by the URL like: `https://alfresco.hltsys.pt/share/proxy/alfresco/auth/get-username`. This way, we can obtain a JSON object with the information, through a simple *GET* to the URL.

4.3.3 Document Library Action

The document library provides a UI that allows users run repository actions under the documents present in *Alfresco*. The actions have the appearance represented in the figure 4.8, in this case, an example in a document of *Alfresco*:

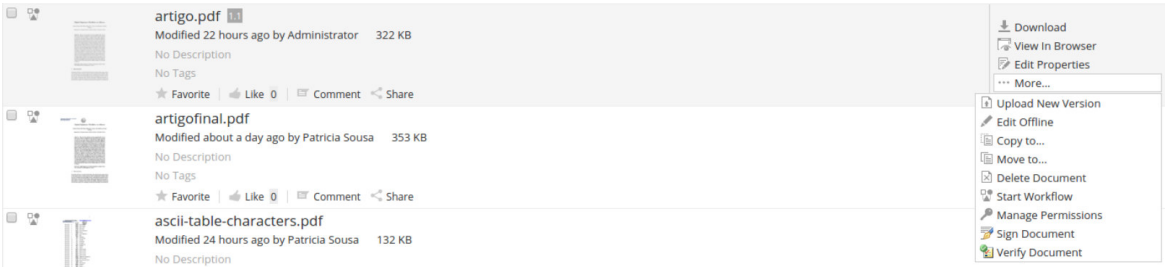


Figure 4.8: Actions of a folder

The action appear on the right side of the figure 4.8, in the options of the document. The actions appear on the right-hand side of the document list page.

Actions can, for example, launch a web page already created, a web page created with *Surf* or return some information provided by web scripts or for a simple JS code. The actions are controlled with XML file and can call a JS code to define a functionality or an action-link (a link of a web page).

We create two actions in this project: sign document and verify document. The sign document action opens a web page created with an *Aikau* library and uses the Web Scripts framework. Thus, with the *Aikau*, it is easy to have a page with the same customisation of Alfresco, providing a way to put the header and footer. We made this web page to a single signature with the own user and, this way, an user does not need to create a workflow if an user just want to sign the document himself. Thus, the page shows the respective document that has the action, and that the user wants to sign, and can sign with a simply and quickly way, without the need to create a

workflow. The page only shows the document and a button to sign. When user press the button, is displayed the PIN pad to the user insert the secret digital signatures PIN of their Citizen Card. The verification of a document is made through an action that opens a box with the information about the validity of the document. We can see an example of the actions in the figure 4.8.

4.3.4 Workflow Management

For creation of a business process more efficient, adaptive and effective to accomplish business tasks, BPM provides methods and techniques for this [45].

As *Alfresco* allows add-ons, we took advantage of this feature and we integrate all the process to signing a document as one module/add-on that can be integrated in the *Alfresco*.

To create an workflow, we start by create new options to the signature workflows using XML. Inside each XML for a workflow option, in the start execution listener, we call a Java file to create the fields to the signatures in the document attached to the workflow. To create the fields, we use the library *itextpdf*. We read the content of the file, we calculate the page size and then, we calculate the position of each field in relation to existing ones. For each field, we create a rectangle with some characteristics including the field name, to compare with the username in *Alfresco* of the signer. This preserve a field for each user that is added to the task, with their user name. This field is where will show the signature of the respective user when signing. *Alfresco* has the option to cancel the workflow, so, in the XML for a workflow option, we also have, in the end execution listener, a call for a Java file to delete the fields that are in the document attached to the workflow.

In the figure 4.9, we can see an example of our workflow process:

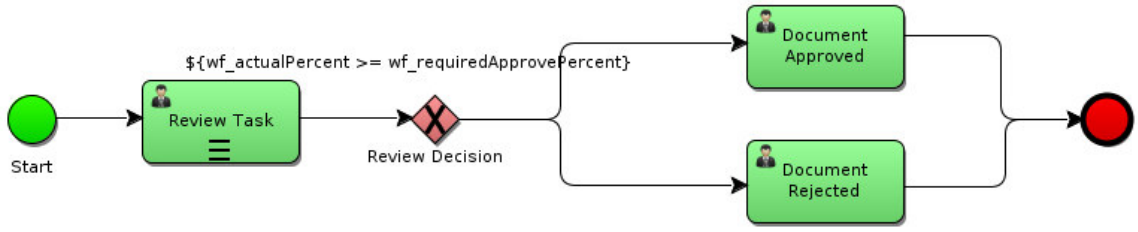


Figure 4.9: Business Process Model and Notation

In this diagram, we can see that we have a circle that represent the state that indicates the start of your business process. Then, we have an user task that should be used when human interaction is required for the business process, for example, when details are to be filled or verified by a human. The review decision is represented by an exclusive gateway, that is used when we want to proceed with one path from the multiple paths defined. So, we can compare the exclusive gateway to an if-else statement of the programming concept. In this review decision, we can define a condition, that if it is true, the document is approved through the user task, otherwise the document is rejected trough other user task. In the two cases, we advance for the final state that represents the end of the business process.

We have three types of workflows as described in this section. One of the workflows, users can sign in parallel, in this way, can sign in any order. The other workflow, users have to sign with a specific order, for example, first signs employed X, and only when X signs, employed Y can sign and in the final the director of the company accepts the task. The last workflow, a group of users can sign in parallel. *Alfresco* has a feature that allows the creation of user groups, so, we can associate a group to the workflow, without the need to associate one person at a time.

In all workflow processes its possible to reassign the people invited to the workflow.

This diagram represents the BPMN that we create for this work. In the initial state, we have a form that we can choose the title of the workflow (*bpm_workflowDescription*), a due date (*bpm_workflowDueDate*), a priority (*bpm_workflowPriority*), the reviewers (*bpm_assignees*) that we want to sign a specific document (or more than one document *packageItems*) and the required approval percentage (*wf_requiredApprovePercent*). This way, the percentage of people that have to sign the document for workflow can be approved by the owner of that workflow. We have the possibility of send an email to the reviewers that are attached to the workflow with the link of the task to review and with the link(s) of the document(s) attached to the workflow (*bpm_sendEMailNotifications*). When the workflow is started, the document is placed in *My files* of the assignees of the workflow. Initially we obtained the error *duplication of child nodes*, and this error was due to the fact that the user who is creating the workflow already have the document used to create the workflow in *My Files*. With that, we check if the current user that is creating the workflow is the owner of the document and, in this case, the file cannot be created in the users own area because the user already has this file in the *My files* area. When the workflow starts, also is created in the document(s) attached to the workflow, one signature field for each reviewer attached to the workflow. Each field has the corresponding user name of the reviewer who will sign this field. After the

initial state, the review task consists in send a task, to each reviewer attached to the workflow, for the reviewer signs and therefore accepts the task. If the user rejects the task, then it does not agree with the document, therefore, does not signs. To review the task, a form is displayed to the reviewers, with the info of the task: title/description (*message*), owner (*taskOwner*), priority (*bpm_priority*), due date (*bpm_duedate*) and identifier (*bpm_taskId*); progress with the status of the task (*bpm_status*): not yet started, in progress, on hold, cancelled or completed; the items attached to the task (*packageItems*) and a comment (*bpm_comment*) that if it is written, is put in the digital signature reason. The result of the review task is identified by *wf_reviewOutcome*. The signature is made through the Citizen Card. When the user hit the button *Accept and Sign* is shown a PIN pad [5] to insert the signature PIN. When the signature is placed in the document, in addition to the reason of the signature, is placed in the same field: the name of the person who signed the document and the date and time.

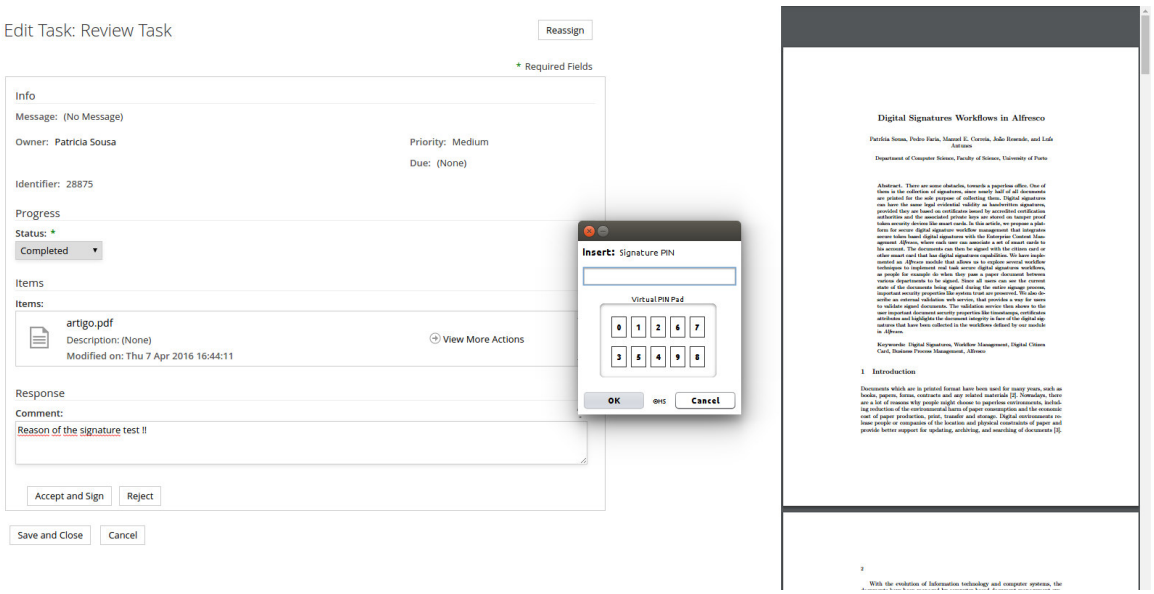


Figure 4.10: Example of a signature page

We can see in this figure 4.10 the components of a review task that we describe. The reason of the signature is placed in the signature, below of the name and the data. In the standard workflows of the *Alfresco*, we see a limitation to the signature process, and we develop a feature. The standard workflows does not have the preview of the document placed in a review task. The person that is signing a paper document

[5] Image of a PIN pad:
`https://www.datacard.com/images/products/id-product-detail/encoder-ingenico.png`
[Online; Accessed 2016-3-31]

with a pen, can constantly see it. The document is placed in front of the person and the person signs "above" the document. With this, the signer is always seeing the document that is signing until the last moment of the process and can review everything that is written, in the same time that is making the signature. In this case, to make the user more emerged in our service, we have to simulate this scenario so that the person does not feel uncomfortable and even unsafe to sign something that is not seeing at the moment because in the case of the standard workflows of *Alfresco*, the person would have to click on the file name to open a viewer on a new page of *Alfresco*. Besides being annoying, this may be quite uncomfortable and unreliable for the signer. This way, we ensure greater comfort for the person who is signing, with the preview of the document side-by-side with the form of the review task. In addition to the review-task page, we also added a preview of the document in other pages such as task details, workflow details and workflow edit. The task details page appears after the document is signed in the review-task, so we think that was also important to include the preview so that the person can view the document signed soon after the process. The pages workflow details and workflow edit is also important to include the previewer as well, whenever users see details about the workflow or if users want to change it, users always see the document side-by-side.

To implement this, we use the *NodeRef* of the document. A *NodeRef* object is a unique reference to a Node, the object that represents locations and resources within an Alfresco repository. As *NodeRef* is used in the pages that previews the document, we have to make something similarly.

In all pages that we described before, we add to the URL of the page a query string with the *NodeRef* of the document that we want to preview such as `...&nodeRef=workspace://SpacesStore/9f2d50eb-c71c-4b10-b9c9-c03cc75c32d3`. We have access to the *NodeRef* of the document that is associated with the workflow by the workflow information.

With this information, we add to the page a *div class* side-by-side with the form and inside this *div*, we put an object with the preview of the document.

The PIN pad is also an interesting feature, in this case, of the signature application (section 4.1), since, as already mentioned in section 3.2, an attacker can use a *keylogger* to be able to record what is being pressed in a keyboard. With a PIN pad, at least, we ensure that it can not be used, as we enter the numbers with a mouse. With the mouse, a more complex intrusion is required to gain access to the PIN.

As an alternative to PIN pad, if the person has a PIN pad on the card reader is even

better, no need to enter anything on the computer. However, we have this alternative to have more security, if the person does not have a card reader with a PIN pad.

Final, an example of a signature that is placed in a document, is showed in the next figure:

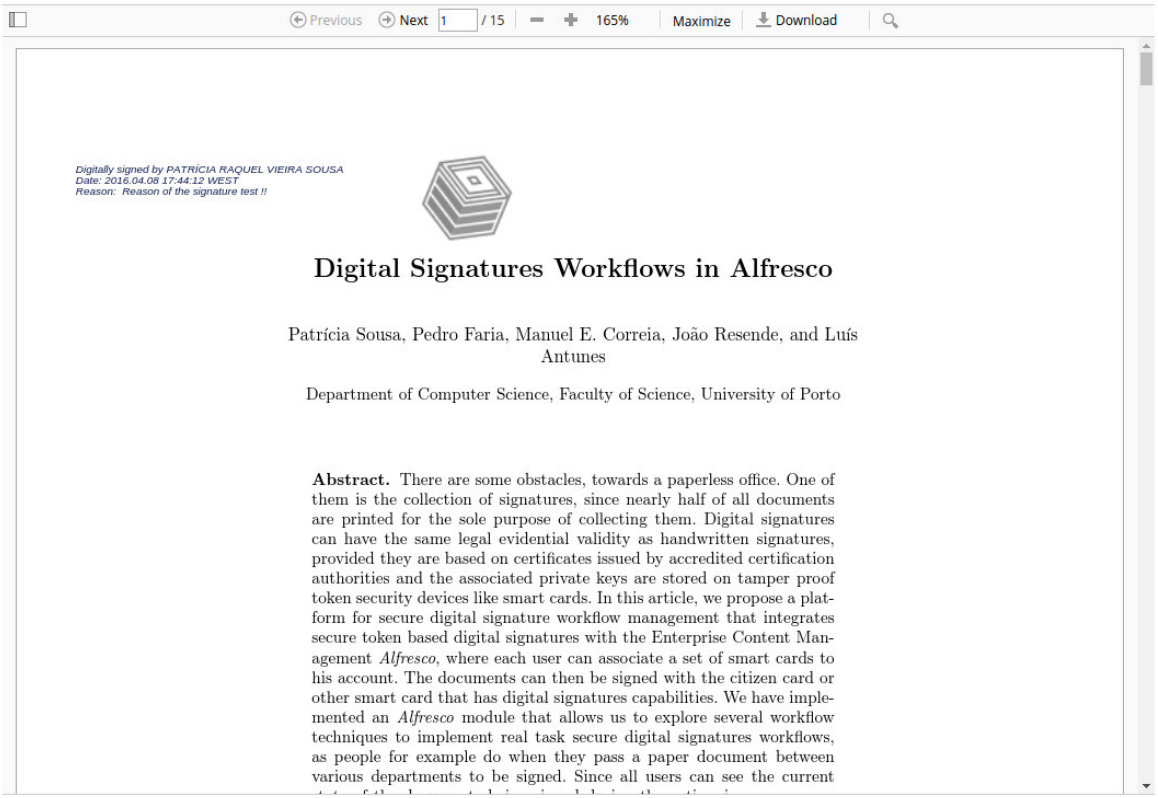


Figure 4.11: Example of a signature in a document

In this figure 4.11, we can see that, the reason of the signature matches with the reason writed in 4.10, as well as the name of the person. The date is the date/time of the moment that signature is made.

After this process, the condition *$wf_actualPercent* $\geq$ *wf_requiredApprovePercent* is tested for the review decision. The *wf_actualPercent* is the percentage of reviewers that sign and accept the document and the *wf_requiredApprovePercent* is the required approval percent, filled on the form, previously described in this section. If the condition is true, then the document(s) can be approved by the workflow owner.

Algorithm 1 Count percentage of reviewers that approve the document

1: **if** $task.getVariableLocal('wf\_reviewOutcome') ==' Approve'$ **then**
2: $newApprovedCount := wf\_approveCount + 1;$
3: $newApprovedPercentage := (newApprovedCount/wf\_reviewerCount) * 100;$
4: $execution.setVariable('wf\_approveCount', newApprovedCount);$
5: $execution.setVariable('wf\_actualPercent', newApprovedPercentage);$
6: **end if**

The Algorithm 1 is called whenever a user approves a task. After this, the owner ends the workflow through a form, even if it is approved or rejected and can do a comment to the workflow. The form has info of the workflow: title/description (*message*), owner (*taskOwner*), priority (*bpm_priority*), due date (*bpm_duedate*) and identifier (*bpm_taskId*); progress with the status of the task (*bpm_status*): with the same choose status then the task review form; the information of outcome: number of reviewers, reviewers who approved, required approval percentage and actual approval percentage; the items attached to the task (*packageItems*) and a comment (*bpm_comment*) that owner can put in the workflow.

As the signatures are made with the Citizen Card, each user has to associate the card to their user profile. The system makes a check if that card already belongs to someone else profile, for security reasons. If the user has no smart card in the profile, when the user tries to sign a document through a workflow, it is required to associate the Citizen Card to their profile. To facilitate the use of the service, we have another way of association of the card to the profile. The users can associate the card without leaving the current workflow task through a button that makes the direct association of the smart card to the user profile.

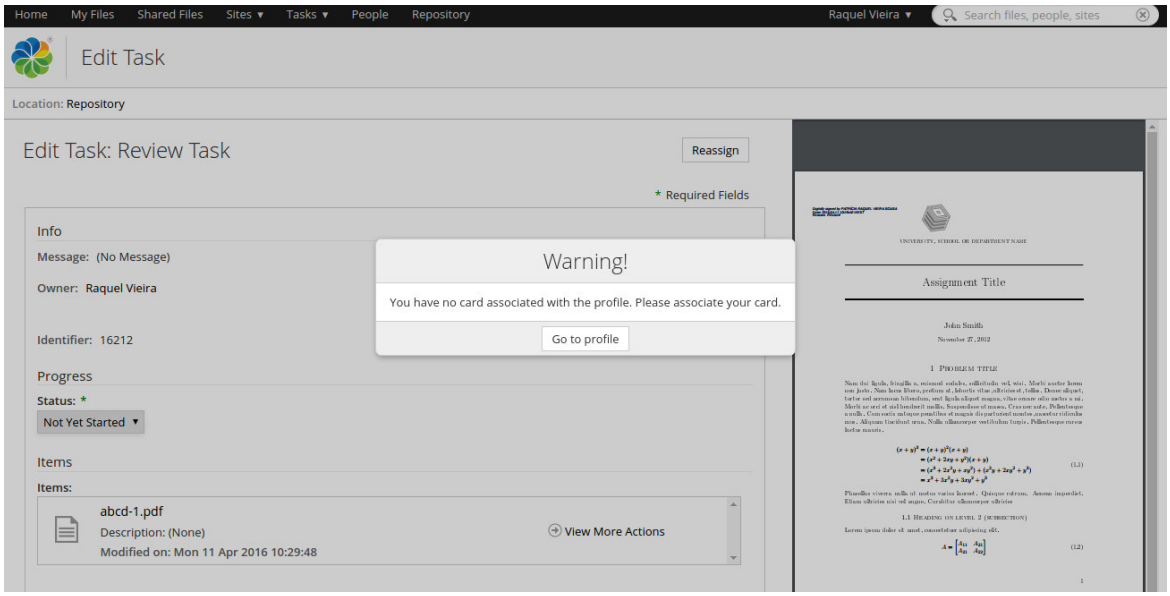


Figure 4.12: No card associated to profile notification

We can see in this figure 4.12 that we sign with *Patrícia Sousa* profile the document, and the signature is in the document (we can see in the figure 4.12 in the preview of the document in the right side). In the *Raquel Vieira* profile, we do not have any card associated to the profile, so, when we try to sign the document that is "shared" with this profile, through the invitation to the workflow signature of this document, we cannot sign because we do not have a card associated to this profile. Is given the notification *You have no card associated with the profile. Please associate your card.*. We have a button *Go to profile* to help the user and go directly to the profile, which is where users can associate their card to their own profile.

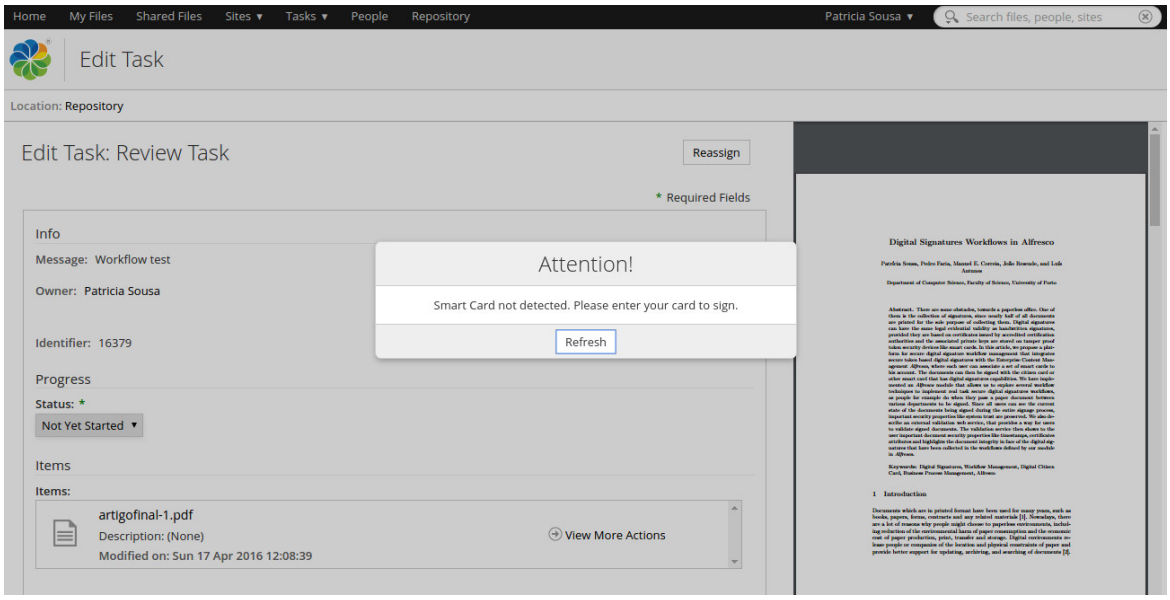


Figure 4.13: No card inserted

In the case that there is no card inserted in the card reader, a notification is given to the person inserting the card and while the person do not insert a card in the card reader, if the person press *Refresh* button 4.13 is given the same notification *Smart Card not detected. Please enter your card to sign.*. When the card is inserted, the notification disappears.

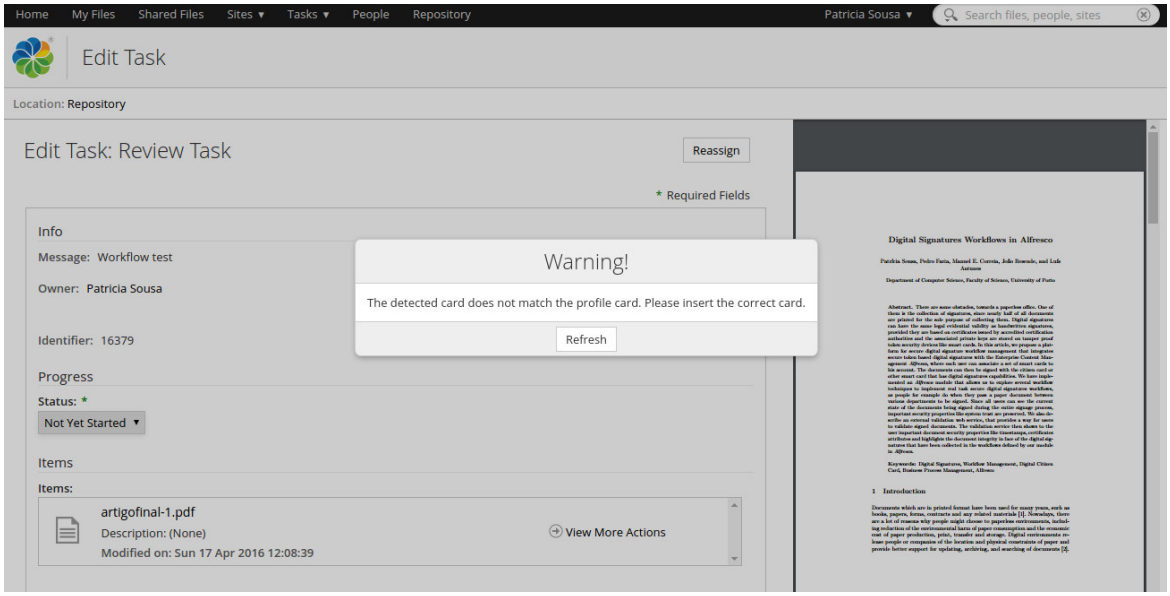


Figure 4.14: Incorrect card inserted

Finally, we give a warning if the card inserted in the card reader, is not the same as the associated to the profile of the current user logged in *Alfresco*. We can see this situation in the figure 4.14. Is given the message *The detected card does not match the profile card. Please insert the correct card.*.

In addition to the Citizen Card, we decided to also give the possibility of users associate other smart cards to their profile, instead of only Citizen Card. If the users work in a hospital, they can associate their hospital card profile. So they can, for example, sign hospital internal documents with the hospital card and human resources documents with the Citizen Card. It gives the possibility of the person to choose which card wants to use to sign the documents.

One of the other biggest capacity of our system is the provision of information about the signature fields for each document. Through an action button, which is one of *Alfresco* capabilities, that calls an external *web service* we offer the user the possibility to validate the document and which fields that are already signed and if the signatures are valid.

4.3.5 Configure e-mail

The default e-mail configuration of the *Alfresco* has this aspect:

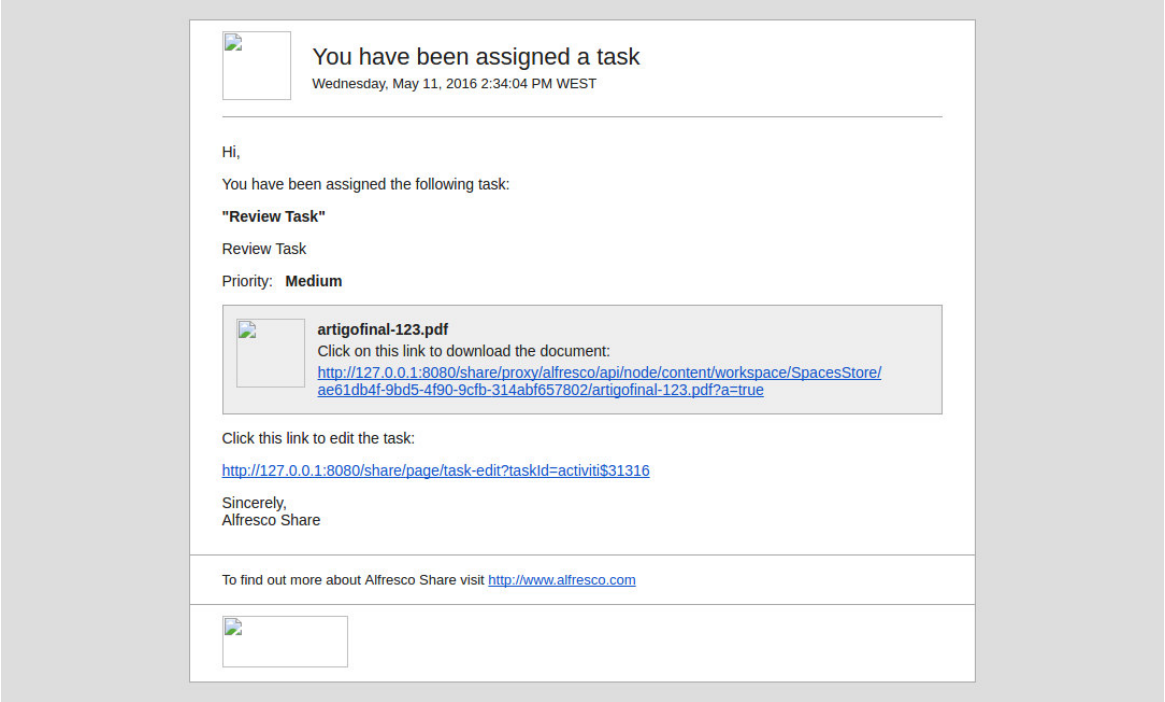


Figure 4.15: Example of email notification default

We want to adapt this to our signature workflows. For this, we have to modify the URLs sent in the content of the email notification, such as we made in the pages of the workflows and tasks, as we described in a previous subsection 4.3.4. To implement the previewer of the document side-by-side in the review task form we have to access to the *NodeRef* of the document. As we described in the previous subsection 4.3.4 too, we have to create a *NodeRef* query string to get access to the *NodeRef* of the document attached to the workflow.

In the case of the email notification, we change the link of the task-review page with the query string too because in the default email notification we does not have this.

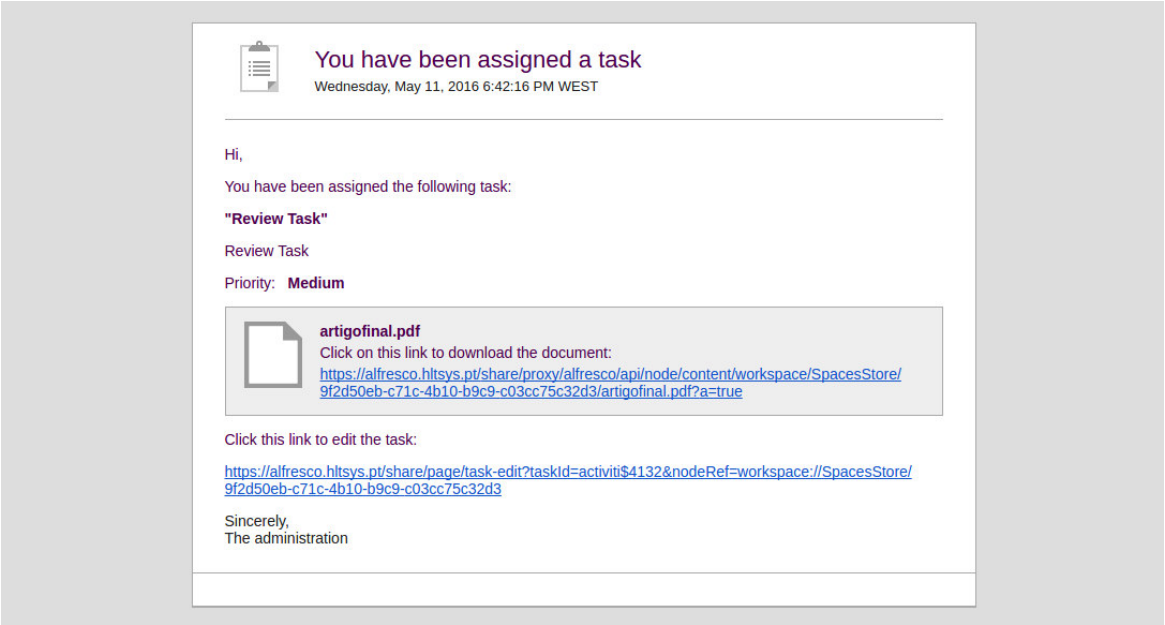


Figure 4.16: Example of email notification adapted

The new email notification adapted has the aspect of the figure 4.16.

The easiest way to modify the email notification content is in the files of repository browser, placed in the *Data Dictionary* folder of the *Alfresco* repository. We change the link of the task-review in the files of the *Workflow Notification* folder part of the set of the *Email Templates* placed in *Data Dictionary* folder. However, if we change directly the *Alfresco* files in the repository browser, when we want to use the module that we create, in another *Alfresco*, the changes made in the email notification content are not part of the module, so we can not move this changes in the repository browser to another *Alfresco* only with module, at this time.

To solve this problem of load this content into the Alfresco Repository in the module, we use bootstrapping. *When installing a module into the Alfresco system, developers probably want to import some stuff like spaces, templates, scripts etc. For this purpose, developers can use the bootstrap mechanism.* To bootstrap content means to load it once into the repository.

4.3.6 Validation Service

We decide to make an external *web service* to validate the signatures. We create the *web service* with *Spring* Model View Controller (MVC) and a CMIS client. The CMIS

client provides a way to access to the session of the *Alfresco* and connect to them and access to the documents presents in *Alfresco* repository. In the *Spring web service* the connection to the session is made with a unique identifier, in the case of *Alfresco*, with a token and ticket sent by each session. The link of the Validation *web service* is something like: `https://alfresco.hltsys.pt/ValidationPDF/Validation?filePath=/examplepdf-1.pdf&ticket=TICKET_2e6fdfe2bd0355acbcc0a3ebb75f39213dd848b2` where */examplepdf-1.pdf* is the path of the file, that we want to validate, placed in the home directory of the *Alfresco* represented by "/" and the ticket (`ticket=TICKET_2e6fdfe2bd0355acbcc0a3ebb75f39213dd848b2`) is the ticket sent by each *Alfresco* session to authenticate with *Alfresco*.

The fundamental purpose of this service is: we have *Alfresco* and the validation service, the customer wants to ensure that the document signing is being properly assessed on their product. To make sure that the validation is done correctly, the validation service has to includes a signature in their answer that the customer can validate and have the security that *Alfresco* is not changing any validation response.

The information that *web service* returns is the number of *revisions*, the number of *fields*, the *status* (empty, partial or full) and the number of signed fields (*nsigs*). For each field the information is the name of the *field* and if it was *signed* or not. If it was signed we have the information if the signature cover all the document (*whole*), if it has been revised (*revision*), *date*, the certificate of the Citizen Card (*certserial*), the integrity of the signature (*integrity*) and the response that *web service* gives (valid or not) is stored on the *validation* variable, then in the client side, we test the conditions again and we compare this variable, so, we can see if the result by the server is correct.

```
{
   nsigs:1,
   revisions:1,
   fields:2,
   list:[
      {
         field:"field_patriciasousa93",
         signed:false
      },
      {
         field:"field_patricia93sousa",
         signed:true,
         signeddata:{
```

```
            date:"2016-06-27 16:41:57.928",
            integrity:true,
            certserial:5683882791040671000,
            whole:true,
            validation:true,
            revision:1
        }
    }
  ],
  status:"partial"
}
```

Code 4.1: Example of signatures validation document JSON

We can see one example of a JSON response by the validation service, with parameters that we need for a document signed by *patricia93sousa* user but unsigned yet by *patriciasousa93* user.

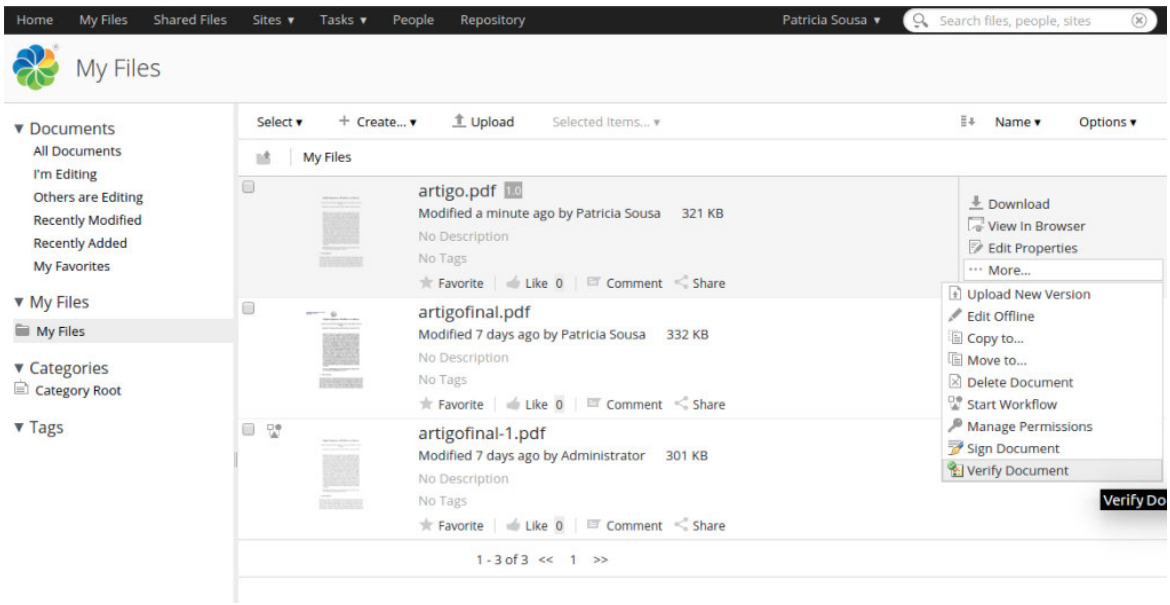


Figure 4.17: Validation action button

To see the validation of the document, we create an action button, as we can see in the figure 4.17.

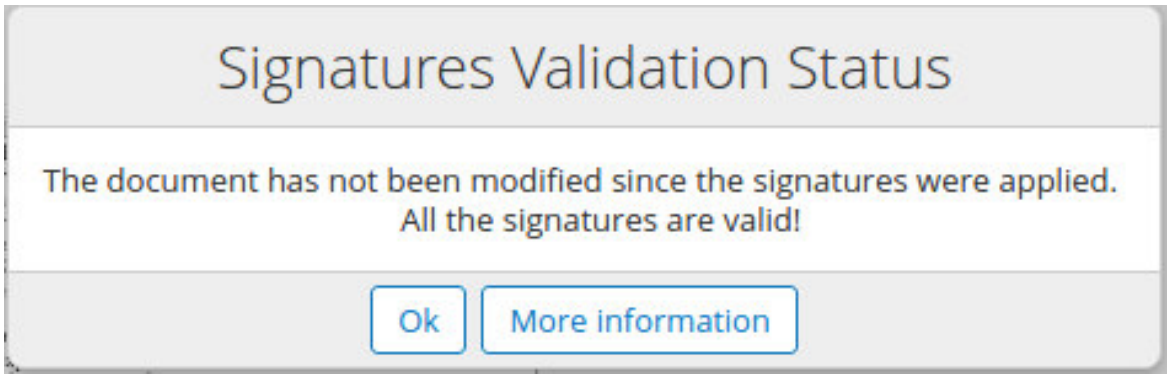


Figure 4.18: Pop-up example with signatures validation status

Figure 4.19: Pop-up example with more information of validation

We can see in the figure 4.18 and in the figure 4.19, the pop-ups that we use to show the information to the client. The first pop-up is showed when the user clicks on the *Verify Document* in a specific document, if the document has been modified since the signatures were applied and therefore, if the signatures are valid or not. With *More Information* button, we can see an example, on the second pop-up, of the information that is showed. We give the information of the number of fields that exists on the document, how many fields are signed, the name of the fields and if each field are signed. For the signed fields, we show which is the date/time and the integrity of the field. In this case, we show in the figure 4.18 *The document has not been modified since the signatures were applied. All the signatures are valid!*, in other words, all the signatures made are valid. In the figure 4.19, we can see the validation response that

corresponds to the JSON object represented in the Code 4.1, in this case, the field *field_patriciasousa93* is not signed, and the field *field_patricia93sousa* is signed in the *2016-06-27* data and *16:41:57.928* time. The integrity is true, so, despite the first field is not signed, the document is valid in it.

Figure 4.20: Pop-up example with number of fields created/signed

Figure 4.21: Pop-up example with no fields created

Another possibilities to the signatures validation status are presented in figures 4.20 and 4.21.

In the figure 4.20, we can see an example of the validation, if we create a workflow but people does not sign any field yet. In this case, 1 field was created and 0 was signed yet. For this pop-up, we do not put the button more information because it is not necessary in this case, since there is no signed fields and therefore there is no information about the signed fields to see.

Finally, in the figure 4.21, we can see an example of the validation if the document does not has any field created yet, in other words, there is no workflows running on this document. In this case, with the same situation as above, there is no button for more information because does not exist created fields and consequently, does not

exist fields to display information. Therefore, in these situations it appears the *OK* button to dismiss the notification.

4.4 Middleware - Communication between components

The middleware is used to group all software technologies that are among the final application and data providers to this end application. Our middleware makes the communication between all components through the API of each component.

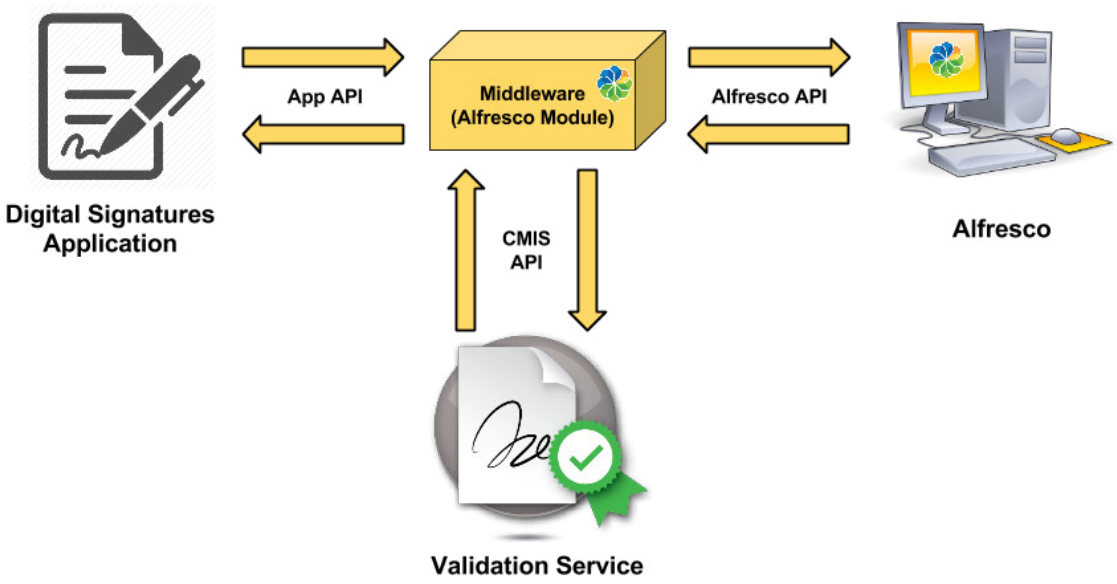


Figure 4.22: Communication between components

When an user, in *Alfresco*, creates a digital signature workflow, each task of the workflow has the functionality of the signatures. When an user wants to sign, *Alfresco* has to communicate with signature application. In the digital signature application we have an API with callbacks. This way, we can use our middleware (*Alfresco Module*) to communicate with the digital signature application, calling an specific callback to create the signatures (example in Code 4.2). The communication between these components is made with JS.

```
1         middleware.signpdf({
2             openjpeg: false,
3             autosignpdf: true,
4             signPDFinput: "",
5             signPDFurl: fileURL,
6             signPDFfield: "field_" + Alfresco.constants.USERNAME,
7             signPDFreason: comment,
8             signPDFlocation: "",
9             before: function() {
10                Alfresco.util.PopupManager.displayMessage({
11                    text: "loading... please wait"
12                })
13            },
14            success: function(data) {
15                response(data.substring(28));
16            },
17            error: function(message) {
18                Alfresco.util.PopupManager.displayMessage({
19                    text: "Sign pdf error: " + message
20                })
21            },
22            complete: function() {
23                Alfresco.util.PopupManager.displayMessage({
24                    text: "Sign PDF complete!"
25                })
26            }
27        });
```

Code 4.2: Example call digital signatures application

In this example, we call the *signpdf* function, that has the functionality of create a signature using some parameters. In this function we have some parameters among which are *signPDFinput* that creates an opportunity to the developer has an input that will fetch the document in the local PC, for example. Contrary to the *signPDFinput*, we can use an URL to fetch the document through the parameter *signPDFurl*, which in this case is what we use to get the document attached in a workflow of *Alfresco*. In the *signPDFfield* parameter, we put the name of the current user that is signing the PDF through the *Alfresco.constants.USERNAME*. The optional parameters are the reason and location of the signature. We chose to use the *signPDFreason* and allow users to write a comment in the task review that is placed in the reason of the signature.

When an user wants to validate a document, *Alfresco* has to call the external validation

web service. As we already described too, the communication between *Alfresco* and the validation service is made by CMIS (described too in the subsections 2.4.6 4.3.6). To implement the action (subsection 4.3.3) to the validation, we create a JS file in the middleware that makes a *GET* request of two URL (web scripts) to get the ticket of the authentication and the file path of the document to be verified. This two parameters are necessary to the link that we construct to call the verification service (`https://alfresco.hltsys.pt/ValidationPDF/Validaton?filePath=/examplepdf-1.pdf&ticket=TICKET_2e6fdfe2d0355acbcc0a3ebb75f39213dd848b2` where */examplepdf-1.pdf*). In the external web service, through the CMIS, we start by get a CMIS session with the connection name (defined by developer) and the ticket of each session *Alfresco* that is given by the URL that we create in the middleware. Then, we get the document to be validated, through the session that we get and the file path of the document to be validated. After this, in the middleware, we receive the response of the server in a JSON like the example of the Code 4.1 to display in *Alfresco*.

Chapter 5

Conclusion and Future Work

In this digital age, distresses us to see many documents, which are common in companies and public entities (such as minutes, orders, opinions, etc.) to be printed, signed on paper and then scanned to be distributed by email. This approach, that we can call "semi-digital", has several problems. First, is expensive, worn paper and ink cartridges. In terms of time (printed, signed for all people and then scanned) and in terms of space (scanned documents take up much more space), the "semi-digital" method is inefficient. Another inconvenient, the users cannot too search within scanned documents with an efficient way.

We are certain that, this is an advance in the paperless technology, and this contributes to a different society, more ecological. Furthermore, this will avoid not only the spend of the paper, as well as decrease the document forgeries that are being made to paper and damages of the paper, since all the document are saved in an ECM online and the versions can be controlled by the people, also during the signature process and can be verified. This preserve some security properties like system trust.

5.1 Research Summary

During this work, we studied how to create a system capable of made a digital signatures workflow in documents online. We already have a digital signature application and we search any ECM's existent. We had to create a middleware between the digital signature application and the selected ECM *Alfresco*.

Herewith, we described a fully working integration of digital signatures with an ECM

that has the capability of workflows, enhanced with a solution (still in development) of redacting. We also describe and we take inspiration from several other proposed digital signatures systems (with and without workflows) and some redaction systems. This search of systems already done, serves to help understand the weaknesses of other systems and making this work better. Also in relation to the interface can have more sense of where we need to improve what already exists in the market.

We believe that we have accomplished the main objectives that we define in the beginning of this thesis.

There is work still in progress to add to the current state of implementation. What we have done so far is a proof of concept, is a first step to develop more compatibilities in digital signatures and add features in the future.

5.2 Current implementation limitations

The redacting is being developed and it is not yet ready to be tested or to be used by users, so this is still a limitation of our system. Therefore, in the future, is the first thing we want to finish.

This system only allows digital signatures in a PDF file, therefore, as a future work, we plan to add electronic signatures and biometric signatures and give the user option to choose between different types of signatures. It is also desirable to implement the signatures for other types of document that not only PDF such as XML, for example. Since, at this time, the system only supports signatures with smart cards. In the future, we want to support other type of tokens that can even be safer, as yubikeys.

5.3 Future Work

During the development of this thesis, we plan to do the integration of the smart cards as an authentication token, for example, we want that users may insert a smart card and the *Alfresco* authentication recognise their name automatically. *Alfresco* made login with the profile of the respective user, recognisable by name and smart card.

Then, we can adapt this feature to yubikeys and other tokens.

In terms of the signatures, we want to store a certificate in the yubikeys and other

tokens, and be possible to sign with them.

We also want to increase the support of the limitations present in the previously section 5.2 turning the system with more compatibility with other types of files to be signed.

5.4 Conclusion

Nowadays, does not yet exists a well-accepted solution to the problem of the paperless and this can contribute for an active area of research.

When a paper document passes between multiple departments, the document can be falsified or tampered with. In the system proposed in this work, as already mentioned, important security properties are preserved like system trust, since all users can see the current state of the documents being signed during the entire signage process, it's possible to verify the document at any time and see the modifications in real time. The digital signatures perverse too the security properties integrity, authenticity and non-repudiation. This all together, is an advance in paperless technology, since the signatures in addition to being integrated into a workflow, the signatures workflow are integrated in an enterprise content management, in this case, *Alfresco*.

Appendix A

Development notes

A.1 Programming languages used

We use multiple programming languages to develop this middleware.

- Java
- Javascript
- XML
- FTL
- HTML
- JSP

A.2 Software used

- ***Alfresco***
  - Description: Open-source Enterprise Content Management that we use to integrate the signatures and redaction.
  - Version: Community 5.1.x
  - Website: `https://www.alfresco.com/`

- **Java ™ SE Development Kit**
  - Description: Free and open source implementation of the Java Platform, Standard Edition (Java SE).
  - Version: 8, Update 91 (Java Development Kit (JDK) 8u91)
  - Website: `http://openjdk.java.net/`
- **Signatures and redaction applications**
  - Description: The base applications that we use to integrate with *Alfresco*.
  - Version: 1.0
  - Website: `https://demos.hltsys.pt/`
- **CMIS**
  - Description: We use the *Alfresco* CMIS to make the connection between *Alfresco* and the validation web service.
  - Version: 1.0
  - Website: `https://www.alfresco.com/cmis`
- **Apache Tomcat**
  - Description: Open-source web server and servlet container capable to run Java Servlets, JSP and more.
  - Version: 8.0.33
  - Website: `http://tomcat.apache.org/`
- **IntelliJ IDEA**
  - Description: An open-source and free limited IDE that we use to develop and use the RAD to this development.
  - Version: Community Edition
  - Website: `https://www.jetbrains.com/idea/`

Appendix B

Manual

B.1 Login Process

1. Access to the website `https://alfresco.hltsys.pt/share`. The first page that appears is the page of login:

Figure B.1: Login page

When user makes the login, the page displayed to the users is the dashboard.

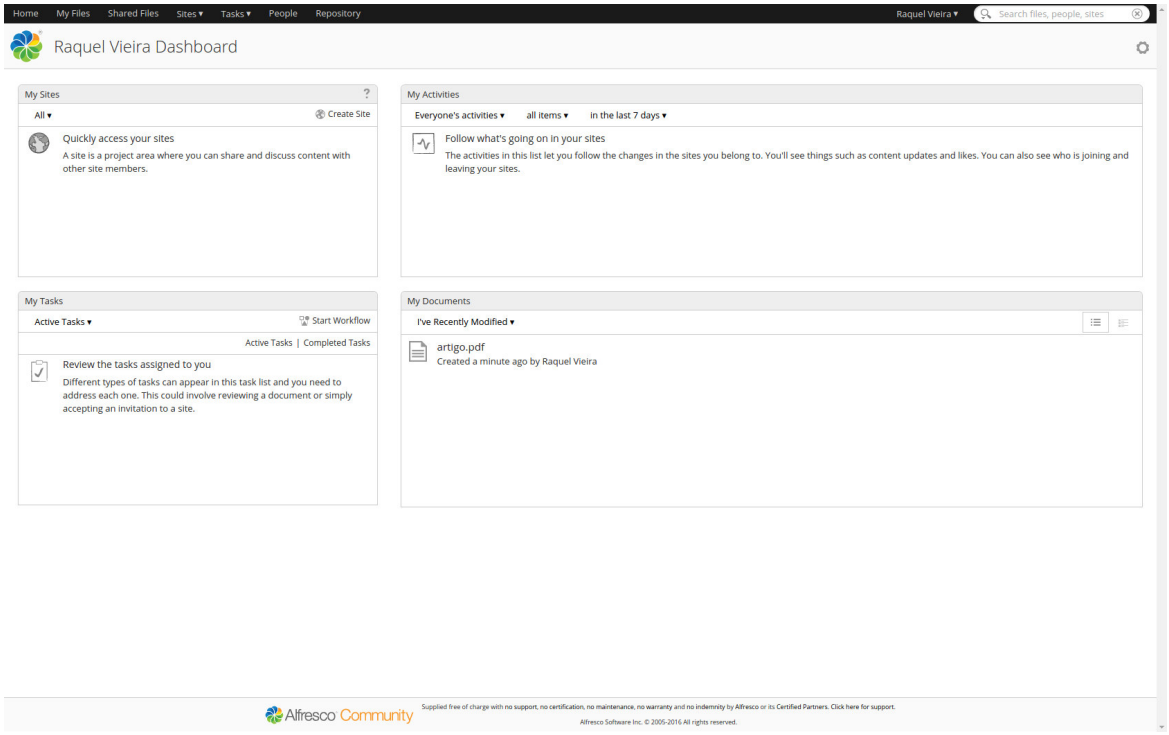


Figure B.2: User dashboard

2. In this dashboard users have some tabs. In the tab ***My Files*** or ***Shared Files***, a user can put files. In the case of ***My Files***, the files are only shared with people that the user wants. In the ***Shared Files*** the files are shared with all people registered in this *Alfresco*.

B.2 Workflow signatures process

When a user place a file, that user can select one of several options:

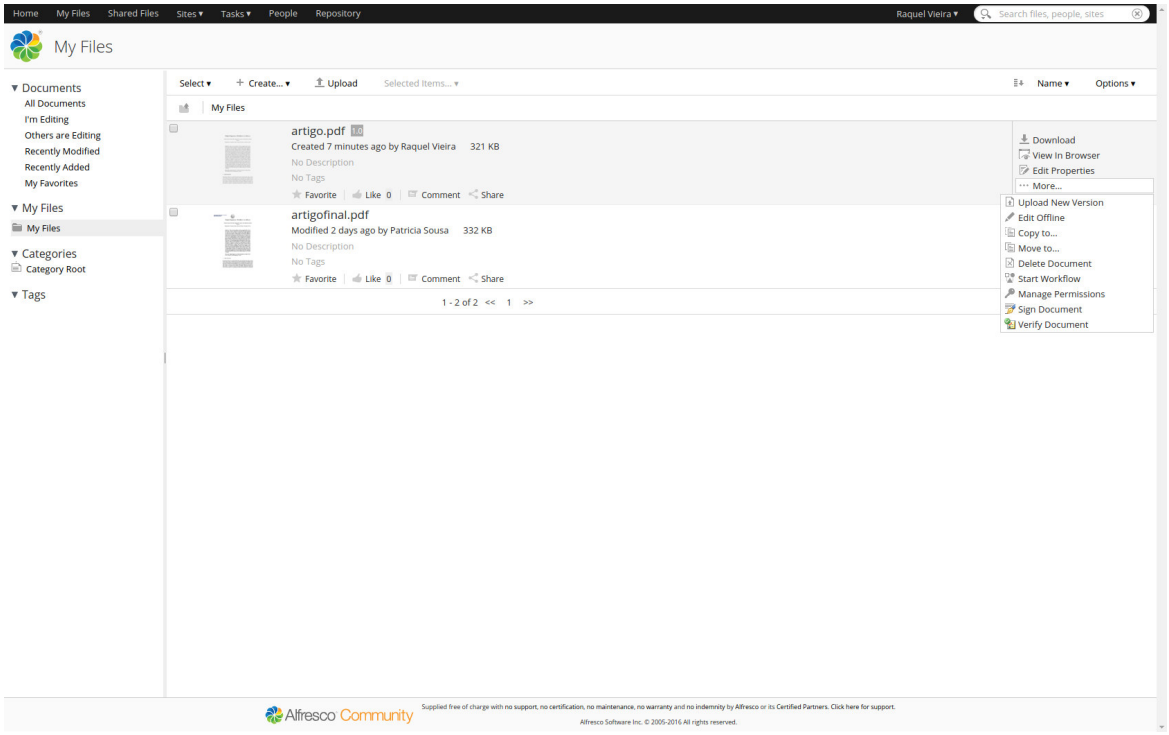


Figure B.3: Actions of the PDF files

3. If you click in the ***Start Workflow***, its opened a page where user can select the type of the workflow that you want.

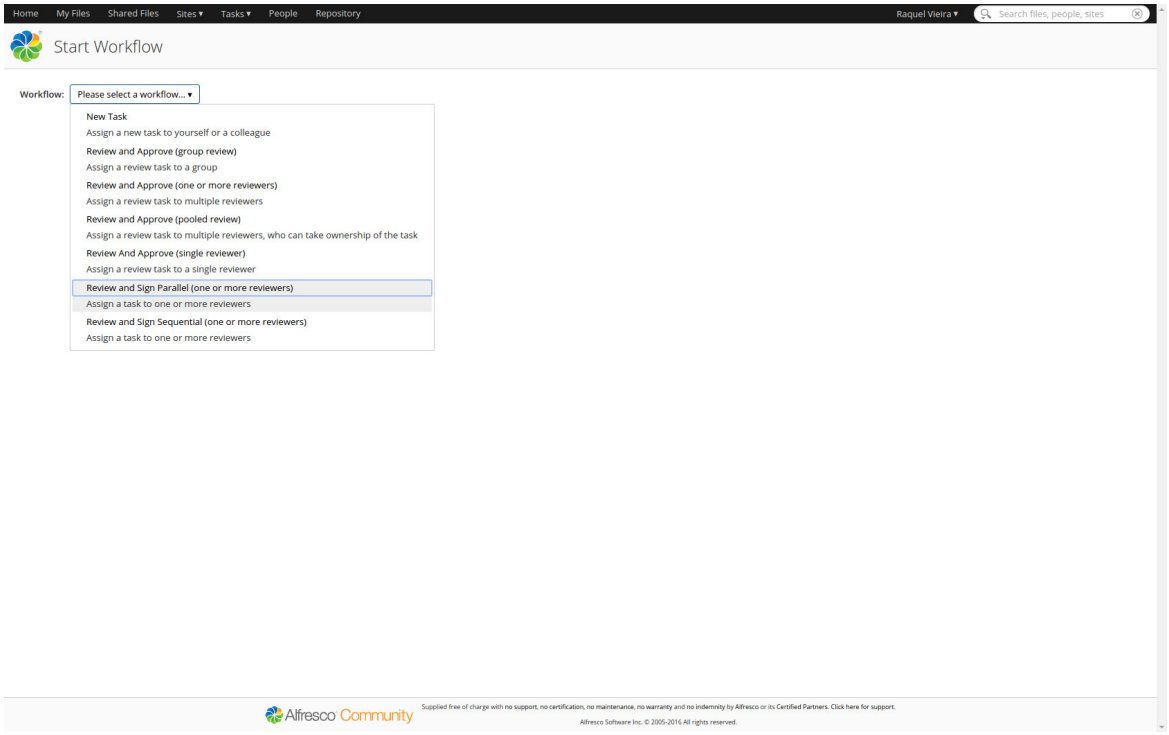


Figure B.4: Select workflow page

In this case, to demonstration, we select our workflow ***Review and Sign Parallel*** to one or more reviewers.

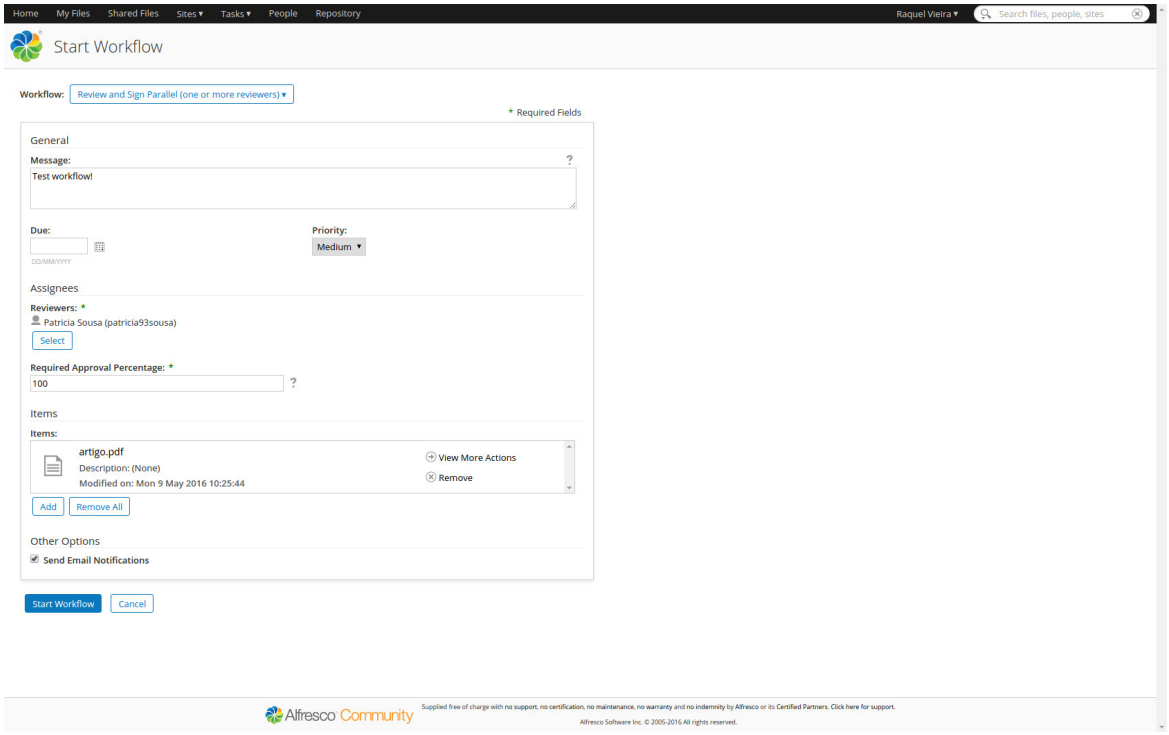


Figure B.5: Start workflow page

In this form, users can put in the field ***Message*** the title/description of the workflow that he wants to start. In the ***Due*** he can put the data/time of the workflow or of the limit and can put the ***Priority*** of the workflow. In terms of assignees, user has to select the ***Reviewers*** that the owner of the workflow wants to sign a document and he has to give a ***Require Approval Percentage*** of the assignees. The both fields are required fields. The ***file*** is selected automatically when users clicks on the action ***Start Workflow*** in an specific file. Another option is to ***Send Email Notifications*** to all assignees.

In this case, for demonstration in the figure B.5, we put a message *Test workflow!*, a reviewer Patricia Sousa (*patricia93sousa*) and a require approval percentage of 100%. The user can select more than one reviewer. We put also the option to send an email notification to the user *patricia93sousa* assigned to the workflow.

4. After that, user may makes logout and login with an assignee account. In this case, we make this to demonstration.

5. After the creation of the workflow, the assignees have three options to access the task review. The first one, probably the most quickly way and the way in which the user is quickly notified is the email notification the link sent to the

users is equal to this link of the dashboard, i.e., goes to this same page of the task review.

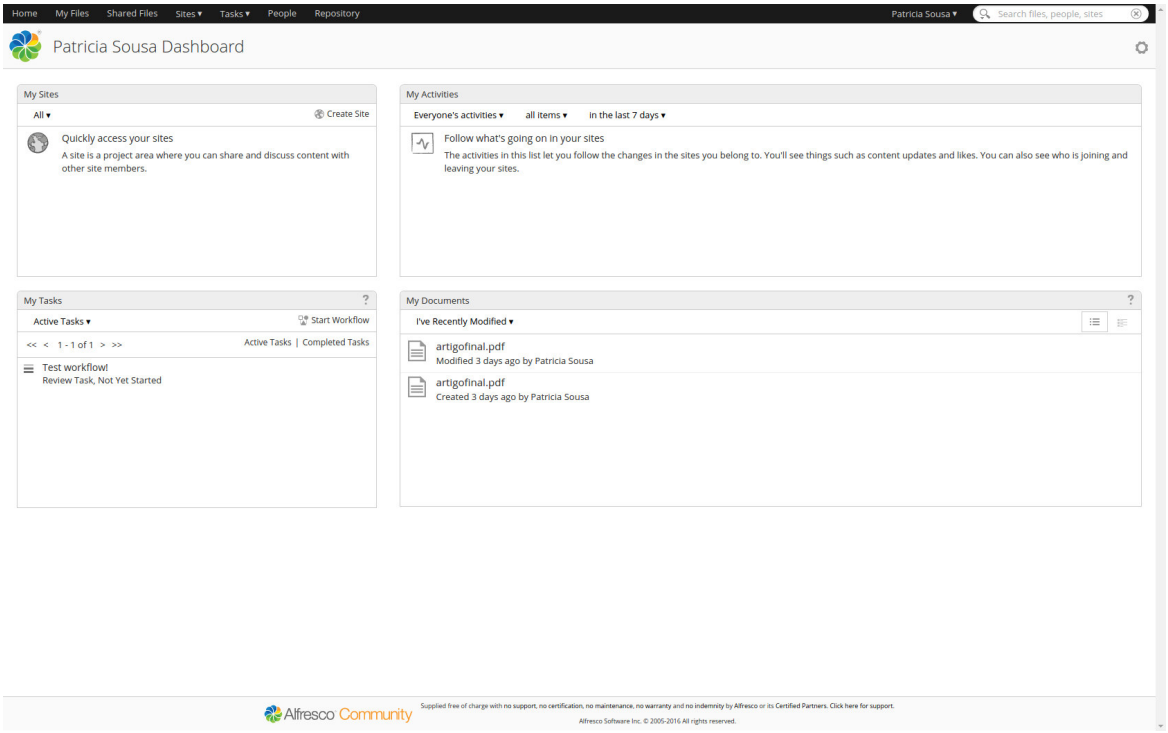


Figure B.6: Task notification in the dashboard

Another direct option is in the dashboard of the user invited to the workflow, is placed the task notifications. This way, users can go to the workflow task.

6. In the workflow task, appears one form that users has to fill. The comment is optional, it's placed in the reason of the signature.

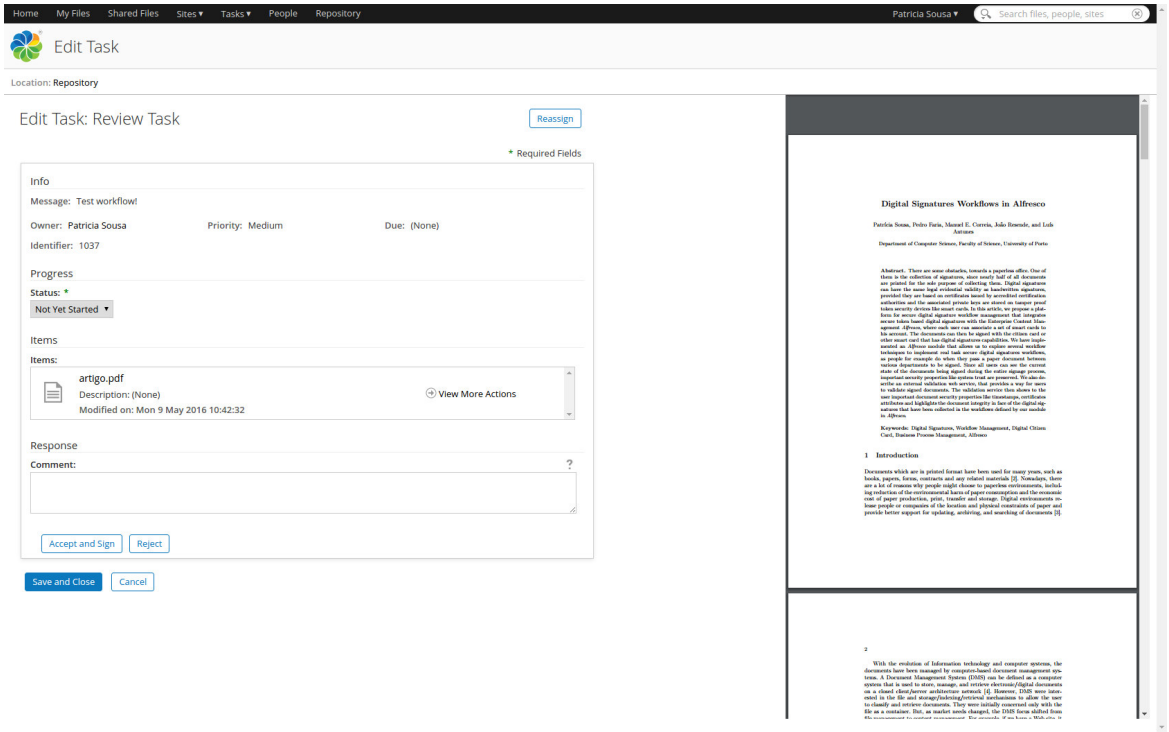


Figure B.7: Review task

After this, users have to click in Accept and Sign if they want to sign the document. After the click, appears a PINpad on the screen and users have put the signature PIN of their smart card.

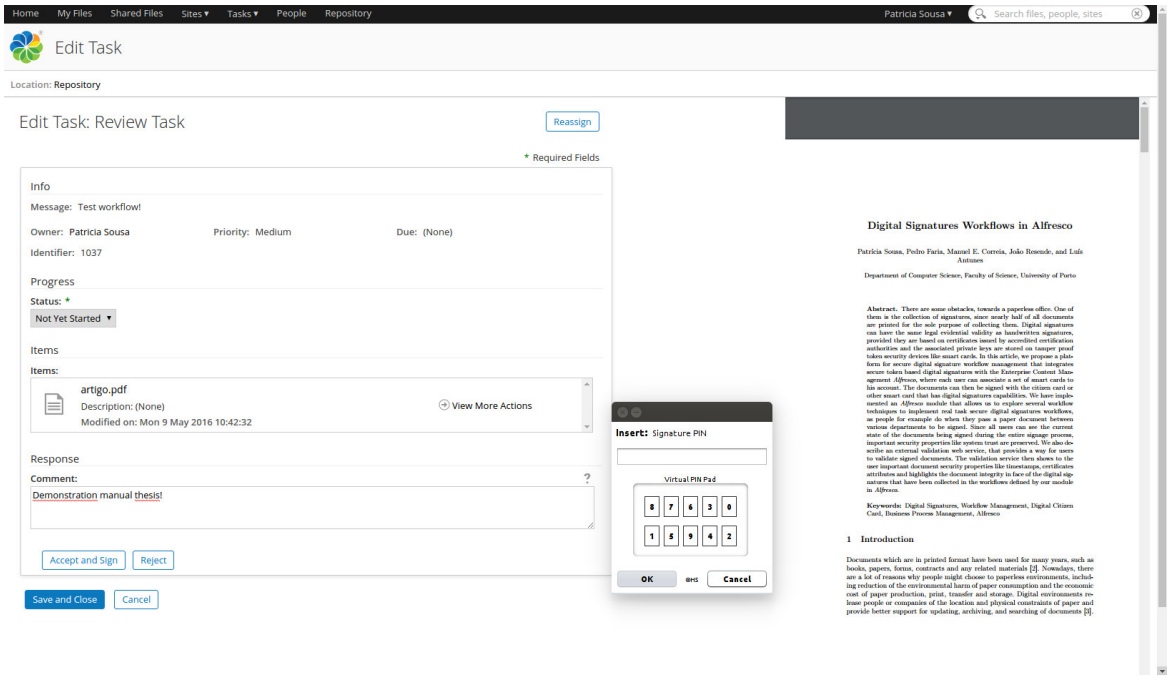


Figure B.8: Review signature task

After the signature, is shown to the user the document with the signature made in the task details page as well as information about the workflow.

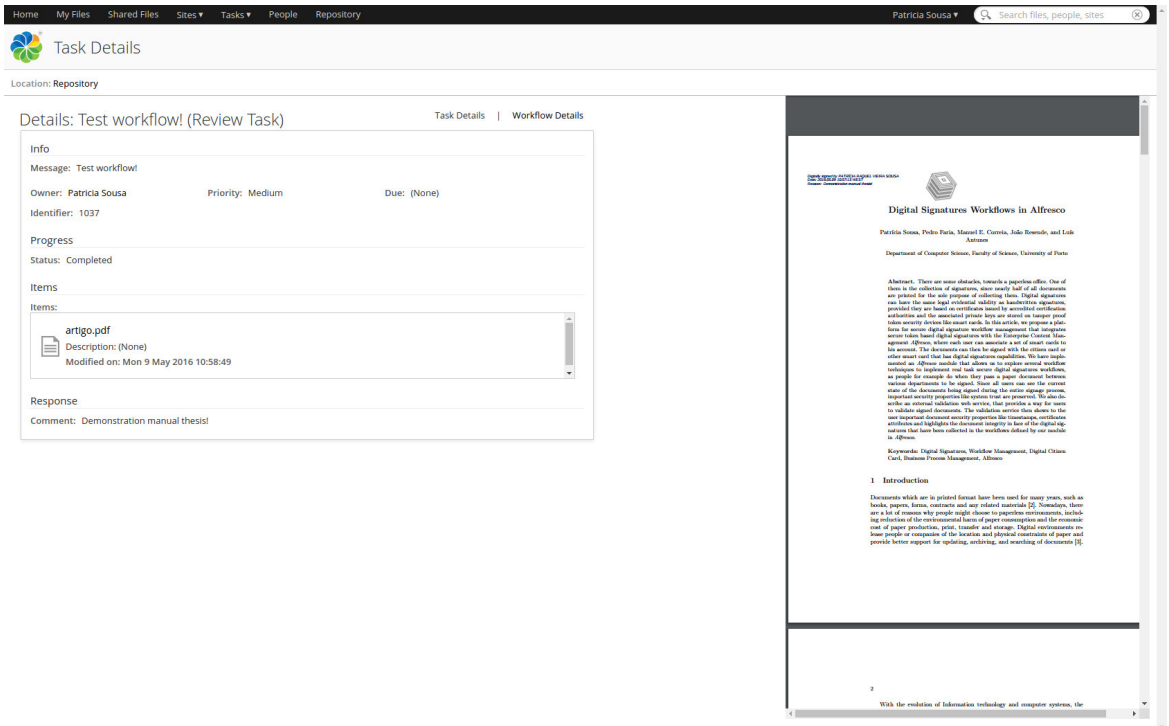


Figure B.9: Task details

7. If a user want to see the workflows that he starts and that have not been completed or accepted yet by him after everyone has done the task. Users can view all current workflows in the tab *Tasks*, the option *Workflows I've Started*. ***Workflows I've Started***.

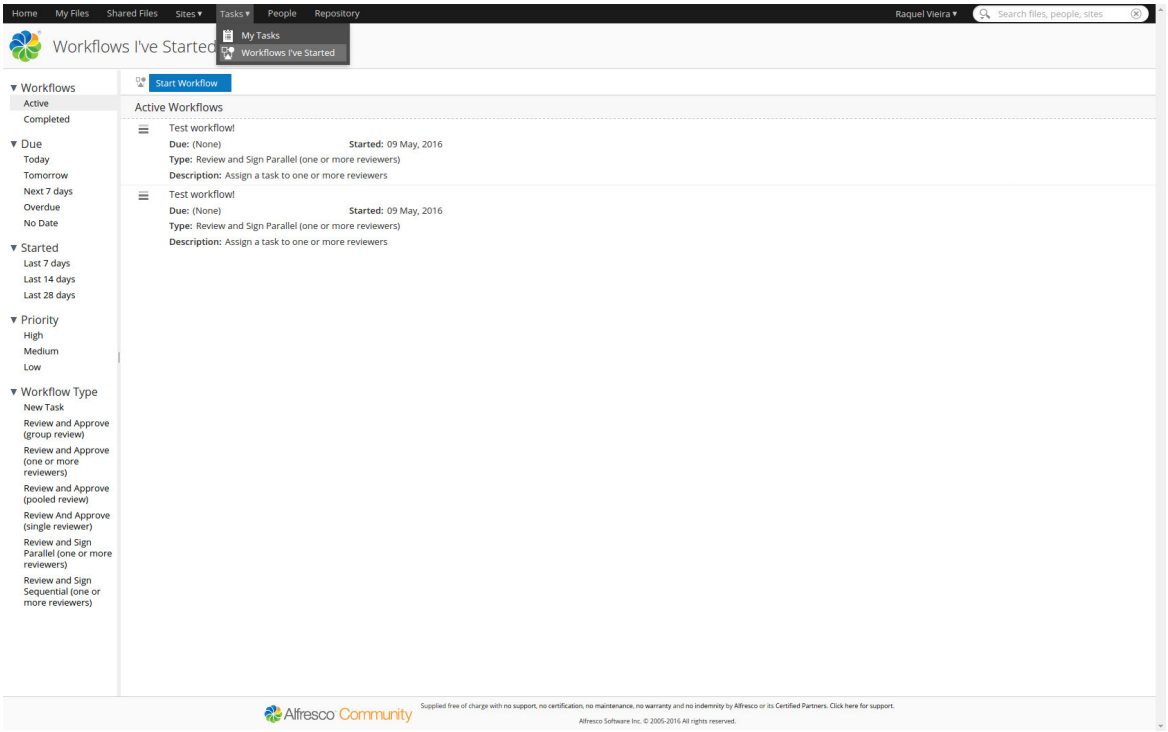


Figure B.10: Workflows I've Started page

8. Another option to enter in the task review, is click on the tab ***Tasks*** and goes to the option ***My Tasks***.

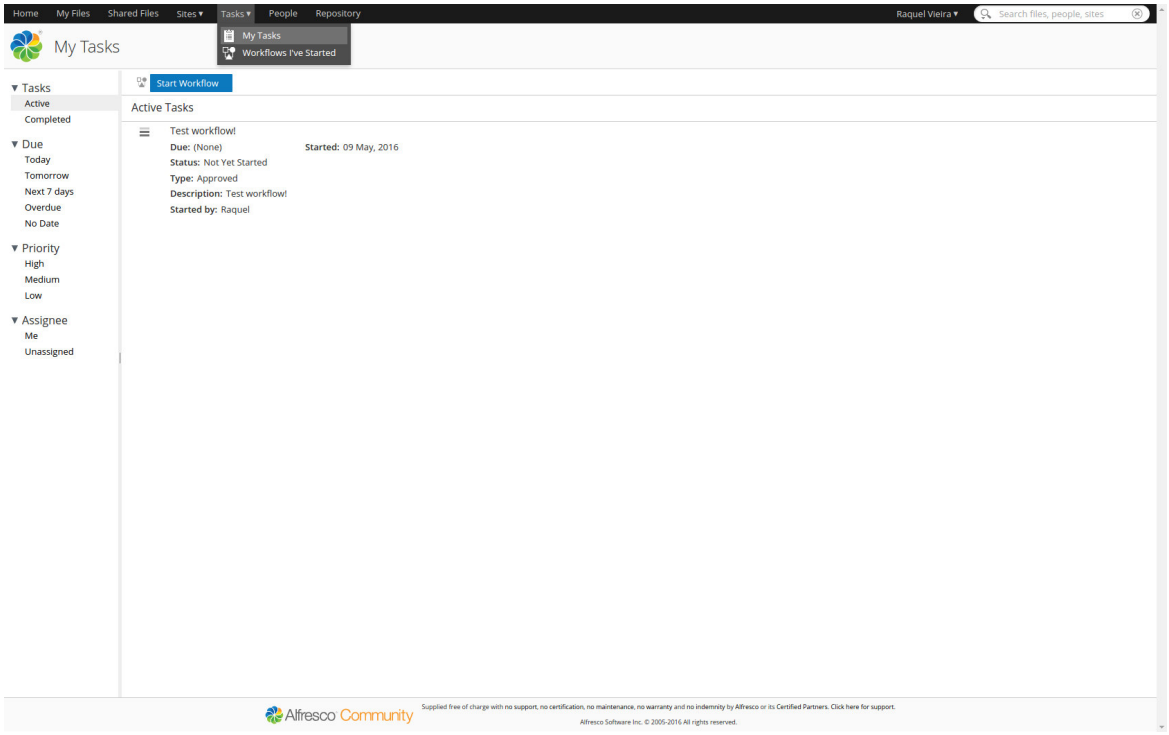


Figure B.11: My tasks page

B.3 Associate card process

1. In the case of the users smart card were not yet associated to their profile, they **have** to do it! If an user tries to make a signature without associate the card, appears a notification to associate the card in the profile.

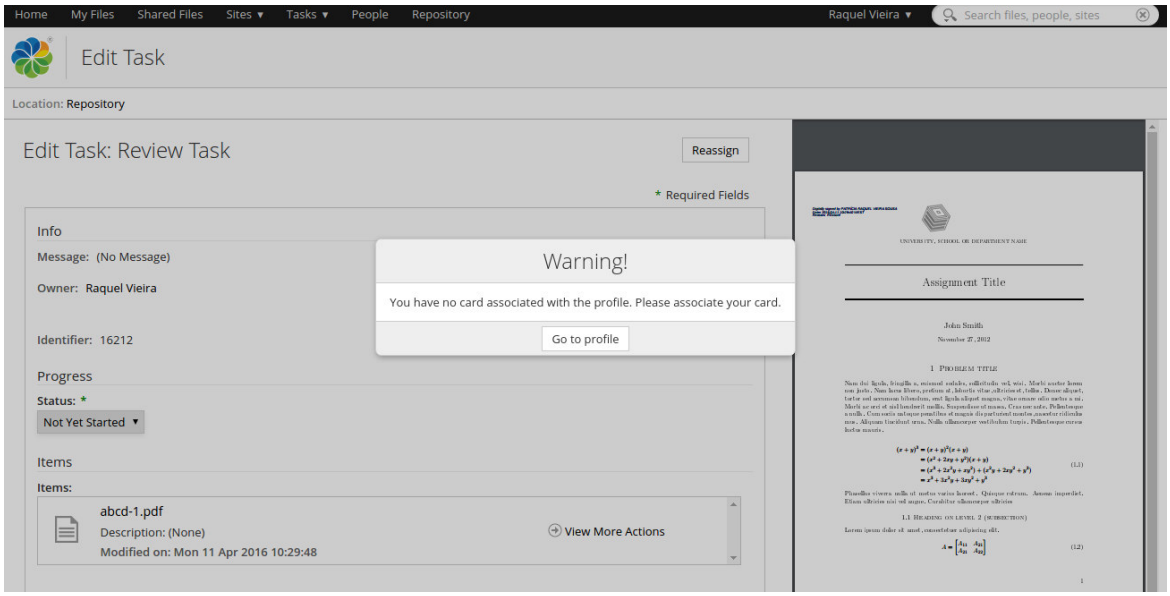


Figure B.12: Task details

In this case, such as we described and show in the figure 4.12, users may hit **Go to profile** and it will be redirect to the page show in the figure B.13.

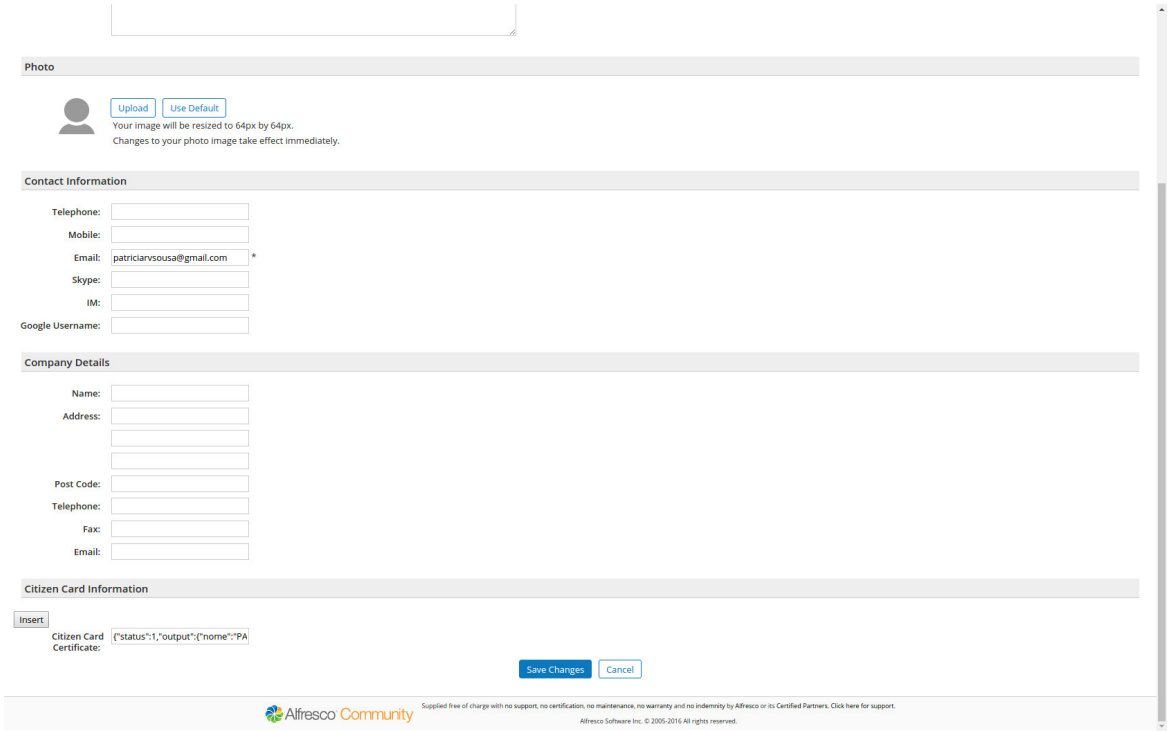


Figure B.13: User profile edit

If users want to associate to their profile their smart card, users can go to the

tab of their username:

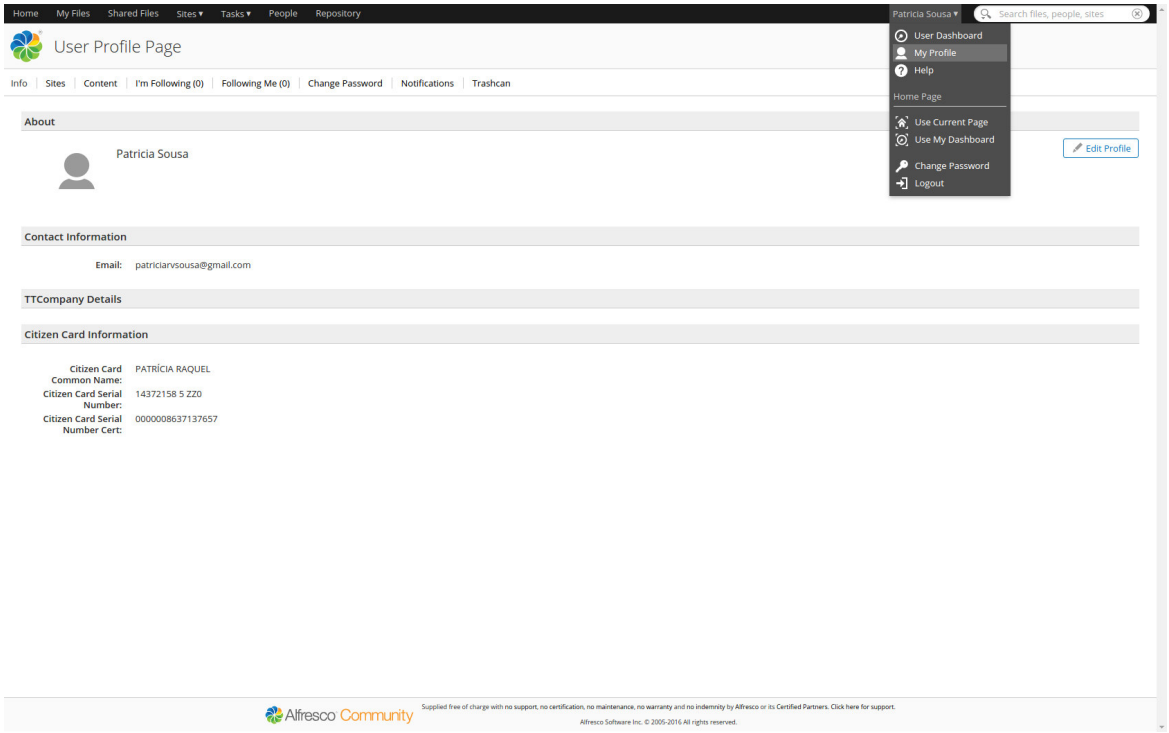


Figure B.14: User profile page

And then, it opens the user profile page. In this page, users have to click in the ***Edit profile***, and it opens a form that users can fill with some information about them.

You have an option "insert" to associate the card, and then, user only have to save the changes made.

B.4 Validation signature process

1. If the user wants to see the validation of a file, at any time, the user have to press the ***Validation Document*** B.3 option in an specific file. In the next figures, we can see an example of the process and an example of validation pop-ups. In the figure B.15 we can see the action that we have to click to access to the signature validation status in the figure B.16. If the user wants ***More Information*** about the fields and the signatures can click in the button ***More Information*** and obtain a pop-up equivalent to the figure B.17 with the information about the file where the user clicks on the option ***Validation Document***.

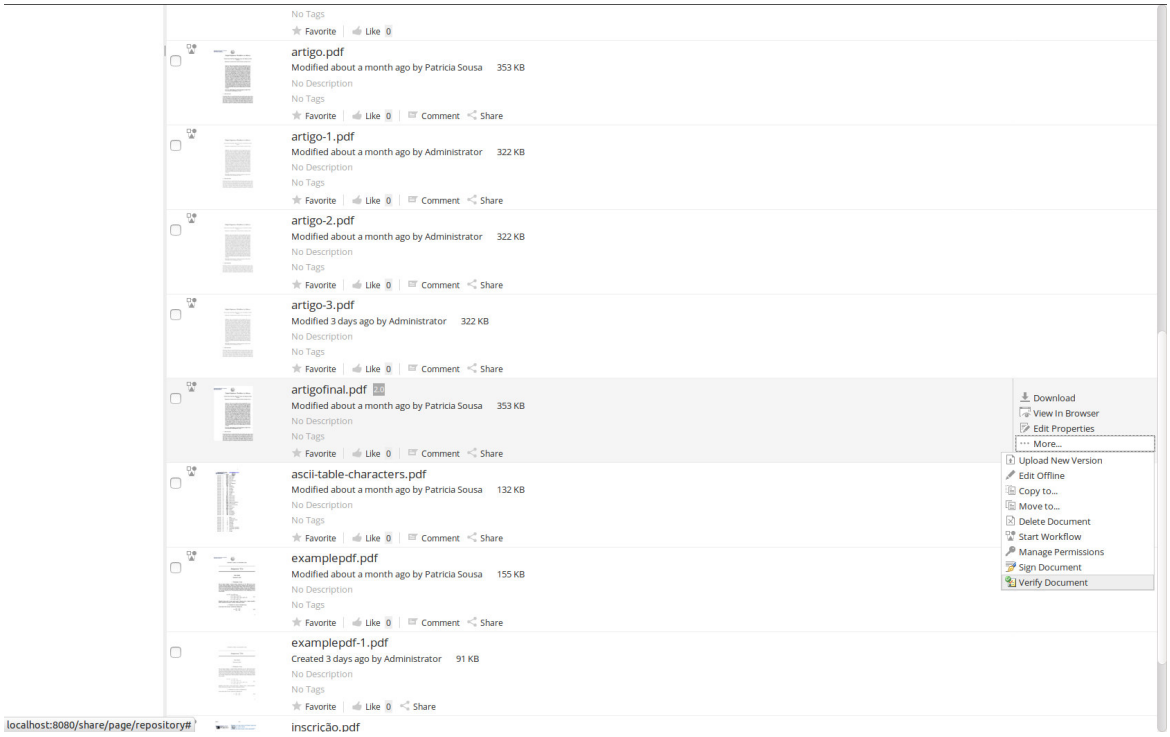


Figure B.15: Validation action

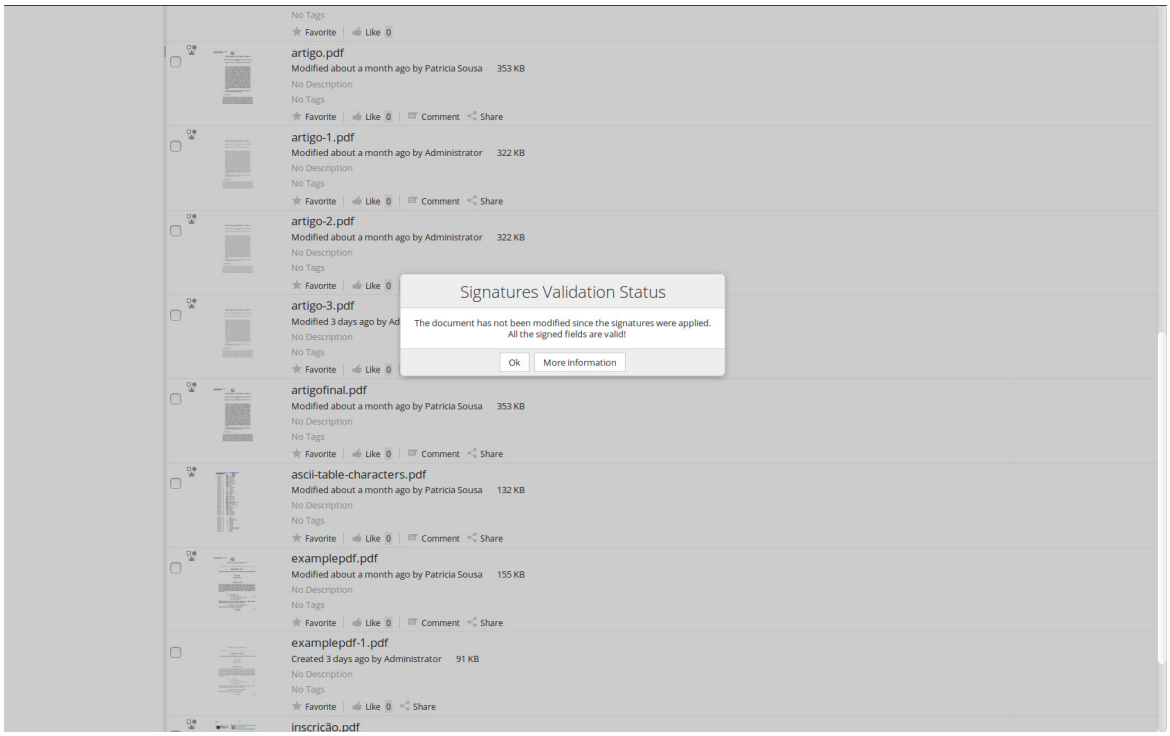


Figure B.16: Pop-up example of signature validation status

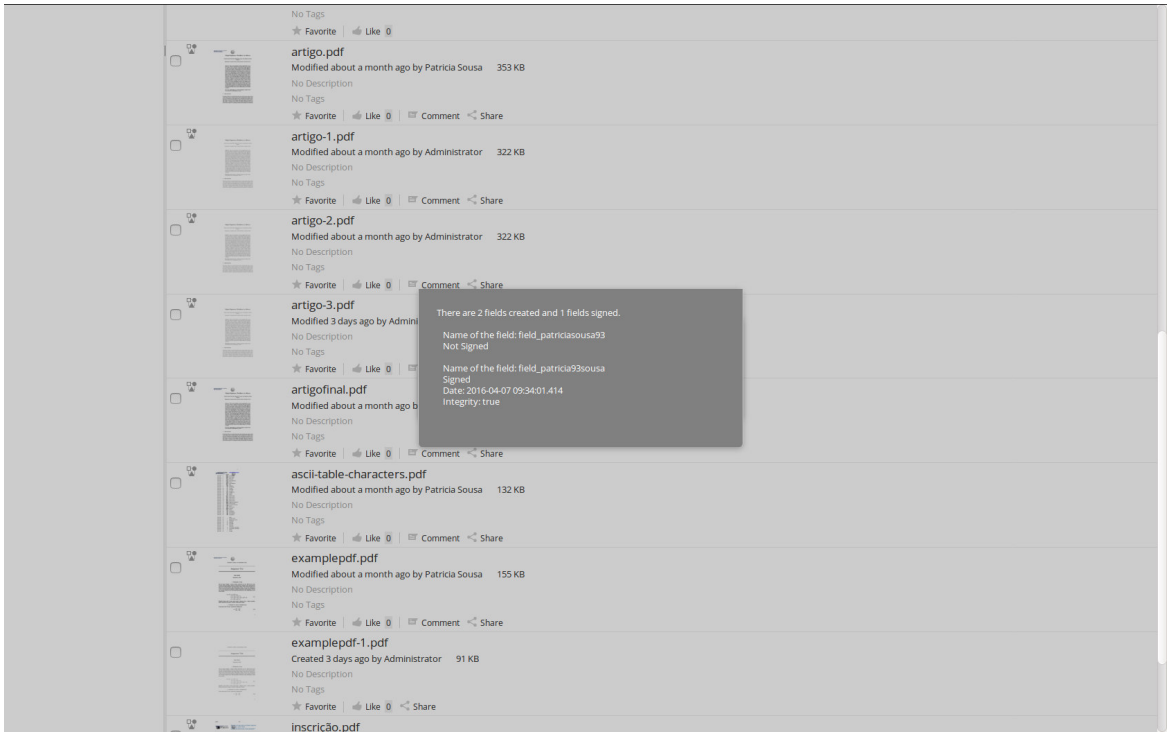


Figure B.17: Pop-up example with more information of signature validation

If an user wants a single signature, the user don't has to create a single workflow. The user can go to an action ***Sign Signature*** in the options of a file that opens a page to sign the document.

References

[1] David Caruana, John Newton, Michael Farman, Michael Uzquiano, and Kevin Roast. Professional alfresco: Practical solutions for enterprise content management. pages 39–42, 2010.

[2] Harika Asili and Omer Ozgur Tanriover. Comparison of document management systems by meta modelling and workforce centric tuning measures. *arXiv preprint arXiv:1403.3131*, pages 1–2, 2014.

[3] Beryl Plimmer and Mark Apperley. Making paperless work. pages 1–2, 2007.

[4] Hind Zantout and Farhi Marir. Document management systems from current capabilities towards intelligent information retrieval: an overview. *International Journal of Information Management*, 19(6):1–2, 1999.

[5] Intergraph. Enterprise content management (ecm) overview. Technical report, Intergraph, 2007.

[6] Majid Vesali. Paperless office. pages 1–7, 2012.

[7] Richard Harper and Abigail J Sellen. The myth of the paperless office. page 17, 2001.

[8] James Nye. Issues and disadvantages of moving to a paperless office. *Issues*, pages 4–5, 2009.

[9] C Sam Charles Devanand. Importance of electronic document/information management systems in modern architectural, engineering and construction projects. pages 1–6, 2015.

[10] M Michael Serbinis, Daniel Leibu, Evan V Chrapko, and Valerian Pappes. Internet document management system and methods. pages 1,16–18, June 24 2003. US Patent 6,584,466.

[11] Heather A Smith and James D McKeen. Developments in practice viii: Enterprise content management. *The Communications of the Association for Information Systems*, 11(1):2–3, 2003.

[12] AJ Van Niekerk. The strategic management of media assets; a methodological approach falta. 2006.

[13] Improve Patient Care. Enterprise content management. pages 4–5, 2001.

[14] Nuxeo. Enterprise platform 5.4. Technical report, Nuxeo, 2013.

[15] John Honeycutt. Digital signature module marketing materials. Technical report, Sharesquared, inc., 2013.

[16] Martin Bergljung. *Alfresco CMIS*. Packt Publishing Ltd, 2014.

[17] Dierk Koenig, Andrew Glover, Paul King, Guillaume Laforge, and Jon Skeet. *Groovy in action*, volume 1. Manning, 2007.

[18] Acklyn Murray, Geremew Begna, Ebelechukwu Nwafor, Jeremy Blackstone, and Wayne Patterson. Cloud service security & application vulnerability. pages 5–6, 2015.

[19] Ephraim Nikoi. Collaborative communication processes and decision making in organisations (advances in human resources management and organisational development). pages 51–54, 2013.

[20] Babette P Aalberts and Simone Van Der Hof. Digital signature blindness analysis of legislative approaches to electronic authentication. *EDI L. Rev.*, 7:7–8, 2000.

[21] Rocci Luppicini. Evolving issues surrounding technoethics and society in the digital age. page 186, 2014.

[22] Je-Gyeong Jo, Jong-Won Seo, and Hyung-Woo Lee. Biometric digital signature key generation and cryptography communication based on fingerprint. In *Frontiers in Algorithmics*, pages 38–49. Springer, 2007.

[23] Samuil Gorelik, Vitaly Lyaper, Lyudmila Bershadskaya, and Francesco Buccafurri. Breaking the barriers of e-participation: the experience of russian digital office development. In *Electronic Government and the Information Systems Perspective*, pages 175–176. Springer, 2014.

[24] Jo ao Pedro Bernardo Gonçalves. Cartão de cidadão: Autenticação de papéis do cidadão. 2010.

[25] Hrvoje Brzica, Boris Herceg, and Hrvoje Stančić. Long-term preservation of validity of electronically signed records. *INFuture2013: Information Governance*, 4:143–158, 2013.

[26] Jonathan E Stern. The electronic signatures in global and national commerce act. *Berkeley Technology Law Journal*, pages 396–397, 2001.

[27] Symeon Xenitellis. The open–source pki book. *Open CA Team*, pages 34–35, 2000.

[28] Andrew Nash, William Duane, and Celia Joseph. *PKI: Implementing and Managing E-security*. McGraw-Hill, Inc., 2001.

[29] Joon S Park, Ravi Sandhu, et al. Smart certificates: Extending x. 509 for secure attribute services on the web. In *Proceedings of 22nd National Information Systems Security Conference*, pages 2–3, 1999.

[30] S. Farrell S. Boeyen R. Housley D. Cooper, S. Santesson and W. Polk. Internet x.509 public key infrastructure certificate and certificate revocation list profile. RFC 5280, RFC Editor, May 2008.

[31] Jonathan Katz. Digital signatures. pages 22–23, 2010.

[32] Prakash C. Gupta. *Cryptography and Network Security*. PHI Learning; 1 edition (November 1, 2014), 2015.

[33] Kenneth H Rosen. *Handbook of discrete and combinatorial mathematics*. CRC press, 1999.

[34] Johannes Buchmann. The digital signature algorithm (dsa). pages 4–8, 2011.

[35] Rolf Oppliger. *Secure Messaging on the Internet*. Artech House, 2014.

[36] Gautam Bommagani, Vinit Kumar Gunjan, Apoorva Paidipelli, and Pooja Singh. Security enhancement of voip protocols using ecc. *International Journal*, 3(12):5–6, 2013.

[37] Disha Shah. Digital security using cryptographic message digest algorithm. *International Journal of Advance Research in Computer Science and Management Studies*, 3(10):2–4, 2015.

[38] Milan Marković, Zoran Savić, and Branko Kovačević. Secure mobile health systems: Principles and solutions. In *M-Health*, pages 15–16. Springer, 2006.

[39] Clifton A Ericson et al. *Concise encyclopedia of system safety: definition of terms and concepts*. John Wiley & Sons, 2011.

[40] R Smith. Cryptography concepts and effects on control system communications, schweitzer engineering laboratories, inc, 2009.

[41] Borko Furht. *Handbook of augmented reality*. Springer Science & Business Media, 2011.

[42] Mabel Vazquez-Brseno et al. Using rfid/nfc and qr-code in mobile phones to link the physical and the digital world. *Interactive Multimedia. Dr. Ioannis Deliyannis (Ed.) InTech*, pages 231–242, 2012.

[43] Choompol Boonmee, Rattapol Chatchumsai, and Sunet Boonmee. Development of electronic correspondence letter time-stamping service using oasis digital signature services. pages 3–4, 2011.

[44] Ian Goldberg, David Wagner, Randi Thomas, Eric A Brewer, et al. A secure environment for untrusted helper applications: Confining the wily hacker. In *Proceedings of the 6th conference on USENIX Security Symposium, Focusing on Applications of Cryptography*, volume 6, pages 1–1, 1996.

[45] Zakir Laliwala and Irshad Mansuri. Activiti 5. x business process management beginner's guide. pages 26–27, 2014.