TÉCNICO LISBOA

qChain: A Blockchain Solution Applied to Citizen-Centric Document Automation Platform

Alexandre Gaspar Manso

Thesis to obtain the Master of Science Degree in

Information Systems and Computer Engineering

Supervisor(s): Prof. Alberto Manuel Rodrigues da Silva

Prof. João Paulo Pedro Mendes de Sousa Saraiva

Examination Committee

Chairperson: Prof. Paolo Romano

Supervisor: Prof. Alberto Manuel Rodrigues da Silva

Member of the Committee: Prof. João Carlos Ferreira

November 2021

Acknowledgements

I want to express my gratitude to both my thesis supervisors Professor Alberto Manuel Rodrigues da Silva and Dr. João Paulo Pedro Mendes de Sousa Saraiva for motivating me onto finishing this work and giving my best efforts, for always believing in my intentions and that I would, at least, try my best to overcome my challenges.

I also want to thank my parents João and Zara for supporting me through it all, as well as my brother Diogo for all the encouragement and relief they provided, helping me with exhaustive processes in my daily life outside of work and college, and making it possible to not only better manage my time but to keep my mental health.

Most importantly I want to give my deepest gratitude to my wife, Marta, for all that she has done for me, for every word of encouragement and every bit of love that would give me hope to keep pushing through. Also, would like to thank her for believing in me, for making me feel better about my accomplishments and for always standing by my side and help me overcome every situation alongside me. Without her, I could have not been any successful.

I want to thank God for all that He provided, especially in these months, for all the love he has put in my life. I also want to thank my community, for all their prayers and all their understanding and support.

I must thank my most precious friend Miguel, which was one of the most gentle and respectful of people that I encountered during this journey. I also give my gratitude to my closest friends: Catarina, Gonçalo, Isabel, José, João, Pedro and Daniel for all they put up with and endured during the stressful situations that happened in the last months.

Lastly, but not least, I want to thank to my colleagues at Cybersafe, for all their help, my boss Nelson, which provided me a lot of learning opportunities and did everything he could to make me feel welcome and happy during my first job.

Abstract

Documents are an important method of registering human's data and actions of public and private interest that involve one or more authorities. Document automation technology is useful in facilitating processes related to every stage and aspect of document digitalization and management in a rising digital world. Document automation platforms have been progressing to public use, focusing on critical certificates and agreements, very present in legal departments and government authorities, but slowly, they are migrating to wider audiences, focusing more on easing citizen daily lives. Alike any paper document that needs to be authorized, signed, sealed according to importance, and stored securely, these operations performed in network-based systems, need that same treatment digitally. qDocs is a citizen-centric and multi-organization document automation platform designed for citizens and friendly use, focusing on reducing unneeded and exhaustive processes while assuring digital safety and validation. This feature is the concept and implementation of this thesis, called qChain, which intends to integrate the qDocs platform with a blockchain-based system, as an immutable database and integrity validator of the system's reliability and authenticity. qDocs does not implement any security measures regarding document management, hence the need to timestamp every event from creation, to signing to access that alters the document's information. qChain provides integrity validation mechanisms to assure the conformity and eligibility of the documents. The research is evaluated by developing templates representing different case studies that evaluate the solution to its extent.

Keywords

Document Automation, Blockchain, Multichain, Smart Contracts, qDocs, qChain, Security

Resumo

Os documentos são um método importante de registar os dados e ações humanas, quer de interesse público ou privado que envolvam uma ou mais autoridades. Software de automação de documentos revela-se bastante útil ao facilitar os processos relacionados com todas as etapas e aspetos de digitalização e gestão de documentos num mundo de crescimento digital. Estas plataformas têm vindo a progredir para o uso publico, focando-se em certificados e acordos de teor mais critico, bastante presentes em departamentos legais ou autoridades governamentais, mas que, com o seu tempo, têm vindo a agradar a uma população maior, focando-se mais em ajudar o quotidiano dos cidadãos. Semelhante a qualquer documento físico em papel que precisa de ser autorizado, assinado, selado consoante a sua importância e guardado de forma segura, estas operações realizadas por um sistema com base na rede, precisam igualmente do mesmo tratamento de forma digital. O qDocs é uma plataforma multi-organizacional de automação de documentos centrada no cidadão projetada para o uso fácil por parte do cidadão, reduzindo a dependência de processos exaustivos e ao mesmo tempo assegurando segurança digital e validação de informação. Esta última propriedade é o conceito apresentado e implementado neste documento de tese, denominado qChain, sendo a integração de uma plataforma na rede com base na blockchain, servindo de uma base de dados imutável e um validador da integridade da informação e por conseguinte da resiliência e autenticidade do sistema. O qDocs não implementa medidas de segurança no que toca a gestão de documentos, por conseguinte, é necessário registar todos os eventos desde a criação, assinatura e acesso aos documentos, eventos estes que alteram a informação do documento. O qChain oferece mecanismos de validação da intetegridade que asseguram a conformidade e ligitimidade dos documentos. Este estudo é avaliado com o desenvolvimento de templates adaptados a diferentes casos de uso que procuram testar a solução em toda a sua extensão.

Palavras-Chave

Automação de Documentos, Blockchain, MultiChain, Smart Contracts, qDocs, qChain, Segurança

Table of Contents

Acknowledgements ... **iii**

Abstract ... **v**

Keywords ... **v**

Resumo ... **vii**

Palavras-Chave ... **vii**

Table of Contents ... **ix**

List of Figures ... **xii**

List of Tables ... **xiv**

List of Acronyms ... **xvi**

1. Introduction ... **1**

1.1. Motivation ... 2

1.2. Objectives ... 3

1.3. Proposed Solution ... 4

1.4. Research Methodology ... 5

1.5. Dissertation Structure ... 6

2. Background ... **8**

2.1 Document Automation ... 8

2.1.1 qDocs, a Citizen-Centric Document Automation Platform ... 9

2.1.2. Other Document Automation Platforms ... 10

2.1.2.1. MHC Document Automation Software ... 11

2.1.2.2. HotDocs ... 11

2.1.2.3. Documate ... 11

2.2. Blockchain Technology ... 12

2.2.1. Blockchain Platforms ... 14

2.2.1.1. BitCoin ... 15

2.2.1.2. Ethereum ... 15

2.2.1.3. NXT ... 16

2.2.1.4. Hyperledger Fabric 17
2.2.1.5. MultiChain 18
2.2.2. Discussion 19
3. Related Work 22
3.1. Open Law and Reuters 22
3.2. Azure's e-Document Verification with Smart Contracts 23
3.3. ActiveDocs 23
4. qChain Architecture 25
4.1. qDocs Requirements 25
4.2. System Infraestructure 26
4.3. Summary 27
5. qChain Implementation 30
5.1. Development Environment 30
5.2. Modules 30
5.3. System Functionality and Specifications 32
5.4. Used Tools/Libraries 38
6. Evaluation 41
6.1. Test Environment 41
6.2. Test Results 41
7. Conclusions 43
7.2. Contributions 43
7.3. Future Work 44
References 45
Appendix A 49

List of Figures

Figure 1 - qDocs Infrastructure Flow Diagram 1
Figure 2 - qChain Integration Infrastructure Flow Diagram 4
Figure 3 - qDocs System Architecture 10
Figure 4 - qDocs/qChain Infrastructure Architecture 26
Figure 5 - Detailed Operation Flow Diagram of qDocs/qChain Infrastructure 27
Figure 6 – Final Flow Diagram of Operations for qDocs/qChain Integration 31
Figure 7 - Visual Presentation of the new UI for Document Template Creation 32
Figure 9 - Checkbox Value Persistence as a Boolean in DocumentTemplate Class 33
Figure 10 - Stream's Identifier Addition to Curator's Organization 34
Figure 11 - Document's Persistence in qChain after initial Creation 34
Figure 12 – Business Process of Creating a Document in qDocs 35
Figure 13 - Code Implementation for an Open Document Integrity Verification 36
Figure 14 - Code Implementation for a Closed Document Integrity Verification 36
Figure 15 – Business Process of qChain’s Integrity Checking for a Single Participant Document .. 36
Figure 16 - Business Process of qChain’s Integrity Checking for a Multiple Participant Document 37

List of Tables

Table 1 - Comparison of the researched blockchain technologies 20

List of Acronyms

CA	Certification Authority
DA	Document Automation
HTTP	Hypertext Transfer Protocol
PoS	Proof of Stake
PoW	Proof of Work
SaaS	Software as a Service

1. Introduction

This chapter explains the motivation behind this project, presents a brief description of the requirements for the solution and its initial prototype. We also refer the objectives of the project, their state of development and its research methodology.

We discuss how document automation has become more relevant nowadays. This refers to a particular sector for automation of critical computer processes such as managing , signing and storing relevant documents in digital format [1]. The main purpose of document automation is to help reduce time wasted on these processes while being transparent to the everyday citizen that benefits with this service. Public document automation platforms have then been created, focusing on citizen's needs, but still, the biggest implementations for automation of documents are private, residing primarily within legal firms [2].

Parallel to the growth of the document automation (DA), blockchain technology grows to be a sustainable solution for safe transactions of digital assets in the internet, which justifies its versatile integration with various IT sectors [3][4]. New platforms supporting blockchain technology are increasingly appearing constantly with various appliances to modern era, offering trustless distribution and integrity and authenticity protocols to critical process applications.

qDocs [5] is a citizen-centric document automation platform, created to promote this environment of transparently providing an easier alternative for document management to the citizen as a remote service, all in a user-friendly web platform. Organizations, called curators, representing official notaries, are responsible for the creation of form-like templates for fast document creation which are then stored locally, close to the official owner, the curator itself, while ubiquitously and unfailingly accessible to the client without any constraint [6]. Nevertheless, if the platform is also responsible for upholding the document's management, storage, and traceability, it is important to implement good security measures to assure the effectiveness of the service, without compromising performance and platform usability.

qDocs is a document automation platform that intends to ease document management in the public sector, accessible to every citizen. A simple diagram of the platform's purpose can be seen below in **Figure 1**.

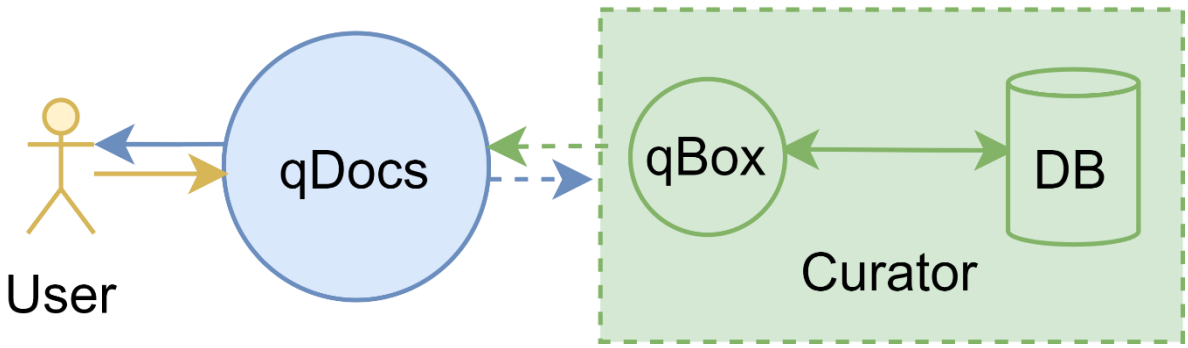

Figure 1 *- qDocs Infrastructure Flow Diagram*

Figure 1 describes the flow of information between the inner components of the qDocs overall infrastructure. In the context of this system, the information flow is in the form of digital documents which are in constant traffic from the user to the curator's databased, where they are stored. The user interacts directly with the qDocs platform from which he/she creates and manages documents and stores them alongside the curator, in their database, using an intermediary, qBox. qBox resides between the qDocs main server and the curator database and it is used for document retrieval, translating requests for a standardized format holding logical and essential information for document fetching.

This research focuses on the integration and interoperability of qDocs with a blockchain system relying on this technology's security properties as an immutable database offering a wide array of advantages over a normal database such as smart contracts and trustless distribution [3].

Therefore, qChain shall be a blockchain-based system that intends to extend and provide qDocs with secure operations related to document management performed users of the platform that are considered critical and important, offering asset integrity, non-repudiation of user operations, secure transactions and storage processes, in a decentralized network with non-dependance on trusted participants [7].

The concept of smart contracts is also useful to qChain. Smart contracts are codified conditions deployed in a running instance of a blockchain platform [8], which try to transparently secure the ongoing transactions without compromising its performance and without depending on a trusted third-party to officialize the overall agreement between peers. Smart contracts are particularly useful when exchanging digital assets between different peers considered critical and relevant to the participants involved, without compromising any information and promoting trustless distribution of the network [9]. Therefore, in the context of qDocs/qChain integration, when signing agreements or contracts with escalated importance and multiple participants, smart contracts help executing these securely and respectfully to everyone transparently.

1.1. Motivation

The motivation behind the integration of document automation and blockchain is to secure citizen centered processes such as managing and signing important documents.

A satisfactory solution is deploying a blockchain-based network and integrate the platform operations with the security benefits of this technology while taking advantage of the performance offered by its distributed nodes capabilities.

Consider, the use case of creating a critical digital document using qDocs, in which two public companies wish to emit a partnership contract. This contract also needs government approval and acknowledgement. Two managers from each company have an account in the platform and often use it to create digital documents not relevant enough to be issued in the blockchain. Nevertheless, this document is deemed important and therefore its information should be secured using proper measures. A template may be created, with indications to use blockchain security, be attached to both companies and only for that type of document, using this template, additional security properties are conferred.

We can also consider the use case of a simple citizen card issuance, which the content should not be meddled with after its creation, and consequently should be stored in a blockchain-based environment. The card is signed by the person wishing to create the document and further stamped by the government's official authority as a seal of approval. The two operations This will not only confer stronger integrity enforcement and authenticity but reduce identity fraud attacks, time consumed in a citizen's shops' queues and in mailing them home and increase the throughput for these operations.

These use cases, although different in complexity, can surely benefit from an architecture able to reliably automate the stages related to the time-consuming processes of document management. As explained, qDocs, is a document automation citizen-centered solution, aims to mitigate the long operations that normally are performed by the human hand and, therefore, automating every process document related. However, qDocs does not offer the security measure already discussed in the previous paragraphs that a blockchain-based system surely does, hence the need to integrate both technologies, document automation and blockchain, and promote a more secure and reliable system, with automating capabilities and integrity checking properties while not impacting the user's experience.

1.2. Objectives

This thesis aims to research how to extend and adapt the qDocs platform with blockchain technology, taking advantage of its security properties. Therefore, multiple blockchain technologies were studied, document automation platforms were analyzed and further integrations of both were analyzed, all to assure the best efficiency in interoperability for this integration.

The main objective for this project is to secure document related operations in the qDocs platform, registering every event from creation and modification to access in a distributed and trustless blockchain-based system, without compromising the normal functionality of qDocs and user experience and conferring integrity, authenticity, non-repudiation, and reliability to qDocs

The solution considers the transparency of the user experience, focusing on maintaining the platform's friendliness and applicability without impacting the user's experience, as the citizen does not even have to be aware of qChain. He/she must only know that extra security properties were offered to his/her service and feel safe about it.

The integration also desires be light and not overload the system with heavy usage of the platform weighing the system's performance and reliability, being highly maintainable, customizable, and expandable.

Lastly, license free libraries are used with huge online support by developers and communities, helping future maintenance and error solving.

Overall, the objectives line up to create a reliable security mechanism for document management as a citizen-centered service.

1.3. Proposed Solution

The Blockchain as an intermediary between the platform and the curator's database, in which the digital documents were upheld. Therefore, every operation in the domain of document management will be registered in an immutable database if the documents involved are deemed to have extra security measures by the curator.

The blockchain technology excels at conferring security measures necessary for the system's wellbeing without compromising its utilization, such as integrity of information with repetitive cycles of checking the information conformity and reliability and immutability of storage, timestamping every operation, forever. Therefore, blockchain-based networks possess integrity persistence mechanisms which confer proof of temperance to the digital assets that are part of the network [4][10]. It is highly unwanted that the documents are altered by external attackers with malicious intentions or that data conflicts happen within the database compromising the system's reliability and corrupting the content which is precious to the user.

We decided to call this solution qChain. Complementing the **Figure 1** above, which represented the ordinary flow between the inner components of the qDocs entire infrastructure, **Figure 2** can be observed as the new system after the integration with a blockchain-based system, with slight changes to the flux of operations. qChain holds text tokens representative of specific events that happen within the qDocs server, as form a form of timestamping and registering every operation performed. This feature is essential for information integrity and system reliability, and it is only turned on for types of documents in which the respective curator decided that it needed additional security measures.

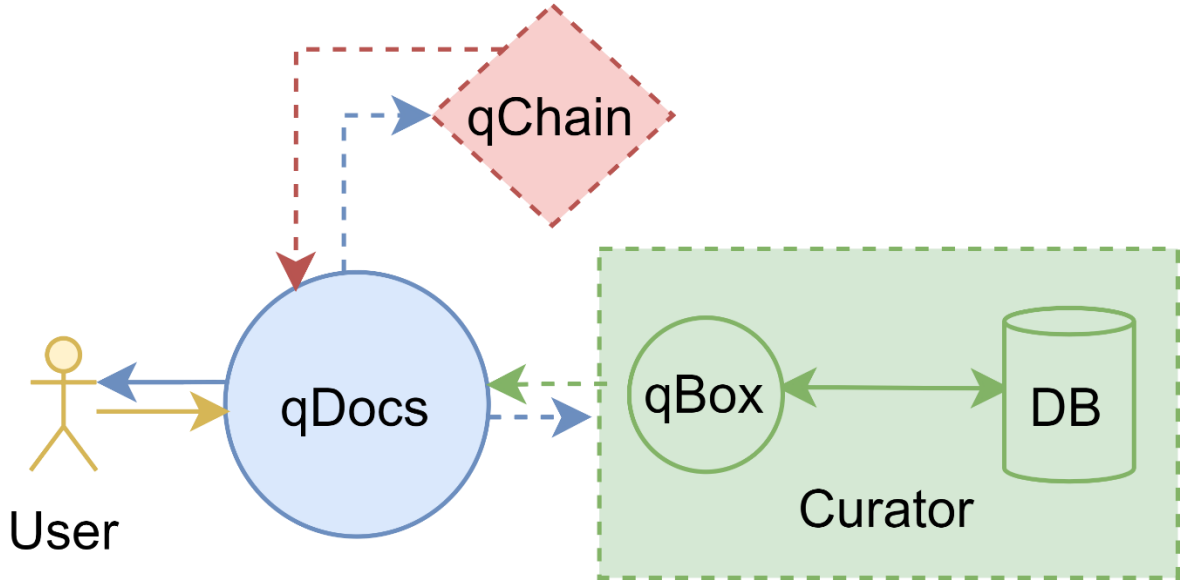

Figure 2 *- qChain Integration Infrastructure Flow Diagram*

qChain aims to secure operations related to documents in need of the properties offered by blockchain as a service. Not all documents flux through this extension of the platform, only the types which the curator decided that were critical enough to be traced and registered.

1.4. Research Methodology

This research follows the Design Science Research (DSR) methodology. The DSR is an iterative methodology that combines principles, practices, and procedures. It provides guidance for research in Information Systems (IS) as well as other disciplines [11], [12]. Design Science lays emphasis on systematic, testable, and communicable methods [13].

For instance, Hevner et al. propose a set of **guidelines** for the application of DSR in the information systems area, namely including the following aspects [11]:

Design as an Artifact: The research shall produce a viable artifact in the form of a construct (e.g., software application or tool), a representation (e.g., new language or extension of a previous notation), a technique (e.g., a process or method), or an instantiation (e.g., a case study that applies such artifacts).

The relevance of the problem: The basic objective of DSR is to develop technology-based solutions to relevant and significant business problems.

The design evaluation: The quality (e.g., measures in terms of utility or efficacy) of the design artifact shall be demonstrated rigorously through a well-executed evaluation method.

Research contribution: Effective DSR shall offer a clear and demonstrable contribution in the area that the design artifact is applied such as design foundations and or design methodologies.

Research Rigor: The DSR depends upon rigorous methods application in both evaluation and the construction of the design artifact.

Design as a search process: The search for an effective artifact depends on the use of the available ways to reach desired outputs while the rules in the problem environment are still satisfied.

Communication the results: The research presentation shall be effective from both the technology- and the business-perspective.

These guidelines drive the research but can be better translated into the following **phases** [12]:

Problem identification and motivation: We identified the problem as qDocs not implementing security measures relating to document management and retention, and therefore, it is useful to integrate a security mechanism, such as a blockchain-based system to confer stronger authenticity, integrity, and reliability for qDocs. Security is essential in a system that holds valuable information such as documents and contracts, with personal and critical information, hence the need for the integration.

Define the objectives of a solution: The solution aims to achieve security properties such as conferring integrity to the information, trustless decentralization of operations, timestamping of events and transparent implementation without impacting user experience. It is a very advised solution in present society to use blockchain-based networks as intermediaries to information flow within companies holding valuable digital assets, therefore, we know such integration is possible with qDocs, existing multiple blockchain implementations adaptable to this platform.

Design and development: We decided to develop the subsystem qChain as an extension to the qDocs platform, as a complementary and non-mandatory service, which works on demand by the curator wishing extra security measures for their documents. The existing code implementation was studied and

modified to efficiently satisfy the objectives implementing various libraries that easily integrated the service.

Demonstration: The qChain system was not fully demonstrated due to implementation issues, nevertheless the integration works manually, although not being an ideal solution.

Evaluation: We evaluate the solution by creating multiple types of documents with different layers of creation, which are stored in qChain and priorly visited for integrity checking testing. No proper testing could be done due to connectivity issues.

Communication: This thesis and an extended abstract serve as communication papers for this integration following the project proposal delivered early this year.

1.5. Dissertation Structure

This report is structured in the seven chapters, as follows:

Chapter 2 discusses the background for both technologies, document automation and blockchain, the most popular implementations and comparisons between each.

Chapter 3 references previous work related to the integration of document automation and blockchain technologies.

Chapter 4 describes the architecture of the solution and the initial requirements and challenges of its implementation, along with the structural changes adopted as we came across such challenges.

Chapter 5 describes the technical details of the project, not only including the requirements and challenges imposed but also the libraries, software and configurations used.

Chapter 6 refers the tests conducted on the platform along with their results.

Chapter 7 concludes the thesis with last thoughts about the overall project, its implementation and integration in present and future.

2. Background

Throughout this chapter, the background research results for both technologies, document automation and blockchain will be analyzed and displayed, along with examples of platforms relevant to the history of each market and their highlights.

Both technologies have been around for a while now and are still striving with the constant appearance of innovative solutions, platforms, and integrations as sometimes they reveal to be a lot more useful than we think, considering that both sectors focused on a privative sector of legal firms.

We studied the most relevant document automation and blockchain solutions to not only be familiarized with the main aspects of a public and successful document automation platform but also to pick the best blockchain implementation to be integrated with qDocs. We already had proposed Multichain as the best candidate, but other technologies deserve to be mentioned.

We considered how easy it is to learn these technologies, to integrate them with existing platforms, how effective and vast the solutions are, and the integrity's transparency to the everyday user's operations. It is equally important how the amount of documentation and community support is present, and the legal license for each technology for maintenance and system sustainability purposes. Therefore, it is expectable that in each technology I gave my opinion in these matters.

First, we discuss the background of qDocs on its motivation, system architecture, operation basis and how it is relevant today and to whom. Although platforms will be mentioned, implementation in details, related work and studies will only be addressed with more detail in the next chapter.

2.1 Document Automation

Documents can be referred as the representation of a piece of information which an individual or a group of individuals intend to emit and share with one another. For years now documents are increasingly getting through a process of digitalization, rising adopted by more peers in present society. In the reality we live in, due to the current pandemic, these digital mechanisms, which strive to help citizens in long, mundane tasks, have proven particularly useful. Document creation and management benefits exponentially from the properties offered by document automation, such as:

- Increasing service up time and reduce time wasted in queues.
- Ubiquitous platforms in which we can access from everywhere, to every document at the tip of your finger.
- Reduce in paper waste due to increasing digitalization of these assets [14].
- Mitigate illegible documents due to paper becoming old and yellowish and characters disappearing due to ink smudginess because of bad preservation.

Nevertheless, digital documents also have their issues and limitations, primarily the fact that they are more accessible to online attackers, also known, as hackers. These look up to damaging digital property by meddling with the information or even replicate it for personal use. Therefore, a well thought solution for preventing this kind of situation had to be implemented and integrated with the ongoing platform, so,

as discussed in the previously presented proposal, that solution is to store document metadata in the blockchain, assuring the platform's integrity and persistence.

Another problem would be to lose the sense of materiality and ownership for not possessing a physical version of the document, which can result in a loss perception of importance towards the digital document [15].

Before we discuss two of the most famous document automation platforms used today, let us first reference qDocs as, until now, a document automation citizen-centric platform offered as a public service, which before this research, did not implement any security measures.

2.1.1 qDocs, a Citizen-Centric Document Automation Platform

qDocs is a solution for public citizen-centric document automation developed by the MDSS company. It has three components: a curator server with a database responsible for storing the documents, as close as possible to the official owner entity a main server which processes the entire platform's operations and a user interface, accessed via browser, in which curators, citizen or users, and administrators, are offered a wide array of operations and permissions to the platform.

The platform resumes to a peer-based service, citizen-centered, but managed by admins and seeded by curators. In these platform, the documents can be something simple such as a citizen card to an academic record to a certification of open financial activity [6].

Designers assigned from the curator's affiliation, define the architecture of the document using templates. These are fillable forms, with multiple input types, which produce a digital asset equivalent to the physical one, in shape, appearance and legitimacy. The input types vary from specific fields for decimals, text, signatures, or drop-down lists with multiple choices, all declared by the curator as important. After the form is properly filled and the document created, the latter is stored onto the curator's database. Simultaneously, the qDocs server saves metadata relevant to document fetching performed by qBox, an intermediary to the document retrieval [16].

When the user wants to access an existing document he owns, or create a new one, both the existing document and respective template are stored within the curator that legally issues it. As said before, qBox serves as an intermediary for document fetching from the platform to the document's storage, upon request, using metadata present in qDocs, such as its reference in memory or the document's file name.

The curator's database helps supply security and legal ownership of the documents as they are kept close to the respective legal issuer. Curators are responsible for officially issuing these documents, legally binding them to their owner while being aware of the importance that is to not only store the document securely but take careful measures about its retention [17]. The qDocs architecture, in more detail, can be seen in **Figure 3.**

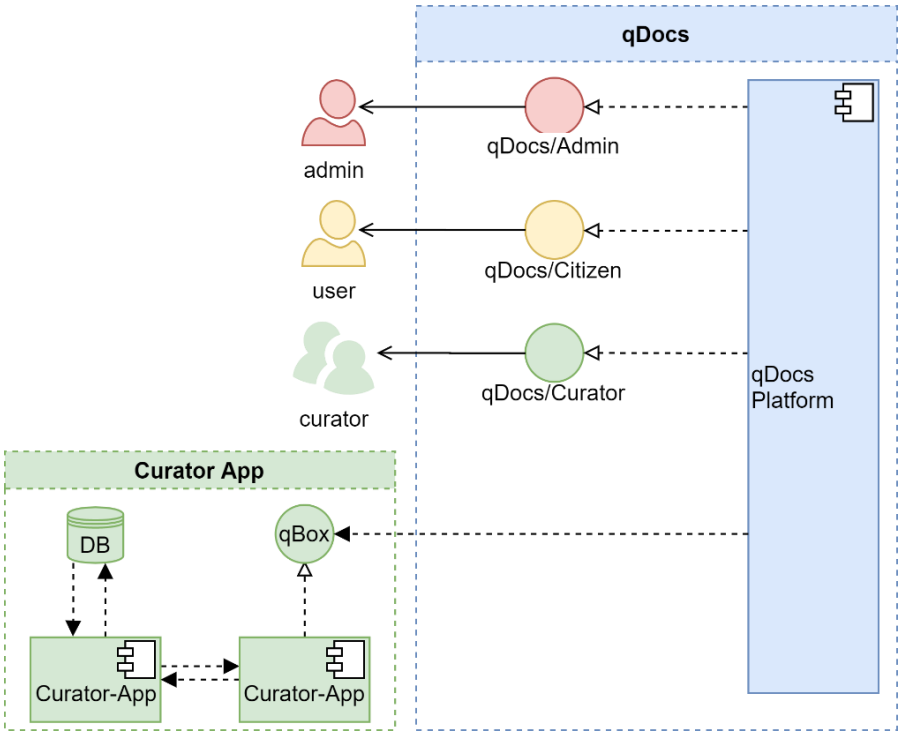

Figure 3 *- qDocs System Architecture*

We quickly refer that this project's intention is to integrate an integrity checking mechanism, such as blockchain, between the platform and the curator, securing the documents' related operations and processes from their creation, management and sharing.

Before we get acquainted with the blockchain's background, basic functionalities, and famous implementations, let us first discuss two present document automation platforms which may be comparable to qDocs, as a helper to further understand how this technology affects us today.

2.1.2. Other Document Automation Platforms

Most solutions of Document Automation (DA) that support promoting major digitalization and easiness of document management and more importantly, this technology is moving further to the public sector, focusing on citizens and their needs.

Document Automation (DA) platforms were initially used for companies to which document preservation and digitalization was useful, not only for management purposes but to reduce paper waste, recycle costs and help the planet's environment [14]. Therefore, these felt particularly restrict and not ideal for a large scale of citizens who would be willing to use these applications instead of waiting in lines to get a physical copy instead, by themselves.

From the three platforms presented in the next sub-sections, the first two are of public use and the last is privative. In the end, a little table will be displayed for easier comparison between implementations of document automation.

2.1.2.1. MHC Document Automation Software

MHC developed a platform for automation of processes related to business-critical documents. They offer a customizable and scalable service, with options for full automation or sometimes partial, for more delicate operations with post-creation reviewing needs [18].

Their software aims to simplify networks withing enterprises by removing time wasted on document filling and archiving, promoting digitalization and automation of processes that otherwise would take specific roles withing the company to fulfill. Therefore, they focus on efficiency and wasted time reduction, resulting in a faster and more organized flux of operations internally and between organizations.

The documents are always all centralized and accessible to the user conferring a reliable SaaS in which, similarly to qDocs and most document automation platforms, documents are presented as fillable PDF's. These also have the possibility for scheduling the periodicity of documents such as monthly reports and customize the data aggregation for document creation. There are also multiple kits and use cases that can be automatically assembled in packages for a full experience, all in a user-friendly, intuitive interface.

2.1.2.2. HotDocs

HotDocs is a software as a service developed by AbacusNext that focuses on building, on demand, cloud-based solutions for Legal and Accounting firms, aiming to secure cloud-based operations, in this case document automation related [19]. Their system is designed to be reliable and ubiquitous performing tasks very similarly to MHC's solution but with the addition of providing integration with complex calculations and conditions within the document's template filling process.

Designed to be easily integrated with tools and platforms in use on ongoing operations within companies, such as business process, customer relationship and document management systems. AbacusNext main goal is to increase productivity by optimizing every process their application or solution aims to integrate and implement, and in the context of HotDocs, it is document management. Offered as a customizable service, clients can alter templates to better fulfill their experience and adapt to the company's needs. Furthermore, compliance is highly enhanced using built-in regulations in templates drastically reducing risk exposure and preventing eventual data leaks.

2.1.2.3. Documate

Documate proclaims themselves to be the most secure and user-friendly document automation platform, having equal properties to its competitors. Their goal is also to reduce wasted time on creating documents, with the premise that people should stop worrying so much about these processes and focus on other tasks that matter the most. Therefore, Documate offers complete automation and independency of document management, as a useful service for easing the user's daily life.

The platform started in legal aid, composed of lawyers and engineers that grouped up to understand better what, technology wise, would satisfy the client more and what rights and regulations would be

behind critical documents and contracts. They support non-profits, through partnerships with the American Bar Association and the Legal Services Corporation.

The company started by helping the poor and victims in society which needed legal help to get out of problematic situations. Soon they moved on and realized that their platform would be useful for anything that needed a template, creating PDF and Word forms.

2.2. Blockchain Technology

Blockchain strives, nowadays, as a solution for a world governed by economic, legal, and political systems, in which multiple entities are bonded together through contracts and various organizations perform numerous transactions of currency and digital assets through the world wide web [20]. Net worth as been one of the closest concepts of empowerment, while operations performed that help the digital flux have been the source of that power. Therefore, the blockchain was successfully designed as a mechanism to oversee the economy's digitalization when other solutions could not, and when governs realized the criticality of these processes, recording all of them properly was a priority.

This technology was first openly proposed by Satoshi Nakatomo in 2008 through cryptocurrency by the creation of Bitcoin as a way to securely and in a decentralized form, store transactional data of a virtual currency network [21]. Nevertheless, it was in 1991 that the concept first appeared by the names of Haberr and Stornetta [22], who thought of it as a solution to solve timestamping issues, by registering each data asset's time in a way that was fully proof of tampering and non-repudiation.

Involving multiple ledgers in a distributed network, it is considered a distributed ledger technology in which trustless operations are the key, i.e.: every peer participation in a transaction or action in which timestamping is important enough to be permanently registered, does not have to trust each other for the transaction to stop being secure. Therefore, the parties involved in a transaction, do not have to be trusted or trust each other, as the blockchain uses mechanisms to abstract that part from its users, in one way, using smart contracts as a set of codified rules to validate transactions securely. These will be discussed further ahead.

Nowadays, we have more public projects using public blockchains, but a significant percentage are privately used within higher classified companies or entities that meddle with critical processes and information. These all refer to the permissions configured for the network in which the platform is implemented, declaring which peers should or not take part in the various operations the blockchain performs to maintain its secure properties. Two implementations, which will be referenced further ahead, like MultiChain or Hyperledger Fabric, have developed configurable fine-grained permissions to their blockchain solutions.

Blockchain stores typically metadata for an action in the network associated with an asset of value and places it in blocks, in which each block references the previous block by containing a cryptographic hash of it, creating a growing chain [10]. Hashing is a one-way function producing a unique code for each input, therefore, the probability of collision between hashes of different assets is virtually zero. Each blockchain implementation has its configurations declaring multiple properties of the network such

as block size, consensus algorithm, and more, as each platform has either base standard configurations or some more specific and unique to the that implementation.

As explained, the chain of blocks manages to achieve immutability using hashes. Since being a one-way function, they confer a certain degree of toughness and integrity to information, as it is easy to check if a file was tampered constantly comparing its previously stored hash with a newly calculated one, if they do not match, then someone altered the file's content. In a large scale such as a network with a constantly growing chain this can become particularly useful. Every-time a block's content was altered, and a validation confirms it, the entire chain needs to be updated, but when a new block is added before this can be done, the entirety of the chain's block's content will be protected again. Therefore, the blockchain is resilient to malicious intent if the time it takes to generate a new block is smaller than the estimated time it takes to tamper with the block and revert the blockchain.

The fact of being a distributed network also helps to not only decentralize it but to confer integrity checking mechanisms to the platform, increase responsiveness and therefore, the system's uptime and reliability as it can be configured as fault tolerant and be prepared to work even if something fails or a node is shut down or erroneous. It also promotes the transparency of functionalities to the client.

When an asset is issued onto the chain, it is also broadcasted to every node to keep all peers actualized, while these are constantly mining new blocks for storing metadata or for generating noise blocks, increasing the chain, and increasing the time it takes to revert it. Whenever the data is check, a consensus must take place for the information to be validated and still considered secured. This is done through an algorithm used for nodes to argue the information's validity in numerous ways, using a majority of votes, like the Proof-of-Work (PoW) or Proof-of-Stake (PoS) do, which are the most used consensus mechanisms in the blockchain implementations.

Proof-of-work's concept was invented in 1993 by Cynthia Dwork and Moni Naor to avoid DoS and other malicious intents in email headers by requiring work from a service requester that would compute mathematical equations to prove itself towards the platform [23]. Later, in 1999, Markus Jakobsson and Ari Juels formalized the concept which would be used in Bitcoin [24]. It consists of one node that attempts to validate itself to the other nodes through computational efforts such as calculation problems or puzzles that can be easily verifiable.

When applied to blockchain networks, this consensus algorithm dictates the behavior of nodes towards the platform and but their insertion and relationship with the rest of the network. Nodes that perform tasks using this consensus typically behave as miners which take efforts to constructing the chain, by generating new blocks and securing information in them.

Every node is then under the consensus obligations and must take part on the challenges imposed by the PoW algorithm which can be defaulted or configured to something different, with various levels of difficulty according to the criticality of the information. Therefore, the difficulty level of the challenge is proportional to the amount of time it takes to find the solution to the problem. The node can be asked to try to find a value hidden in a hash, per example, and in this case, the longer the hash's value is the longer it will take to discover the hidden digit and so the harder the challenge is.

We may have the situation where more than one node tries to complete the challenge, taking part in a race to win the rights for mining a new block, in which, it will be registered that the miner took efforts in doing so. The first node(s) that manage to win the race will broadcast their solution while the others validate the answer, also validating the block, which will begin the process of mining that new block, if not, then the block will be discarded. A new block will be accepted when nodes start a new race, and so one, a computationally exhaustive process keeps running and feeding the chain, generating virtual currency as reward for helping the construction and maintenance of the chain.

PoW increases the blockchain's resiliency to attacks because an attacker must control most of the network's hashing power to have an impact on the network and that is why blockchain integrated systems benefit from the distributivity of the network. However, as explained, this method is very demanding for miners and very power consuming therefore multiple nodes tend to group together to join computational power and increase their chances for winning the race, forming what is called, mining pool which may control most of the network, which disrupts the whole concept of decentralization.

Therefore, Proof-of-Stake or PoS was created by Peercoin in 2012 as a solution to all the drawbacks present with PoW, with a similar concept but different methods. Whereas in PoW nodes would need to prove their worth in the form of computational work, with PoS nodes have in their pose a stake of the network and use that to gamble between them using tokens in the hopes of being selected as righteous miners for the new block.

Nevertheless, both consensuses eventually decentralize the network, as with PoS, the richest node increases its chances when gambling for the block, whereas using PoW, a more expensive system or server, with more computational power, also reduces the time of answer for the challenge. Therefore, these two consensus algorithms, although still often used today, do have major issues compared with more recent proof methods used today [25].

Blockchain also possesses financial attributes, as discussed, considering that most networks using this technology have their own virtual currency which, as explained, is used for holding patrimony over the network and is rewarded by working towards the blockchain's growth. Transactional data is registered in the network permanently conferring non-repudiation for every action. Although important and useful for applications that work in this market sector, it is not an interesting feature for this project and, therefore, we do not focus on this blockchain's property.

2.2.1. Blockchain Platforms

This chapter discusses the technologies studied for this thesis deeper understanding of the blockchain's functionalities and benefits while taking account for fresh solutions that might improve upon others or not.

In total, six implementations are presented in the further subchapters: Bitcoin, Ethereum, NXT, Hyperledger Fabric and MultiChain. Although two are very rudimentary and could be ruled out for improper purpose and uselessness to the project's integration, they are still important for comparing different metrics and properties between all of study involved platforms. All of them contributed to the construction of the right project's mindset.

At the end of the section a table will be presented, resuming all the different blockchains benefits and metrics for specific desired fields.

2.2.1.1. BitCoin

Nakamoto, responsible for the creation of blockchain, as discussed above, focused on integrating financial networks with trustless and distributive properties, and since ever, these have been the grands pillars of blockchain platforms. This implementation aimed to secure digital currency transactions without the need for a third party and total independence from trusted peers, while mitigating the double-spending problem.

BitCoin [21] was the first public implementation of the original conceptualization of timestamping every event, originally created by Haberr and Stornetta. This method was also the solution Nakamoto implemented to counter the double-spending problem. Nevertheless, it is a very simplified form of this technology, which is already outdated by its successors. This technology behaves identically to the properties we described above, as a pillar to the blockchain community, it is expectable that it possesses the basic functionalities a blockchain platform would.

The implementation uses PoW and relies solemnly on miners, which, as discussed, tends to decentralize the network, and reflects on later issues, such as large costs in energy consumption and hardware assembly. These problems are recurrent in older technologies, that in time of conception did not have the same values as newer platforms started appearing and claiming their place in the podium. Nevertheless, still a big reference to the world of financial digitalization and distributivity.

Although still extremely popular, BitCoin is considered a close economy, as it is increasingly difficult to earn power in this network due to the stage of progression the network already is, with already approximately 90% of the total system's currency mined and distributed, which was capped by its creator at 21 million.

The next platform is a recent technology that attempts to mitigate the issues related to BitCoin, but also varies in purpose, focusing more on integration with existing applications conferring them distributivity and reliability.

2.2.1.2. Ethereum

Today this is one of the most used blockchain technologies as it has various functionalities to which blockchain platforms take great benefit from. It was founded in 2015 by Vitalik Buterin to help with the distributivity of applications on the blockchain using a versatile and simple architecture accessible by anyone. Its platform is one of the most powerful compared with concurrent technology as it uses a Turing-complete virtual machine, increasing overall performance in computation and operation security. It uses the Proof-of-Work consensus algorithm [26].

Ethereum is a decentralized open-source, public blockchain featuring the original concept of smart contracts as their most important functionality. It highly focuses on application's decentralization, while trying to apply all the uniquely powerful properties to the system's infrastructure and function, while

keeping transparency to the user. It supports applications, called DApps, that run on the Ethereum blockchain in a decentralized environment taking advantage of the blockchain as a database.

As referred, Ethereum implements Smart Contracts which are sets of codified conditions that run on the blockchain itself and mimic the behavior of a normal contract. For that, the platform uses Solidity, a contract-oriented language developed specifically to write smart contracts. These work by defining rules as intermediary secure measures before registering anything onto the chain. Any asset must conform to those rules and only then is officially issued and secured.

Smart contracts offer trustless distribution to the platform because peers do not have to worry about another's legitimacy, the contract will do that for them. All of this happens in a deterministically as the behavior must always be the same despite issues in the network, therefore, smart contracts are extra secure measures that can be particularly useful.

However, Ethereum has its own cryptocurrency, and it is an expensive technology to implement to larger infrastructure, due to the power needed to keep the Ethereum VMs up and running on multiple nodes and with the computational specifications they have, but at the same time weights with the potentiality of the platform to be an effective and reliable system.

2.2.1.3. NXT

NXT is a public blockchain which also uses smart contracts such as the previously discussed technology and like bitcoin, has a huge basis in cryptocurrency in a payment network as a service. It combines both purposes of Ethereum and Bitcoin as it was specifically conceived as a platform to build applications and financial services [27].

Launched on Sept. 28, 2013, by BCNext, an anonymous developer, as opensource on a forum called Bitcointalk.org with a total and initial stake of one billion coins as second-generation cryptocurrency. Initially, BCNext requested insignificant donations, fundraising the platform for 3-week period and then determine how distribute the stake between ledgers.

NXT uses the Proof of Stake model instead of Proof of Work due to the decentralization that happens with the latter. This choice was supported because with the growth of the network, the reward is split between more peers which causes them to lose interest in mining.

Therefore, the technology implements a new model which considers each coin in an account as a tiny mining rig, so the more tokens the user has, the bigger the stake of that node and the chance to be rewarded the block generation. When the node manages to earn the block generation, it will be awarded the sum of the transaction fees stored on that block. NXT does not generate new tokens because of block generation. These tokens are only redistributed by the block creators receiving transaction fees.

Furthermore, while mitigating the drawbacks imposed by PoW algorithm, NXT ended up developing a similar technology to smart contracts, so, in 2015 a company called Farla Webmedia launched the project, NXT Asset Exchange and a half year later they would launch Smart Transaction templates, a close implementation of Smart Contracts. Similar in purpose but limited in flexibility. The smart contracts implemented by NXT, or so called, Smart Transaction templates, are extremely limited and rudimentary,

never achieving a desirable or even accepted level of complexity and integrability and therefore, the platform's integrations are specific.

2.2.1.4. Hyperledger Fabric

Originated by Linux Foundation and IMB strongly advised and endorsed by Intel, Hyperledger Fabric, is a great open-source solution for blockchain-based distributed ledgers. A few months later in early 2016, the project started accepting proposals for integrating other technologies as core elements and incubation of codebases which then originated Fabric as one of the first solutions by combining work by Digital Asset, Blockstream's libconsensus and IBM's OpenBlockchain [28].

Fabric focuses on distributed applications for digital data storage for, per example, institution records, without the need of a central administrator for the whole purpose of improving performance, reliability, and security of these systems. It excels in interoperability with existing platforms and supports JavaScript as programming language oriented for web applications.

It is a permissioned blockchain designed for enterprises with the particularity of having nodes with distinct roles in the network and completely different tasks for better organization of the infrastructure. There are plentiful of tasks involved in the ongoing and exhaustive processes of the chain generation and maintenance, so, by splitting these through different nodes, Fabric achieves another level of complexity and increases scalability. There are nodes for broadcasting and keeping the network updated, nodes for mining and nodes for registering the network's events [29].

Another big feature of Fabric is the fact it supports Smart Contracts with the same level of efficiency as Ethereum and calls it chaincode. Furthermore, it has the capability for multiple consensus algorithms which are configurable [30]. Additionally, there is no need for a virtual currency reducing cost of implementation, operation, and maintenance.

Fabric often uses Practical Byzantine Fault Tolerance consensus algorithm, a great solution for malicious exterior intents to tamper with data and software errors caused by data inconsistency and network issues [31]. This algorithm makes sure every peer in the network is constantly broadcasting and acknowledging every action and every transaction processed and registered, therefore, the whole flux of the network is constantly being questioned for constant updating, assuring non-repudiation, distributivity and integrity. Consensus algorithms from this family have proven to be dependable over the years.

There are already multiple tools helping with the deployment of blockchain Fabric-based applications like Hyperledger Composer or Hyperledger Explorer. The first tool is a collection of tools for building blockchain networks in larger scale for business purposes, allowing blockchain applications through a graphical user interface called Playground. Hyperledger Explorer is a blockchain module hosted by Linux Foundation, and it focuses on the easy creation and deployment of user-friendly web. These two tools combine, have the capabilities of developing complete blockchain-based applications.

The community forums and teams on the Hyperledger domain are heavily populated with information, case studies, and documentation that help understand the different scenarios in which this technology may be applied and constantly updating the state-of-the-art for the platform with various market sectors.

Nevertheless, Fabric can be difficult to learn and use. The level of complexity and efficiency achieved by this technology is high, but in the end, we want a transparent, easy to learn and easy to maintain solution.

2.2.1.5. MultiChain

MultiChain is also an open-source platform for easy creation and deployment of private blockchain applications supporting multiple operating systems. It has Bitcoin as its base but with the purpose to reduce the issues about participants' permissions over the network. MultiChain tries to mitigate how decentralization problems, such as the ones presented above, affect the entire network by letting peers overpower it. The particular aspect that stands out with this technology is easy configurable fine-grained control of permissions for the participants with easy maintenance and monitorization of such measures [32].

MultiChain is quite easy to use and develop with. Uses a simple API and command-line interface and communicates with the backend server using JSON-RPC commands via HTTP from an executable client. The technology possesses a vast variety of commands like permission granting, mining configuration, requests for network, asset and block information, consensus re-run, and more. All of these are properly documented in the MultiChain's web page.

This blockchain implementation does not have a native cryptocurrency and does not force the user to. Supports multi-currency and creation of virtual fictitious tokens for private simulations or enterprise encapsulation of digital currency. The most important of all, the use of data streams to store data with greater sizes without weighting too much on the blockchain's performance and not strictly focusing on currency transactions. Moreover, it is highly configurable for easy personalization of the network on permissions, block size and more, being adaptive of the environment it is deployed to.

The most salient key feature of MultiChain is data streams. Data streams behave like private channels in which the blockchain writes information to and can be configured along with the peer's permissions for a finer accessibility. In a way, this feature partitions the blockchain into segments that each can be associated with an ID, but overall, the information will still be stored in the same chain, as this serves more as a tag for better organization of the information and faster retrieval. Nodes of the network may subscribe to streams to participate in operations related to that stream such as block mining, read, or write to that stream, as referred, taking advantage of the permission features of this technology.

Unfortunately, MultiChain does not implement smart contracts but instead has also a particular feature they produced called smart filters. The developers felt it was unnecessary to have code running on top of the blockchain, weighing the system's performance and impacting the overall efficiency and reliability. Therefore, smart filters are less complex than smart contracts but the purpose of such are similar, but instead of influencing the network directly, it behaves as a simple piece of code for validating the assets before their issuance which is invoked whenever something is about to be registered onto the blockchain. This can be useful in terms of scalability, as the chain grows it will take longer and longer to re-validate and check the conformity of the information.

Smart filters are written in JavaScript's language and can be associated with a data stream as a fine-grained sifter, filtering the JSON assets that are published, by checking their fields and performing proper evaluation of the information. Data assets are published to the stream in JSON format with desired fields holding the data asset's metadata or specific fields of interest to the platform.

Multichain uses a distributed consensus like Practical Byzantine Fault Tolerance, like Fabric. However, whereas in the latter, we had multiple validators per block, in MultiChain's approach, there is only one validator per block, working in a round-robin strategy. These validators are picked according to the initial configurations. If one fails in a round-robin type scheme, the system performs a re-round picking a new validator that may even step in from outside of the initial round-robin rotation and become part of the new round set. If not, then there is a fork in the system resolved by the validator picked in the next rotation.

Everything is properly documented on their website, thoroughly explained for developers and MultiChain provides the executables and configuration files necessary for fast deployment of the blockchain network. Furthermore, the community forums are continually active and seeded with FAQs, latest updates discussions, library integrations and resolution for common problems.

2.2.2. Discussion

Table 1 shows a comparison between each technology according to various aspects, as follows:

- Accessibility: whether it is public.
- If it possesses smart contract technology.
- Support for native digital currency.
- Consensus algorithm often used.
- Block size limit.
- Hash algorithm often used.
- Programming language often used.

It is important to refer that the key features in choosing a good supporting technology for qDocs are the transparency to the user while maintaining a proper functionality; easy integration; and ease of maintenance. As more platforms arise in popularity it is important to argue which is best to integrate for a desired application and which purposes does that blockchain has for improving its functionalities.

The aspects present in **Table 1** refer the accessibility of the platform, the existence of smart contracts, the obligation for a native currency, the consensus used and other features, mainly the programming language used, which plays an important role when integrating with existing systems.

We consider these aspects, weighting the properties of each technology against one another. Nevertheless, they are either expensive to implement like Ethereum, or too simple, like Bitcoin and NXT. Most of these technologies support the implementation of smart contracts, and while some do not, others, alike MultiChain, implement similar solutions.

Table 1 *- Comparison of the researched blockchain technologies*

	BitCoin	Ethereum	NXT	Fabric	MultiChain
Accessibility	Public	Public	Public	Fine-grained permissions	Configurable permissions
Smart Contracts	No	Yes	Limited	Yes	No
Currency	Yes	Yes	Yes	No	Configurable
Consensus	PoW	PoW, PoS	PoS	Multiple	Round-robin
Block size limit	1MB	Unlimited	42KB	Configurable	Configurable
Hash algorithm	SHA-256	SHA-256	Curve25519	SHA3	SHA-256
Languages	C++	Solidity	JavaScript	Go, Java, JavaScript	JavaScript

We also studied private blockchains with promising features that are of interest to be integrated with qDocs, like permissioned blockchains for both Fabric and MultiChain, or high performance and complexity achieved by using Fabric's solution, or in the case of MultiChain, easy to learn, use and integrate.

A study was conducted comparing Ethereum, Fabric and MultiChain uses for the healthcare sector, about issuing medical recipes and bills [33]. The study approaches these three platforms while considering their learning difficulty, complexity of solution, performance, easy development, and transparency, and MultiChain excelled at some of these aspects despite being the single one not implementing smart contracts as its competitors.

In that the study there was a significant difference in security and complexity, being MultiChain the one implementing weaker mechanisms for lacking the technology of smart contracts, but despite of that, it was secure enough for the matter. The Multichain is still very recent in development for achieving the same stages of its competitors but has interesting features, such as smart filters and data streams, that the latter do not and might be of benefit to specific projects, in this case, qDocs.

The next topic discusses the state-of-the-art of integrating document automation and blockchain technologies, with examples of projects that notoriously have been implementing the same concept discussed throughout this research.

3. Related Work

Document automation and blockchain have had increasing popularity in recent years, being the latter widely used. Although the objectives of integrating both are not new, there are not many projects combining these technologies, or they are privately confined solutions, residing in a close environment within legal firms or university campus, helping reduce time wasted in document related processes while not losing their sense of criticality.

Legal firms and government authorities work with substantial amounts of documentation which, when in physical form, occupy a great amount of space, hence the preference for major digitalization and consequently proper security measures. Like any paper document that needs to be securely stored in a drawer after signed, similarly these assets must be secured when converted to a digital format. The solution often adopted is to secure them in a blockchain-based network keeping a timestamp for each event document related.

Smart contracts are useful in this context, since document related processes rely on more than one entity, obliging the participants to be in conformity with the document's specifications and jurisdictions, such as digital signing or paying a transaction for the document's closure.

Although blockchain was not meticulously designed or focused entirely on storage but on currency/asset transactions, it was always considered a secure way to store immutable data and serve as a counter to tampering of information attacks and memory conflicts.

In the next sections, we present projects that combine these technologies with the same purpose as qDocs, offering service reliability and tampering-proof properties to a document automation solution.

3.1. Open Law and Reuters

Thomson Reuters Corp and Open Law cooperation is, respectively the integration of a legal document automation platform and a deployable smart contract solution as a service.

Thomson Reuters Corp. is a media company that provides information for professional markets in five different sectors: corporations, government, legal services, news and media, professional services, and accounting [34].

OpenLaw is a provider of a smart contract-driven solution for convenient integration with existing platforms that wish to adopt blockchain to secure their operations.

Reuters often manage critical and important documents from their legal sector, therefore, they decided to add security to their processes, hence this integration. The solution was to deploy a blockchain, driven by smart contracts, by merging OpenLaw's solution for decentralization of legal agreements and Contract Express, a document automation tool for producing legal documents from Thomson Reuters. This was achieved by translating the contracts from Contract Express into smart contract code, assuring more security and efficiency to their operations.

This integration used Ethereum's blockchain with a different protocol developed by OpenLaw, proving that smart contracts would be easily incorporated within their legal document templates. An important

matter was to do this transparently and in a way that so their users would learn how to use the system without difficulty.

3.2. Azure's e-Document Verification with Smart Contracts

Azure created a development kit to work with blockchain platforms and integrate them onto ongoing projects [35]. The goal is to sign and store digital codified documents in a transparent and distributed manor. Equally to the design purpose of blockchain, every event, agreement, process, task, and every payment is timestamped and securely stored to be further validated in every access [36].

Microsoft provides serverless technology to maximize efficiency and reliability in a distributed network designed for enterprise use. Alike qDocs, Azure's solution concept describes a use case student related, about a certificate requested by a student and issued by an authority, in which the digital content of the certificate and its metadata are hashed and stored in the blockchain producing a unique token. This action is easily validated by the token, conferring non-repudiation and authenticity to the service.

The development kit is designed to be easy to use, deploy and integrate with applications with the premise that there is no need for a central authority or server to mediate every process. This feature helps understand better qDocs moto of operations and analyze the case of signing documents in the blockchain.

Lastly, this solution is open for contributors under the agreement of a license, assuring solution continuity, sustainability, and maintenance.

3.3. ActiveDocs

ActiveDocs is a document automation benefits highly from blockchain-based technologies. They use smart contract features, such as codified agreements to potentially help the secure and reliable creation and management of legal documents [37]. They also explain that this technology brings conformity and standardization to agreement processes, depending on the globalization of technology.

The platform believes that despite of document automation being already highly efficient and scalable, the execution of a contract or agreement still depend too much on trust between participants and long authority processes. Therefore, blockchain is ideal to mitigate these obstacles in peer interaction by removing the trust factor and the worry of its existence, relying only on the good functioning of the blockchain-based network and its features.

Furthermore, ActiveDocs' acknowledges that this solution should be spread out through wider groups, gathering contributors, and trying to satisfy more people. Therefore, they constantly follow up this technology's state-of-the-art and growth with the purpose of bringing document automation features and blockchain security measures to their clients wishing to adopt digitalization.

4. qChain Architecture

This chapter details the qChain architecture. We considered qDocs general requirements to facilitate the integration and implementation of qChain and rigorously evaluate the solution. Nevertheless, we came across two challenges when implementing the project's infrastructure.

The first challenge was choosing where the blockchain network should reside inside of the existing structure. Considering that qChain serves as an intermediary to document integrity and conformity validation, it does not make sense to implement it between the curators and the qBox. Since qChain is managed by the same machine or set of machines running the qDocs server, then the qChain should be deployed and operating between qBox and the qDocs server, registering and timestamping new entries on the blockchain before the documents are stored in the curator's database and validating the information's integrity for those documents.

The second challenge was how the MultiChain would be deployed and how would the server communicate with it. We chose to deploy the blockchain encapsulated in two Docker environments, one for the documents with multiple stages of creation and another for documents deemed as closed. Documents with more than one stage This setup was preferred because MultiChain does not implement smart contracts, and although it has a similar technology, we thought would be more secure if we could register when, per example, a document must be sign by multiple peers, asynchronously, and every time it does, we wish to register the event. This will be discussed in more detail in the **Section 4.1**.

More technical challenges will be also answered in **Chapter 5.**.

4.1. qDocs Requirements

qDocs had three requirements that were taken in consideration when we chose the blockchain technology.

First, the implementation should be easily maintainable with ongoing updates and developers and community support. Fortunately, MultiChain is heavily documented and has already a strong community filled with integrations and related common issues.

Second, we must assure that the implementation would be transparent to the end-user, since qDocs is a citizen-centered platform that should be reliable and efficient. Therefore, the normal use of the platform must not be interrupted, and the end-user does not have to know much about the backend functionalities of the integration. End-users will surely be notified when documents have extra security measures without being aware what qChain is doing in detail.

Third, this solution also considers the learning difficulty for the technology used, in this case, MultiChain within docker capsules. In some case studies researched, MultiChain excelled at these characteristics as one of the easiest blockchain technologies to understand, learn, and develop compared with technologies like Ethereum and Fabric [33]. The platform is easy to deploy in many environments, being adaptable to programming languages with light wrappers and integration libraries. Although the communication is done using RPC requests in JSON format via HTTP, there are already

multiple wrappers in various programming languages that manage to translate MultiChain commands using direct calls to the blockchain's API. Therefore, we use an existing wrapper for communicating with the blockchain via HTTP in C#, since the main server of qDocs was developed using that programming language and the method calls for interactions with the blockchain are performed through the main server.

The next section describes the structural specifications for the qChain integration with qDocs.

4.2. System Infraestructure

The system's architecture is observed in **Figure 4** below**,** representing two blockchains for the entire qDocs server: (1) one provisional for unfinished documents with multiple stages of creation such as multiple participants filling some components and signing, producing more than one version of the doucment due to asynchronity, and (2) a final blockchain for documents that are stored after automatic approval from the issuing authority or that have been already completely filled with all conditions and signatures met.

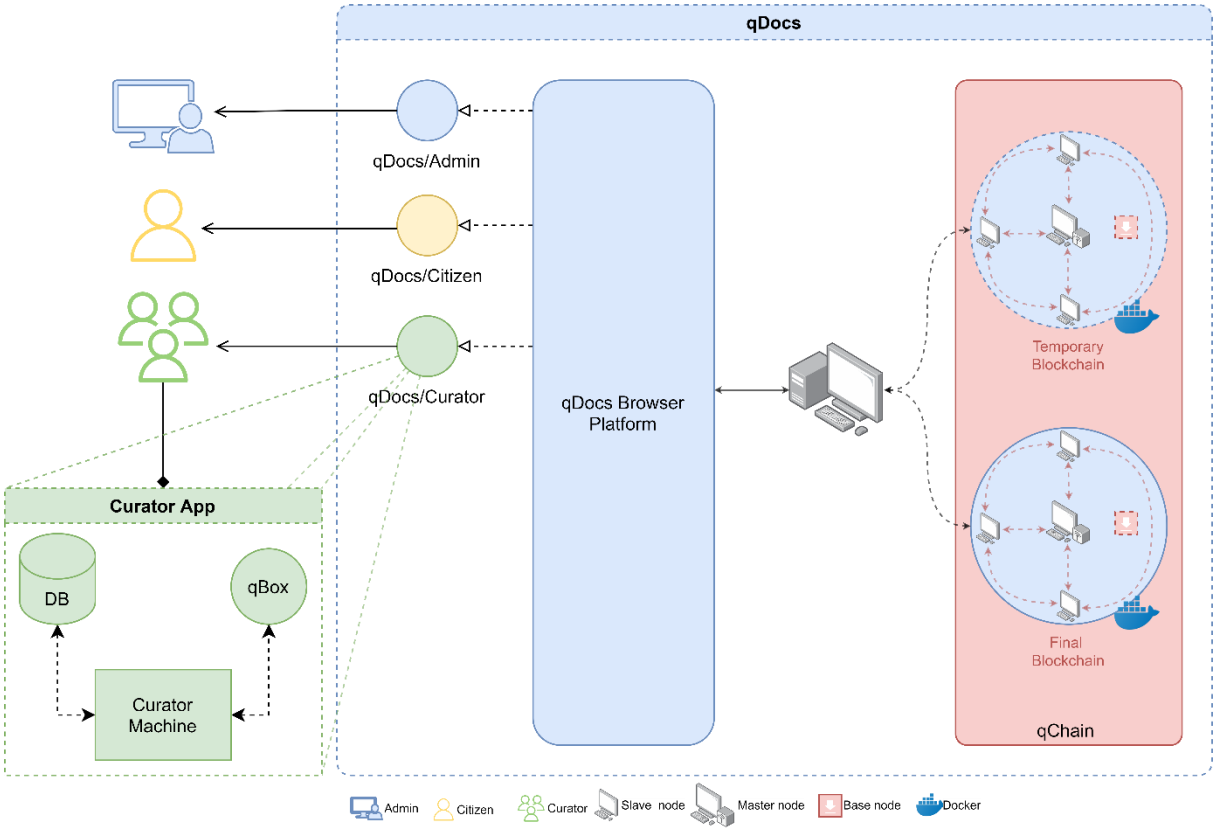

Figure 4 - *qDocs/qChain Infrastructure Architecture*

As explained, this choice was based on the lack of smart contract technology from MultiChain, because although Multichain has a similar, but simpler, technology, called smart filters, we decided that having a blockchain for registering intermmidiary stages of the document's full generation, was important, since we can't control the document's flow and trustless operations from multiple participants with the finesse smart contracts do.

Smart contracts would hold the document in a latent state waiting for codified conditions within the document's structure to be met by all participants with no need for a trusted parthy or for them to trust one another, and only then would the document's creation be registered in the blockchain. Therefore, since Multichain does not implement such complex mechanism, we decided to register each time an operation is performed in each document, until the document is complete, ensuring integrity, authenticity and reliability to the document from creation, to signing and finally, completion.

As illustrated in **Figure 4**, each of these blockchains has four nodes considering the Byzantine Fault Tolerance model [31], which was already discussed above in **Section 2.2**. Ruling out the base node which has no operational impact whatsoever in the platform's functions, having four nodes achieves the minimal number of nodes required for mitigating byzantine failures in the network. Byzantine Fault Tolerance systems admit that there must be N nodes, where ***N = 3f +1***, in which is admitted a maximum of ***f*** faulty nodes.

qDocs's main functionalities are preserved, the platform will be only responsible for forwarding to the MultiChain all requests that rely in the document management domain, using the wrapper's methods. These requests are the following: creation of a new curator/organization triggers the creation of a new data stream identified by the respective curator's name; creation, modification, and signing of a document triggers new publications in qChain; document access and visualization requests qChain for the document's hash stored for integrity validation.

4.3. Summary

The network deployment and distribution had to heavily consider the communication flow between each component to correctly position each module in its righteous place, therefore, it was the right decision to let the blockchain-based system, or qChain, reside as an optional feature to be used, when requested by the curator, as a security extension of qDocs.

Therefore, **Figure 2** can be changed into a more complex diagram, resulting in below **Figure 5**, which better describes the use of qChain, its architecture and the flow of its inner components.

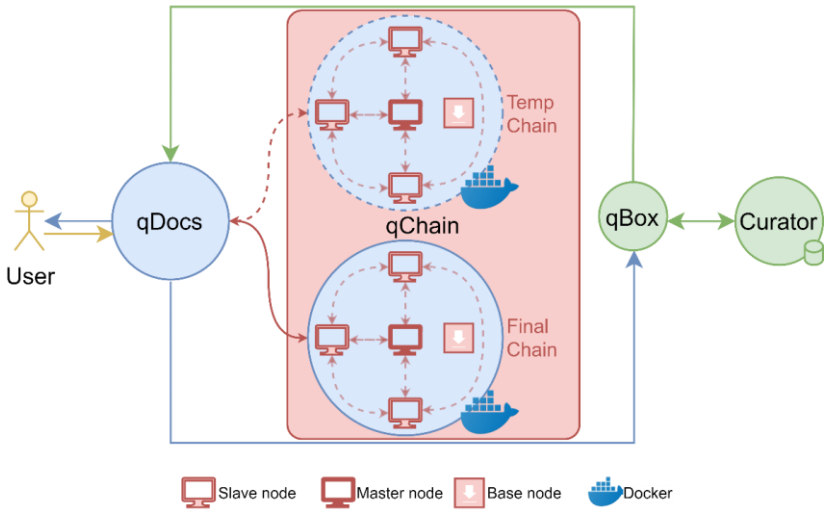

Figure 5 *- Detailed Operation Flow Diagram of qDocs/qChain Infrastructure*

We can observe the communication between qChain as service that does not operate all the time but on demand by the curators' specifications in the template.

The next chapter details the implementation of the blockchain as a service and as a software complement to the platform, displaying the functional and graphical changes to the platform.

5. qChain Implementation

This chapter discusses the in-depth configurations and specifications for the blockchain network as well as the backend communication with the qDocs server. We also refer the modifications made with the existing code of the platform, what behaviors were changed and what was observable in the old structure that might have been optimized in the new one.

5.1. Development Environment

qChain was encapsulated within docker images, in which there are two containers each with four work nodes: a master node which was the first to be set up and to which the others connect to, three slave nodes responsible for block mining and management. Additionally, there is a fifth node which only performs an initial setup, download and installation of the MultiChain libraries. In total, all four operating nodes manage the network, performing mining operations and integrity validations.

The development environment is implemented in an uncomplicated way due to resource accountability and simulation, since the qDocs server is not openly in production. The development and further testing of the platform's implementation were conducted in a home computer after proper tool and libraries installation.

Visual Studio was used as a text editor to adjust code to better fulfil the qDocs requirements and the communication with the platform, therefore, we did not introduce major changes in the code, at most, we deployed method calls for the functions present in the wrapper's library. We only altered a minimal number of lines, adding fields to further complement and persist the database implementation. These changes will be further detailed in the next sections.

Every process of qDocs infrastructure was initiated and tailed through MobaXterm, a toolbox which lets us open multiple command lines in parallel and monitor their status while the data was persisted in a home POSTGRESQL database from pgAdmin's interface. As referred, since qDocs is not actually in full production, the development environment was simulation-based.

The next section briefly discusses the different modules that were present in the solution and the additional ones implemented in this research.

5.2. Modules

The qDocs solution has multiple modules, that successfully communicate between. Therefore, it is essential that each module has the capability to commit to the system's purpose and interoperate without any internal individual failures or network connectivity issues.

Before the integration, the system had three key modules suggested in **Figure 5** as follows: (1) qDocs server. (2) qBox, and (3) Curator and respective database. Altogether, there are three roles, the Curator which is responsible for issuing and officializing the documents and creating templates for new

ones, the Administrator that manages all configurations in the qDocs server and, lastly the Citizen that will rely on the platform's operations.

We added two more modules which rely in the middle of the infrastructure: two docker containers each of five nodes, containing (1) a provisional blockchain and (2) a final blockchain. Each docker container, as already explained, has only four workable nodes that communicate with the rest of the network while the fifth node only serves to install the MultiChain's libraries. These nodes start the network by mining a fixed size of blocks generating a bigger and stronger chain as an initial security measure.

Every module communicates using HTTP, even the MultiChain does so with the wrapper that performs the JSON RPC calls. Considering the development environment is close to a simulation, it facilitated the encapsulation of each module and further, easy access to each logging and debugging that had to be done. The communication between the qDocs server and the nodes is done through a wrapper is a community library developed by the GitHub user JonathanCrossland as an open-source implementation.

When a there are no specifications for the use of qChain's validation, qDocs will only use qBox for document retrieval. It is important to clarify that qBox will always be used to assist document fetching directly from the curator's database and that qChain will only store tokens representing key events of each document, such as creation, signing, closure and sharing. Each time there is a change to a document, an updated version will be produced and therefore stored safely in qChain if that was deemed by the curator.

Whereas not closed documents are stored in a provisional blockchain, closed documents are stored in a final blockchain, in which the only updates the document may have been the last accessible date and sharing with another citizen. Consequentially, **Figure 6** below considers this condition that further increments the complexity of qDocs/qChain's integration.

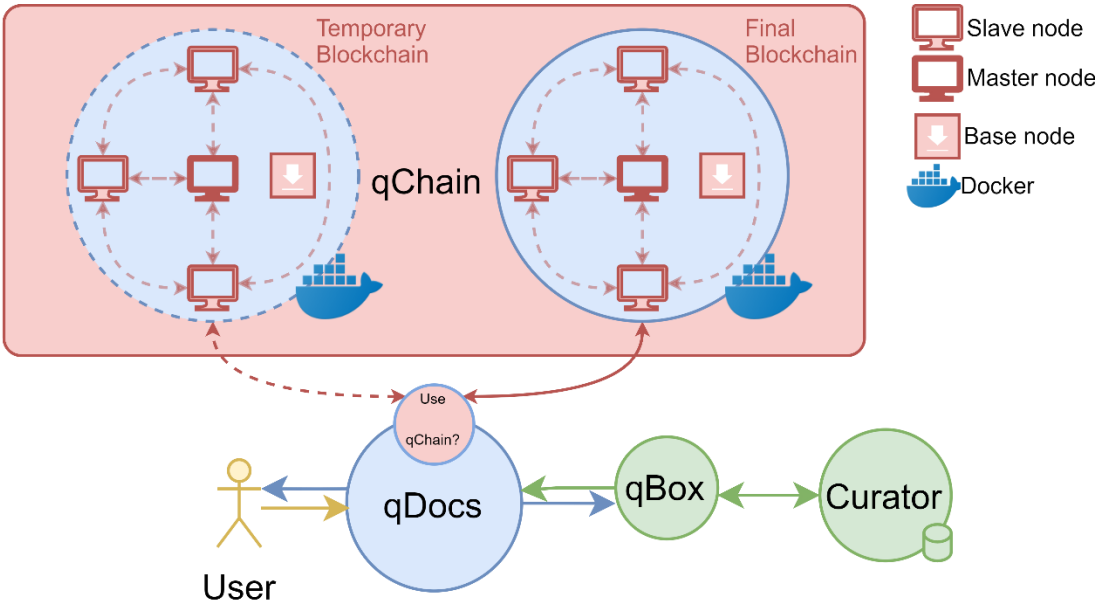

Figure 6 *– Final Flow Diagram of Operations for qDocs/qChain Integration*

5.3. System Functionality and Specifications

This subchapter details the system's operations with MultiChain's architecture along with screen shots, diagrams and examples of each process and stage of document creation and access.

Firstly, the document's hash is only stored on qChain if the curator, when designing the respective template for its creation, deems that this should have additional security measures provided by the qChain.

This is requested to the curator at first glance when choosing the template's name. The curator is presented with a checkbox along with the words qChain Protection, and for this matter of fact, the presentation file for the first page of document template creation was changed to include the checkbox along with the DocumentTemplate's class in which a new variable of Boolean nature was added, called MultiChain. This is all then persisted in the database as a new column named *MultiChain* in the DocumentTemplate's table in the database.

The new interface was slightly modified to include a simple checkbox under the name qChain Protection which is not mandatory to be filled and does not harmfully impact the curator's experience with the platform. The visual representation of the checkbox for storing the document in qChain during template creation is presented in **Figure 7**.

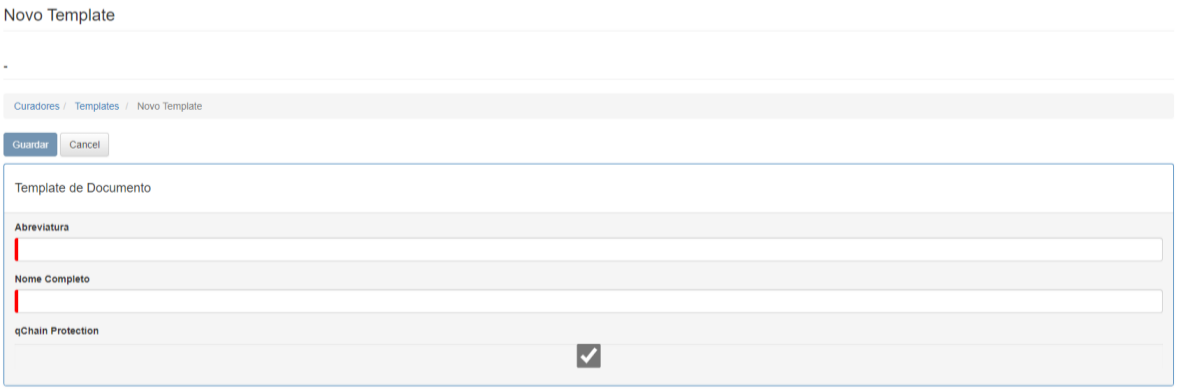

Figure 7 *- Visual Presentation of the new UI for Document Template Creation*

Lastly, the checkbox is referenced by a Boolean variable which is persisted in the document's template class and further added to the database as a new entry with the column *MultiChain* with the value *true*. The modification of the DocumentTemplate class is presented in **Figure 8**.

```
namespace dDocs.Models.TemplateTypes
{
    [Table("Document")]
    public class Document : DDRemovableEntity
    {
        public DocumentStatus Status { get; set; }

        [ForeignKey("DocumentTemplate")]
        public long DocumentTemplateId { get; set; }
        public DocumentTemplate DocumentTemplate { get; set; }

        [ForeignKey("ApplicationUser")]
        public string CCNumber { get; set; }
        public ApplicationUser ApplicationUser { get; set; }
        public bool Favorite { get; set; }
        public Guid? guid { get; set; }
        public DateTime CreationDate { get; set; }
        public DateTime CloseDate { get; set; }
        public DateTime LastAccessedDate { get; set; }
        public ICollection<SharedDocument> SharedDocuments { get; set; }
        public String qChainToken { get; set;}
    }
}
```

Figure 8 *- Checkbox Value Persistence as a Boolean in DocumentTemplate Class*

The platform is responsible for only letting the document be created if all fields are filled properly. If the curator deems that document should be stored on qChain via the indication on the template the process, on this point on time, will start being parallelly followed by qChain. Furthermore, if the document needs more than one stage of creation, such as awaiting confirmation or signature from one or more participants, then it will not be stored in a final blockchain, but instead, in a provisional instance where documents with intermediate processes will be secured. Not needing one, the document's metadata is stored in the final blockchain, which, as explained, is meant to store documents that do not need further operations.

Secondly, we implemented one of the MultiChain's features already referred in the Background chapter called Streams. These reside in the curator, having each curator a stream identified by their full name and referenced by their abbreviature. Streams serve as channels in which documents will be attached too, like tags, and facilitate the retrieval process for documents.

Furthermore, it is here that another feature, smart filters, is deployed, as gates at the entrance of the blockchain, validating the entry's fields and specifications. For now, no smart filters will be developed, because these should be adapted to the curators' needs and therefore, similarly to them being responsible for designing the template, it is also the curator's responsibility to negotiate with the qDocs development team, the desired verifications for documents using that template.

The smart filters are written in JavaScript and are quite simple, behaving in a similar fashion to smart contracts. This service provided by qDocs team will be provided on demand to the curator. When creating a stream channel, the aggregated process will produce a token identifying the stream, which is then used to connect that stream and retrieve its items and their metadata. This token was also added to the Organization class as a string and further added to the respective table in the database as a new column.

The changes to the Organization class are presented in the **Figure 9** below.

```
namespace dDocs.Models.DataSources
{
    [Table("Organization")]
    public class Organization : DDRemovableEntity
    {
        public Organization()
        {
            Domains = new List<Domain>();
            DocumentTypes = new List<DocumentType>();
            DataServices = new List<DataService>();
            FormObjectGroups = new List<FormObjectGroup>();
            Snippets = new List<Snippet>();
        }

        [Required]
        [StringLength(300)]
        public string ShortName { get; set; }

        [Required]
        [StringLength(300)]
        public string FullName { get; set; }

        public string StreamToken { get; set; }

        public OrganizationType Type { get; set; }
```

Figure 9 *- Stream's Identifier Addition to Curator's Organization*

Every time a new curator representing an authority or organization is added to the platform, a new stream is also created in the qChain's MultiChain with the already discussed details and identifications.

Lastly, we altered the Document and Persistency class. The Persistency class holds all methods for operations of the platform that will eventually persist information in the database. When a new document is created, there is not a direct call to the qChain, because the template for that document is only associated with the document after its creation, therefore, in the process of document creation, after the template filling, the document is presented correctly filled to the user, and that is when it is secured to the qChain, in the method call GetDocument before the document has any token associated with it. In this class we implemented the calls for creating streams amid the creation of new organizations, new entries after new documents are filled and updates prior to every document being created, along with integrity checking for each retrieval. A snippet of the code is presented in the **Figure 10** below.

```
// Document's first interaction with qChain after creation
if (document.qChainToken.Equals(null))
{
    if (document.DocumentTemplate.MultiChain)
    {
        byte[] value = Encoding.ASCII.GetBytes(document.GetHashCode().ToString());
        Task.Run(async () =>
        {
            String documentName = document.CCNumber.ToString() + "_" + document.DocumentTemplate.DocumentType.ShortName.ToString() +
            "_" + document.DocumentTemplate.DocumentType.Organization.ShortName.ToString() + "_" + document.CreationDate.ToShortDateString() +
            "_" + document.LastAccessedDate.ToShortDateString();
            response = await _Client_Temp.Stream.PublishAsync(document.DocumentTemplate.DocumentType.Organization.StreamToken.ToString(), documentName, value);
            document.qChainToken = (JsonConvert.DeserializeObject(response.Result)).txid;
            Console.WriteLine(response.Result);
        }).GetAwaiter().GetResult();
    }
}
```

Figure 10 *- Document's Persistence in qChain after initial Creation*

Furthermore, **Figure 11** represents the logical process of creating a new document in qDocs, in which this document does not need more layers of creation and its hash is immediately stored in qChain and secured. The operation waits for approval from the qChain's task of creation and produces a result in JSON format with a token. This token is the registration of the event and is added to the document's

class and persisted in the database process of creating a document, that, in this case, does not need any signature, using qChain's blockchain to store it.

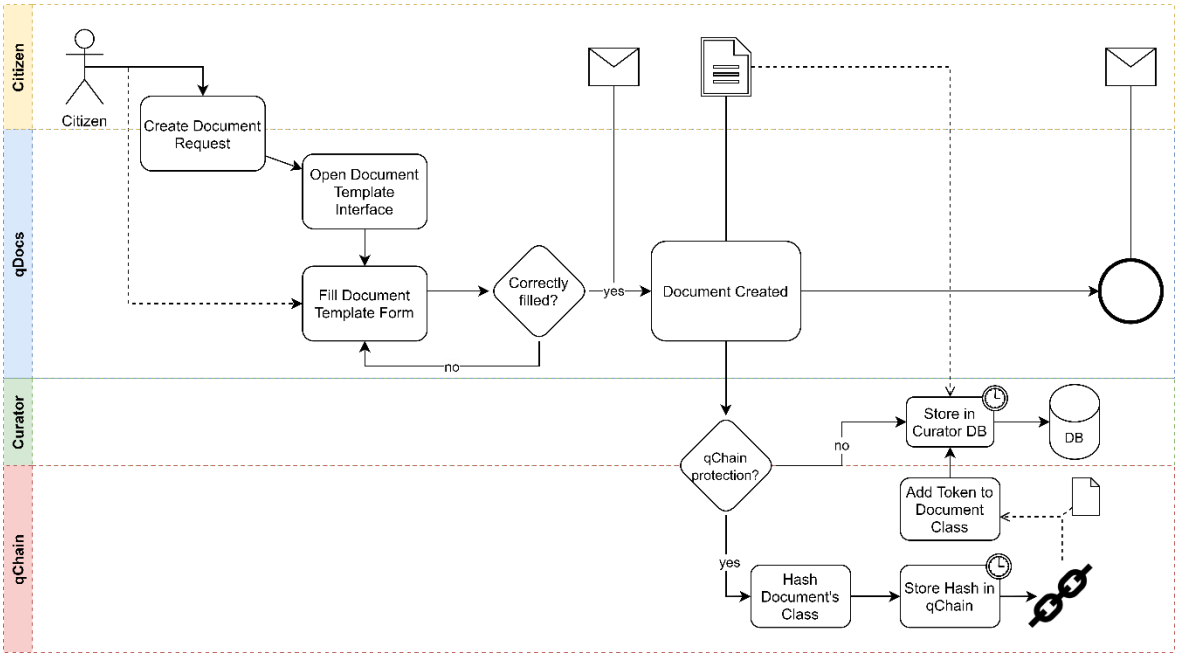

Figure 11 *– Business Process of Creating a Document in qDocs*

The document is stored under an alias with the following format *CCNum_DocTypeAbb_OrgAbb_CreaDate_LastAccDate*[1], which is not important because the document's hash will be retrieved from qChain using the token generated amid its creation and the stream token of the respective curator.

It is also in this method that the document's integrity verification is performed, by fetching the document, also through a wrapper call, and parsing the JSON from the block, retrieving the hash stored in that asset. After the hash is obtained, we hash the document's class we obtained from qBox, before updating the last accessed date, so it does not produce a different output. We then compare both hashes. Assuming they match, the document can be considered secured and the system dependable.

The code for the operation detailed above is represented in the **Figure 12** below, representing the integrity validation executed by qDocs amid the document's access by the user prior to its creation. The document is considered open until its closure is requested and the document considered finalized, therefore, it is stored in the temporary MultiChain instance.

[1] = CCNumber_DocumentType.ShortName_Organization.ShortName_CreationDate_LastAccessibleDate

```
if (!document.qChainToken.Equals(null) && document.DocumentTemplate.MultiChain && document.Status.Equals("open")){
    Task.Run(async () =>
        {
            response = _Client_Temp.Stream.GetStreamItem(organization_token,document_token,true);
            String documentHash = (JsonConvert.DeserializeObject(response.Result)).txid;
        }).GetAwaiter().GetResult();
    if (documentHash.Equals(document.GetHashCode().ToString())){
        Console.WriteLine("Document is secured thanks to qChain's integrity validation.\n");
    }
    else{
        Console.WriteLine("Document was tampered with in an intermediary stage.\n");
        return false;
    }
}
```

Figure 12 *- Code Implementation for an Open Document Integrity Verification*

The last accessible date of the document is updated after this verification and persisted in the database.

In the **Figure 13** below we have similar code for verifying the document's integrity, but in this case, the document it does so for closed docuements, therefore, the document's class hash is located in another MultiChain instance, the final chain.

```
else if (!document.qChainToken.Equals(null) && document.DocumentTemplate.MultiChain && document.Status.Equals("closed")){
    Task.Run(async () =>{
            response = _Client_Final.Stream.GetStreamItem(organization_token,document_token,true);
            String documentHash = (JsonConvert.DeserializeObject(response.Result)).txid;
        }).GetAwaiter().GetResult();
    if (documentHash.Equals(document.GetHashCode().ToString())){
        Console.WriteLine("Document is secured thanks to qChain's integrity validation.\n");
    }
    else{
        Console.WriteLine("Document was tampered with after its closure.\n");
        return false;
    }
}
```

Figure 13 *- Code Implementation for a Closed Document Integrity Verification*

In **Figure 14** below, we can see represented the logical operation of a simple document retrieval and integrity checking described in the previous paragraph. This operation uses the token representing the stream which is stored in the organization's class and the token referencing the document, which was also stored in the creation process, both essential for qChain's asset retrieval.

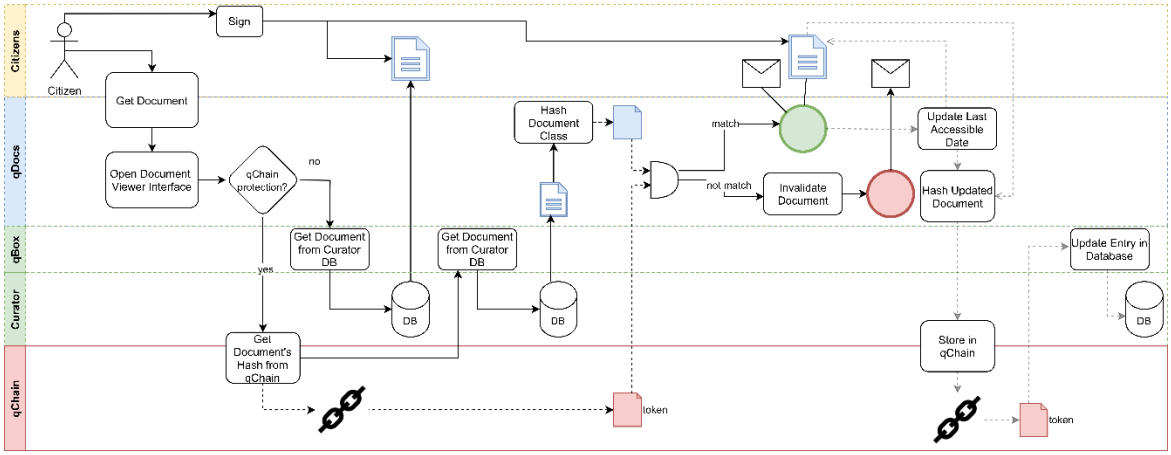

Figure 14 *– Business Process of qChain's Integrity Checking for a Single Participant Document*

Lastly, the platform displays a message confirming the successfulness of the operation, letting the user now that he does not have to worry about the solution's resilience and the document's safety. Next time the user attempts to access this document, the whole operation of the **Figure 14** above will take process to not only help argue the consistency and reliability of the system but to serve its main purpose of securing the document's integrity and operations, from creation to storage to access.

As explained, MultiChain's implementation is a little different due to lack of complexity offered by their smart contract solution, or as they call it, smart filters. Still, we want to assure that a similar mechanism can be present and offer the same level of security, therefore we developed the solution of separating closed documents with no need for modifications prior their creation and signature from documents that need multiple layers of alterations before officially being issued, such as multiple signatures from more than one participant, performed asynchronously.

Furthermore, Smart contracts would only be used, in the qDocs domain, to hold assets, in this case, documents, to be signed properly, waiting for the signatures of as many people had to sign them or to wait until a ransom was paid for the agreement. Therefore, we still decided to implement this solution with the primary purpose of assuring immutability and integrity to the database. Furthermore, MultiChain is simple to learn and implement and has too many other useful features from which qDocs can surely benefit from. Since we cannot implement this feature, we register every stage of the document.

Further, we have the example of creating a document that is meant to be signed by n participants, which can be observed in **Figure 15**.

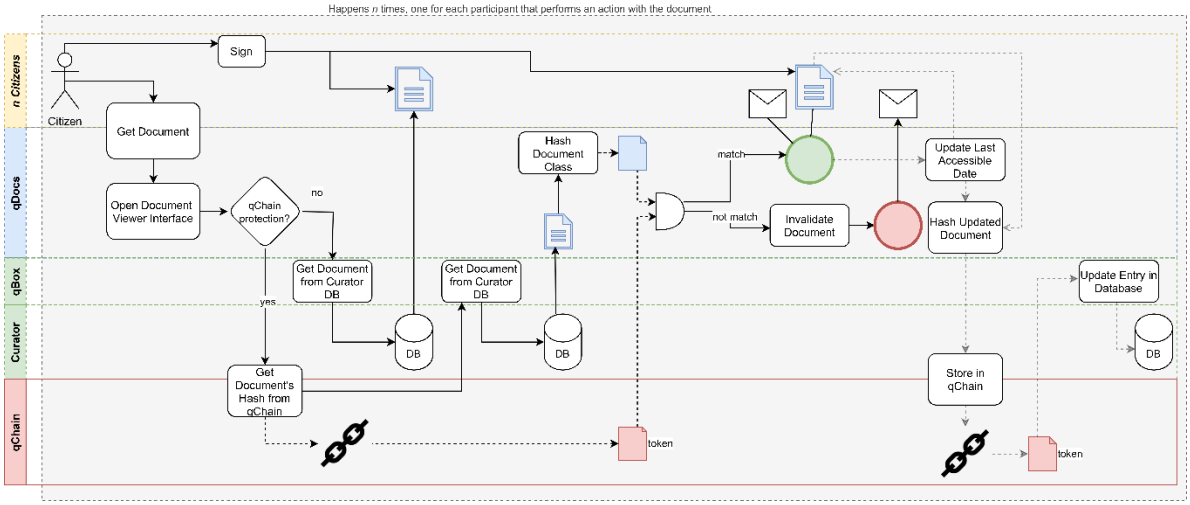

Figure 15 - *Business Process of qChain's Integrity Checking for a Multiple Participant Document*

This operation will happen as many times as participants need to sign the document in which the curator himself may be one of these participants, to issue its officiality as an authority. Every time a participant signs the document, the platform checks if everyone has already signed. If not, a new updated version of the document is created, stored on the curator's database and its metadata stored again on the provisional blockchain repetitively. Finally, after qDocs confirms all signatures and that the document is completed and ready to be stored, this will be secured in the final blockchain as a closed document.

Therefore, creating a document that the curator overrules the need for a signature or that has not yet been closed, will have its metadata stored on the provisional blockchain. The qDocs platform will be responsible for knowing when all the signature fields are filled correctly for the final document to be stored in the final blockchain. Because in this approach we do not control the upholding of this asset the same way a smart contract would, every time the document is modified, a new asset will be written onto the MultiChain's provisional chain. Therefore, every time a participant signs the document, it will be stored as provisional, until the platform confirms that all three signature fields are indeed signed.

This solution confers increased security to documents with multiple stages of creation because when the document is fetched for signing, its integrity will be checked and assured by the provisional blockchain. This happens as many times as the document needs to be modified, assuring security not only on the final stage of creation but also in every process throughout the full generation of the document. The final stage will produce an entry in qChain of different name, instead of aggregating the LastAccessibleDate at the end, it will add CloseDate and store it in the final chain.

5.4. Used Tools/Libraries

This section is reserved to discuss the tools used and libraries implemented in the project. Although not all of them are completely updated, they are complete enough to satisfy our development requirements. We also add the fact that some components of the qDocs server prior to the integration had to be updated and version checked for proper interoperability between modules and to avoid conflicts.

Firstly, we used MobaXterm's ability to open multiple tabs in parallel with different command lines. It was through this tool that the platform was also downloaded, installed, and configured. MobaXterm allows us to monitor the logging of the two components executed for the system's functioning: the curator application and the qDocs server. This platform also allows us to easily start and shutdown the services which can be useful in test environments and new implementations. Initially, MobaXterm was used to install the qDocs services using Linux terminals for library installation, system configuration and database populating.

Secondly, the code's study, visualization and adjustments were made through Visual Studio text editor, with no additional plugins. VS is a solid software for this kind of work, and it proved particularly useful when debugging the platform using breakpoints.

After these two components were enlisted, we proceeded to the integration, installing the docker software for scalable and containerized production, allowing fast and easy deployment of network-based systems. Through the library provided by the GitHub agency Kunstmaan, called docker-multichain [38], which was advisable by the MultiChain's official webpage, we downloaded the configuration files for the deployment of a MultiChain blockchain instance with four nodes: one master, two slaves and one explorer.

We decided to take out the explorer node, since its purpose is visualization, but we already have enough metrics in the docker's interface. We also decided to add one other slave node for reasons already explained above, such as conformity with system reliability using a Byzantine Fault Tolerant

network model. We also added a base node for downloading and installing MultiChain's configuration and executable files. This library was the base infrastructure of the qChain, in which two instances of the loadout described above were deployed and kept running for extensive periods of time, generating noise blocks, and providing system's health feedback.

Lastly, as detailed, we used GitHub user JonathanCrossland's C# wrapper library [39], called LucidOcean.MultiChain , to forward API calls to the qChain's blockchain directly from the qDocs server code. Although it stopped development and maintenance two years ago, since MultiChain was still the most appropriate blockchain platform to integrate with, and so a wrapper using the same programming language is also advisable to use.

We consider all tools to work correctly and to be properly documented, with ongoing updates and forums supporting discussions and error resolution. We also consider that the tools chosen were the proper ones for the development of this project, with the necessary specifications and programming languages, resulting in an easier integration and better interoperability between the old system and the new extensions added.

6. Evaluation

There are two main case study to evaluate the system's functionalities and operations, which simulate as close as possible to real ones. No extensive testing is made due to time shortage, but the creator of the C# MultiChain's wrapper, developed unit tests in his repository regarding the use of the wrapper and proves its functionality's success. The two-case study developed are the ones described as follows:

Case Study (A): Consider a simple document, an opinion poll for the semester satisfaction with the curricular unit's measures. This document only needs to be filled up and issued by the college administration. This document shall be stored on the final blockchain, as it will be deemed as closed after the college administration authorizes it. In this case, the curator is the administration services, and the participant is the student.

Case Study (B): Consider a more complex example, involving multiple participants to sign a document minute of a thesis defense. Let us say that after the student has had his dissertation's final discussion, the president of the jury creates a document of the minute using qDocs, and each member of the jury shall sign that document asynchronously. Only after multiple stages and all participants sign the minute, is the document closed and stored in the final blockchain. The college administration also represents the role of the curator and every member of the jury and professor present in the thesis defense and the student serve the role of participants

After these two case studies are repeated with various documents fitting to each case study' criteria to properly analyses the system's operational consistency, we collect metrics of the blockchain's usage.

6.1. Test Environment

The test environment is particularly simple and mostly hardcoded. We want to evaluate not only the successfulness of creating documents and storing them, but the connectivity and uptime of the qChain as a service. Therefore, we use qDocs platform to create templates that satisfy the purpose of the case studies described above and mark them as deemed for "qChain Protection."

We proceed to create such documents using the templates and access those multiple times to check the output for the validation of the document performed by qChain.

As for the second case study, we must take extra steps, adding participants and having them sign the document asynchronously to produce various versions of the document, as many as participants signing it, and further produce an unalterable version with the closure of the document.

6.2. Test Results

The tests were not fully conducted due to connectivity issues with the docker and misconfiguration of the network which led to a heavy load of requests with timed out exceptions. Some features could be evaluated directly in the docker terminal which proved the system to work, but the communication between the qDocs server instance and the docker program was being denied by the docker's firewall.

The docker software presents metrics as graphics for memory and CPU usage for each container, which would have been useful to observe when used extensively with the tests.

7. Conclusions

We started this project with the objective of integrating citizen critical processes in the document sector, automated digitally, with a secure way of registering events related to these operations. This solution was designed for public, everyday human use, defocusing on enterprise levels of infrastructure and more on the target population that also benefits from these features securely but does not have them yet.

We stumbled across many challenges such as learning this recent technology, despite of it being easy to use and connectivity problems with the docker containers and the MultiChain's inner components. Nevertheless, we agree that MultiChain is an appropriate platform for the scope of this project, not focusing only on financial statuses and transactional data, and being optimized with easy deployment and configuration for storage and validation and system resiliency.

Therefore, integrating a blockchain-based network with operations related to documents is a promising idea if the purpose is to secure them from malicious intents or conflicts that may compromise the document's information. Mostly this technology benefits the application it integrates with using distributivity properties and non-dependance on trust over the network peers. Considering we are dealing with operations performed in a network-based environment, and of public use, this is ideal for this market.

We also realize that this project could have traveled further in development and be more complete, but unfortunately, due to time shortage and programming challenges, this could not be done as we intended. Still, not only was a useful implementation developed, but we also researched deep into the world of this integration, which, despite rising, it is still poorly used in the public sector. Future tests may be enough to realize the extensibility of the solution, as it does not depend on a specific document, but on the demand from the curator, since the templates are all in the same format.

7.2. Contributions

This research can be useful in future studies related to similar integrations, combining a platform that manages critical assets with a secure decentralized system that aims to trace and register every event and action. Such security mechanisms help providing better conflict troubleshooting in information's content, increase system's resiliency and reliability and assure authenticity, non-repudiation and freshness to actions. Since the world is increasingly governed by digitalization and network connectivity and empowerment, securing digital assets, within companies or larger authorities, has become a priority.

The concept of this research, even if not fully implemented, is enough to serve as a basis to further implementations in various fields dealing with digital markets, in which, the main goal for the integration of document automation with a blockchain-based system was to confer security to citizen-centric operations related to documents, in the public sector, accessible to any citizen while not compromising its experience with the platform. Consequently, that's the main purpose for automation of services: to

facilitate time consuming operations, whereas blockchain intends to secure flows of events in an infrastructure.

Concluding, we hope to contribute to the concept of securing automation processes in a rising digital world without compromising to this solution as the only one viable.

7.3. Future Work

Future work may involve finishing the full integration with proper feedback to the user and fixing connectivity issues still present in the communication between qDocs and the qChain. Additionally, further extensive evaluation should be performed to better analyze the results produced from the platform.

The code should also be optimized for proper organization and documentation, but also providing better troubleshooting and no redundancy of data. We may also develop documentation over this new integration, describing every action taken and adjustment, for maintenance of the platform.

There may also be upgrades to the qChain functionalities by integrating codified conditions in the document creation events and associate them to each document or types of documents, compartmentalizing document's behaviors as extents to their complexity. Therefore, these conditions should be defined by the curator on demand and negotiated with qDocs operators for efficient implementation of what could be a future integration of smart contracts with qChain. Although MultiChain might not support smart contracts but smart filters, which do not behave the same, a future integration should implement such technology as an extent to qChain, residing between the latter and qDocs.

References

[1] J. A. Menezes, A. R. Da Silva, and J. De Sousa Saraiva, “Citizen-centric and multi-curator document automation platform: The curator perspective,” in *Proceedings of the 28th International Conference on Information Systems Development: Information Systems Beyond 2020, ISD 2019*, 2019, pp. 1–12.

[2] R. Lankester, “Implementing Document Automation: Benefits and Considerations for the Knowledge Professional,” *Leg. Inf. Manag.*, vol. 18, no. 2, pp. 93–97, 2018.

[3] V. Gatteschi, F. Lamberti, C. Demartini, C. Pranteda, and V. Santamaría, “Blockchain and smart contracts for insurance: Is the technology mature enough?,” *Futur. Internet*, pp. 1–16, 2018, doi: 10.3390/fi10020020.

[4] J. Yli-Huumo, D. Ko, S. Choi, S. Park, and K. Smolander, “Where is current research on Blockchain technology? - A systematic review,” *PLoS One*, vol. 11, no. 10, pp. 1–27, 2016, doi: 10.1371/journal.pone.0163477.

[5] MDSS, “qDocs , Citizen centric document technology,” *White Paper*, 2018. .

[6] J. António Trocado de Saldanha Sousa e Menezes, “qDocs : Citizen-Centered and Multi-Curator Document Automation Platform : The Curator Perspective,” Universidade de Lisboa - Instituto Superior Técnico, 2019.

[7] S. Priyanka and A. Nagaratnam, “Blockchain Evolution - A Survey Paper,” *Int. J. Sci. Res. Sci. Eng. Technol.*, vol. 4, no. 8, pp. 1–4, 2018, [Online]. Available: www.ijsrset.com.

[8] D. E. Mik, “Smart Contracts: Terminology, Technical Limitations and Real World Complexity,” *Law, Innov. Technol. 9.2*, pp. 1–26, 2017.

[9] M. Alharby, A. Van Moorsel, and A. Aldweesh, “Blockchain-Based Smart Contracts : a Systematic Mapping Study,” in *International Conference on Cloud Computing, Big Data and Blockchain (ICCBB)*, 2018, pp. 1–6, doi: 10.5121/csit.2017.71011.

[10] M. J. W. Rennock, A. Cohn, and J. R. Butcher, “Blockchain Technology Regulatory and Investigations,” *J. Litig.*, pp. 1–11, 2018, [Online]. Available: https://www.steptoe.com/images/content/1/7/v3/171269/LIT-FebMar18-Feature-Blockchain.pdf.

[11] A. R. Hevner, S. T. March, J. Park, and S. Ram, “Design Science in Information Systems Research,” *MIS Q.*, vol. 28, no. 1, pp. 75–105, Jan. 2004, doi: 10.2307/25148625.

[12] K. Peffers, T. Tuunanen, M. A. Rothenberger, and S. Chatterjee, “A Design Science Research Methodology for Information Systems Research,” *J. Manag. Inf. Syst.*, vol. 24, no. 3, pp. 45–77, Dec. 2007, doi: 10.2753/MIS0742-1222240302.

[13] F. Gacenga, A. Cater-Steel, M. Toleman, and W. G. Tan, “A Proposal and Evaluation of a Design Method in Design Science Research,” *Electron. J. Bus. Res. Methods*, vol. 10, no. 2, pp. 89–100, 2012.

[14] R. Iswanto, “Paper Waste Reduction Efforts Through Digitalization,” vol. 4, no. 2, 2019.

[15] M. Fritze, A. Eisingerich, and M. Benkenstein, “Digital transformation and possession attachment: Examining the endowment effect for consumers’ relationships with hedonic and

utilitarian digital service technologies," *Electron. Commer. Res.*, vol. 19, Jun. 2019, doi: 10.1007/s10660-018-9309-8.

[16] D. José Matias Caramujo, "A Citizen-Centric and Multi-Curator Document Automation Platform : The qBox and Further Interoperability Aspects," Universidade de Lisboa - Instituto Superior Técnico, 2019.

[17] LexisNexis, "Document Retention & Destruction Policies for Digital Data: What You Don't Know Can Hurt You," *Hum. Rights*, pp. 1–6, 2004.

[18] MHC, "DOCUMENT AUTOMATION BUYER ' S GUIDE DOCUMENT AUTOMATION BUYER ' S GUIDE," pp. 1–5.

[19] H. W. Paper, "Building Interactive , Data-Gathering Forms for BPM-Defined Workflows The Document-First Approach," pp. 1–7.

[20] T. Aste, P. Tasca, and T. Di Matteo, "Blockchain Technologies: The Foreseeable Impact on Society and Industry," *Computer (Long. Beach. Calif).*, vol. 50, no. 9, pp. 18–28, 2017, doi: 10.1109/MC.2017.3571064.

[21] S. Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System," pp. 1–9, 2008, doi: 10.1162/ARTL_a_00247.

[22] D. Bayer, S. Haber, and W. S. Stornetta, "Improving the Efficiency and Reliability of Digital Time-Stamping BT - Sequences II," 1993, pp. 329–334.

[23] C. Dwork and M. Naor, "Pricing via Processing or Combatting Junk Mail," in *Advances in Cryptology --- CRYPTO' 92*, 1993, pp. 139–147.

[24] M. Jakobsson and A. Juels, "Proofs of Work and Bread Pudding Protocols(Extended Abstract)," *Secur. Inf. Networks. Springer, Boston, MA, 1999*, pp. 258–272, 1999, doi: 10.1007/978-0-387-35568-9_18.

[25] L. M. Bach, B. Mihaljevic, and M. Zagar, "Comparative analysis of blockchain consensus algorithms," in *2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, 2018, pp. 1545–1550, doi: 10.23919/MIPRO.2018.8400278.

[26] V. Buterin, "A next-generation smart contract and decentralized application platform," *Etherum Whitepaper*, 2014. https://ethereum.org/en/whitepaper/ (accessed Nov. 29, 2020).

[27] "Nxt Whitepaper," *Whitepaper*, 2014. https://nxtdocs.jelurida.com/Nxt_Whitepaper (accessed Dec. 03, 2020).

[28] S. Aggarwal and N. Kumar, "An Introduction to Hyperledger," *Hyperledger Whitepaper*, 2018. https://www.hyperledger.org/wp-content/uploads/2018/08/HL_Whitepaper_IntroductiontoHyperledger.pdf (accessed Dec. 05, 2020).

[29] W. Ma *et al.*, "Hyperledger Architecture, Volume II," *Hyperledger Whitepaper*, 2018. https://www.hyperledger.org/wp-content/uploads/2018/04/Hyperledger_Arch_WG_Paper_2_SmartContracts.pdf (accessed Dec. 05, 2020).

[30] D. R. Joseph, “Hyperledger Architecture, Volume I,” *Hyperledger Whitepaper*, 2018. https://www.hyperledger.org/wp-content/uploads/2017/08/Hyperledger_Arch_WG_Paper_1_Consensus.pdf (accessed Dec. 05, 2020).

[31] M. Castro and B. Liskov, “Practical Byzantine Fault Tolerance,” *OSDI. Vol. 99*, pp. 1–14, 1999, doi: 10.1007/978-0-387-95982-5_25.

[32] G. Greenspan, “MultiChain Private Blockchain - White Paper,” 2015. http://www.multichain.com/download/MultiChain-White-Paper.pdf (accessed Dec. 04, 2020).

[33] H. Yu, H. Sun, D. Wu, and T.-T. Kuo, “Comparison of Smart Contract Blockchains for Healthcare Applications,” 2019. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7153130/pdf/3202134.pdf.

[34] S. Augustin and J. Russel, “Thomson Reuters and OpenLaw Combine Blockchain-Driven Smart Contracts with Document Automation Technology,” 2019. https://www.thomsonreuters.com/en/press-releases/2019/october/thomson-reuters-and-openlaw-combine-blockchain-driven-smart-contracts-with-document-automation-technology.html.

[35] Y. Wang *et al.*, “Formal verification of workflow policies for smart contracts in azure blockchain,” *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 12031 LNCS, pp. 87–106, 2020, doi: 10.1007/978-3-030-41600-3_7.

[36] S. Tempesta, “Verify e-Documents with Smart Contracts in Azure Blockchain Development Kit,” *MSDN Magazine Issues - March 2019 - Volume 34 Number 3*, 2019. https://docs.microsoft.com/en-us/archive/msdn-magazine/2019/march/blockchain-verify-e-documents-with-smart-contracts-in-azure-blockchain-development-kit.

[37] M. Srubar, “From Paper to Blockchain: How Your Automation Turns into Smart Contracts,” *Industry Insights*, 2018. https://www.activedocs.com/industry-insights/2018-11-14-paper-to-blockchain-automation-to-smart-contracts.html.

[38] Kunstmaan, “docker-multichain,” 2015. https://github.com/Kunstmaan/docker-multichain.

[39] JonathanCrossland, “LucidOcean.MultiChain Assembly,” 2018. https://github.com/JonathanCrossland/multichain#lucidoceanmultichain-assembly.

Appendix A

Study Methodology and Thesis Schedule

DSR Phase	Task	Period
#1	**Problem identification and motivation**	
	Analyze of the research context and motivation.	Oct
	Understand and analyses the background of document automation, blockchain and smart contracts.	Nov
#2	**Define the objectives of a solution**	
	Analyze the Related Work: existing blockchain platforms.	Nov
	Analyze the Related Work: related projects	Nov
	Define the research objectives	Dec
#3	**Design and development**	
	Iteration-1: Design and development: Design a more detailed and technical description of the solution.	Apr-May
	Iteration-2: Design and development: Start the implementation of only one simple Multichain blockchain instance.	Jun
	Iteration-3: Design and development: Start the implementation of both blockchains, provisional and final.	Jul-Aug
	Iteration-4: Design and development: Work on the systems accuracy and performance and reliability related tweaks.	Sep- Out
#3	**Demonstration**	
	Iteration-1: Demonstration: Focus on the first prototype usage, with one blockchain.	
	Iteration-2: Demonstration: Focus on presenting the solution implementing the two blockchains.	
	Iteration-3: Demonstration: Demonstrate the reliability, accuracy, and consistency of the final improved solution.	

#4	**Evaluation**	
	Iteration-1: Evaluation: Case Study (A)	
	Iteration-2: Evaluation: Case Study (B)	
s#5	**Communication**	
	Write the research project report	Dec-Jan
	Present and discuss the research project report	Jan
	Write a research paper to submit to a conference or journal	Out
	Write the MSc dissertation	Aug-Out
	Present and discuss the MSc dissertation	Nov