Universidade do Minho
Escola de Engenharia

Pedro Francisco Sousa e Silva

Immutability of Health Data: Private Blockchain Approach

Master's Dissertation

Integrated Master's in Engineering and Management of Information Systems

Work carried out under the supervision of **Professor Manuel Filipe Santos, Ph.D.**

and

Professor Tiago Guimarães

October of 2024

Universidade do Minho
Escola de Engenharia

Pedro Francisco Sousa e Silva

Immutability of Health Data: Private Blockchain Approach

Master's Dissertation
Integrated Master's in Engineering and Management of Information Systems

Work carried out under the supervision of **Professor Manuel Filipe Santos, Ph.D.**
and
Professor Tiago Guimarães

October of 2024

Copyright and Terms of Use by Third Parties

Acknowledgements

In this section i felt the need to provide a Special words of gratitude to the many persons who have assisted me along the process toward bringing this thesis to completion.

I would like to express my most sincere and great respect to my advisor, Tiago Guimaraes, for his inspiring guidance, valuable insights and endless patience. His technical contributions and encouragement were of extreme relevance in shaping this work, and I am deeply indebted as regards his investment of time and effort in my growth as a researcher.

Also, i would like to mention deeply indebted to Centro Algoritmi for affording me the means and facilities necessary for my work. I would like to thank [Funding Organization] for the financial support that made this work possible.

On a personal note, I would like to thank as well my family and friends for their unwavering support and encouragement throughout this long and often difficult journey. The hard work with which my parents, Pedro Silva and Helena Silva, passed to me always made me feel like it always pays off in the end, so thanks. Also, the same applies To my close friends within the University context, Tiago Martins and Jonas Barbosa, which made my life at the university easier, bringing motivation, which was an invaluable asset at the most difficult moments.

I would, lastly, extend my appreciation to all those who have, one way or another, contributed toward the completion of this thesis. Without such help, it could not have been completed by me. I shall always be grateful for that fact and i hope this is another set of firm steps to what could be a successful career.

Abstract

This thesis investigates the benefits of adopting private blockchain technology, among other recent innovations, within the health sector. Even though organizations have been concerned about security for a long time, there is a trade-off between security and the exchange of data between entities across the network. The primary attribute of blockchain is thus ensured to provide better integrity, security, and flexibility in sharing information securely across a group or consortium of organizations.

During this project two main use cases where created: integration of **Hyperledger Fabric** with **IPFS** and optimization of management of blockchain infrastructure by means of contemporary technologies. In further sections, each of use cases is deeply elaborated to give an extended look at their practical usage and results.

The first use case would assume the benefits of using **Hyperledger Fabric** in combination with **IPFS** for secure file storage, where a private network of **IPFS** would store the files and record hashes on the blockchain for integrity.In the second use case on the other hand, there is the design and benchmarking of several network infrastructures of blockchain seeking an optimal configuration. Besides that, there is a discussion about the development of UI tool for monitoring and analysis of the status of the network. To this precise use case,more work is expected in the future to enhance this use case further, to refine even more its robustness, operability and maintainability.

This work will demonstrate such use cases and illustrate how blockchain can enhance data security and operational efficiency in healthcare while resolving challenges in large-scale implementations.

Keywords: Business Intelligence; Health Care; Data Warehouse; *Misericórdia*;

Statement of Integrity

I hereby declare that I have acted with integrity in the preparation of this academic work and confirm that I have not engaged in plagiarism or any form of misuse or falsification of information or results at any stage leading to its completion. Furthermore, I declare that I am aware of and have adhered to the Code of Ethical Conduct of the University of Minho.

Contents

List of Figures

List of Tables

List of Abbreviations, Symbols, or Acronyms

IPFS - Interplanetary File System

Hlf - Hyper Ledger Fabric

GDPR - General Data Protection Regulation

HIPAA - Health Insurance Portability and Accountability Act

DSR - Design Science Research

ICT - Information and Communication Technology

CDBS - Central Bank Digital Currency

ICP - Internet Computer

ACM - Access control mechanisms

EMR - Eletronic medical records

PHR - Personal health records

SWOT - Strengths,Weaknesses,Opportunities and Threats

KYC - Know-Your-Customer

AML - Anti-money Laundering

TLS - Transport Layer Security

CA - Central Authority

MSP - Managed Service Provider

PoW - Proof of Work

PBFT - Practical Byzantine Fault Tolerance

CFT - Crash fault-tolerant

P2P - Peer to Peer

DHT - Distributed Hash Table

CID - Content Identifier

DNS - Domain Name System

HTTP - Hypertext Transfer Protocol

NFT - Non-Fungible Tokens

JSON - JavaScript Object Notation

IPV4 - Internet Protocol version 4

IPV6 - Internet Protocol version 6

IPNS - Interplanetary Name System

AC - Access Control

IoMT - Internet of Medical Things

VM - Virtual Machine

LXC - Linux Containers

API - Application Programming Interface

IP - Internet Protocol

SDK - Software Development Kit

CPU - Central Processing Unit

VPA - Vertical Pod Autoscaler

HPA - Horizontal Pod Autoscaler

FQDN - Full qualified name

PV - Persistent Volume

PVC - Persistent Volume Claim

NFS - Network File System

K8 - Kubernetes

DCT - Docker Content Trust

Seccomp - Secure computing mode

CI - Continuous Integration

CD - Continuous Delivery

BPG - Border Gateway Protocol

ARP - Address Resolution Protocol

NDP - Neighbor Discovery Protocol,

LAN - Local Area Network

MAC - Media Access Control

JWT - JSON Web Tokens

mTLS - mutual Transport Layer Security

SPIFFE - Secure Production Identity Framework for Everyone

URI - Uniform Resource Identifier

TCP - Transmission Control Protocol

TPS - Transactions per second

HTML - Hypertext Markup Language

CSV - Comma-Separated Values

KPI - Key Performance Indicator

UI - User Interface

CRUD - Create,read,update and delete

REST - Representational State Transfer

gRPC - Google Remote Procedure Call

IAM - Identity and access management

SSO - single sign-on

RBAC - Role-Based Access Control

OIDC - OpenID Connect

MFA - Multi Factor Authentication

SMS - Short Message Service

TOPTP - time-based one-time password

RPC - Remote Procedure Call

DSR - Design science research

EHR - Electronic Health Record

OTP - One time password

RMI - Remote Method Invocation

XOR - Exclusive OR

UTP - Unshielded Twisted Pair

SCTP - Stream Control Transmission Protocol

SFS - Self Certified File System

ICE - Interactive Connectivity Establishment

NAT - Network Address Translation

PKI - Public key infrastructures

CRL - Certificate revocation lists

HSM - Hardware security models

TX - Transaction

RAM - Random Access Memory

IEEE - Institute of Electrical and Electronics Engineers

1 Introduction

1.1 Context and Motivation

The healthcare industry is in the landscape of a radical modification, dragged by indulging advances in technology, converting expectations from patients and a highly complex regulatory environment. From digital health solutions to electronic medical records, telemedicine, and connected devices, the proliferation of data is happening in healthcare at rates never seen previously. This explosion of data opens up new opportunities and creates challenges. While on one hand, access to huge volumes of patient information has the potential to transform diagnosis, treatment and preventive care - all to improve patient treatment - this could be problematic because the increasing complexity of healthcare systems coupled with rising concerns on data privacy and security creates considerable barriers to benefits realization.

The healthcare organization, therefore, finds itself between two poles of extremes: driving innovation through data on one hand, and on the other hand, ethical and legal imperatives of maintaining patient privacy. This gets even more imperative as most healthcare systems across the world are fragmented in nature, encompassing a broad range of stakeholders that include hospitals, laboratories, insurance companies, regulatory bodies, and pharmaceutical firms among others, all of whom will have to share closely guarded information. Ensuring that the data remains accurate, secure, and accessible only to authorized parties stands as one of the most formidable issues associated with healthcare today.

Traditional medical data management systems are prone to inefficiencies and vulnerabilities. Centralized databases remain at the mercy of even basic cyber-attacks, data breaches and manipulation from internal sources. Moreover, many old health management systems face operational difficulties, such as non-interoperability, which complicates the free flow of information between different agencies. The above limitations signify the urgent requirement for more secure, flexible, and trustworthy systems for managing healthcare data.

But working amongst all these critical challenges, the blockchain technologies have inspired confidence. What was ostensibly conceived as the underpinning technology for cryptocurrencies like **Bitcoin**, blockchain itself-in core concepts of decentralization, transparency, immutability and cryptographic security-has kindled interests in industries way beyond finance. Particularly in healthcare, blockchain holds real promise to offer an indestructibly secure way of recording transactions and managing data where trust and security are paramount. Blockchain can provide a decentralized solution that removes some of the associated risks of using centralized databases by distributing the data across a network of nodes, none of which has unilateral control over the entire system.

Within the sphere of blockchain frameworks, **Hyperledger Fabric** shows up and shines as it is considered one of the most convenient for the healthcare sector. **Hlf** is described as an open-source project covered and performs as a permissioned blockchain framework for enterprise usage, thereby perfect as well for this thesis but this will be covered more ahead. Unlike public Blockchains, which allow everyone to participate, **Hyperledger Fabric** is designed for participation in environments where participants are known and trusted. This is an important permissioned model necessary for healthcare, where regulatory compliance-like **GDPR** and

HIPAA, along with patient privacy is a non-negotiable requirement. Another very important aspect of the framework is the modular architecture, which provides great flexibility because an organization can adapt the system to suit their needs in any area of concern that it deems fit-whether record keeping, tracing drugs, or settling claims.

The thesis is aimed at discussing the application of Hyperledger Fabric as a transformative technology to approach the most important challenges faced by healthcare systems. The key focal area in this research is how Hyperledger Fabric can be mobilized to advance the qualities of data integrity, trust, flexibility and coordination in the healthcare ecosystem. Therefore with this, we will answer the very research questions which are "How can a private blockchain network be utilized to improve data integrity,security,flexibility and collaboration within the healthcare ecosystem?" and "What kind of infrastructure design is necessary to support a blockchain solution in such a vast and complex environment as healthcare?" By implementing blockchain through a distributed ledger, cryptographic algorithms and consensus mechanisms, Hyperledger Fabric is poised for Layer 1 utilization in building a novel model of healthcare data management that engenders trusted expectation among stakeholders in the assurance of sensitive healthcare information being controlled against unauthorized access and tamper-proof.

We will start with data integrity in healthcare, which at its core wants correct and tamper-proof records to form the very bedrock of diagnosis, therapy, and even billing. The immutability ledger in **Hyperledger Fabric**-finally cryptographically linked and unable to be altered-offers a big advantage over traditional databases. The feature will ensure that the data can't be tampered with and will provide a clear, transparent audit trail to verify if any information at any given time is genuine. Letting patient data, medical history or prescription be recorded in a secure way, **Hyperledger Fabric** may become the technology capable of excluding frauds of data manipulation and unauthorized access altogether. The other focus is data flexibility and interoperability, which have been a nightmare in health care for quite some time. Often, there are different systems within different health organizations that are incompatible with one another and usually lead to data silos, which cannot easily facilitate smooth information exchange. The modular design of **Hyperledger Fabric** allows innovation in health care to build a blockchain-based solution that interfaces with existing systems and is agile to adapt to future technologies.

This flexibility not only greatly eases migration from older systems but also promotes innovation in that new functions can easily be added as the needs of the industry develop.

More importantly, this research will go a long way in addressing the need for collaboration among healthcare stakeholders. Since the data is scattered across hospitals, insurance companies, pharmaceutical firms and regulators, the ability to share information in a secure manner has never been so critical.

Permissioned network and granular access control mechanisms are provided by **Hyperledger Fabric** to enable wide collaboration among health organizations and ensure that sensitive data is inaccessible to unauthorized parties. Preceding with such control provides a trusted setting where stakeholders are free to share their information, knowing it will help achieve greater advances more quickly, improve patient care, and give way to smoother processes. Beyond both of these key themes, consideration is given to the specific use cases of **Hyperledger Fabric**

in healthcare: supply chain management, identification of patients, and clinical trials. Each of these has unique problems that blockchain technology can help solve.

For instance, origin and journey tracing in supply chain management will detect the risk of counterfeit drugs reaching the market. In identity verification, **Hyperledger Fabric** will help the patients to be in control of their health records and grant access to specific providers with permission instead, hence advancing privacy while reducing the risk of identity theft.

In this respect, the thesis will discuss the technical deployment of **Hyperledger Fabric** by utilizing state-of-the-art technologies such as **Kubernetes**, mainly in on-premise environments where healthcare organizations would wish to have full control of their infrastructure. Thus, using **Kubernetes** for the deployment of a **Hyperledger Fabric** will provide the ability to assess the scalability, security and performance of this deployment and then compare it against the conventional systems in operation within the context of today's healthcare. The technical assessment will also include an analysis of the advantages and disadvantages of such deployment, thereby offering practical insights into any healthcare organizational considerations towards blockchain implementation. Finally, the research will try to provide a holistic pathway on the expected work regarding the future of the second use case project, considering potential risks and pitfalls that might arise, areas of investigation, and more. While huge, the benefits of blockchain are not devoid of problems in application, including most especially, those related to governance, regulatory compliance, and integration into the existing healthcare system. As such, this thesis tries to give equal attention to considerations in an effort to balance the thesis as regards to the role **Hyperledger Fabric** can play in transforming healthcare. To sum up, the present thesis will critically assess the magnitude of transformational impact Hyperledger Fabric may have on the healthcare industry through the in-depth review of its applications and respective technical and strategic consequences. As we forge ahead into a future characterized by data-driven innovation, collaboration, and trust, blockchains could provide that much-needed tool in reshaping the health environment to ensure not only the secure but also efficient and ethical management of patient data.

Consequent upon the rapid development and increase in health data volume, security, and privacy concerns, and efficiency, the healthcare industry currently finds itself at an extremely critical juncture. As health care providers increasingly adopt digital health solutions-such as electronic medical records, telemedicine platforms, and connected devices-the overall volume of health care data resources is developing at an exponential rate. This data outrun instigates a vast number of opportunities to improving systems that help in taking care of patients, but it also compiles a serious deck of challenges in data management, security, and sharing. Perhaps the most relevant of this deck is the concept of data integrity, which remains fragile in healthcare systems as in any worldwide realm.

These inaccuracies, unauthorized accesses and data changes may critically lower patient safety and impact the quality of care. Traditional data management infrastructures, often centralized in architecture, have proven very vulnerable to many forms of data breaches, cyberattacks and insider threats. What could represent an example of this shameful situations is, in the healthcare organizations sphere, this being the biggest target for ransomware attacks, where many have resulted in an unfortunate financial and reputational disasters. In 2021 solo,

more than 40 million health record breaches were declared in the United States. Furthermore, fragmented health systems disappoint further by creating silos of data where different organizations have incompatible systems, creating barriers to collaboration and limiting interoperability, furthering vulnerabilities. Other than security challenges, demands for efficient, flexible and interoperable systems have never been more urgent.

Health care organizations struggle with outdated, legacy systems that do not permit seamless integration across providers, insurers, and regulatory agencies. All these latter systems are frequently incompatible with each other, making it very difficult to facilitate the real exchange of critical health data in real time. This lack of coordination obstructs innovation and operating efficiency, delaying treatment, inflating health care costs, and adversely impacting patient outcomes.

Moreover, the healthcare system relies heavily on trusts: both in data shared between stakeholders and in security around that data. Patients, providers, insurance companies, and regulators all need to know that when they collaborate, their information is exchanged correctly, securely and in a tamper-evident manner. Obviously, many systems today do not ensure this level of trust, with all the associated benefits of collaboration and cooperation that would enable improved patient outcomes. Abstracted, to engage against regulatory frameworks such as **HIPAA** and **GDPR** and gather new solutions and have in mind restrictions related to the usage of traditional providing of data or even cloud storing that may suffer from scalability, there comes the urgent of improving the current technology landscape presented in the healthcare facilities.

1.2 Finality and Objectives

If putted into deep thinking, in the sphere of the modern new world, blockchain technology has loom as an auspicious solution to the given number of challenges faced by the healthcare industry. It's enormous traits can range from cryptographic security, transparency and immutability and ,by retaining such characteristics, it also seeks influence in securing and managing sensitive data. It aims to safeguard the gathering of data across a networking of nodes, making it tamper-proof and erasing the traditional single points of failure. In another deck of terms, blockchain can enable smooth data allocation between heterogeneous entities and yet making certain that only conceded parties have privileges to specific and maybe confidential pieces of information.

Within the vast array of blockchain frameworks, **Hyperledger Fabric** looms as particularly well-suited to addressing the specific needs of the healthcare environment. It provides the flexibility, scalability and security that serve as the genesis block for enterprise-level applications. Unlike public blockchains such as **Bitcoin** or **Ethereum**, **Hyperledger Fabric** permissioned model allows more control over access to the network, ensuring that only trusted participants are allowed to validate and access data. This is a crucial feature in healthcare, where strict privacy regulations and the need for secure, authorized data sharing are very important, not speaking about the need of on premise implementations which are also very possible with this framework.

Hyperledger Fabric's modular architecture also enables organizations to customize their networks to meet specific needs. It can be used for medical record management, pharmaceutical

supply chain tracking, or patient identity verification, the platform can be adapted to a variety of use cases. This flexibility not only facilitates innovation but also supports the integration of blockchain technology with existing legacy systems which reduces the friction associated with technological adoption in healthcare settings.

The research and project is motivated by the growing recognition of the transformative potential of blockchain and also by the major other main technologies used currently in the market. With frequency, lots of people discuss the matter of decentralization, but lack on the sphere of trying to solve the problem in hands. When there is a depth relation within the creation of an conjunction between centralized and more decentralized technologies, we get the means to empower healthcare institutions to face their real problems. While blockchain has already been explored in sectors such as finance and supply chain management, it's application in the healthcare sector remain relatively fledgling. Nevertheless, early trials and implementations have been showing promising in aprimoring data security, interoperability and collaboration between multiple organizations or even stakeholders.

The reliance from the healthcare on trust makes it an ideal candidate for the blockchain landscape, even more if there is careful thinking about the need of securing and verifying sensitive patient data. However, deploying such complex system in the healthcare context does not cease to be very challenging and demanding. A lot of possible barriers to achieve this can be the Technical, regulatory, and operational spheres of the sector. Scalability is also another dual knife problem, and remains as so in large healthcare environments with enormous and vast amounts of information and high transaction volumes. This is a huge concern in both worlds since, within a blockchain there is the need of careful thinking such that there is the concern regarding the degree of consensus which delays requests out. Furthermore, the integration of blockchain with legacy systems and compliance with existing regulatory frameworks must be thoroughly evaluated.

This thesis is driven by a desire to explore these challenges and opportunities in depth. Specifically, it seeks to understand how Hyperledger Fabric can be designed, implemented, and deployed to address the critical needs of healthcare organizations. By investigating the ways of deploying a private blockchain network within healthcare systems.

This research aims to answer the following key questions:

1. **"How can a private blockchain network be utilized to improve data integrity, security, flexibility, and collaboration within the healthcare ecosystem?"**

2. **"What kind of infrastructure design is necessary to support a blockchain solution in such a vast and complex environment as healthcare?"**

This questions are imposed, so that it becomes more evident in a qualitative way the achievement of the result in mind. Further assumptions about actual quantitative goals must be ensured as the thesis goes along but answering this questions is the basis to come up with a suited solution in practical but also in theoretical knowledge terms.

As obvious as it may be, every research must have it's own objectives and this one could not be different, specially this one where there is the possibility to gather multiple objectives due to the countless numbers of opportunities and challenges that could be faced regarding this

procedure of recursion to the blockchain. This becomes even more clear according to the area of investigation which is healthcare, a institution that depending of the department becomes a high level risk sector to face, not only because of that but also mainly because of it's many problems, it's public/private nature that is always changing, the regulations required to be apart of the procedures, the continuous innovation, the fact that is researched by a vast number of different areas, the number of stakeholders involved, the number of organizations involved and the change resistance. With this in mind, the objectives covered will be the most general possible and they will aim to solve the majority of problems that the project may face and with this there are a lot of risk as well but those will be also described further in this work. The objectives that are intended to be reached in the realm of this work are the following:

1. Answer Research Questions

Answer Research Questions is the basis of the project, if answered this question will generate practical and also theoretical insights that will foster innovation. The questions mentioned here, are the ones that were acknowledged in the section 3, being those **"How can a private blockchain network be utilized to improve data integrity, security, flexibility, and collaboration within the healthcare ecosystem?"** (question 1). And **"What kind of infrastructure design is necessary to support a blockchain solution in such a vast and complex environment as healthcare?"** (question 2), respectively. As it will be shown further in this thesis, to gather the information to answer such answers, there will be mentioned the existence of 2 major use cases: One works more as a extension, altought inspected firstly and the second one, which is conducted using **DSR**(Design Science Research), is what is required as genesis block for creating use cases that can then me extended using the first use case. In other terms, the first one is mainly theoretical and the second one try's to solve a real world problem within the healthcare realm. Answering those questions, not only is important for the thesis itself but also for the healthcare sphere and other contexts that could adopt the same logic to solve their problems, being essential answering them as such occurs where both practical and theoretical inceptions are very important. Making the second project more appealing due to having proven results to achieve a given solution. Despite this being the main objective, inside of the second use case there are others that are more quantitative, thereby more concrete. Some can be used as metrics in other contexts, others may not but the importance of these questions remains important altought not that much specific when it comes to the refereed metrics and even the requirements disposed in the use case section.

Both questions are answered by using the conjunction of both research prior use case and by delving into the use case itself, altought it is believed that the use cases represent better the potentialities of this work because the theory by itself sometimes is not sufficient to cover vast cases, such the case of the sector under study, more concretely, the healthcare environment.

2. Contribute

Another objective that stands out with honor is the honored **contribution**. Despite this being obvious and also be part of the **DSR** methodology, it is still important to mention concretely this as an objective. Contribution is very important and avoiding specifying to much in presenting the knowledge produced is also something that should be done with a lot of consideration, since by

specifying to much, there is the probable occurrence of not giving to much for other's to work in their own context or situation, therefore making the contribution only suitable in the landscape that is being targeted in this precise research. More concretely, it is founded that creating clear and effective contribution that could be applied to various contexts is another objective that is intended to reach and that is the main reason why there are specific understandings relatively to the use cases presented but also in a general context, such that other researches can feed them with the specific cases but also understand deeper what was reached by using the global view presented in the final of the thesis.

Document Structure

This dissertation is divided into several chapters: "Introduction,State of Art,Methodologies and Tools, Use Cases,Discussion,Conclusions," and "References.".

Introduction: This chapter is reserved to give a contextualization and expose what is about to be delved. This is accomplished by explaining the context, motivation, finality,objectives and document structure.

State of Art: Within this chapter there is the supply of the most important theoretical concepts that are pretty relevant for the case under scrutiny.

Methodologies and Tools: In this chapter, there is the explanation of DSR and how it is being used currently in general terms, the explanation of SWOT analysis, the expected project plan, a list of risks, and the relevant tools divided by area of concepts (blockchain, microservices, analytics and web development).

Use Cases: On this section, there will be mention of two major use cases that were conceived during the project, standing as the biggest font of knowledge of this work. In each section, objectives, plans, discussions, and even conclusions will be conducted. Also, one of the use cases is more practical, having topics that are more specific to it's sphere, such as implementations, solution comparison, approach, and final architecture.

Discussions: This represents the discussion of the whole project and not a single use case. This is important to reflect over the results obtained as a whole and not only specific to a single use case.

Conclusions: In the conclusions section, there will be plenty of views around final considerations, project limitations, and future work. This will be done in the context of the general project, since every use case already speaks about such things.

References: The references chapter will be around the bibliographical references that were used along the dissertation.

2 State of Art

Speaking about the state of Art, there is a need to explain the current landscape surrounding 4 major subjects: **Blockchain**, **Hyperledger Fabric**,Scalability and **IPFS**. Each one of this topics is important to the given research, since they are considered in at least one of the mentioned use cases.

The first one is more about the current general state of the blockchain in the healthcare environment,focusing in the overall current play of the blockchain in the world, it's influence in healthcare, its benefits, requirements, applications and even the challenges that it is facing.The second one is more concretely about the **Hyperledger Fabric** network which is the framework that is being used in both use cases, having more merit than other private solutions due to its time on the market, the enterprise behind it and the number of works that are already published around this technology which makes it ideal for this project.The third is Scalability, a crucial huge concern in enterprise level contexts because it is the major concept for high availability and meeting the demand of a vast number of users. Finally, the forth is about **IPFS** that plays a role of complementing the blockchain capabilities.

2.1 Blockchain

Blockchain technology, introduced in 2008 through the cryptocurrency **Bitcoin**, has been envolving and expanded its influence across numerous sectors within **Information and Communication Technology (ICT)**, with its usage on the rise in recent years. Its adoption has been in majority due to the phenomenon of **cryptocurrencies** [1] ,**CDBS** (Central Bank Digital Currencies) [2] , **capital investments** and **substantial interesting use cases in blockchain startups** which gathered interest and development of this technology.

This technology operates by storing information in records distributed in a decentralised manner to all computing devices within the blockchain infrastructure. The system functions within a peer-to-peer network consisting of interconnected nodes. Every node retains its individual copy of the ledger, documenting all transactions within the network. The transactions are organised into blocks, each cryptographically linked to the next, forming a chain of blocks known as the Blockchain. Thanks to its advanced features, this technology offers multiple services, such as traceability, integrity, and security, while maintaining public and decentralised information storage, thus preserving privacy. Blockchain technology holds the promise to transform various sectors, including healthcare. Despite the associated challenges, ongoing research and developments actively address these potential hurdles. As blockchain technology progresses, its potential applications and advantages in the healthcare sector are just beginning to unfold.

2.2 Types of Blockchain

Blockchain technology primarily exists in three forms: public (permissionless), consortium (public permissioned), and private (permissioned). The main differences between each type of network rely on who can access, write, and read data on the Blockchain. In order to come up with a deep understanding over it's characteristics, it's worth exploring the specific features of each type [3].

Due to their transparency and openness, public blockchains have no barriers to who can use and participate in the process, which makes them more decentralized in a certain sense. The blockchain has publicly visible records that anyone on the internet can verify and use to add a transaction block. Examples of public blockchains include major project networks like **Bitcoin**,**Ethereum**,**Tron**,**ICP** and **solana** where even its use case can be very difference since,

as example, **Bitcoin** is meanted to only monetary transactions of information while **Ethereum**, due to it's difference architecture can even run logic over it by leveraging smart contracts [4] [5].

On the other hand, private blockchains (permissioned networks) are blockchains in which an organisation or entity centralises writing permissions while reading permissions can be public or arbitrarily restricted. This means that the configuration of the network is of the responsibility of a single organization. Additionally, they allow mining and provide high privacy due to writing and reading permissions restrictions. The biggest advantage of private blockchains over public ones is that participants can adapt their rules and logic according to their business altought a given number of members must agree with each other about this rules depending of how the network is configured, allowing them to religiously set what they see as a valid transaction within their organization. Moreover, participants and its components are known, restricting any addition of forged blocks to the chain. Additionally, altought not recommended manual intervention can correct faults and chain participants can control the maximum block size, addressing scalability issues or their proper use case. Verified participants only verify transactions, leading to lower processing power and cheaper transactions. This type of blockchain is similar to a standard distributed database. **Hyperledger Fabric** is the most mentioned permissioned [6].

Finally, let's explore the consortium blockchain. This type comprises a set of organisations or entities and is partially decentralised. Unlike assigning power to a single entity like in permissioned ones, the consortium blockchain divides this power among a group of people or entities known as consortia. Only this defined group can verify and add transactions, which need to be acknowledged by consensus nodes. A small portion of nodes is selected to determine consensus. This number of nodes are defined by the rules shared across heterogeneous organizations in contrast to only permissioned blockchains, precisely because this participants share a given flow in their normal activity which makes it easier to exchange information in a more reliable way. **Hyperledger fabric** can also be an example of a consortium blockchain and so as **R3CEV**.

Now, let's examine the following table, which provides a comprehensive overview of the properties exhibited by each previously discussed blockchain.

Property	**Public Blockchain**	**Consortium Blockchain**	**Private Blockchain**
Consensus determination	All miners	A selected set of nodes	Selected Organisation
Read Permissions	Public	Public or Restricted	Public or Restricted
Immutability	Nearly Impossible	Could be tampered	Could be tampered
Efficiency	Low	High	High
Centralised	No	Partial	Yes
Consensus Process	Permissionless	Permissioned	Permissioned

Figure 1: State of art: blockchain types

Each type of blockchain, irrespective of its classification, presents distinct advantages tailored to its particular use case. Public blockchains may be suitable for certain situations, while in others, it may be necessary to ensure private control through consortia or private blockchains.

The choice of which type of blockchain to implement ultimately depends on the specific service or application for which it is needed.

2.3 Blockchain and Healthcare Sector

The healthcare sector is a problem-oriented domain with intensive data usage and personnel involvement, where the ability to access, edit, and trust data emerging from its activities is crucial for the sector's overall operations. Currently, healthcare institutions face a growing demand for data from the industry and research organisations. Simultaneously, unauthorised sharing, breaches, and theft of sensitive data continually undermine public trust in these institutions, which can be fatal in certain critical level information's. A third issue involves bad and non standardized practises in the healthcare ecosystem, such as problems with medications, procedures, competencies, and patient falsification. Given this set of challenges, it is necessary to explore alternative approaches to the existing ones. With key attributes such as decentralisation, distribution and data integrity, without needing third parties, blockchain technology can surge as a potential allie to fight this problems. In addition, this technology can also ensure and improve existing interoperability, information sharing, access control, provenance,immutability and analytics of data among stakeholders, along with an extensive set of features. This way, the healthcare sector is moving towards a new infrastructure capable of building and maintaining trust among all stakeholders speeding up innovation [7].

The immutability offered by blockchain is a crucial option for health data, safeguarding health records and clinical outcomes and ensuring regulatory compliance when standardized practises get achieved. The use of smart contracts demonstrates how this technology can support business logic and ,by taking leverage of other technologies, potentially enhance processes of real-time monitoring of patients ,medical interventions, enhancing patient data security, reliability, privacy, availability and management of the infrastructure. It has the power to manage patient data, enabling the exchange of medical records between different healthcare institutions while maintaining a record and control over access to this data. Another application is related to the pharmaceutical supply chain and developing measures against counterfeit drugs. Despite the substantial costs of developing new drugs, including trials assessing the drug's safety and efficacy, smart contracts facilitate the informed consent procedure, improve identity management, and enhance data quality. Providing patients with access to manage their identity also allows the integration of the informed consent procedure, ensuring the privacy of individual health data [8].

According to **IBM**, 70% of healthcare leaders predict that the most significant impact of blockchain in the healthcare domain will be in improving the management of clinical trials, ensuring regulatory compliance, and establishing a decentralised framework for sharing electronic health records.

2.4 Key Blockchain Benefits/Features When Applied to the Healthcare Sector

Speaking about blockchain influence within the healthcare sector. To understand why this technology can be viable for biomedical and healthcare applications, the key benefits and advantages

it presents to the sector are respectively data integrity, data traceability and data security due to the logic introduced in smart contracts and it's cryptographic schema and the flexibility of supplying data, because the ledger is like a puzzle and when you share data the recipient can check if the data is actually valid using this puzzle. Furthermore, with the ongoing evolution of the healthcare landscape, blockchain technology plays an increasingly vital role, providing innovative solutions to tackle various challenges within the sector [9] [10].

A. Authentication

Blockchain provides secure storage and ensures the authentication of records or other private information stored in blocks along the blockchain. This authentication process requires a specific private key linked to a public key to initiate the creation, modification, or viewing of information stored on the blockchain.

B. Immutability

This technology supports only creation and reading functions, making it challenging to alter data or records. As a result, blockchain is well-suited as an immutable ledger for recording critical information, including insurance claims records and confidential patient data. Immutability is established via a consensus mechanism whereby a network of nodes must collectively validate transactions before they are appended to the blockchain. Transactions are linked to each other which make a block and a block is also connected to the previous block, all cryptographically, thus creating a chain of blocks that reinforces the security and integrity of the recorded data.

C. Provenance

The owner can only change ownership in the blockchain, following cryptographic protocols. Furthermore, the origins of assets are traceable, allowing for verifying sources and data records. This traceability enhances the reuse of verified data. Therefore, the blockchain is well-suited for managing critical digital assets like patient consent records. This traceability is achieved through cryptographic techniques, ensuring a secure and auditable history of asset ownership changes.

D. Robustness/Availability

Each node in the blockchain has a complete copy of the historical data record. This characteristic makes it well-suited for the preservation and continuous availability of important records, such as patients' electronic health records. Data distribution among nodes ensures redundancy, contributing to the blockchain's resilience and reliability for long-term data preservation. Combined with other existing technologies, this concept of Availability can be even more improved, making single nodes available even tought some failures occur.

E. Security

Blockchain technology can be a powerful allie of healthcare security, since it leverages advanced encryption, distributed consensus and immutability if everything is well designed. Putting all features together, it may represent a strong resource when it comes to cyber threats, malicious actors within the organization and lack of integrity in the possessed data. To put it more clear,

advanced encryption preserves data confidentiality, distributed consensus can prevent malicious actors from modifying the data if well designed and immutability, due to this encryption linkage that is usually assoated to a puzzle, can indeed make sure that the data is not tampered.

F. Privacy

Privacy comes along as another big pillar that, not only is important within the healthcare landscape, but also plays as a huge concern in every realm of information management. Blockchains usually come with a certain number of features that ensure privacy. In some types of blockchains the privacy is achieved by not knowing which person owns what (public), while in the other hand the privacy is achieved by **ACM's** (Access control mechanisms). Regarding the second way to achieve privacy,more concretely in the healthcare environment, this achievement is filled by facilitating secure sharing of medical information among authorized parties, enabling granular control over the access of healthcare data and by ensuring pseudonymity in transactions which by extension gives the blockchain usage benefits like authentication,immutability,provenance,robustness,availability,secur and privacy where all together improve security,transparency and privacy.

2.5 Implementation Requirements

To successfully implement a blockchain, a certain number of aspects must be fulfilled: interoperability, security, consistency, integrity, data immutability, cost and resource effectiveness, trust and transparency, and complexity [11].

These requirements are a must for establishing the groundwork for a resilient and efficient blockchain system tailored to healthcare needs which means that they are shared across multiple blockchain implementations. The detailed examination of each requirement will provide insights into the necessary considerations for successful implementation in the healthcare domain.

A. Interoperability

Any organization and ,more specifically, healthcare systems encounter challenges ensuring interoperability, which represents a major barrier in health management systems. The lack of universal standards contributes to this issue. Altought standards are discussed every day, blockchain technology can mitigate this issue by enabling the interoperability of electronic health systems with current healthcare systems.

This integration comes facilitated by blockchain due to the flexibility of most of the implementations(ex: **Hyperledger fabric**), where health information can be exchange across different systems, according to a before hand setting of rules, fostering a more interconnected and efficient healthcare ecosystem in a controlled and secure environment.

B. Data Security

In cases where multiple entities are involved, there must be a way to ensure that data is secure. In order to ensure that this data is actually safe, there is the surging of Blockchain because it can ensure higher security, privacy, and trust among multiple parties compared to traditional healthcare systems. This is mainly because of the characteristics that were mentioned before.

To mention the main reason for this to happen, increased privacy and trust by all entities is achieved by a consensus of nodes where the rules for establishing who can do what and what is determined as a valid transaction is also something that must be agreed between a consortium of organizations, where the number of organizations that must agree in these definitions depends also on the agreement of the involved. This creates transparency, is more easier to setup than by using a conventional technology because everything is just well and previous thought and enhances the overall security of patient data, instigating for a system where every action within the consortium or even outside is verifiable , traceable across the network and thereby valid.

C. Data Integrity

Lack of data consistency of medical data is a major problem in the modern landscape. Data inconsistency can cause delays and higher costs in completing the overall health process for any patient, not speaking about research bad results since researchers that want to improve patient service rely on this data. Therefore, a blockchain-based healthcare system must ensure that health data is consistent and cannot be altered by unauthorised entities or malicious actors.

By correctly leveraging blockchain technology we can possibly guarantee data consistency, integrity and immutability by creating a tamper-resistant record of all transactions. This enhances health processes, providing a secure and unalterable history of each patient's medical data and even possibly setting rules to accept or dismiss information if the network provides something such as a smart contract, therefore providing integrity at the max level.

D. Cost and Resources

Currently, healthcare systems are consuming more resources in terms of costs, computations, time, personnel and machinery. For example, in most transactions, the presence of intermediaries may lead to more delays or resources required to perform specific tasks. One of the key requirements when implementing a blockchain-based system is the possibility of reduction of costs and transaction delays that can be caused by third parties, multiple entities involved or any other reasons.

Blockchain technology increases processes effectiveness by eliminating intermediaries and possibly reducing the overall resource consumption, resulting in a more cost-effective and efficient healthcare system.

E. Trust and Transparency

With the current healthcare system, it is necessary to build trust among various stakeholders to ensure more secure data storage and sharing processes. Maintaining complete trust and transparency becomes a significant challenge as data is stored and shared among multiple parties. A blockchain-based system eliminates the overhead of intermediaries and creates a deterministic source of data thus building more reliable and transparent healthcare systems. By eliminating intermediaries, blockchain establishes a system where each transaction is verifiable, traceable, and transparent, building trust among stakeholders and preserving the integrity of health data.

This becomes all possible due to the cryptographic nature of blockchain, which creates less human interaction and by extension creates error prone processes and also removes probable bad actors that may be within those processes, which creates deterministic operations by extension.

F. Complexity

Current healthcare systems involve multiple stakeholders and are more complex in storing, sharing, and processing medical data and other billing-related information. Therefore, one of the requirements for a blockchain healthcare system is to have fewer complex processes to prevent unnecessary complexities and delays at various stages of the process. Simplifying processes within a blockchain-based healthcare system streamlines the overall workflow, reducing complexities and enhancing efficiency. This ensures smoother stakeholder interactions and minimises delays in critical stages of healthcare processes.

Altought the processes can become simpler, there must be a conscious thought from the organization that depending of the design of the blockchain solution the system may become complex. Standardized approaches, operational guides and recovery plans usually are ways to also remove this complexity more into the nature of managing the network.

2.6 Implementation Challenges/Issues

Issues and challenges must be considered when contemplating the implementation of a blockchain-based health system. The challenges identified in the research are listed and detailed throughout this section.

A. Security and Privacy

Looking to the security aspect, there is a concern about ACM's (Access control mechanisms like for example ACL's),authentication and non-repudiation which can be a extreme barrier when it comes to offer overall data integrity,confidentiality and availability of this sensitive information. Restricted access to controls, querying ,writing and provide a correct logic to handle information are the main mechanisms to making sure we achieve the wanted result, which can be challenging since there other problems associated with this. As a example, non-standardized encryption protocols create heterogeneous processes between organizations and even within the same organization, imposing a challenge when it comes to access information [12].

B. Interoperability

Interoperability is a crucial requirement for the healthcare sector since the proper organization or multiple organizations needs to access this data, either for own usage,investigation,giving information to a supplier or any other entity that requires the usage of a given slice of data. This involves sharing and transferring data between different sources, which is not optimal. The biggest issue with interoperability is , in the current methodology, that data storage is done in different sources within the healthcare institution. This is due to the fact of how Centralized data storage operates and it poses a problem for healthcare providers as it consolidates all health records in a single point but with multiple data sources. This becomes a issue arising from fragmentation of health data, slow access, and the quality and quantity of data for medical research.

Many records are generated daily and stored centrally in different hospitals. Since the data

is stored across various hospitals this may result in the loss of data and the patient needs access to the contained data. Blockchain technology can come into play to ensure interoperability among all hospital systems, since it helps sharing data in a secure and effective way. With its decentralised,distributed ledger and the possibility to implement various rules and business logic in the processes, blockchain faces the challenges of centralised data storage because it allows more efficient and secure health data exchange between different healthcare entities, ensuring data integrity and accessibility.

C. Data Sharing

Data sharing is a huge problem when considering healthcare records. Sharing these can be challenging, as unfortunately the records of an individual may be stored in various locations. In order to a Patience have a comprehensive view of it's state, behind the hoods, it must gather records dispersed across multiple healthcare units which sometimes is not feasible and,also unfortunately, this also applies to healthcare providers as they need access to up-to-date patient data if records are located elsewhere. Since this records are dispersed in a vast number of institutions, this becomes a challenge for seamless sharing, due to bureaucracy and because it's hard to gather information from a vast number of points.

Now, this becomes even worse if some data is not meanted to be shared to certain entities. As instance, hospitals and insurance payers may be reluctant to share data too easily with other entities. One reason could be that the hospital wants to keep the prices at which it gets the products for it's activity. Thus, mechanisms must be created in order to tell what which entity can access from where. Blockchain offers this if well implemented, creating a enhance data sharing in healthcare by establishing a secure and transparent mechanism. It enables a decentralised approach to data storage and sharing, ensuring that stakeholders have controlled access to relevant and up-to-date information while maintaining data integrity.

D. Mobility

Since there is a substantial rise in smart devices,sensors and other kind of technologies, mobility is a important characteristic that the data must have. This becomes very hard since by moving data we must make sure that it remains secure and that it complies with legal requirements. With this in mind, blockchain is a suitable solution to have this moving of data respecting the business logic, empowering users with real time access to their medical records, ensuring the best security standards.

E. Transparency and Confidentiality

The transparency of information during a transaction is also very important, altought often seen as a drawback. In a blockchain, everyone can see everything, resulting in higher transparency but lower confidentiality. Even despite using hash values for anonymity, users can still potentially be identified by analysing transaction publicly available transaction information. This poses a threat particularly for critical for healthcare applications where patient-related data is sensitive and confidential.

F. Speed and Scalability

A real issue is real-time healthcare applications based on blockchain. This is because transaction times can vary depending on the protocol used, number of nodes within the consensus, network, current volume of transactions and other constraints. In other words, there must be careful thinking in terms of what it is needed and the amount of decentralization that we want so it does not affect the available computing capabilities and the volume of medical transactions required for such healthcare systems. Also, by recurring to horizontal scaling there may be a possibility of increasing the number of components that serve requests without actually increasing the number of members of the consensus, which could be valuable in certain situations but further benchmarking is required to see until each measure it can help.

G. High Development Costs

Healthcare systems based on blockchain can incur high development and operation costs. Altought the solution excels in flexibility, refactor the way the things is currently done takes resources. With this in mind, understanding how can it be done at the minimum usage of resources is something that is seem as challenge but there must be careful thinking on this matter.

Standardisation

Because there are a lot of uncertainties along the way, it's always important to seek standardization. This is very important, since standards usually occur from given situations that were faced before and it becomes precedent to actually use already well established best practises that faced the same exact problem, precisely because they were proven to solve the given problem. With this in mind and ,specially concerning the healthcare, there must be specially defined what types of data, size , format and other kind of characteristics can be recorded into the blockchain. Also, there must have clear guidelines that which data should be stored on-chain or off-chain. Having on hands these kind of information is more than greater to ensure integration between blockchain technology and the healthcare systems.

Cultural Resistance

Stakeholders that were used to other ways of managing data may present resistance to change. Is it either by relying on paper-based procedures and Electronic Health Records or other online services, this personal is even more skeptical to actually have a platform that makes sharing of data something common, which can be revealed as being a hell of a challenge, also because there was also times were a hybrid paper-based and online services were actually useful, because there were cases where the online services stopped working and because they had the paper from older stakeholders, they actually were able to function almost at 100% besides reveiling all of the benefits we also must present blockchain as something that hardly could become unavailable and availability itself is something as much harder as changing people's minds. To put it clear, it is important to gain broadly acceptance of this new way of doing things but also deserve the trust from the stakeholders.

2.7 Applications of Blockchain Technology in the Healthcare Sector

Since this technology is emergent, it has been still gathering interest by the researchers, but fortunately more and more research use cases are surging and this may result in further future solutions that can benefit the healthcare landscape. The most discussed applications of blockchain in this precise space is for information exchange,transactions between patients,transactions between providers,transactions between payers and transactions between a whole other number of relevant parties. The potential of this is huge and there is plenty of expectations regarding the further usage of this concept. With this on scope, and because there is some research about this matters, there is mentions to the exact current known applications more ahead.

A. Electronic Medical Records

A current usage of the blockchain technology is for management of eletronic medical records (or EMRs), often also called personal health records (PHRs). These records are usually meanted for creation,storage and management of the patient data. Blockchain, if implemented, can take a huge responsibility in the management of this data, determining exactly how it is shared, how it is processed and how it is used. Now, because of the nature of the immutability of this technology, the risk of tampering becomes almost zero because once recorder neither the doctors or the patients can alter this data. Also, because of cryptography and business logic, data will for sure only be recorded if it is actually valid thus enhancing the integrity of the data in question not speaking about this could actually influence the security and privacy of the data since the data can only be decrypted by using the patient private key for example, but this depends of the cryptographic scheme that is in place. Key attributes such as decentralization, immutability, data provenance, reliability, robustness, smart contracts, security, and privacy make blockchain an ideal solution for storing and managing EMRs. As an example, there is a application of this in MedRec, a decentralized record management system developed by Azaria, Ekblaw, Vieira, and Lippman. This solution enables patients to access their medical information in real time across different healthcare services and treatment locations. It gives the power to patients to securely share their records and decide who the access of it is guaranteed. Besides this and ,because of the modular design that integrates existing local data storage systems, there is interoperability for both doctors and patients [13].

B. Clinical Data Sharing

Clinical data sharing is another crucial application of blockchain in the healthcare sector. It enables secure data exchange facilitating data sharing among various entities in the system. Confidential and crucial patient information is stored within electronic health records (EHRs) and electronic medical records (EMRs), highlighting the need for secure storage, processing, and access. Sharing this data among healthcare stakeholders is essential for improving service quality, but it also requires robust transparency and accountability measures. By leveraging blockchain these challenges are gone, because there is a record of all this transactions in a distributed ledger creating security and transparent data sharing. All participants can view the transactions, which creates a friendly environment for managing clinical data in a secure way.

An example of this is applied to clinical data sharing is HealthVerity. This solution leverages

blockchain to enable secure and transparent clinical data exchange of data between healthcare organizations,payers and research organizations which promotes further collaboration in research and clinical analysis. With this, the organizations can deal easily with data access, interoperability and privacy. By creating this environment, this project supports research,analysis and data-driven innovation within the healthcare sector which influences several future work that enhances the healthcare services [14] [15].

C. Global Data Sharing

The most frequently spoken subject while speaking about blockchain is data sharing, but a even more extensive subject is when we apply this at the global context. If applying this concepts regionally is hard, imagine trying to expand this standardized solution to multiple organizations across the globe, the potentiality is immense but so it is the complexity of such systems. Imagine if a patient wants to be treated in a foreign healthcare provider, this means lots of bureaucracy just for this institution to have access to this patient data which takes time that sometimes lacks. By recurring to blockchain, all of this bureaucracy can still be done but quickly as fast as the procedures in a chaincode can be, depending of the implementation in use of course, and still give patients the control to decide who can access it's data, which is a requirement to ensure safe and effective treatment abroad. In simple terms, this technology can deliver optimal care regardless of the location.

An example of this is Health Wizz, a platform focused on healthcare and it leverages blockchain for supplying secure and transparent solutions for storing and sharing healthcare data across the globe. User have the opportunity to store and manage their health data, which includes medical history, test results, prescribed medications and other resources. Health Wizz creates immutable and tamper-proof records, ensures reliability and authenticity of information, data integrity, privacy and secure sharing between participants [16] [17].

D. Medical History Maintenance

Having a medical history is something particularly interesting and important, because it keeps track of patient records which gives insights about, for example, when a patient visits multiple hospitals and there a need to avoid redundant analysis over the patient like for example doing the same analysis multiple times. If this history is well maintained and shared across institutions, then there will be the reduction of redundant processes when treating the patient, reducing costs,risks and time needed for the observation.

Blockchain offers a reliable way to maintain and store this history and , at the same time, offers more power to the patient which chooses or not to share its personal information. This increases privacy and consent management. In other words, it offers a extra layer of trust preserving the integrity of medical records over time, preventing as well tampering and unauthorized alterations. Medicalchain, a technology company, leverages all of this to offer accurate data record with meticulous updating of the data and its history. This is very valuable if a patient goes to multiple healthcare institutions.

E. Health Data Access Control

The more data retained, the more difficult it is to secure it and because of this there is a need to

create mechanisms to make users aware of who accessed their data, because by only giving them power to decide who sees it is not enough since sometimes who can use the data can be inside of a given group and their access can be camouflaged making the user more vulnerable than ever. Therefor there is a need of giving this power to users: allow or not allow some entity to access its data, check who is seeing its data, check who is storing its data and check who is sharing its data. Blockchain helps by giving potentially all of this to the user, simply by leveraging smart contract to enforce rules and permissions.

An example of this application is Coral Health, which uses blockchain to make easier the secure sharing and access to patient information. Patients can trust that their medical records remain the same and the platform also connects them to other participants in the healthcare service such as doctors,scientists,lab technicians and public health authorities. Using the blockchain is a huge life savior in this situation because besides of the benefits that were mentioned earlier, there is also administrative automated processes and deterministic operations that are error prone for accurate data and treatments for the given patients.

F. Billing/Payments

Payment system may be complex and prone to fraud, leading to requiring more resources and time than needed. To face this problem there are usages of blockchain to simply and speed up the payment process when comparing with traditional payment methods. Using blockchain, invoice processing and payment processes flow faster and this removes barriers from the activities, there is no need for intermediaries, prevents fraud, helps stakeholders access the data and makes the data more secure,private and less prone to cyberattacks.

Blockchain has a powerful tool called smart contracts - Self-executing agreements with terms encoded directly into the system. Doted of this, the automation of processes is possible because smart contracts provide all the business logic needed to come up with what is needed in that flow and by extension it also reduces administrative burden, minimizes errors and disputes, contributes to more efficiency, gives more transparency in the payment ecosystem, provides security and fraud-prone payments.

As an example there is the project PokitDok, a platform which uses blockchain to facilitate apyments and transactions in healthcare. Within this project DokChain was developed, a private blockchain that references off-chain file systems that, by incorporating smart contracts retrieves data from health insurers and payers in real time, enabling real-time decisions on insurance claims. After a patient treatment, all of the information about the treatment is available to all stakeholders and thereby there is a calculation of the given costs by leveraging smart contracts. Also, the platform makes calls to API's to assess a patient's payment risk, enabling patients to schedule and pay for healthcare services, automating payments and verifying benefits instantly.

G. Biomedical Research and Education

When it comes to do research or even gather information that make people conscious of a given reality for the sake of education there are problems that must be faced. Especially in healthcare, there are questions regarding the acceptance of the participants for them to land their data. Also, in cases where the sources of data are not that valid, data can give bias to those that analyse it, which unfortunately is good for a certain number of malicious actors that

just want to make sure that their point is correct where in reality it isn't. To go against this problems, not considering the analysis of the source and doubting the author in question and it's purpose, there is the possibility of using blockchain which can serve as a powerful allie against falsification,reducing the omission of unfavorable clinical research outcomes and maintainability of data integrity. Also, by using historical data which the blockchain maintains, there is the possibility of questioning current surprising results, giving us the possibility to question what caused such variability in them, thereby aggregating hints about the possible bad introduction of those inputs.

An immutable record of patient consent enables regulators to monitor trial patterns and ensure compliance with informed consent regulations. With this, there is also the possibility to detect fraudulent form consents, which are a form of clinical fraud, resulted from record tampering and fabricated patient consent. If implemented well, blockchain can make sure that only patient permitted participants, can access or use in multiple ways subject to also permits it's data. Thereby, for this concrete situation, there must be mechanisms to consent,revoke consent,see which entities are taking advantage of the consent and this must be per item that the user retains. This not only gives power to the patients, but it removes a lot of bureaucracy in releasing their data for better purposes like research and it also helps for auditing clinical teams,researchers and regulatory authorities.

As an example, there is this project called Shivom Project, which aims to implement blockchain in genomic data technologies. With this in mind, users of this platform can store, share and monetize their genetic information. By leveraging this large genomic datasets, the project aims to make research and treatment discovery easier, while participants maintain control over who uses its data or not [18].

H. Pharmaceutical Supply Chain

When it comes to the supply chain, there are a lot of mentions to blockchain but in a particularly more focused in healthcare perspective, there is the case of the pharmaceutical industry. This is because, unfortunately there are cases of counterfeit or substandard drugs that can cause harmful issues to patients.

By using blockchain, every transactions that involves probably risky products is recorded from the production,distribution and the delivery to the end consumer. This occurs because, in case of malicious intent within this chain, there are records that give hints about where the product came from, thereby actions can be taken in order to solve this kind of issues. With this in mind, there is a barrier against drug alterations and modifications within the chain, counterfeit drugs and stealing. With this, poor quality drugs are easily tracked until their source, helping healthcare institutions to meet safety and regulatory standards in pharmaceutical supply chains.

As an example, **IBM** launched projects with pharmaceutical manufactures,distributors and hospitals to demonstrate the feasibility of blockchain-based solutions [19].

I. Equipment tracking

Equipment tracking is closely related to supply chain, the difference is that this concerts the life after the delivery process and it concerns who retains what within the organization or what does the stakeholders retained before. Because of the immutable nature of the history inside

of blockchain records, blockchain stands as a huge countermeasure against theft,loss,movement and reordering of medical devices. By preventing alteration and deletion of the location history, there are plenty of dollars that are saved, making this solution a tamper-proof against this unfortunate situation.

Additionally, in case the usage of this technology is guaranteed, there are less costs associated with personal and less errors occur due to their deterministic nature making the life easier for those that require this equipment's for their work like nursers,porters and staff support. As an example, Chronicled exists as a project that focuses on supply chain solutions but that also offers solutions around product tracking. What makes this product more interesting is that product tracking works as a extension of supply chain tracking which goes in accordance to the closely related relationship that was mentioned before. Putting in another words, this product leverages blockchain to not only track products in its supply chain but also to track the products after the delivery, which create immutable,transparent and valid records that provide reliable audit trail for each product [20].

J. Health Data Analysis

Data has become something that is more valuable than gold, because it can help us understand the best way to operate in various realms of the human devour. It spots behaviors, characteristics about something that no one spotted and helps businesses to meet optimal solutions for difficult problems. In order to achieve this, not speaking only about data itself, analysis must be done and, in order to make good analysis, data must be also of fine quality in terms of integrity, verificability and ownability. If done correctly, data analysis can revolutionize the healthcare field in such a way that patterns can get identified, medicine can be tailored to patience characteristics and preventive measure can be taken.

Blockchain can take a huge responsibility in terms of making this possible, because it safeguards the integrity and privacy of the data, addressing also concerns around security and confidentiality which can lead healthcare institutions to significant breakthroughs in medical research and patient care becomes better. As an example, Nebula Genomics, makes analysis of genomic data that comes from a blockchain and , in this precise way, reshapes the genomic data market. It creates a market where buyers, such as sequencing facilities, drug design companies and health organizations, can access genomic data. Individuals use this platform to discover their genetic variants,disease predispositions and get compensated by allowing third parties to access their data, while still maintaining privacy [21].

3 Methodologies and Tools

Since the methodology used has a bearing on the success of the project in hands. It is of high importance to choose which practises should be adopted to effectively move forward. By means of this, investigation under which arsenal were to use is counseled. After a lot of digging, it was decided that the number of different practises should not be to vast, otherwise it would be a burden to put them to work together seamlessly.

Having a research methodology is full of advantages like: helping **plan**,**structure** and **conduct** the research in a way which is structured and objective.

Research methodologies can largely be categorized into **quantitative**,**qualitative** and **mixed**. **Quantitative** focus on gathering numerical data and performing statistical analysis to draw conclusions. **Qualitative** on the other hand, is more suited to specify detailed insights such as behavior,motivations and experiences, which means that is more theoretical while the **quantitative** is more empirical. Finally, there is also the concept of **both** which stands for having both **quantitative** and **qualitative**.

During this thesis, in terms of research methodologies, there will only be space for two practises: **DSR(Design Science Research)**, which in terms of nature of findings is hybrid, thereby categorized as a **both** approach like mentioned in the last paragraph and **SWOT** which is more **qualitative**. The first one is more complete and more suited for the projects while the second will be used more to drive the conclusions of the project to further analysis of it's pros and cons of usage of the project in cause. With all of these settled, lets move to deeper thoughts about each.

3.1 DSR

As an important gadget for the project, **DSR** was the embraced tool. Considering it's traits, resembling extreme focus on solving real problems, it represents indeed a powerful tool for going on with this thesis. Some people gain knowledge through reading while other's achieve so by earring,practising or even speaking with other's. It's a serious challenge understand how a team will work best but,due to the nature of the team, **DSR** proved to be the best choice for it's phenomenal practise and rebuild model. In a distinct set of terms, this methodology is notably better for individuals which praise more the practical way of things and by it's landing into the creation of artifacts for solving the current landscape of problems, creating this way solutions and still gaining theoretic knowledge, it does outpace other methodologies making it better for the expected aspiration of this project.

1. Goals of DSR

For the verbal reference of the **DSR** goals, there is a colossal number of concepts that could be spotted as one. The gross of it, outcomes from the requirement of appeasing certain field such as information systems,engineering and others. This project is within the information systems realm, which means that these values are more than fit for the project and thereby qualified to be mentioned here. By means of this, the most captious goals will be pronounced here.

At the heart, **solving practical problems** is one of those goals. This is reached by the design and creation of such research artifacts that are implemented into practise, where those are seen as the form of models,frameworks,methods,processes,systems and even tools. With accordance to this means, practical problems are solved and theoretical knowledge comes as extension.

Besides that, there is also the goal of **creating innovative artifacts**. It's presence is requested when it comes to push the horizon of existing knowledge and capabilities. As Innovation takes precedence in any kind of project, it could not be different when it comes to the realization of this research, standing as a must for the **DSR**.

Tertiary to this, surges the **contributing to Scientific Knowledge**. Which is conven-

tional, if further thinking is done through the definition and name of the methodology. Knowledge occurs by extension when depared with the solving resolve. Thus, the more knowledge extended bigger the contribution and new perspectives and ideas may flow, creating a avalanche of future projects, instigating development and furthermore improving sectors.

Onward to the next objective, there can be **Evaluation and Validation**, since the evaluation itself is a pillar for the loop created within the **DSR** realm and so it is validation but, altought valid, it does not mean that it should be final. On this assumptions, rigid considerations must be accommodated, however, solutions keep maturing which explains the continuous satisfaction of this precise objective therefore nothing remains fully complete, which is expected from the point view of innovation.

Another objective important to consider is **Balancing Rigor and Relevance**, where it stands glamorously within **DSR** to make sure that nothing becomes to much theoretical until the point there is no practical implementation. This is a risk in the research world, where by applying to much importance to the theory category, the project starts loosing color into it's true serving purpose. With this in aim, exists there the importance of instituting a balance between the two dimensions, emphasing both the theoretical inception but also the practical. This becomes the best approach, since the practical determines the verocity of the empirical sphere.

Lastly, there is the existence of another aspiration, denoted by **Generalizability of Solutions**. It's priority resides within the sphere of making sure that, knowledge produced within the project can be adopted into broader contexts, from a deck of various industries or scenarios. This bearers a huge relevance, because by having countless scenarios of usage,new use cases are produced, thereby relevance of the knowledge becomes enormous and so it's contribution for the research world, which affects the world by extension, main desire of the research itself.

2. Artifacts in DSR

Artifacts are a paramount in the **DSR** methodology. It is labeled as any **designed object**, that is conceded for deal with practical problems that a research may be centered in. Relatively to it's existence, it takes care of not alone living as functional tool for combat problems, but also as a source of new knowledge. The achievement and evaluation of **artifacts** is what most diverge this methodology with others that only concern understanding and explanation. In this section, it is planned to delve into each of the **Artifacts**. They all have their own particularities, but they are described as the following:

A. Constructs

Constructs are the genesis block, for describing elements and solutions in the sphere of the problem. They provide a way of communicating the building models and solutions.

As an example, within the realm of information systems, constructs could include the terms "**user**","**data integrity**" and "**security protocols**".

B. Models

Models on the other hand, act as an abstraction representation or even framework, that illustrate how things play within a given realm. They exist, so there is awareness, predictiveness

and even cleanness of phenomenons, thus helping into guide developing new solutions.

As an example, there is the inception of a **model that can optimize the supply chain operations, which may include insights into inventory levels,lead times and even order quantities**.

C. Methods

Methods are procedures or tactics, that have the power to dictate how to come up with a solution for a given problem. They serve well, when it comes to standardized common and multi used tasks. By following this guideline, tasks are made with quality assured, making the execution more effective.

As an example, within the software realm, a **new algorithm** can be given with distinction for ordering data in a more desirable way, by taking into accordance the needs of the scenario. This algorithm may not be the most efficient but the most efficient for this precise case, making it tailor-handed specifically to the scenario in question.

D. Instantiations

Instantiaton is a occurrence, which results from the application of a given artifact. By having a instance of what was conceded during the project, there is a certain data gathering for the understanding of the artifact usefulness. In another words, this is where the evaluation takes notice and thereby there is the gathering of knowledge from it that can occur, either in a successfully instance or failure. In each case, since this is working under the **DSR**, further thinking must be taken to understand until each measure it can have incrementations or fixes. Putting in another words, this is the artifact that is responsible for feedback over the solution, making it a valuable resource within the lifecycle of the methodology.

A software application that implements a new inventory management system in a retail environment is an instantiation of the designed artifact.

E. Frameworks

For the purpose of contributing with a set of high-level blueprints, all designed for understanding interaction between systems, resides the concept of **Framework**. This represents a well formed approach for combining concepts,models and methods.

As an example, this can be a **framework for decision-making in healthcare**. It might give insights about patient data,clinical guidelines and doctor input but if this becomes actually effective for medical decisions, that's what a real world incrementation of this product may reveal with the given usage. Also, like mentioned before, it can be incremented something more into it, since a product cannot always be considered as final. // ROLES

3. The DSR Process

Entering into the process of the **DSR** realm, there must be a understanding that this is a structured and iterative methodology, focused on construction,evaluation and improvement of **artifacts**. The process is extensive, but it covers all the steps necessary from problem identification to evaluation. Since it works around incrementation, this process is within a loop processing

always new form of solutions and knowledge, since it produces both. Without further delay, this section will be going through each step, covering,giving examples and also refer to its objectives.

Problem Identification and Motivation refers to the first step. Here, there is a preoccupation into picturing a real-world problem which seeks solution. To glean such information, various perspectives must be attended such as the current state,context and impact on the project under scrutiny. The spotted problem should be doted of meaning and with a given degree of relevance. The objective here is to gain power to articulate the problem clearly and make sure the team has the motivation for solve it. A good example to achieve this is in **healthcare, where there could be observed a problem with inefficiencies in patient management**.

As a second step, there is **Define Objectives of the Solution**. After problem identification, it is of huge gravity to think in a given set of objectives for the future solution. The objectives could be qualitative or quantitative, even thought the quantitative are better, establishing them in the beginning is hard,sometimes it is not possible and there are also situations where requirements already give them away. The objective in this step is to set expected outcomes and conduct obligations for the artifact that will be forged. A example for this would be setting as an objective, **creating a system that should be capable of reach 50 request per second. But it could a more simpler thing like giving good data for decision making in a healthcare facility**.

Design and Development of the Artifact surges as the third step. When landed in this step, there is work to develop the **artifact** (model,method,system or framework) that will be managed to meet the objectives created on the second step. With the problem and objectives on hand, the team must create,conceptualize and refine it's artifact to better meet the expectations. In another words, the objective is create a artifact that addresses the problems. **Developing a prototype of a healthcare data management system** can be a good example of what is expected in this phase.

Moving to the forth step **Demonstration**, here the **artifact** is assigned to work, by either creating a simulation or simply putting it in a real world system to check if it does meet the expectations. This stage is critical, because it serves as proof of concept to inquiry if it has utility or not. Regardless of the result, knowledge is generated and the **artifact** has room for growth because of the feedback created in this phase. The objective is to demonstrate if the problem can be solved or not with this implementation by a practical setting. One example of this would be the **deployment of a healthcare management system**.

Evaluation on another hand, corresponds to the fifth step, where the evaluation of the **artifact** takes place. In this phase, the product is evaluated. Rigor,effectiveness,efficiency and overall impact are considered, taking into consideration the problem and that the evaluation method may defer relatively to the nature of the given product. This step has the objective of measure if the **artifact** solves the problem and meets the objectives settled in the earlier stages. As an example, the **evaluation of the performance by tracking the key metrics** can be considered.

At the realm of the sixth step, **Iteration and Refinement** is the step right after the evaluation. This is because, based on the observations, there may be areas where the **artifact** still lacks which compel for improvement. With this in mind, iteration is done and evaluation

and demonstration occur multiple times until there insignificant improvement. As an objective, in this phase, there is the max refinement possible of the **artifact** recurring to iteration. To illustrate this, there is the possibility of image a case where **after testing, the system might be updated to include additional data fields or enhanced user interface features to improve usability**.

Finally, in the sept and last step there is the **Communication** phase. Within this, the research findings are shared, artifact design and theoretical contributions are released and the form it takes are **academic papers**,**presentations** and **reports to stakeholders**. Deconstructing this, here there is the documentation and dissemination of the findings, including both practical and theoretical contributions. As an example, there is the **Publishment of a paper describing some development in healthcare made by a given research**.

4. Challenges in DSR

Since this is a very important methodology within the scope of this project, it was very important to inspect probable challenges that could occur. Despite powerful, **DSR** also comes with it's achiles heel's. By recognizing it's limitations, there is plenty of room to achieve that the outcomes are actual suficient to satisfy the need of practical and academical value. A set of challenges will be described below:

A. Balancing Rigor and Relevance

Balancing Rigor and Relevance is the first probable challenge. Since this evolves a research, there are always 2 major risks to be shown: Presenting to much rigor and less practise or presenting to much practise and no theoretical presentation not soever. This becomes a problem because if doing research, the desirable outcome is to have both contributions and in case of to much rigor, there may be lack of practical implementation and, in another hand, to much practical could lead to few theoretical representations thereby no knowledge is passed. Reaching this balance imposes a burden but if managed harmoniously, it benefits not only the case under scrutiny but it also benefits others that may want to use the same idea in another context,influencing this way other sectors and gaining more recognition from the work.

B. Problem Complexity

When it comes to **Problem Complexity**, **DSR** does go along with it but with extreme caution. This happens by virtue of sometimes becoming to hard to use, due to the hardness of understanding which problem is within a to broadly concept situation. Determining the appropriate scope of the problem is demanding and sometimes it can lead to a over complex artifact but there are countermeasures that can possible bring up a cure. This remedy can range from deconstruction the problem to high level representations but the idea here is to understand that this does come as a limitation of the methodology, thereby the need to address this challenge when a complex situation surges.

C. Artifact Evaluation

Evaluating something is not always a smooth task, since sometimes even simpler tasks are a burden for consensus reach. With this in mind, **Artifact Evaluation** is considered also a

challenge but even more in the context of various factors, such as environment detained of much intricacy's, vast variables and the incapacity of controlling external conditions. Also, when it comes to evaluate, there must be specified methods to do so, like mentioned before. This requires time and careful thinking from the team, creating burden and representing a hard obstacle.

D. Iterative Nature of the Process

As stated before, **Iteration** is a important characteristic of the **DSR** methodology. Thus, the normal workflow for which is, in this precise order, **artifact design** and **artifact evaluation** where this is looped all over again for refining. With this perspective in mind, several cycles occur and managing such extended process may become a objection to the project. In behalf of this, refining may represent messing with other existing parts of the **Artifact**, side effects that the management was not taking into account and also the possibility to not completely understand when the **artifact** is already good enough. By taking this into consideration, the team could understand when it is fine to left the loop, in which measure it is safe to implement features and what can be refined with effectiveness.

E. Generalization of Findings

This challenge, comes from the need to dodge the generation of specific knowledge. This is imposed because, while you are conducting research, there is the objective of generating knowledge that can affect society and if this information becomes to much specific, it becomes harder to apply in various sectors having less relevance by not being applied to broader contexts. Ensuring that his does not happen and that the knowledge can be applied to various fields represents a major challenge.

F. Interdisciplinary Nature of DSR

In some **DSR** projects,notably those exchanging interactions with sensitive spheres like healthcare, education or public safety, ethical and social matters can present challenges. Ensuring that the artifact aligns with ethical guidelines, respects privacy and evicts unintended social negative effects is critical but a burden.

G. Ethical and Social Considerations

DSR often requires knowledge from various spheres, particularly when corrective measures for solving complex and real-world complications is included. This vast sphere and integrative approach, while immeasurable valuable, can impose a barrier, as researchers need to develop different sides of view, methods, and frameworks which can represent to communication barriers or conflicting approaches.

I. Time and Resource Constraints

DSR projects often mean long crafting into the design, development, and evaluation phases, all of which require compelling resources (time, funding, expertise). Atoning the requirement of a relevant and effective research process with the boundaries of project timelines and budgets can be a burden.

5. Usage during this project

The usage of this methodology will be more comprehensive during the second use case. This is because the first one is more theoretical and the second is more practical. Since this approach goes around creating knowledge through a practical multiple loop iteration, it does require that the project is dotted of such.

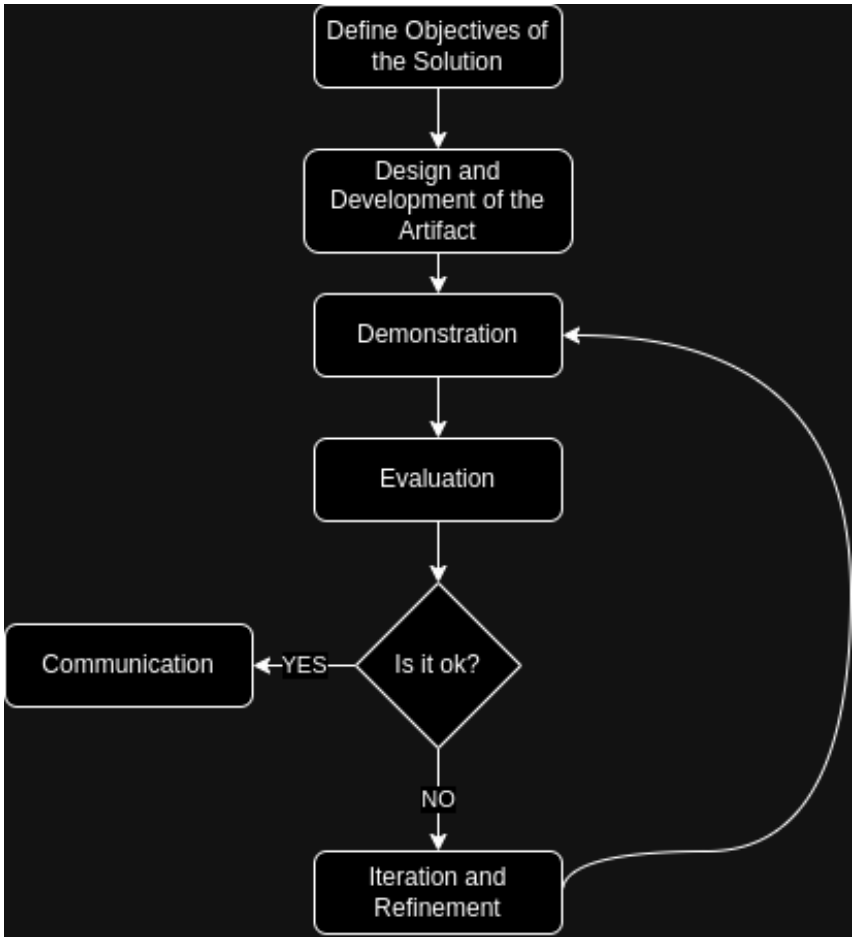

Figure 2: DSR Life-cycle

With this in mind and better discussed further, firstly, within this work there is a definition of objectives for this solution. Secondly, there is the design and development of the artifact, which is a blockchain network. Thirdly, this network passes by a demonstration to effectively see how it behaves. Forthly, evaluations around the solution are made, and in case the solution is already good enough, communication is made; otherwise, there is once again a process of iteration and refinement that starts once again in the demonstration, creating this in a way an infinite loop until the condition of the solution satisfying the project needs is made.

3.2 SWOT analysis

SWOT analysis is a strategic planning tool which is often related to business. This is because it is a simple and yet powerful overviewer, that empowers organizations to set a plan for that precise moment. The acronymom **SWOT**, stands in English for **Strengths**, **Weaknesses**,

Opportunities and **Threats**, which represent the key factors that influence the organizations success or failure. If done multiple times and by several people, it can gain multiple new perspectives that help as a decision factor because it can probably give a deeper understanding of the company competitive position, identifying potential areas for growth and also empower the staff to think better in strategies to address their challenges. The purpose is to provide a well structured framework for decision making, which accesses internal factors like strengths and weaknesses and also external factors like opportunities and threats. Despite it's current usage being more directed to business, it also helps in other kind of projects such as personal career planning and even research. In this project there's the intention to use it. This is because the use cases are complex, having a tool like this can help others understand the current situation, can help the stakeholders of the research to work according to the books and it can give a hint to those interested on the project to discover if it is doing well or not. With this in mind, there is detailed information about it further, explaining each particularity of this valuable tool.

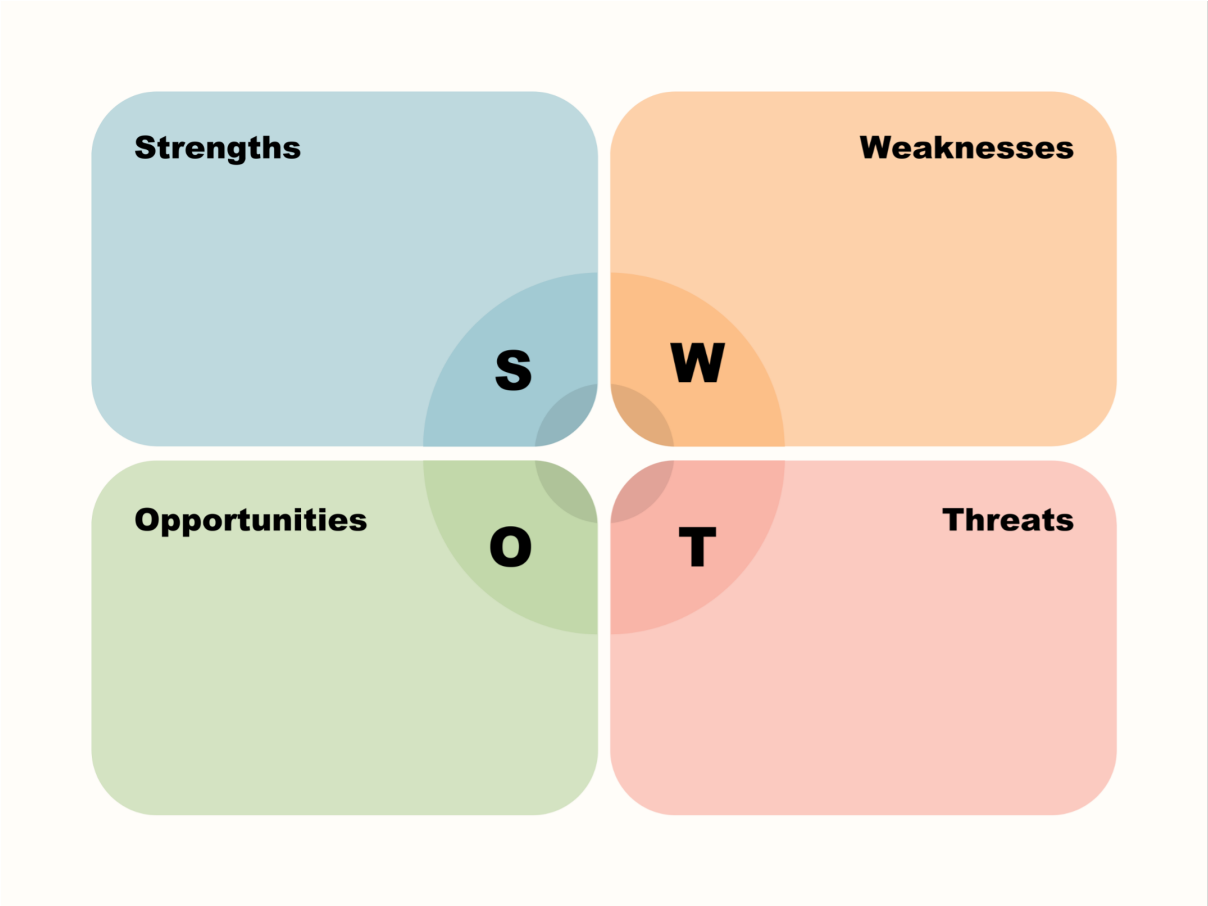

Figure 3: SWOT

1. Purpose of SWOT Analysis

The purpose of **SWOT analysis** is to serve as a decision-factor for organizations. This is done by analysing internal and external factors, providing a clean environment to understand the organization competitive position and it's ability to achieve it's objectives. Putting it in more deep terms, the key purposes include:

A. Strategic Planning

To help developing strategies where the strengths will have more emphasis, weaknesses will be strengthened, opportunities exploiting will be taken and threats will be mitigated. This can be done if the organization aligns their internal strengths and weaknesses with external opportunities and threats.

B. Identifying Competitive Advantage

To outperform competitors, the organization must focus on areas where they have clear advantage. This is done by emphasizing internal strengths, such as capabilities,resources and unique advantages.

C. Addressing Weaknesses

For leaders to address challenges and reduce potential risks, **SWOT** offers insights about where the organization is vulnerable.

D. Capitalizing on Opportunities

For the recognizing and to take advantage of the market trends,technology advancements and emerging markets that could be beneficial for the growth and innovation of the organization, **SWOT** can identify the external opportunities.

E. Mitigating Risks

In order to offer risk mitigation that negatively affects the impact performance of the organization, **SWOT** offers a way to identify external threats, such as competition,regulatory changes or macro-economic downturns. This enables careful thinking about strategies that must be taken.

F. Informed Decision-Making

For developing new strategies, it is important that there is plenty of knowledge about all aspects of the internal and external environment. **SWOT** helps in this matters because, it makes sure that all aspects are considered, therefore this creates more error prone new initiatives or facilitates when it comes to enter new markets.

2. Internal Factors: Strengths and Weaknesses

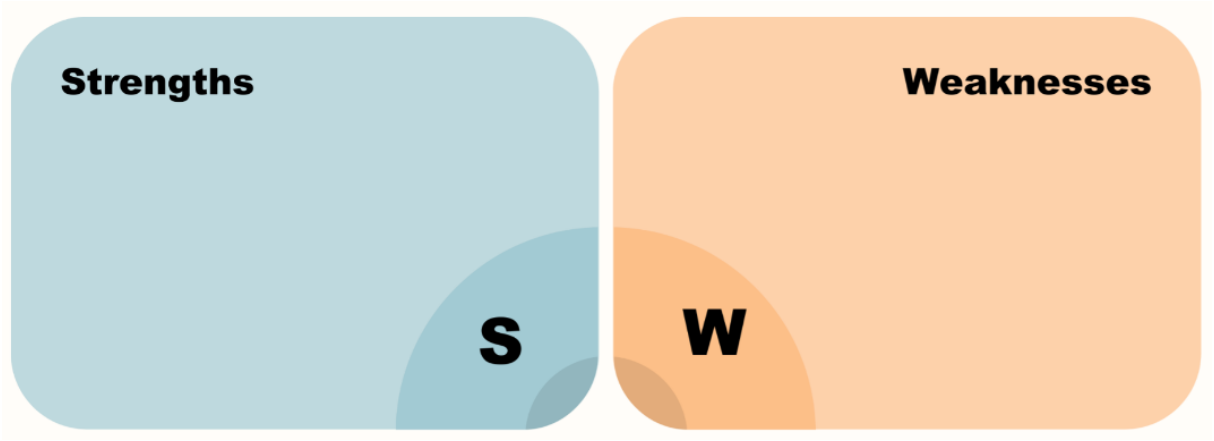

Figure 4: SWOT: Strengths and Weaknesses

Representing the upper part of the **SWOT** standard visualization, there is the internal factors **Strengths** and **Weaknesses**. This is what influences the most, the ability to the achievement of the organization objectives. Understanding both strengths and weaknesses is crucial as a building block for enhancing existing capabilities and check out areas where the organization usually outperforms or the opposite. To put in other terms, the organization has the capability for control and actively manage this two, in a way where the focus is to improve but the decisions to do so are not always deterministic into that direction. With this in mind, let's make a deep dive into each to understand it further:

A. Strengths

Strengths are the internal contributions, so that the organization has a certain number of attributes,resources and capabilities for enterprise level competitive advantage. These are the areas or aspects where the company can outperform others, that can be used to collect opportunities and even defend against possible threats. Identifying strengths is something very valuable for organizations, because it helps in focusing on the best attributes of the organization, thereby fixing their market position. As an example of Strengths, it could be something like: **Strong brand Reputation**, where a well established brand is widely recognized and trusted by consumers. **Skilled Workforce**, in cases where there is a qualified and experienced team who drive innovation and efficiency. **Proprietary Technology**, which are unique or patented technologies that provide a competitive edge in the market. **Operational Efficiency**, for processes and effective cost management that lead to higher profitability. **Customer Loyalty**, a loyal customer base that provides consistent revenue and positive word-of-mouth marketing and finally, more concretely for research, **Having more than enough resources**, which stands for having such resources that makes the research something more than feasible which usually is a huge challenge. By levering this strengths, an organization can maintain or even enhance it's market position, thereby maximize its competitive advantages in relation to others, improving by extension its overall economic performance.

B. Weaknesses

Weaknesses are the core stone of the internal given limitations, for areas where the orga-

nization actually underperforms relatively to competitors which by extension can prevent the organization from achieving it's objectives, by representing a obstacle to the capitalization of the existing opportunities and even by exposing the company to risks presented in external threats. **Weaknesses**, if spotted correctly, allow the organization to take measures for correctness and minimization of the impact that this unfortunate characteristics may pose in future iterations. Addressing this is essential for vulnerabilities mitigation, improve performance and prevent competitors from exploiting these causalities. As an example, there is a lot of thinks that can be mentioned like: **Outdated Technology**, because by relying in older systems or tech can slow down processes,limit capabilities and even possibly increase costs for maintenance. **Limited Financial Resources**, where insufficient capital to invest can pose a problem to initiatives,R&D and further expansion if desired. **Inefficient Processes**, in the idea of resource wasting, higher costs resulted from activities and delays in delivery. **Weak Brand Presence**, which represent a weakness in the essence of reduced market share and finally **High Employee Turnover**, where talent within the company is hard to maintain, thereby results in higher recruitment and training costs.

C. Both combined

By understanding and analysing **Strengths** and **Weaknesses**, there is the possibility of creating a good synergy between both which can result in organizations positioning themselves to achieve their goals. **Strengths** help creating and sustaining competitive advantages and **Weaknesses** ensures that there are no barriers for further success. In other words, considering both interchangeably will imply future success, therefore it is very important to careful think in all of those and afterwards try to see it's relations.

3. External Factors: Opportunities and Threats

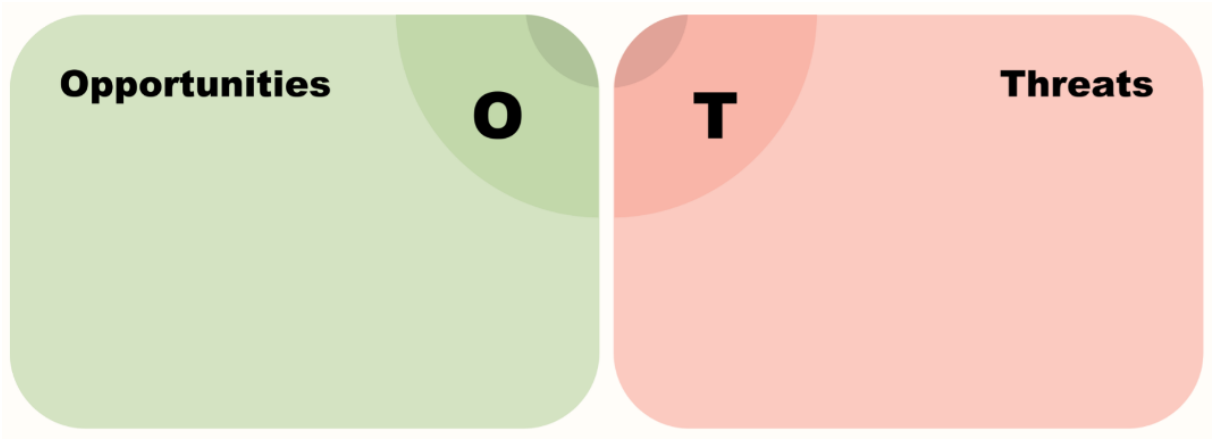

Figure 5: SWOT: Opportunities and Threats

When it comes to external factors, there should be informed that those factors are the ones that the enterprise cannot control, since they don't own them and they are external to their direct manage. This means that despite the organization not being able to coordinate them, they can still take actions to make sure that those causalities don't affect them that much or

that don't affect them in any way. These aspects are **Opportunities** and **Threats** and they play a huge paper in the **SWOT analysis**. Putting in different terms, those are the existences that are based in outside situations that are beyond their control, such as market trends, macro-economic indicators, competition and others. **Opportunities** provide a path for growth, while **Threats** present a potential challenge that could negatively impact the company. A deep dive into each factor will be done in this section:

A. Opportunities

By identifying opportunities, organizations are doted of taking proactive steps toward expanding their market presence or even aggregating more into their competitive deck of cards. To achieve this, the organizations must attend to their external circumstances in order to effectively search over exploits that they can take into accordance to represent a new advantage. These, if used correctly, can help the organization grow, innovate and improve performance. As an example, there is the following possible aspects: **Emerging Markets**, which stands for entering into a new geographic or demographic market, that offers potential for growth and expansion. **Technological Advancements**, that leverage new technologies that improve efficiency, number of product offerings or new innovative services.**Regulatory Changes**, related to new policies or regulatory shifts that if,taken into favor, can benefit substantially the organization in scope. **Shifts in Consumer Preferences**, that aims to answer to new consumer and trend behavior that can be targeted by a new product that may be on release and finally **Strategic Partnerships**, where forming alliances and collaborations with other companies can open new revenue streams. By paying attention to opportunities, organizations can gain a competitive advantage, enter new markets and drive long-term growth.

B. Threats

Threats on the other hand, are challenges or risks that could affect negatively the organization. Some of these are expected, but the worst ones are those that no one is expecting and because of this there must be a lot of thoughts about it, because a lot of this comes from disguised situations. Being able to spot such things, is a important skill when it comes to project maintainability and therefore it is specially important to pay attention to, because it can be the meaning of the project ceasing and catching them early can help to take preventive measures that could minimize or even remove the effects from it. Due to the external nature, the enterprise has low control over this kind of situations but being interested in such matters, can reduce significantly the downturns. As an example, threats can be represent as **Increased Competition**, where the increase of offering of similar products can potentially remove some market share. **Macro-economic downturns**, where the economy slows down for some reason which results in less demand for the project. **Regulatory changes**, which can result in the enterprise suffering by politics that may be against it or that revoke previous given benefits. **Changing Consumer Preferences**, that happens when the consumer shifts it's interest in the current product and benefits the competitor leaving the organization behind and **Technological Disruptions** where the advances in technology actually removes the need that the enterprise was serving, therefore making the organization service/product obsolete. In case well informed, organizations can adapt and overcome this challenges, by preparing defensive strategies that will

mitigate risks,ensure business continuity and market share protection.

4. How to Conduct a SWOT Analysis

In order to come up with a **SWOT Analysis**, a systematic process of evaluating the internal and external factors is required. In other terms, the **Strengths**,**Weaknesses**,**Opportunities** and **Threats** are something that is in constant change, because there may be variables that were not putted into custody, new perspectives and internally or externally there is always something that may be changing because a organization is something dynamic. With this in mid, here's a comprehensive step-by-step guide to come up with a **SWOT**:

A. Define the Objective

As a first step, a thematic should be specified. This makes sense since a organization is a very complex environment, thus there is a need to specify a specific matter within the company like a specific project,business decision,product launch,market entry or a specific strategic plan.

B. Gather Relevant Data

Moving to the second step, there should be collected the most data possible for gather the most important internal and external factors. These factors, based on the data can be collected through using financial reports, employee feedback and operational metrics in the case of the internal and market research,industry trends and competitor analysis in the case of the external. The data sources range from financial performance,operational efficiency,staff skills,technology and others in the case of the internal and market trends,competitor behavior, macro-economic events and customer preferences in the case of the external.

C. Identify Strengths

As a forth step, there must be a focusing in spotting the strengths. To actually get to the point where the organization can extract these strengths, a certain number of questions should be answered to carefully revise our perspectives:

1. What does the organization do well?
2. What are our unique resources or capabilities?
3. What advantages do we have when compared to the competition?

By answering this questions, there must be already sufficient ground to come up with actual valid answers.

D. Identify Weaknesses

In the fifth step, internal weaknesses are revised. Like mentioned before, recognizing them it's very important as other aspect of the **SWOT** but, like the threats, sometimes this weaknesses are not that easy to spot and lack of self feedback should be a barrier to come up with answers. Alike the previous step, there are questions that can help with this, being that questions the following:

1. What are the areas where we need improvement?

2. What resources are we lacking?

3. What processes are inefficient?

Like the previous step, the same applies when it comes to questions. By answering them there should be support enough to discussion.

E. Identify Opportunities

As a sixth step, now in the external realm, there must be answers relative to the opportunities. Likewise, but now for growth,innovation and performance exploitation, there are also a certain number of questions that could make the answer easier:

1. What external trends can we capitalize on?

2. Are there any upcoming regulations or technological advances we can take advantage of?

3. What market gaps or consumer needs can we address?

F. Identify Threats

In the realm of the septh step, on the other hand it comes the threats, another external factor but related to risks. These may include competitive pressures, macro-economic downturns, changes in regulations and others. Likewise again, there are a set of questions that could help with this:

1. What obstacles do we face?

2. Are there external factors that could cause problems for our business?

3. What are our competitors doing that could threaten our market position?

G. Analyze the Results

In the eight step and ,now that all the results were reunited, there's is plenty of capabilities to make a structured analysis. This analysis is precisely to make a correlation between all of the aspects and ,this way, play around with what could be strategies that could help in the given areas. It is important to mention that because they are related to which other, what could improve one side could make the other worst, which means that in the same proportion what could be good can actually have side effects that the organization cannot berry. What a organization must do in this kind of situation is: **understand how internal strengths can help on gathering opportunities** and **understand how weaknesses might expose the organization to threats**. As an example of this, in the case of the strengths to exploit opportunities there is the usage of the strong brand to expand to new markets. In the case of the weaknesses to avoid threats, in the other hand there is the update of technology to mitigate the risk of competitors offering more advanced options.

H. Develop Strategic Actions

In respect to the step eighth, this is the part where there is a creation of a given set of strategies for each element of the **SWOT**. This is challenging because they are correlated, which creates fuzz but by thinking on the effects of this strategies, bad side effects can be avoided or atleast can be taken in account.

I. Review and Update Regularly

Finally in ninth, there is a comprehensive review of what has been done so far, contributing to what was mention before with the constant change of these aspects. Either because your strategies are already making effect or simply because of the dynamic nature of a organization but the main idea here is that the **SWOT** will be continuous updated.

5. Applications of SWOT Analysis

SWOT analysis is something that can be applied across various fields, because of it's ease of using and versatility. With this in mind, in this section, it will be delving into some of those sectors for understanding until each measure can this be useful.

A. Business Strategy and Planning

To achieve long-term objectives,grow market share and enhance competitiveness, there is a widely usage of **SWOT** by the businesses by developing and refining their strategic plans based on this. It's application ranges from **Identifying strengths to levare in competitive markets**,**Addressing weaknesses to improve operations or product oferring**,**Spotting opportunities for market expansion or new product development** to **Recognizing threats from competitors or economic shifts**.

B. Product Development and Innovation

To respond to shifts in customer preferences, improve existing products and introduce new products or services, the usage of **SWOT** is also relevant. It's application ranges from **Identifying product strengths (e.g., unique features, superior quality) that can differentiate it in the market**,**Highlighting weaknesses such as high production costs or design flaws**,**Spotting opportunities like untapped customer segments or technological innovations** to **Evaluating potential threats from competing products or changing customer demands**.

C. Marketing and Brand Positioning

To reaching target audiences,develop strategies and understand the organization brand is being positioned in the market there is also the respective usage of **SWOT**. Some of its applications in this sector are **Analyzing brand strengths such as strong customer loyalty or a high-quality product line**,**Addressing marketing weaknesses, like lack of digital presence or poor brand awareness**,**Identifying new marketing opportunities through social media platforms or influencer partnerships** and **Anticipating threats from new competitors or shifts in consumer preferences**.

D. Competitive Analysis

SWOT can be also used to comparison with other organizations. With tangible characteristics shared among them, there is the possibility to identify competitive advantages and potential areas where competitors are a actual threat to the business. It's applications are enormous in this sector like **Assessing the company's strengths relative to competitors**,**Identifying weaknesses that competitors may exploit**,**Evaluating opportunities in the market**

that competitors have not yet pursued and **Identifying threats posed by existing or emerging competitors**.

E. Project Planning and Management

SWOT can also become a huge allie when it comes to project planning and management. This is because, it ensures that projects are well prepared to succeed by knowing the internal resources and external risks. More concrete applications of this are situations like **Leveraging strengths like skilled teams or efficient workflows to meet project objectives**,**Identifying internal weaknesses such as limited budgets or insufficient manpower that could affect the project timeline**,**Recognizing opportunities, such as new technologies or favorable market conditions that can enhance project success** and **Anticipating threats such as resource shortages, regulatory hurdles, or changing customer demands that may derail the project**.

F. Risk Management

To mitigate risks more effectively by identifying both internal weaknesses and external threats, **SWOT** can also become a good allie within this matter. This is because it identifies and addresses potential risks. Applications on this matter using this tool are something like **Identifying weaknesses that might expose the organization to financial or operational risks**,**Analyzing external threats such as economic downturns, supply chain disruptions, or technological obsolescence**,**Using strengths to build resilience against risks** and **Developing contingency plans to mitigate identified risks**.

G. Personal Development and Career Planning

In case a personal development plan is something desired, **SWOT** is a good tool to spot threats to make good career choices by spotting strengths, weaknesses and even opportunities. This means that this tool is flexible, until the point that it is not applied only to organizations but very relevant for other types of cases. Some of the applications within this concrete case are something like **Identifying personal strengths such as skills, qualifications, or experiences**,**Addressing weaknesses such as skill gaps or limited experience**,**Recognizing opportunities like new job openings, networking possibilities, or educational opportunities** and **Evaluating external threats like industry changes, job market competition, or economic uncertainty**.

H. Nonprofit and Social Sector Strategy

For identification of sustainability and impact opportunities and to deliver their mission, nonprofit organizatons and social enterprises also use **SWOT**. How it is applied is listed from **Evaluating internal strengths such as dedicated volunteers or unique fundraising channels**,**Identifying weaknesses like limited financial resources or operational inefficiencies**,**Spotting opportunities such as emerging social trends or partnerships with other organizations** to **Addressing threats such as changes in public policy, donor fatigue, or increased competition for funding**.

6. Advantages of SWOT Analysis

There was been a lot of **SWOT** coverage so far. Like the purpose,it's factors,how to actually use it and its major applications. With this in mind, there will be also the coverage of its advantages in a more direct way so there is comprehensive understandability about the meaning of using such tool. Without further delay, there will be a coverage of the main ones below:

A. Simplicity and Ease of Use

The first advantage is it's ease to use, which is no surprise due to its straightforwardness. It is simple to understand therefore simple to use, making it feasible for a range of users like businesses from small to big and even for individuals.

B. Comprehensive Overview

The dual approach, internal and external section of factors, ensures that, in a holistic way, there is coverage of all critical aspects for decision making. This is a key factor for making decisions since it gives less burden when analysing for coming up with a vision of the current state of a given project.

C. Encourages Strategic Thinking

SWOT creates a environment for strategic system, because it requires the organization to constantly change the aspects they have been putting into it. By constant changing, this obligates management to look beyond day-to-day operations and think more in the broader picture, focusing on long-term goals and strategies.

D. Identifies Core Competencies and Areas for Improvement

By identifying strengths and weaknesses, it helps organizations recognize their key competences and areas where they can do better, specially within the internal realm since the external realm can only be tangible through taking action from inside. This conscious, leads to a obligation for the management side to take more focused strategic decisions.

E. Supports Informed Decision-Making

For supporting data-driven decision-making, **SWOT** is widely used for revealing key insights about the organization position and the surrounding environment. This is essential for a organization to take well-informed decisions by having in their possess structured assessment of internal and external factors.

F. Flexible and Versatile

It has been proven that **SWOT** is flexible and versatile, because has it was mention before, it can be applied to a vast numerous of cases like business planning,product development,marketing,project management,risk assessment,personal career planning and even non-profit strategy development.

G. Fosters Collaboration and Team Input

Since the data is kind of bias, having different perspectives can be a valuable insight for the

organization and this is what creates a environment for comprehensive collaboration and input from the staff. By involving multiple people across the organization, **SWOT** reunites all of this insights and it helps in covering all the aspects for a well-rounded strategy for the future.

H. Promotes Strategic Alignment

Since after the last step, there is continuous monitoring of the situation and by extension mutation of the **SWOT**, alignment is expected to occur, because by having the internal and external aspects a well-coordinated strategy gets formed, thus all get to work into the same direction.

I. Risk Identification and Mitigation

Another advantage of the **SWOT** is the prior risk detection. This is very important, due to the fact that if there are notices of deviation, organizations will develop strategies that will mitigate those risks, thereby making them prepared for challenges and uncertainties.

J. Cost-Effective and Efficient

SWOT is cost-effective method for evaluating business environments. This is evident, due to the fact that this analysis don't require significant financial investments or sophisticated tools like other approaches. Putting in another words, it is not expensive to use this approach and it does serve its purpose which is perform a quick and yet effective analysis of the current situation of the organization, creating means for further decision making.

7. Limitations of SWOT Analysis

While **SWOT analysis** is a valuable strategic tool, it has several limitations that can affect the effectiveness of the insights. By getting to know the limitations that this tool offers, it is possible for us to delve into the realm of contra-measurements or good practises that in return can help to avoid this nuances and make a fair analyse. After deep thinking, there is below a coverage of the most relevant limitations that this framework can find:

A. Lack of Prioritization

Altought this tool reunites all of the most important information, it does not offer a structured way to prioritize them. Without a clear ranking of which factors are most critical, it becomes fuzzy for the management to take action due to the struggle of knowing which issues should be taken into custody first.

B. Subjectivity

SWOT analysis is unfortunately very subjective, as it relies in perspectives and opinions, which leads to a analysis. This can introduce bias or incomplete perspectives depending of the personal, leading to a vision that may not be the true representation of the organization situation.

C. Over-Simplification

Simplicity is seen as a advantage in a lot of situations, but this becomes a burden in more

complex situations where factors may not fit in the given categorizations or ,even worst, they may be a strength and a weakness at the same time.

D. Static Snapshot

Despite the **SWOT** analysis being updated frequently, they become quickly outdated in case the organization is to much dynamic, therefore, actions must be taken in order to make sure that this remains updated which is very difficult in those kind of situations.

E. No Quantitative Data

Another limitation of this tool, is the fact that the data is largely qualitative, which means that it does not contain hard data or any kind of metrics. This lack of quantitative data makes it difficult to measure significant in some points and also even more harder to establish goals which usually are better with numbers.

F. Limited Actionable Insights

SWOT reveals itself as limited in the actions to take, this is because it only presents a analysis and not any kind of recommendation regarding actions to take over certain situations. In other terms, the tool is limited into giving insights of the action to take in facing common problems, which means the most it can do is simply present a simple a analysis.

3.3 Project Plan

The following tables are conceded in respect to the activities and milestones that will be comprehended during the dissertation. It shows per month where the activity was done or is expected to be finished.

Activity 1 (A1) : Literature Revision (use case 1)

Activity 2 (A2) : Article (use case 1)

Milestone 1 (M1) : Article Delivery to Conference (use case 1)

Activity 3 (A3) : Conference Presentation (use case 1)

Activity 4 (A4) : Literature Revision (use case 2)

Activity 5 (A5) : Development

Activity 6 (A6) : Interim Report

Milestone 2 (M2) : Prototype demonstration

Milestone 3 (M3) : Interim Report Delivery

Activity 7 (A7) : Dissertation

Activity 8 (A8) : Article (use case 2)

Milestone 4 (M4) : Article Delivery (use case 2)

Activity 9 (A9) : Dissertation Review

Milestone 5 (M5) : Dissertation Delivery

Activity 10 (A10) : Dissertation Presentation

	April	**May**	**June**	**July**	**August**	**September**	**October**	**November**	**December**
A1	X	X	X						
A2		X	X						
M1			X						
A3				X					
A4				X	X	X	X	X	X
A5						X	X	X	X
A6					X	X	X	X	
M2							X		

Table 1: Monthly plan 2023: General

	January	**February**	**March**	**April**	**May**	**June**	**July**	**August**	**September**
A4	X	X	X	X	X	X	X	X	X
A5	X	X	X	X	X	X	X	X	X
A6	X	X							

M2				X					
M3			X						
A7			X	X	X	X	X	X	X
A8							X	X	X

Table 2: Monthly plan 2024: General (part 1)

	October	**November**	**December**
A5	X	X	X
A7	X	X	
A8	X		
M4	X		
A9		X	X
M5			X

Table 3: Monthly plan 2024: General (part 2)

	January	**February**	**March**	**April**
A5	X	X		
A9	X			
M5		X		
A10				X

Table 4: Monthly plan 2025: General (part 3)

3.4 Risk List

In the following table, the risks that directly or indirectly influenced the development of this dissertation are listed. Depending on the identified risk, a mitigating action is proposed in order to minimize the impact of the risk. The scale for the Probability and Impact columns is measured from 1 to 5, where 1 corresponds to a very low risk and 5 to a very high risk.

Nº	**Description**	**Probability(P)**	**Impact(I)**	**Seriety(P*I)**	**Mitigation Action**	**Verified**
1	Unreachable deadlines	3	5	15	Planning better the next events and gain experience from previous deliveries;	Yes

2	Requirements of the project change	2	4	8	Make frequent communication with stakeholder;Check well the limitations imposed by the project;	No
3	Lack of Time	4	5	20	Implement only what is actually needed	Yes
4	Resources	4	5	20	There are no mitigation measures	Yes
5	Incorrect understanding of data	2	5	10	Spend more time analyzing the data; Make sure that the data is relevant for the object of study;	Yes
6	Mistakes	1	5	5	Don't do tasks in automate mode; Focus more in the tasks;	Yes
6	Security implications	1	5	5	Follow standards; Try to learn more as the time passes;	No
7	Complexity of the project	2	3	6	Document Everything; Simplify everything;Join Communities;	Yes
8	Unknown Errors	2	4	8	Join Communities;Arrange more debug mechanisms;	Yes
9	Acceptance	4	5	5	Make the best solution possible;Make clear the need of the solution;	No

Table 5: Risk List: General

3.5 Relevant tools

Before delving into the use cases and what was done in practical terms, there is a need to explain seamlessly which tools were used during the project. This is because, by explaining what was used, by extension, an assumption by the reader for the reason why certain things were inserted in the project is achieved. Some tools are related to blockchain, since that is the main subject under scrutiny along with the healthcare sector, but others, on the other hand, are extensions of a product that is strong by itself. Reasons for the usage of blockchain were already mentioned in the State of Art; thereby, there is no reason to go much deeper in this aspect, but returning to the other tools, they are needed for commodity purposes, and probable risks resulted by the usage of certain tools will be covered further in this thesis. Also, it is important to mention that the idea around this project is not to achieve excellence in decentralization, to simply use emergent technologies because they are trendy, but to try to solve the existing problems with the best set of tools that are proven to be the best in the current market and perhaps standardize the way private blockchains are designed to gain broad acceptance. Standardization helps with broad acceptance because this way implementations are not that fuzzy like they were before, contributing to ease in implementations and creating a common path between people that, in case of obstacles, there may be a current solution already because someone crossed the same path before. This is essential for a vast spread adoption, and thereby that's our contribution here. There are 3 covered sections within the tools; the list is not at all complete, but there is an effort to cover the most important concepts of technologies that were used during the project. This will empower the reader with the correct view of what has been done so far. There is a Microservices section in which the tools are a part of the other tools that will be covered. They retain an honored representation on this project because their usage is meant to empower administrators with capabilities such as resource management, easier setup of networks, decoupled services, scalability, and high availability. Another important section is analytics; this is very important for any type of robust system, and its aim does not reside only in maintainability. It empowers the project with also mechanisms to detect bottlenecks, detect unexpected behavior, troubleshooting support, debug support, benchmarking, observability, measures to better resource management, and more.

Finally, there is also the web development section, which is used in this project as the core to build extra services that may be required for the normal function of the network. They work mostly for communication support, for ensuring security, and to help automate procedures during the results obtained in this research, but also for future usage while using this in a real environment.

3.5.1 Blockchain

This section will contain all necessary related or close related information about blockchain that is being used in this research. The aim in this section is to give a good fundamental overview about **Hyperledger Fabric**, the private blockchain technology that is used all over the investigation and also to give some hints about **IPFS**, which despite not categorized as a blockchain, has a lot of characteristics that resemble a blockchain thereby despite the section name, it still makes sense to put it here for the purely sake of relation between the concepts

making it easier to digest. The first is considered a general purpose technology, which can be used with all kinds of data but having the capability of working with all kinds of data and at the same time be capable of dealing with all types in a effective way that's a completely different story and, because of this and also to respect to files, there was a need to think in a extension of the blockchain capabilities where **IPFS** can effectively play its rule. Altought intriguing, in this work there was not a measurement of performance regarding this technology and how it would behave in the private landscape, which means that atleast in this realm there are no tangible evidence of effectiveness on our side, which may result in the future measurements while working interchangeably with both the blockchain and this technology for storing files, exacly like describe in one of our future mentioned use cases.

3.5.1.1- Hyperledger Fabric

To understand **HLF** (Hyperledger Fabric), there must be an understandable view of what a blockchain is. In an abstract way, this concept refers to an immutable ledger maintained within a distributed network of peer nodes. Each node maintains a copy of the ledger by applying the same logic to their ledger. If all reach the same result, this view of the world becomes valid, and everyone gets to keep the same state. The process mentioned here is thereby called **consensus**, which is a protocol where there is a validation of blocks that inside contain these tasks that must be endorsed by a collective number of nodes. To actually reach the point where the block is validated, all of it's transactions must be accepted by a number of nodes that depends on the configuration of the network. If a transaction gets no validation, this transaction is not included in the state of the ledger, which means it is not valid.

The first and most widely used cases of this are **Bitcoin** and **Ethereum**. Both serve different but related purposes; both fall into the class of blockchain, but these ones are public ones, like mentioned in the state of the art of this thesis.

As the popularity of these 2 plumed, other derived technologies coming from the same concept started to emerge. With this in mind, a more enterprise-friendly approach started to grow, though many organizations still need performance characteristics that presently permissionless blockchain technologies are unable to satisfy. Also, there are other requirements that also present a challenge for this approach, which are: the requirement of identity for financial transactions where **Know-Your-Customer** (KYC) is required and **Anti-Money Laundering** (AML) regulations; networks need to be permissioned; high transaction throughput performance; low latency of transaction confirmation; privacy; and confidentiality. **Hyperledger** was literally designed to meet these precise characteristics, not needing to adapt like other solutions, and that's what differentiates it from others.

Fabric is an open source permissioned distributed ledger platform designed for use in enterprise contexts, delivering a certain number of capabilities that are imperative in business scenarios. It is maintained by the **Linux Foundation**, and it presents an architecture that is modular and also configurable. It supports smart contracts through general purpose programming languages like **Go**, is permissioned, has pluggable consensus protocols that range from existing default ones to custom, does not require a native cryptocurrency, and offers one of the better performing platforms today of this kind for transaction processing and transaction confirmation latency.

This framework will be intensively used in the provided use cases. With this in mind, this section will be a solid coverage all over the main concepts of this framework.

Relatively to **Permissioned Networks**, this is a concept relative to a type of blockchain. Dissimilar to public blockchains such as **Bitcoin** and **Ethereum**, often called **permissionless** blockchains, these ones restrict access to a set of known participants, making them ideal for applications that require trust and control over the membership. Like mentioned before, **Hyperledger fabric** is one of these types, and so is **Corda**. Both platforms emphasize accountability without relying on a central authority, enabling organizations to construct custom and heterogeneous applications where there is a clear conscience of who can participate in the network [22]. Usually, the usage of one restricts the usage of the other.However, there are also approaches where networks may want both **permissionless** and **permissioned**, referred to as **two-tier blockchain**, where both types are combined to allow both participation and validation, such as supply chain or cross-border financial transactions [23]. For the context of healthcare, **permissioned** is obviously more appealing, such that **operational efficiency** and **transaction security** when combined with restricted trust between participants is needed for **security** and data **privacy** [24].

Furthermore, there is also the modularity of **Hyperledger fabric** to be covered. It has 5 components: the **Peer**,, **Orderer**, **Chaincode**, **Ledger**, **CA's (Central Authorities)**, and **client**. Each component will be covered with more extension in this category.

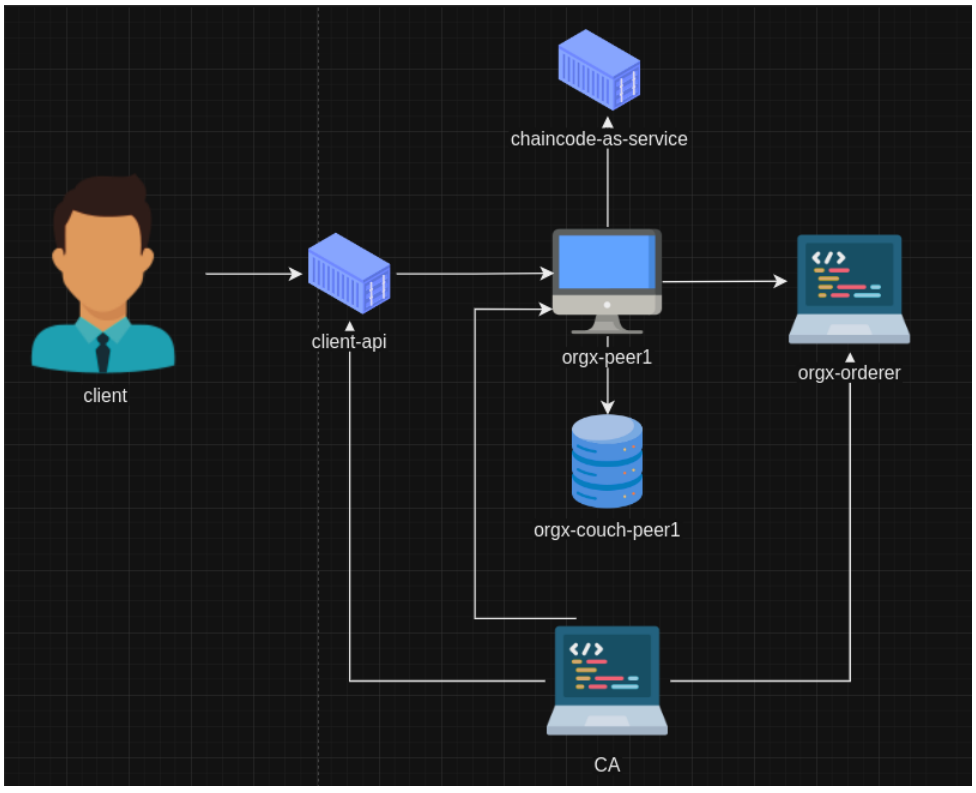

Figure 6: Major components Relationship

As an overview above, there is an image that illustrates not the communications but how components may relate with each other. Also, it should be noted that this is very far from representing a whole system, where usually multiple peers must endorse transactions and multiple

orderers must order transactions and blocks. Putting in another terms, the representation here explained is just for the sake of understanding how individual components work with each other. With this in mind, each component's individual functions will be covered more ahead.

The **Peer** (orgx-peer1) in this representation is the middle point of everything. This is because it is the one that receives the transaction proposals and endorsements that come from other peers by previous request of a client. It also receives blocks from the **orderer**, and it is responsible for endorsement of transactions by using the **chaincode**, and furthermore, after validation through this **endorsements**, it is the one that receives a block containing valid transactions from the orderer (orgx-orderer), for then to include in it's ledger (orgx-couch-peer1). The **client**, on the other hand, manifests in two ways: the client itself as an actor and the **client-api**, which turns out to be the application itself that is responsible for calling the **Peer**. The **chaincode** (chaincode-as-service) simply runs logic, which is what the **Peer** endorses, by running the application against the **ledger** (orgx-couch-peer1). In respect to the **ledger**, it is used as a mock to endorse transactions to see if the output is the same within the vast majority of peers.The **orderer** receives transactions and puts them inside of blocks, and after ordering these transactions and blocks, it delivers the blocks to the peers. Finally, the **CA**(certificate central authority) only gives cryptographic materials so that components can identify themselves within the network. It must be known as well that the chaincode-as-a-service has the option to create TLS connections with the peer. In this picture there is only one CA for purposes of simplicity on this explanation, because usually there can be a chain of CA's and in each level, for purposes of good practices, should have 2 CA's where one is for normal certificates and the other is for TLS. The difference between normal and TLS certificates is that one is for identification and the other is to secure the communications with encryption. Once again, in this representation only normal certificates are considered, but within the network there are both, which contribute to a secure environment. [25].Now that their relations were described, let's describe them in more deep terms for further explanation.

In what concerns the **peer**, this is a fundamental concept within the network since it is responsible for managing **ledgers**, **smart contracts**, **transaction proposals**, and **endorsements**. **Ledgers** and **Smart contracts** will be explained further, but a transaction proposal is a transaction that was signed by some entity that wants to be further endorsed by the peers for then to be included in the ledger if valid. **Endorsements**, on the other hand, are the process of a peer running the logic within the proposal against its ledger. Depending on the configuration of the channel and the number of peers that validated this proposal, this transaction will be included on the block or not (within the orderer side); therefore, it could be or not included in the ledger depending on its validity.

In the **Orderer** sphere, however, there is the responsibility of receiving endorsed transactions, which, depending on the number of signatures of the peers in it, could add the transaction to a block or reject it. In the case of the first, after ordering the transactions and the given blocks, it returns to all the peers this given block. Block this that will be used for the peer to insert its operations against its ledger, therefore ensuring that every single peer has the same view of the state because they are all running the same operations against the same state. To achieve this, the orderer uses what is called a **consensus protocol**. This protocol is a mechanism to

reach agreement in the sequence of transactions, but that will be discussed further.

Another vital pillar for the network is the **Smart contracts**, more commonly called **Chaincode**. This is where all of the business logic runs, and it uses abstractions to communicate with the ledger. Usually, installation is required only once. However, it must be added per channel if that is the intention of the participants within that channel. This means that I can have the chaincode installed, but this chaincode could potentially not be allowed to be called because, for that, it must be proposed to be added and committed when a certain number of participants agree on its insertion. Besides that, there are 2 typologies for usage of chaincode: **normal chaincode installation**, which occurs per peer and the chaincode becomes more like a side container, and **chaincode as a service**, where the chaincode becomes a service and to install it there must be passed metadata for the connection management. The normal way is often used for questions of ease, and, along with it, there is the certainty that the chaincode that is being invoked is in fact the desired one, since it is literally a side container or side service of the peer, therefore also giving more isolation to it. The other way, however, depending on how it is deployed, it can be reused by multiple peers, which means that if an operator does not retain this service within its infrastructure, it may represent a risk since the endpoint could be the same and its application could be changed. Be mindful that, in case the **chaincode as a service** is adopted, depending on how it is implemented, it can be exposed to the common microservices vulnerabilities, and further actions should be taken to prevent these [26].

Ledger, on the other hand, is a normal database like **couchdb** [27]. This is the state that the **Peer** uses as a mock to run its business logic and get an output for later usage for the endorsement procedure. Transactions can be rejected by either violating some business logic or by presenting different outputs among the peers. As an example, let's suppose there are 3 peers. In the first situation, peer 1 rejects the transaction because the business logic says so. The other 2 peers, peer2 and peer3, respectively, accept the transaction. In the case of a majority policy, the state resulted in peer2 and peer3 being included in peer1 as well, simply because the majority wins. Also, let's consider the case where peer 1, peer 2, and peer 3 have different outputs, but the signatures were still reunited because the transaction was not rejected. What will happen is that when this transaction reaches the orderers, the orderers will reject its validity. This is because, despite passing the peer endorsement, the orderer endorsement will fail due to a lack of determinism between the three peers. Also, within the same database, there could be different slices of data for each **channel**. However, the concept of channel is something that should be delved deeper into more ahead.

Speaking about **CA's**(Certificate Central Authorities). This is a service that serves cryptographic material like private keys and x509 certificates, which within a **hyperledger fabric** network form a **MSP** (Membership Service Provider). **MSP** is a mechanism designed for providing a way to offer identities that could be trusted and recognized by the rest of the network. To achieve this, there is the usage of this cryptographic material provided by the CA's where a key pair is offered. One of these pairs, the private key, is used to sign transactions, and the other, which is public, gets shared among the entities that will be communicating with the owner of the private, giving them a way to verify that this signature is actually valid. **Hyperledger** lets the operator either rely on existing public CA's, use their CA implementation, or simply use a

hybrid methodology. The main difference between the public and their implementation is that it's implementation already comes with a given file structure to make the configurations easier and also the aggregation of more responsibility to the operator, but that's something that is not the concern here, thereby it should not be extended.

Relatively to the **client** within the hyperledger fabric components sphere, there is a close relation in how it operates with the chaincode. Like chaincode, both can be implemented with a certain number of high-level programming languages such as **Go**. Both use an interface, attributed to their set of tasks in aim. In the case of the **chaincode**, its interfaces are related to making interactions with the ledger, but in the case of the client, its abstractions are focused on making calls against the peer by giving metadata related to the **channel** and methods to be targeted. Thus, it's more formal name is **Fabric SDK**, where there is the opportunity to use **Node.js**, **Java**, and **Go**. Giving the opportunity to perform various functions, such as creating **channels**, submitting **transactions**, querying the **ledger**, and managing the **chaincode**. Within the **SDK** ,there are various subpackages aimed at serving different purposes, like: **gateway**, for establishing connections to the network and provide access to **channels** and **contracts**. **Network**, which represents a channel that a client can interact with. **Contract**, conceded for iteration with a given smart contract deployed in a **channel**, used for submitting and evaluating transactions, and finally Wallet, responsible for managing user identities and credentials for connecting to the **fabric network**. Further details should be seeked within the proper documentation of **hyperledger fabric**.

Besides all of this, there are also a certain number of extra topics that should be covered for an extensive understanding of **hyperledger fabric**. This topics are, more in depth, **Consensus Mechanism**,**Privacy and Data Confidentiality**,**Channels**,**Scalability**,**Endorsement Policies**,**Governance and Membership Services**,**Real-World Use Cases** and also **Importance in this project**.

Starting with the **Consensus Mechanism**. This is a protocol that originated within the context of distributed systems. The most well-known protocol is **Paxos**. Although not implemented in the context of **hyperledger fabric**, it is of high importance to refer to it. This is because this algorithm was the first and major one of them all, contributing to the creation of further, simpler, and well-designed algorithms. For reference, **Paxos** is a consensus algorithm that originated for achieving agreement between a network of unreliable nodes. It was introduced by **Leslie Lamport** in a series of papers, starting with **The Part-Time Parliament** in 1988 [28] [29]. It's importance comes from ensuring that a vast number of nodes can agree on a single value, even in the presence of network delays, node failures, and other types of faults. Also, it was the building block of major projects such as **Google's Spanner**, **Amazon's DynamoDB**, and **Apache ZooKeeper** [30] [31] [32]. It is considered a building block precisely because, like other algorithms, every single one of these projects has its own implementation according to their use case, which means that all are deviations of the original **Paxos**. However, although very powerful, it has begun to become more obsolete because of its complexity. Putting in another words, though very powerful, it is hard to understand, thereby hard to implement, and by extension, hard to debug or resolve issues, not speaking about other major concerns such as **Performance** due to it's reliance on multiple phases and need for communication with the

majority of nodes and also because of its **Scalability** problems, where this algorithm requires a huge number of message passing, therefore causing an overhead in the network.

Deviating from **Paxos**, other algorithms surge, such as **PoW**,**Kafka**,**Raft** and more recently **PBFT**.

Speaking of **PoW**(Proof of Work), this algorithm is well known for serving decentralized networks like **Bitcoin**. However, there are implementations of this protocol under the **Hyperledger fabric** umbrella, which means that it's practical cases are not just suited for public implementations alone. It's conception exists around solving complex computational puzzles that require significant effort to complete, but at the same time very easy to verify [33] [34] [35].

In second, it's worth to mention **Kafka**, more well known in the hyperledger community as **crash fault-tolerant**(CFT) consensus mechanism for ordering transactions. This one got replaced by **Raft**, which is simpler and will be covered next. CFT is an algorithm that uses **Apache Kafka** in combination with **Apache ZooKeeper** to establish an ordering service that provides total ordering of transactions in a permissioned blockchain network. **Kafka**, with its publish-subscribe messaging system designed for high throughput and fault tolerance, was doing the handling of message ordering and durability by putting the messages into topics and partitions, while Zookeeper was used to manage Kafka brokers, ensuring that Kafka partitions are assigned to brokers, and was also helping in tasks such as leader election [36].

In third, there is the mention of **Raft**. This is the current most-used consensus algorithm, and it's very claimed by its simplicity, giving the same results as Paxos. To achieve this, raft elects a leader among a group of distributed nodes, ensuring that all nodes agree on the same sequence of operations. More concretely, when a leader ceases to exist, every node enters a random timer state where the first peer to send its vote request message usually becomes the leader. This is because the others, by receiving this request, enter a non-candidate state since there is a peer who requested it first. After reuniting the majority of votes, and since each node can only vote once, this node becomes the leader, and every log replication passes by him to endorse. With this, log replication is guaranteed, creating safety and consistency within the network. By offering simplicity and robustness to a distributed system that requires replication, **Raft** has become the most broadly accepted consensus algorithm, having implementations such as **etcd** and **Consul** [37] [38] [39].

Lastly, in terms of consensus algorithm, it surges the most spoken one lately, **PBFT**(Practical Byzantine Fault Tolerance). This algorithm was introduced by Miguel Castro and Barbara Liskov in 1999 and is widely used in permissioned blockchain networks. Although it is older, it has only been implemented in **Hyperledger fabric** now, and it promises to get rid of faulty or even malicious nodes. **PBFT** can tolerate up to (n-1)/3 faulty nodes in a network of n nodes by using a **Three-Phase Commit** approach. Despite it's enormous advantages like **High Fault Tolerance**, **Low Latency**, and **Deterministic Finality**, it still comes with disadvantages like **High Communication Overhead** and **Leader-Based approaches** [40].

Changing headlines, **Privacy** and **Confidentiality** are critical components of information security and data protection strategies. **Privacy** refers to the right of individuals to control their personal information. This is done by giving the power of control over how it is collected, stored, and shared. It is about ensuring that personal data is not shared or used without people's

consent. On the other hand, there is **Confidentiality** which proves to be about the protection of a user's data from malicious, non-desired access, making sure this way that the information remains available only for those that were allowed to have such. In the healthcare context, it is required compliance against the **Health Insurance Portability and Accountability Act (HIPAA)** and **General Data Protection Regulation (GDPR)** [41] [42]. Also, **Hyperledger fabric** helps achieving these regulations by offering characteristics such as **Permission Blockchain Model**,**Data Privacy Through Channels**,**Data Confidentiality With Private Data Collections**,**Access Control and Identity Management**,**Auditable Data Sharing and Provenance**, and **Data Encryption and Secure Data Sharing**. Firstly, **Permission Blockchain Model** helps to fit in the regulation because it offers a network where all participants must be authorized and well known. Secondly, **Data Privacy Through Channels** is also another characteristic that does fill the needs of regulations because **channels** enable private communication between a subset of participants, where each channel has its own segregated ledger. Thirdly, **Data Confidentiality With Private Data Collections** also helps on this because Hyperledger can create private collections within a channel that, despite still fitting in the main ledger, are privately only available for a set of given participants. Fourthly,**Access Control and Identity Management**, because **Hyperledger** uses the **MSP**(Membership Service Provider), managing effectively the roles within the network, creating fine-grained access control, and enabling authorized access to resources. In fithly, there is the concept of **Auditable Data Sharing and Provenance**, due to its offering of logging records, making audit easier; thereby, it complies with **GDPR**, and like mentioned before, due to the providing of a history of events, it shows clearly how records are being managed, spotting probable malicious activity and identifying the bad actor if needed. Finally, **Data Encryption and Secure Data Sharing**, since it allows encryption of data, ensuring that the data is protected against malicious access [43].

Apart from what has been covered so far, there is the concept of **channel**. This concept is very important since it is the existing mechanism that allows you to aggregate a certain number of components, thereby creating a linkage between them and a precise section for them to exchange information. Within the documentation of **Hyperledger Fabric**, there is the reference to channel as a **group of friends**. This is because, exactly like a channel, you have multiple groups of friends on your social media, and their members are different depending on the task or the theme to be discussed within that group. Also, within those groups there is an exchange of information that is private and it's not shared with other group members, which is essentially the same thing as a channel. In a channel, you aggregate a certain number of components that could be part of multiple different organizations depending on what you want to be sharing within that channel, and also within this channel, there is the option to create a **private collection**. Despite the theme being the same, there are participants that may want to discuss information regarding this discussion within the group but just with a set of participants, especially like it would be with the private collection. Though the difference with a friend group is the nature of blockchain. This means that with a private collection, there is the option of creating a subset within the channel, making others able to see that you are communicating with each other but not what they are exchanging. In other terms, a given

organization can be in multiple channels, with multiple different individuals in each, and it can also create multiple private conversations regarding the same theme, but just with a subset of participants in the same channel and exclusively of that precise channel. As an example, let's think of an organization that has the idea of creating a group of medicament's suppliers and other group of medical clinical reports. Obviously, in these examples, there will not be common participants because their nature is different. The organization may discuss within that group that it needs medicaments for X. Supposing they want to buy the cheapest one, they will request it from the enterprise Y. This enterprise, although offering it more cheaply, may not have the quantity to satisfy this need, making this organization forced to go to supplier Z to satisfy the rest of the demand. Maybe, for the benefit of this organization, prices of the product should not be known by the participants within this channel, which means that at least the monetary transactions within this channel should be done in private collections. However, the transactions of products could be done publicly, therefore making it easier for all of the participants to have the information publicly available that this organization is supplying itself from these two organizations, making others in the obligation of trying to bargain to also benefit from this channel. Also, the same applies to the clinical registry, even if it has a different logic or idea. There may be information that can be publicly exchanged between doctors, but there may also be communication that should only be sent within a more restricted set of individuals within that consortium. With channels and private collections, hyperledger obtains data confidentiality,selective data sharing and regulatory compliance [44].

Delving into the **scalability**, this is most concerned matter when it comes to the **blockchain**. Since this is a complex environment to maintain, various factors could potentially harm the performance of such, making it infeasible as a project if not taken into account the following factors that could affect it: **consensus mechanisms**,**network size**, and **network configurations**. **Consensus Mechanisms** could be a problem because there are certain algorithms in consensus that offer more fault provenance but at the same time lose performance because of the number of operations to achieve a more secured consensus, making it, like mentioned, more fault provenance but at the same time slower. There are works concerning the comparison of such [45]. **Network size**, on the other hand can be also a problem to the performance, because when it comes to the blockchain, having a bigger network means more nodes to achieve consensus, therefore causing the network to be slower do to all the communications required to reach the consensus. To address this, there are techniques to make the consensus faster, like side chains or sharding, consisting of making the consensus within a subset of peers, not implying that the whole network reaches a consensus but that a subset does,therefore making the network faster [46][47].Finally, **network configurations** is related more to settings that are presented in various domains of the blockchain. There are enormous parameters that could be settled within the hyperledger landscape. This parameter is categorized by its own domain presence, and changing them could benefit or make the performance worse, but keep in mind that, in order to do that, there is the requirement of understanding the particularities of each deeply; otherwise, changing the default values could imply problems in the future. As an example, there are the categories **Block Size** and **Consensus Settings**. In the **Block Size** sphere and, without mentioning everyone, there are settings such as **MaxMessageCount** for a max number of transactions

within a block and **AbsoluteMaxBytes** for the maximum size of a block in bytes. In the **Consensus settings**, on other perspectives and also without mentioning everyone, and in the case of the **Raft** protocol, there can be found settings like **Heartbeat Timeout** for heartbeat messages to followers interval and **Snapshot interval** for the interval for which there is the need to make snapshots of the current logs [48].

Endorsement Policies are another big dinosaur in the room when speaking about **hyperledger fabric**. These are the rules under which the transactions will need to apply, making the transactions valid in case they succeed and invalid in case they fail to do so. In other terms, these are the policies that define which peers must sign (endorse) a transaction before it can be valid and aggregated to a block by an orderer. It specifies how many and which organizations must approve the transaction before it can be inserted into the ledger.

Piercing to the real use cases realm, there are several occurrences of hyperledger appliances. The sectors affected by this remarkable technology were: **Financial**,**Healthcare**,**Supply chain management**,**Goverment and Public Sector**, and **Insurance**. Within the **Financial**, various banking consortiums were created for processing cross-border payments, managing the financial chain, and enabling transparent asset transfers [49]. In the **Healthcare** there are also a lot of occurrences of it's usage for managing electronic health records (EHR's), which is possible due to the channels and private collection capabilities, allowing sharing of patient data between healthcare providers [50]. In the **Supply chain management**, there are multiple occurrences for the usage of hyperledger fabric for tracking goods. By using this, enterprises were dotted of track food products from the farm until the shelf, reducing the time required to trace the origin of food products in the event of contamination while still offering privacy for sensitive business data using private collections [51]. Also, when it comes to **Government and Public Sector**, it is well known that there are already present in the market implementations for the usage of such technology for managing digital identities, land registries, and transparent voting systems [52]. Finally, there is the usage for **Insurance**, where there are retained implementations that are meant to help claiming processing and reduce fraud. Thus, by levering this, the **Insurance** can collaborate in a shared ledger to verify these claims and manage shared data, reducing duplicated and fraudulent claims [53].

As a final for this section, the **importance** of this tool for this thesis must also be explained. As it has been explained, this technology is very complex; there are a lot of alternatives for it; it's emergent, and there are still problems that must be addressed. However, the urge to choose it over others's comes from the community that it has, the current and continuous work that is being released over it, the fact that it is being used by the major companies such as **IBM**, its modularity, and it's already uncovered advantages within the healthcare sector. This is the aim of the research, which makes it a more favorable choice for the project; therefore, the use cases that will be crossed further will be both based on this.

3.5.1.2- Ipfs

When it comes to **IPFS**, it must be known that this does not represent a blockchain, though its concepts derive from a P2P perspective, which means there are relations between the two. It is a modular suite of protocols for organizing and transferring data [54].

More concretely,**IPFS** is a tool that stands for Interplanetary File System, where it refers

to an implementation that was meant to become the file system of the future.

Their aim is to serve as a decentralized file system where, to achieve that, what is required is having a distributed system that goes through an immense decentralized network of heterogeneous numbers of peers. This way, by adding a file or an image, there will be a unique identifier for that file. In the case of that file being a folder, it generates an identifier for himself and for the respective children's and stores it in an **DHT** (Distributed Hash Table) [55]; otherwise, it will simply create an identifier and store it in the same data structure mentioned before. With all of this done, anyone that wants to serve from that file only needs that the given peer is correctly configured and that everyone that has that identifier (which is called **CID**) have access to that file through a certain **IPFS** gateway, which can be public or private, depending on the situation you are in.

This solution can be interplanetary, because when the first request of a file in the network occurs, it may endure a lot of time, but after getting it, the file will be retrieved faster because it will be stored in cache in the most nearby peer. Thereby, if you plan to travel to another planet, it can be possible to share files in this network, which, depending on the range, may also include other planet files, but this has countless ways to occur. Or either by one of the peers from another planet changing its location to a location that is reached by our network or simply because the connection between those 2 planets is feasible. However, in order for this to work, there is still work related to making sure that the files are pinned; otherwise, they will get deleted later by garbage collection. This ensures that we do not have files permanently. If they are not that relevant, with time, they will simply disappear. To understand **IPFS** more in depth, there will be a vast number of concepts covered, so there is a vast understanding of this technology.

Decentralized Data Storage is one of the features of **IPFS**. It refers to the possibility of storing files across a network of nodes, where each node can also serve requests if they possess the file a given user is requesting, creating like this a distributed web of information. To achieve this, **IPFS** relies on a network of heterogeneous nodes, or p2p, where each user can retrieve files from any node that holds a copy of the requested data. Also, it gives a unique identifier to each file **CID**, making data immutable and verifiable. It uses **DHT** for the routing and discovering of nodes that store content. It breaks files into smaller pieces, where each piece is replicated across multiple nodes. It pins the data to make sure that the data remains on a specific node, guaranteeing its availability in case there are nodes dropping out, and by doing all of this, users that have a **CID** of a file can request it at any node thereby making the network effectively decentralized.

As it's second feature, there is the **content addressing** feature, which is it's capability to assign a CID to each content of IPFS. This identifier is unique to the content and does not depend on where the content is stored. This is due to the fact that the CID is generated by hashing the file, and the retrieval is done by using the **DHT** to find nodes and ask them if they have this hash value.

Data Redundancy and Persistence is another major characteristics within **IPFS**. Since it has the competence of persisting and making available data over time, even if there are failures with the network. Because of it's decentralized nature, it can offer redundancy by ensuring that

multiple copies of a given file are stored across different nodes within a the network and it offers persistence because the file remains accessible and intact even if there are nodes going offline or data get's lost. To achieve this, it relies in a vast number of mechanisms. This mechanisms are **content replication**,**data pinning**,**fault tolerance** and **Data availability and load balancing**. **Content replication** is the mechanism where IPFS divides a file into multiple chunks and distributes them within the network between multiple nodes. It is with this practise that the data remains accessible even with offline nodes. **Data Pinning** however, is the instrument that this technology is doted that consist in marking a file as something that should be stored indefinitely, thereby preventing that a garbage collector cleans it. **Fault Tolerance** refers to having multiple nodes that may have the copy. This means that the system is resilient due to the fact that it makes data available if atleast one node that has this copy is not failing. Finally, there is **data Availability and load balancing**, meaning that by having multiple nodes where the copy got stored the access ends up distributed thought the nodes. Regarding this feature, there is a work that highlights the effectiveness of data redundancy in **IPFS** against a centralized solution [56].

Ipfs also offers **Efficient Data Distribution**. This is possible because of its P2P architecture to distribute the data across the network, which gives high availability and fast retrieval. However, there are still discussions to improve even more the protocol by optimizing its node management and reducing redundant data delivery [57].

Another important thing to mention is the **Censorship resistance** that it has. This is possible due to it's decentralized nature and immutable identifiers. In addition, there is a project that proves how **IPFS** techniques are effective against censorship practises such as: **blocking IPFS gateways**, where malicious actors can use techniques to block the point of access (gateway) from users using conventional practises(DNS blocking,IP address blocking,Deep Packet Inspection,Blocking HTTP headers, etc...).**Filter p2p traffic**, where bad actors filter information from the request or even the response with common practises (Deep Packet Inspection,Trafic Analaysis and Pattern Recognition,etc..).**Preventing IPFS boostrapping** that happens when the attacker blocks ipfs boostrap nodes which are nodes that new nodes connect initially to know others in the network, making unfeasible for new nodes to join. **Targeting Specific Content Hashes (CDI's)** where malicious actors don't retrieve certain CDI's because they may want to sensor that content. **Blocking IP addresses of Known Nodes**, which occurs when bad actors block ipfs nodes or even gateways. **Coercing or Compromising Gateway Operators**, which is a practise where someone makes the operator of a gateway censor determined **CDI's** and **Legal and Policy-based censorship** where the government decides prohibiting others to share certain content [58].

When it comes to the integration of such technology within the blockchain, there are various interesting projects. Firstly, there is the tokenization of storage, where the **Filecoin** [59] project gives the possibility to users to store their data for a fee that they should pay to a storage provider (or miner) in a decentralized environment. Secondly, there is the **NFTS** [60], another project that **IPFS** is very used for, which is for storing metadata that can later be used in a platform such as **Opensea** [61] or even for storing the image itself. This metadata is a standard that is stored as **JSON** [62] in the **IPFS** network, such that the **NFT**, by having the

link to such **JSON** related to it on its smart contract, could have its characteristics mentioned in the **Opensea** store. Thirdly, there is the **Distributed Education Blockchain Network (D-SELI)**, which changes its SELI educational blockchain network by integrating IPFS to handle the storage and accessibility limitations. With this integration, materials are stored in a decentralized way, making them more accessible and robust against data loss or network failures [63]. Forthly, **Blockchain-Based Ride-Sharing Services Using IPFS**, which is a study that proposes a blockchain-based ride-sharing system that integrates IPFS. This aims to share data outside of the blockchain, answering issues around response time and scalability, which with **IPFS** and by sogregating the blockchain of the data storage, there can still be data integrity and traceability while improving storage and efficiency [64]. Finally, **IPV4 and IPV6 for blockchain networks**, where there is an IPV6 blockchain network implemented alongside **IPFS**, which improves privacy, security, and scalability compared to IPV4 [65].

Another important concept when speaking about this technology is its **handling of versioning and file updating**. Like mentioned before, files that are stored in **IPFS** are immutable, which means that there are no ways to change them. However, what can be done is to create a mechanism that supports pointing to a new version. This is because each new version will require a new upload; therefore, a new **CID** will be generated if the content of the file got changed with its previous version, of course, even if it is a small change. Now, if the method for pointing is not established, then there must be a website or something that gives a link to the current version; otherwise, the older link will always display the older version. This mentioned mechanism is called **IPNS (interplanetary name system)** [66], which is an endpoint where a operator could update the file for where it is pointing. This is a powerful tool when it comes to constant updating or simply by creating a certain registry of versions, like in a git release, for example.

Thought very secure, and with **Security and Privacy** features such as **Content Integrity**, **Immutable Data**, and **Distributed and Decentralized Storage**, **IPFS** remains with some limitations. This limitation is **lack of built-in encryption**, which means that by having a given **CID**, anyone can access a given slice of data without restrains. **No Native Access Control**, that refers to the fact that anyone can access without any kind of limitations any file, without mechanisms to prevent so, such as an **AC** (Access Control) and **IPNS security limitations**, that are only protected by a key pair, which means that if a private key gets compromised, the links can be redirected by a malicious actor. Putting it another way, despite having good features, it is highly dependent on custom integrations from those that require more advanced security and privacy. However, there are projects that try to colmatate this kind of issue, such as **Consortium Blockchain Distributed Storage Protocol for IPFS** [67] for dotting the network with encryption and data block optimization. **Scalable Blockchain Model Using IPFS for Healthcare Data Security and Privacy** [68] for integrating blockchain alongside IPFS in healthcare with symmetric and asymmetric encryption, **Blockchain and IPFS Integrated Framework for IoMT Devices** [69], **Enhancing Security and Flexibility in the Industrial Internet of Things** [70] and **Three-tier Storage Framework Based on TBchain and IPFS for IoT Security and Privacy** [71].

When delving into the importance of this tool within this thesis, it will be discussed in

use case 1, an implementation that makes recursion to such a tool. However, only theoretical inceptions around this were done, but there is the possibility to put it into practice since it is an extension of what is done in use case 2.

3.5.2 Microservices

Within this section, there is information about the major high level technologies that were used in this precise research that are related to **microservices**. One of those technologies is almost a obligation for the current landscape of technology, there is one which is a upgrade, one that is necessary in certain concrete situations and there is also a extension but all of them are valid. The one that is almost a obligation is docker, because currently in the tech landscape it is used in almost every system because of its isolation therefore faster for development in big teams. The upgrade is the kubernetes, which manages this **docker** images like they are resources ,making it complex in a certain way but also easier to manage in another after learning it. The necessary in certain concrete situations, in another hand is **MetalLB**, because it is meant to on premise implementations, which means that it is more suited for own infrastructure implementations which is a big help when it comes to healthcare due to their resistance into moving to the cloud. Finally the extension is **Istio**, because it is used to have more control over the network and its communications, but its usage its not mandatory, thereby in this project despite tested it is not used in any implementation. After **Hyperledger**, this are the technologies that more represent how some implementations communicate, therefore is good to present what they are and how they work, so the reader can understand the architectures that will be presented further.

3.5.2.1- Kubernetes

Kubernetes stands as a open-source platform designed for **automating the deployment**,**scaling**, **management of containerized applications**,**management of resources**, **management of secrets** and **management of configurations**. Originally developed by **Google**, **kubernetes** has become a standard for managing containers being a premise to build cloud-native applications. By taking leverage of this, developers can abstract the infrastructure and focus more into the application logic instead of hardware and other operations side tasks. With this in mind, **kubernetes** today is centered as the pilar for building any application, even if it is on-premise(within a organization own infrastructure), due to the fact that currectly every organization seeks a **microservices architecture** which obligates businesses to build their own kubernetes clusters. This is because, microservices itself offers a lot of advantages for their business such as **high-availability** because of deployments, **better resource management** and **resilience** which are crucial for any organization that wishes to remain competitive.

Relatively to it's key features, on the other hand there is the **Automated Container Deployment and Management** for automaticatically deploy and manage large number of containers across multiple nodes, **Scaling and LoadBalancing** for managing instances of the applications such that the resources are spared or for balance the requests per instance,**Self-Healing** for restarting a pod every time it crashes automatically,**Service Discovery** for discover a given service by using it's name that later on maps to a given ip, **Storage Orchestration** for manage and provide storage resources,**Declarative Configuration and Automation** for

users define their application state in configuration files and finally **Rolling Updates and Rollbacks** which is a mechanisms for going back to the previous application version in case of error.

This tool is powerful and reliable and that is the main reason why there is the usage of such tool in the given use cases, thereby this section will be covering it in depth.

Containerization involves packaging an application and its dependencies (libraries, configuration files, and binaries) into a lightweight and portable box. This can be used in multiple environments without problems with dependencies. This is because this box, more often called a container and sharing the same kernel with the main machine, segregates the dependencies and operative system, making it a huge allie in software engineering, and that's why containerization always has deep tights with **isolation**, **portability**, and **efficiency** (because it is lighter than the traditional VM's). The main technologies for containerization are **Docker**, but there are others such as **Podman**, **CRI-O**, and **LXC**. To manage this, there are existing abstractions. The most well known is **Kubernetes**, and the main reason for using it is that it treats these containers as resources, it has abstractions to deploy them, it manages secrets to use on them, it helps in creating horizontal and vertical scaling depending on metrics, it has mechanisms to go back in case the deployment goes wrong, and it ensures high availability by resetting containers every time they crash, though there are other ways.

Kubernetes is a very complex abstraction over Docker. Because of the number of components required to run such infrastructure, there is the possibility of not being big enough as an organization to actually benefit from its features. This is because there is a curve to reach a point where the cluster could actually not be a burden for the performance of the application. The bigger the infrastructure, the less relevant it is having an extra requirement in terms of resources to actually manage containers, which for small businesses may be killing a fly with a big arsenal, not having the strength to handle it. By the effect of such, there should be some understanding of it's components to comprehend what kind of burden a organization may be willing to take. **Kubernetes** is divided into two main parts: **Control Plane** and **Worker Nodes**. The **Control Plane** has 5 components in it: **API Server**,**Etcd**,**Controller Manager**,**Scheduler** and **Cloud Controller**. The **API Server** is the hearth of the interactions with the Kubernetes cluster. It exposes the Kubernetes API, which can be either internal (kubelet) or external (kubectl) to communicate with it's control plane, which is how all configurations are made, such as deployments and scaling, just to give an example of such. **Etcd**, another important component, on the other hand, is a distributed key-value store that stores all cluster data, such as cluster state, configurations, secrets,service discovery data, and other metadata, while ensuring that this data is consistent across the cluster for other components to use. After, there is the **Controller Manager**, which in this case is responsible to manage the life-cycle of resources within the cluster, such as node management, replication, and endpoint controllers, all leveraging controller processes, where each process watches the state of the cluster (via the API server) and makes necessary changes to maintain this desired state specified by the configurations that the user implied. Also, there is the **Scheduler**, which is the one that is responsible for assigning new pods to the nodes within the cluster by taking into account important factors such as availability, affinity (this is what can be created in a configuration file to make sure a given pod

is deployed in a node with a given label), and policies defined by the user, all to make sure that the resources are wasted in a balanced way, not creating huge differences between perceptual usage of a node over another. Finally, within this part there is also the **cloud controller manager**, which stands out for integrating **kubernetes** with **cloud provider API's**, enabling l**load balancing**, **persistent storage**, and **node management** based on cloud infrastructure being specific to cloud environments, enabling communications to it's usage. Relatively to the **Worker Nodes**, on the other hand, this part has 3 components: **Kubelet**, **Kube-proxy**, and **Container Runtime**. **Kubelet** is a core component that runs in each node and communicates with the API server to receive instructions and ensure containers are running as they should, while at the same time interacting with container runtimes (**docker**,**containerd**) to start, stop, and manage containers. **Kube-proxy**, in another perspective to have, is a network proxy which runs also in each node to manage network rules and enable communication between pods within the cluster and with external resources, while using IP tables and IP virtual server rules to route traffic for achieving this for **service discovery** and **load balancing** in the cluster realm. As a final for this part, there is also the **container runtime**, which is responsible for running containers on each node. Typically runtime containers are **docker**, **container**, and CRI-O, like mentioned before, but their main function is to interact with Kubelet to manage the lifecycle of containers, pulling container images and running or stopping them if requested. Additionally, when it comes to the logic flow within this architecture, what happens is that a given user interacts with Kubernetes by leveraging it's API by either using **kubectl** or a SDK like Go Kubernetes API to make its request, such as deploying a container. Second, what happens is the **API server** receives this request and updates the state that is present in **etcd**. In third, the scheduler chooses the best node, if needed, to place resources in it, and the **controllers** monitor the current state and desired state of the cluster, so there are adjustments according to both perspectives. Forth, the **kubelet** receives instructions to push some strings for reaching this desired state, communicating with the container runtime to do whatever changes it must take, and the **kube-proxy** manages the traffic needed. Fifth and final, the **control plane** simply continues monitoring the state of the cluster, ensuring that all runs as expected, taking action if so is needed.

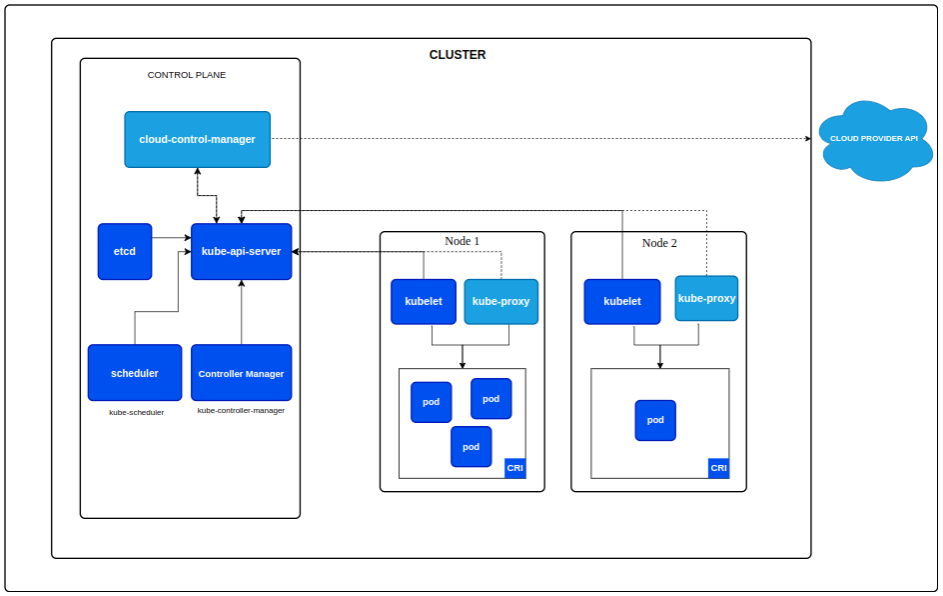

Figure 7: Kubernetes Architecture: High view

Another concept worth mentioning is the distinction between **Pods** and **Containers**. **Containers**, like mentioned before, are lightweight boxes of software that include everything needed to run an application, offering **isolation**, **portability**, and **efficiency**. On the other hand, **pods** are the smallest unit deployable unit within the **kubernetes** logic and represent a logical host for one or more containers, which can be more or less as another box but just to put containers in it. It shares both storage and networking resources, being as well the fundamental building block for running applications and services.

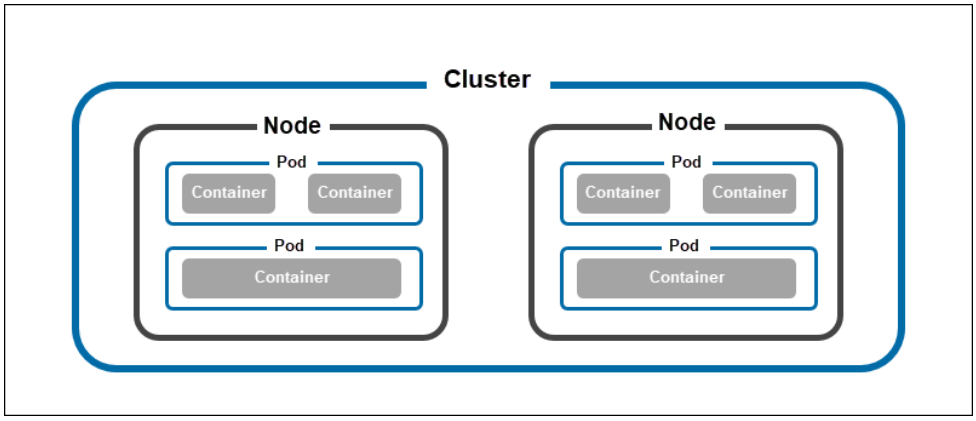

Figure 8: Representation of pods within a cluster

When it comes to **Microservices**, **scalability** and **load balancing** have a notorious complexity and importance. This is elevated to a more extreme level when it is combined to **kubernetes**. **Scalability** refers to the capacity of increasing resources, where this increase could be done in two major ways: **Horizontal Scaling** and **Vertical Scaling**. **Horizontal Scaling** happens when there is increase or decrease of instances of the same application where by using a service can receive balanced traffic and thereby work together for the same goal, which in the case of **kubernetes** are **pods**. As an example, lets think of an web application

in go, which serves a API to retrieve records of something. Within a given time-frame, there is a increase of demand such that a unique instance of this application is not enough. To make the service meet the demand, the operator increases the number of **replicas** of the same service "go-service". Because this service is targeting both instances (replicas) that are actually the same application, when a bunch of users request the records the traffic is spread among this 2 same instances making the server able to use more resources and serve more requests by having two instances. After the demand decreases, the operator can reduce the number of replicas and leave only 1 to save resources. **Vertical Scaling**, on another perspective, is a way that can also be used to meet increase or decrease of demand but more related to hardware. As an example lets say there is the same case described before but this time the application is incapable of serving well its users because it's CPU is already exceeding what he can do. With this practical, the operator can escalate vertically upwards to meet the demand by adding CPU and downwards to save resources in case the demand turns out to reduce once again [72][73][74]. Within this scope it is important to mention that **Kubernetes** has mechanisms for making this scalation automatic by using metrics, which are **VPA**(Vertical Pod Autoscaler) and **HPA** (Horizontal Pod Autoscaler). **Load Balancing**, in another perspective is spreading the traffic among this multiple instances, making sure that none gets overwhelmed [75].

Service Discovery and **Networking** are important things to know when it comes to kubernetes. It's exceptional for making communication between various services and pods of the cluster where it's understanding it's important for deploying applications. The **Service Discovery** is a mechanism that allows applications and services to discover others inside the cluster. To make this possible, it requires methods such as **DNS-Based Service Discovery**, which allows pods to communicate with each other using its fully qualified name **(FQDN)**, such as **my-service.my-namespace.svc.cluster.local** and **Environment Variables**, which are injected in each service, making pods enabled to know the existence of others's by existing settled variables. **Networking**, on the other hand, is a model that abstracts the complex realm of the communication between containers, pods, and services. For reaching this abstraction, it uses 3 components such as **Pod Networking**, where each pod has it's own unique IP address and the containers within the pod can communicate with each other using **localhost**. Containers inside of a given pod can communicate with another container inside of another pod by levering their assigned IP's or by using the component **Service Networking**. **Service Networking** is a way that pods can use to abstract their access, either by using a gateway such as a load balancer or by using the service name within the cluster. Finally, there is a **Network Policies** component that can be used to define rules for ingress and egress traffic between pods. Ingress means the internal traffic, and egress is the external traffic. This is used to control how the internal and external traffic can flow inside of the cluster.

Configuration and secret yielding are a need for every microservice. Is it to retain keys for accessing an API, a password, or even a certificate? This is needed to make connections between a certain number of services while making sure that the services exchange messages with a given degree of security. With this in scope, **Kubernetes** offers configuration management and secrets out of the box. Thus, as responsible for this, there are **ConfigMaps** and **Secrets**, but both serve 2 different use cases. **Config Maps** are used to store non-sensitive configuration in key-value

pairs like environment variables, configuration files, and even command-line arguments that can be passed at runtime to containers without the need to rebuild the containers, serving as an extension to the proper existing environment variables in the pod configuration but without the need of resetting the pod, being changeable at any time. As an example, there is the case where a given property may change in relation to the environment that is under target. A development environment may have its own database and staging another; therefore, there is the capability to change the connection configuration, making it feasible to change without having to restart everything but instead changing just the key value pair in this store. However, in the opposite direction, **Secrets** are deeply related to storing sensitive information, such as passwords, API keys, and certificates. It is stored in base64 format, and Kubernetes has its own mechanisms to securely save this. It is recommended that an organization that wishes to implement this carefully consider the existing security vulnerabilities [76].

For achieving persistence storaging within **Kubernetes**, there are 2 major concepts that must be known. One is for creating volumes that a application may try to own further **PV's** (Persistent Volumes), and the other for making the actual request to own a given persistent volume **PVC's** (Persistent Volume Claims). **PV's** abstract the storage implementation that can be either local, using **NFS**(remote storage) [77] or cloud-based, while still providing various access modes to that volume, such as **ReadWriteOnce**, **ReadOnlyMany**, and **ReadWriteMany**, which specify the type of permissions over it and how many pods can access it. As an example, **ReadWriteOnce** gives read and write privileges, but only one pod can actually mount it.

Like mentioned before, there are a certain group of components responsible for managing the smallest unit that a Kubernetes cluster can have (pods). This components can be used to watch the state of a cluster and make a set of actions to bring this state closer to the state that the user mentioned in it's configuration file, which means that they continuously monitor the cluster and work to make its state converge to the desired state. The existing types for a **kubernetes controller** are: **Deployment**,**ReplicaSet**,**StatefulSet**,**DaemonSet**,**Job and CronJob Controller**,**ReplicaSet Controller** and **HorizontalPodAutoScaller**. As first, **Deployment** is the most used type of controller. It is widely used to deploy and manage stateless applications, making sure that they have high availability and controlled rolling updates. This is done by making sure that a desired number of replicas is maintained for a specific application and that these replicas are running all the time, which usually implies automatically handling scaling, updates, and rollbacks. In second, **ReplicaSet** is what the **Deployment** manages to maintain the replicas, and it is used to ensure that these replicas keep running. **StatefulSet**, on the other hand, is another useful controller that shares characteristics with the deployment, but the difference relies on the type of management required, which according to the nature of the pods in question is different. **Deployment** is for stateless applications, while **Statefulset** is for "Stateful". What this means is that the **deployment** deletes random pods in a scale down and adds pods with random identifiers while the other adds and removes pods sequentially, which helps in certain situations in pods with states. As an example, there are databases that are distributed that are based on a consensus algorithm. To scale down those effectively, there is the need to scale nodes sequentially and not in parallel like in deployments, making the removal and addition of nodes more constant; thereby, in case nodes are removed, the consensus algorithm

will not be unstable and the service will remain available despite this removal. **DaemonSet**, on the other hand, is a pod that is obligated to run per node within a cluster, making it feasible for logging, monitoring, and network cases that require an instance per node. As an example, there is cadvisor that requires an instance per node for collecting data for metrics. For temporary pods, there is also the **Job and CronJob**, which runs pods that are designed just to fill temporary tasks such as backups and data processing. Finally, **HorizontalPodAutoscaler**, however, is a controller that manages the horizontal scaling by looking into metrics such as CPU utilization, memory usage, or custom metrics.

Security within the **Kubernetes** cluster remains one of the major concerns in the industry. This is because of its complexity and widespread adoption, making it a must in organizations, and at the same time, because of its complexity, something that could be targeted from anywhere within its scope. The proof of this is that security for this technology continues to be the most debated topic, promising 12.3% of overall topics, which means that securing this technology is a priority among users in the **kubernetes** sphere [78]. However, there is also some work related to the analysis done around security reports oriented to **kubernetes**. More concretly, according to some works, security breaches in kubernetes are extremely underreported, with 0.79% commits related to security. This reveals that further research must be done to actually spot hidden security vulnerabilities, thereby strengthening security practices [79]. Also, there is an increase in the usage of AI for anomaly detection, which has been showing improvements in threat detection... spot more 92% and reduce response times by 67% compared to traditional methods [80]. Nevertheless, there are also assumptions chattered by Helm charts, evidencing that half of the container images used are outdated and 88.1% of those images are exposed to vulnerabilities [81].

Relatively to the importance of this tool for this thesis, it should be known that the intention is not to simply create a blockchain application but instead gather the best that the market has to offer and create seamlessly coordination between web2 and web3 technologies. With this in mind, all of the use cases displayed here make use of such, taking advantage of all of the best capabilities that kubernetes can offer for such a big organization as a healthcare one.

3.5.2.2- Docker

Docker, on the other hand, is a technology that could be abstracted by the previously mentioned **Kubernetes** and is said to be a tool to build applications in a different paradigm than before. To contextualize, in the past it was something normal to install dependencies directly in your local machine, which created some burden since you may have had different projects with different dependencies versions. Also, the application would mess with the local machine configurations because it was directly using it's resources, having more probability of making adjustments that would compromise other projects by having different configurations. By acknowledging these limitations, Docker came to light because it was conceded to automate deployment, scaling, and management of these referenced applications using a new concept called **containerization**.

Before containers, there was a concept called **virtualization**. This is still widely used, but it became less with the introduction of containers. This is because, despite being very powerful, **virtualization** is more heavy than a container, though offering the same characteristics.

The main advantages, on the other hand, for this tool are **isolation** from the operative

system, the possibility to run **more instances** compared to VM's, **portability**, **more collaboration** within the team, **version control**, **streamline process** capability, and the capability of **scaling** if used along side kubernetes. Thereby, it's main features are: **Lightweight Virtualization**, **Containerization**, **Portability**, **Isolation**, **Version Control and Reusability**, and **Scalability and Efficiency**.

Due to its importance not only within this project but also the current landscape of technology, it is considered a very special tool; therefore, it will be discussed here in this section.

Containerization,like mentioned before is a technology in software engineering to package all dependencies and software all together, creating a environment that shares the kernel with the host but isolates the rest of the resources, while being a lot more light than VM's. Because of this,the application runs in any kind of environment, since this creates isolation. By the cause of such, the main characteristics associated with **docker** are **Isolation**, since processes,memory and file systems are totally segregated, such that multiple containers can run without interference with each other. **Portability**, because it packages everything in a single unit. **Efficiency**, due to the fact that is very lightweight and **Scalability** because it can be scaled up or down easily and there are orchestration tools that make this even more easier such as the previous mentioned **Kubernetes**.

To understand how Docker works, there is the necessity to view how it's architecture is composed. With this in mind, the main core components of Docker are the following: **Docker Client**,**Docker Daemon(dockerd)**,**Docker Images**,**Docker containers**,**Docker Registry**, **Docker Network** and **Docker Volume**. **Docker Client** is responsible for sending requests to a given instance of the Docker daemon. This instance can either be locally or remotely located in another machine. To make such communication there are commands such as **docker build**, **docker run** and **docker pull**. **Docker Daemon(dockerd)** is the one that is responsible for managing containers on the host system while ensuring isolation and resource management between containers. Also, this is also what is responsible for serving requests from the Docker client, making it able to manage Docker objects like containers, images, networks and volumes. **Docker Images**, on the other hand, are read-only templates that define the structure that a container should have. This template contains everything related to an application, like code, runtime, libraries,environment variables, and configuration files, and is usually created accordingly to a **Dockerfile**, where it gathers all the necessary instructions for setting up an image. **Docker Containers** are instances that could run that came from a previously created template (Docker image). Each instance can run without affecting others', but it does share resources over the machine they are running on. They have their own file system, network and process space. **Docker Registry** is another important component, which is where images can be published. This publication is done in a remote repository that can be later used to download and use images as intended in our applications. This registry can either be public or private, giving power to those who use it. **Docker Network** is what enables the communication between containers. This container can be either on the same host or across multiple hosts, having several networking drivers like **bridge**, **host**, and **overlay**. Finally, **Docker Volume** is the persistent storage that a container can have. This is done by mapping the container file-system with the host, which otherwise would be always resetting every time a new container is created, which

may not be feasible in some situations.

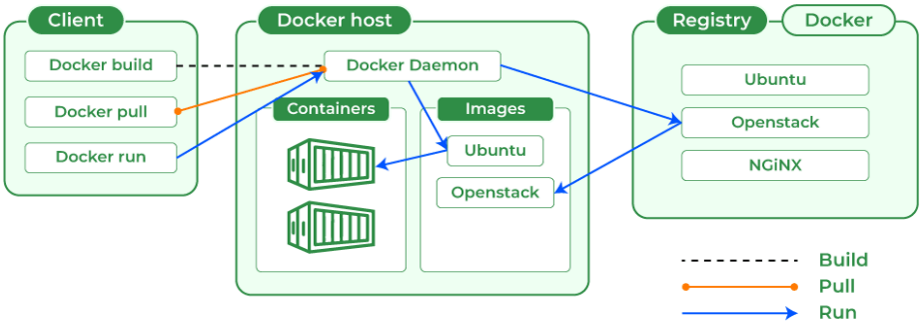

Figure 9: Docker architecture: simple view

In the realm of the **docker security**, it has a lot of features that contribute for this to be ensured. However, there must be configured accordinally, otherwise it may not be as much effective as it should. With this covered, there are several features that docker disposes that actually help establishing this type of securance like **Namespace Isolation**,**Control Groups(cgroups)**,**Read-only file systems**,**user namespace**,**security capabilities**,**Seccomp**,**AppArmor and SELinux** and **Image Security**. **Namespace Isolation**, is a method to isolate even more containers and the host system by sogregating its network,users and mounts. **Control Groups**(cgroups) are groups used to limit and prioritize resources such as CPU,memory and disk I/O among containers, which prevent containers of consuming resources excessively. **Read-only file systems**, is another feature which enables the container of being only allowed to read and not modify it's file system. With this malicious actors cannot modify or inject malicious code during the runtime. **User Namespaces** is responsible for running containers as non-root users, making unfeasible the possibility of malicious actors to scalate it's privileges to another containers, therefore only comprimising a single container. **Security capabilities** are a set of capabilities that the administrator can set for a container to be dotted of. By default this containers already come with a certain number of capabilities that are not harmful. However, the administrator should understand and add or remove them accordingly to it's need to make sure that the application remains the most secure it can be. **Seccomp (Secure computing Mode)** however, its another feature which is composed by profiles that reduce the number of systemcalls that a container can do. This systemcalls are calls that are made directly to the hardware and if profiles are well settled, a good defense against potential exploits is present. **AppArmor and SELinux** are security modules to enforce access control policies, providing mandatory access control on the host system. This ensures that containers can only use resources defined by this policies. Finally, there is **Image Security**, where docker offers a command for scan a image in order to detect it's vulnerabilities. This command is **docker scan**. Also, there are plenty of information regarding good practises in docker such as **Use Official and Trusted Images**,**Least Privilege Principle**,**Enable Read-Only File Systems**,**Limit Network Access and Inter-Container Communication**,**Regular Vulnerability Scanning**,**Usage of Docker Content Trust (DCT)**,**Implementation of Seccomp,AppArmor or SELinux Policies** and **Avoid Storing Secrets in Images** as an example [82] [83] [84] [85].

This tool is core for **CI/CD Pipelines**. **Continuous Integration** (CI) is a pipeline that aims to create a flow for code, build, and test applications, while **Continuous Development** (CD) is a pipeline for release, deploy, operate and monitor applications. To achieve this, **docker** is the tool that is used to build images, to release, deploy, and to monitor. These practices are very important because they speed up the development, thereby making Docker very important from this perspective. Also, another usage of Docker is within the **kubernetes** realm, since this is the containerization technology that is often more used. Kubernetes requires Docker because it manages containers, where Docker fits religiously.

Also, docker is very important in this thesis since it has all of the characteristics covered so far and it's used across all of the use cases.

3.5.2.3-MetalLB

On-premise cluster is an advanced concept related to setting a network of computers within your own infrastructure. In the case of this project, the reference to this is related to a network composed of **kubernetes** nodes. Because of this, it is imperative to mention **MetalLB**, which is a **load balancer** conceded just for **on premise Kubernetes clusters**.

When it comes to **kubernetes**, usually the tendency is to relate it to the cloud. However, as the time progresses, components become more powerful and cheaper, and more and more, the consideration of moving to the cloud or not becomes more a question of cost and responsibility. This is because, though the power of components is becoming stronger, the complexity of hardware is sometimes increased, making organizations choose between being themselves responsible for the maintainability of the infrastructure or simply giving it away to those that are more specialized in the subject, which may or may not reduce costs for them because not only do they have the common infrastructure costs but also the burden of finding and maintaining qualified personnel for this type of job [86].

MetalLB comes into play when it comes to those more audacious to have all the responsibility. As a strong **Load Balancer** for **on premise** networks, it stands in this project as a marvelous tool. for achieving load balancing, which is a concept where the requests are distributed through multiple instances of the same application, balancing this way the load that each may have. Cloud distributors have their built-in support for such, while on-premise networks do not, which makes **MetalLB** a must for setting such capability in a non-cloud environment.

With this in mind, with features such as **Load Balancing for Bare-Metal Kubernetes**, **Layer 2 and Layer 3 Modes (BGP Mode)**, **Address Pool Management**, **Seamless Integration with Kubernetes**, and **High Availability**, **Metalb** is indeed a genesis block when it comes to mirror cloud environments locally, therefore this section will be covering it at its best.

To put it more clearly, the need for **MetalLB** in Bare-Metal Kubernetes cluster has to do with several reasons, such as: **Lack of Cloud Load Balancer Services**, **Direct Integration with Bare-Metal Network Infrastructure**, **Ease of Configuration and Flexibility**, **Avoiding the Need for External Hardware Load Balancers**, **Support for LoadBalancer Type Services**, and **Scalability and High Availability**. In regards to the **Lack of Cloud Load Balancer Services**, this means that on-premise there is no load balancer out of

the box like in a cloud environment. Thus, in the cloud there is no need at all to implement your own, while in your own infrastructure there must be a settled one; otherwise, load balancing will be a feature out of scope. Direct Integration with **Bare-Metal Network Infrastructure** means that **MetalLB** can integrate this functionally directly with the existing infrastructure. This is done either with IP advertising in **Layer 2 mode (ARP)** or with **Layer 3 (BGP) mode**. **Ease of Configuration and Flexibility** because **metallb** it's easy and flexible to configure, like it will be mentioned in future use cases. **Avoiding the need for external hardware load balancers** it's also another good characteristic, which means that, while other organizations resort to the usage of expensive hardware for load balancing or complex software-defined networking, **Metallb** offers a software-based and yet native approach to manage the necessary operations to come up with this load balancing [87]. **Support of LoadBalancer type services** is another big factor, since without a load balancer this is not possible, where **metallb** comes into action. Finally, **Scalability and High Availability**, since it can scale with the number of nodes within the cluster, distributing service traffic evenly across multiple nodes.

Speaking about it's architecture, it should be known that it's simple but yet powerful. It has a brain by the name of **controller** that monitors the Kubernetes API server for services that require a load balancer, thereby deciding how to allocate IP addresses on them, while still being responsible for managing the state for each of the load balancer services, ensuring that the right configurations are established for further well-conducted external connectivity. At it's core, it also has the **speaker**, which is a component that announces the external IP's using protocols such as **Layer 2** or **Layer 3**, enabling this way the cluster to communicate with the external network, making services accessible outside the cluster. As a finishing term, there is also the already mentioned protocol, which is the **Layer 2** mode, where the speakers announce the IP's by levering l2 capabilities such as the **ARP** [88] protocol within the local network segment. This mode is more suitable for smaller clusters, where the **L3** is more suitable for bigger clusters. This is because l3 uses a **Border Gateway Protocol**(BGP) [89] [90] that advertises the IP addresses to external routers while also offering advanced capabilities such as networking policies and fault tolerance.

Within a more deeper analysis of **l2** mode, it must be known that it works under a local area network (LAN). This is the simpler configuration mode supported by **metalLB** and works with the leverage of both **ARP** and **NDP** to advertise the presence of a service IP within the local network. With this implementation, what happens is that a ip address is generated from a pre-defined pool of ip addresses and then, recurring to ARP, it announces a given IP that then is requested to the network and exchanged by a **MAC** address.

On regards to **l3**, on the other hand, it works using external routers. This makes it more suitable for larger clusters or envronments where complex routing and multi-subnet configurations are required. Also, it offers more advanced capabilities such as network policies and fault tolerance. It works by making sure that a given external router is compatible with the speakers, it communicates with them to check the ips to be advertising and makes those available outside of the cluster where any request will pass throught them and get redirect to the correct node in question.

Feature	Layer 2 Mode	Layer 3 Mode (BGP)
Protocol	ARP (Layer 2)	BGP (Layer 3)
IP Advertisement	Local network (broadcast domain)	External routers (inter-domain routing)
Traffic Routing	Directly to nodes via ARP	Through external routers using BGP
Scalability	Suitable for small networks with limited segments	Suitable for large networks with multiple subnets
Use Case	Small clusters or development environments	Large-scale clusters, production environments
Configuration Complexity	Simple to configure	Requires BGP setup and external router integration

Figure 10: Layer 2 vs Layer 3 Mode

To use this extension, it must be known which kind of **Kubernetes** service can be used to expose the traffic. In the case of **MetalLB**, the only service that he does support is the **LoadBalancer** type. However, there are a lot of others that may be discussed forward.

Within a lot of major capabilities, **MetalLB** also offers both high availability and failover. Such ensures that services remain accessible even in the case of node failures or network problems. High availability is achieved with both **L2** and **L3** modes, depending on the network configuration and it's specific requirements. As an example, when a failure occurs in a L2 implementation and one of them goes down, it rapidly selects another node for that precise IP, while in a L3 mode, there can be implemented an equal-cost multi-path routing, which allows multiple nodes to advertise the same IP and switch the IP to another node in case this node goes down.

The importance of such in this project is that most of the healthcare system's relies on an on-premise network. Doted of such, and in case there is a Kubernetes network, it is very important to have a mechanism to create load balancing for some precise components, which will be occurring within the use cases realm.

3.5.2.4- Istio

In this thesis, **Istio** appears more like an extension and not concretely as something that was of extreme usage. However, because it was used to determine whether each measure is useful and also because it is very different from other concepts, explanations over what it is are required because, although not extensively used, it is mentioned in the thesis.

Microservices is an architecture of exalted power. However, what comes with great power also requires great responsibility. Therefore, conducting the management of the communications between services is not a straightforward task. By virtue of such, the more the application expands, the more work there is around this matter, making **DevOps** teams struggle when it comes to make optimization and manage this within the application realm.

By means of this, **Istio** surges as alleviation of such. At the hand of its capabilities, it serves as a **service mesh**, which is nothing more than an abstraction created around the application for providing **Traffic Management**, **security**, **Load Balancing**, and **monitoring**.

As mentioned before, cloud-native architectures and microservices are becoming something permanent. Such that managing it is a challenge. Thus, using a **Service mesh** could potential be something meaningful because it works as an abstraction layer that handles the dynamic networking and communication between microservices. With this brief introduction in mind, thinking in the most important aspects for which this is needed is something intriguing. Firstly, **enhances microservice communication and networking** is one of the aspects for including istio. This is because, a service mesh provides service-centric networking, thereby oferring features that are not native to kubernetes. This is helpful for managing services communications with capabilities such as load balancing,retries,timeouts and circuit breaking [91]. In second, despite current implementations create overhead in terms of resources, due to the proxys and the new side containers, there are service mesh implementations that actually aim to **improve performance and resource efficiency**. This can be done with new architectures like **Flat-Proxy**, which reduce performance bottlenecks and improve resource utilization [92]. As a third motive, there is the aspect of **Observability and Monitoring**. This as to do with the fact that a service mesh collects metrics,traces and logs for each service interaction, providing insights about the behavior of the microservices enabling performance bottlenecks spotting, understading of service dependencies and improving system debugging [93]. In forth, it should be known that it also facilitates traffic management through fine-grained control policies, such as traffic splitting,canary releases and fail-over strategies. However, there are studies affirming that it creates overheads such as 269% latency and 163% CPU usage, emphasizing the need for optimization [94]. When it comes to the final, the **Security and Resilience** is another aspect that serves as a benefit to use it since a service mesh usually offers mutual tls out of the box with certificate rotation [95].

When it comes to the **Istio Architecture**, there are two major main components such as **Control Plane** and **Data Plane**. The **data plane** handles all the network traffic between services. This is furnished using **Envoy proxies**, which is a sidecar container or a container that stands in addition to the main container. This side container catches and controls all inbound and outbound traffic for the service. With such an broker, the given service gains extended network capabilities like load balancing, enforcing policies, collecting telemetry, routing, and observability. The **control plane**, on the other hand, manages and configures the proxies from the data plane. It had several valuable components, such as **Pilot**, **Citadel**, **Galley**, and **Istiod(Unified Control Plane)**. The **Pilot** is the component that is responsible for service discovery, traffic management, and distribution to envoy proxies that translate high-level routing rules into configurations understood by envoy and push them to the proxies. The **Citadel** manages all the security mechanisms, like the security in the communications and certificate distribution, providing strong service identity and mutual TLS communication between services, and automating key and certificate management for service-to-service authentication. **Galley** validates and processes configuration files before distributing them to other components, while ensuring that only valid configurations are applied, reducing this way the errors. Finally, **Istiod(Unified Control Plane)** is the aggregation of Pilot, Citadel, and Galley in the new versions.

In the sphere of **traffic management**, more concretely, referring to the **Istio** landscape.

This is an existing concept for enabling fine-grained control over the flow of traffic among services by leveraging routing rules, controlling the flow of traffic, and implementing fail-over policies to ensure reliability and robustness. To make this happen, there are several resources that can be used within the configurations that could be used to manage such, like **Virtual Services**, **Destination Rules**, **Gateways**, and **Sidecars**. **Virtual Service** defines how requests are routed to a service within the mesh by setting rules that could describe how requests should be forwarded based on characteristics like path, HTTP headers, or traffic weight distribution. **Destination Rule** are the rules destinated to a specific service or a subset of a service, like load balancing settings,connection pool sizes, and outlier detection. **Gateway** configures how traffic enters and leaves the service mesh, enabling to specify the behavior in ingress (incoming) and egress (outgoing) traffic. **Service Entry** is to manage outbound traffic to services that are not apart of the mesh, and **Sidecars** configures the sidecar proxy behavior for a specific workload.

Security measures for such helpers in traffic management are another thing to take into account. To conquer this, Istio offers features for authenticate, authorize, encrypt and provide identity management. This is helpful because it removes the need for the operator to implement such things in the application itself, thereby segregating the application security from the application and making it easier to change settings as time proceeds. It provides **Authentication** by ensuring the identity of both services and users is valid. This is achieved because it has mutual TLS with automatic rotation for both clients and services, while also offering JSON Web Tokens (JWT) for end-user authentication. It provides **Authorization** because it has **Authorization-Policies** that define what kind of actions are actual permitted or denied based on attributes such as source, destination, and content requested, while allowing to apply them at the namespace, workload, and service level. It also lets us change the **mutual tls** behavior, mentioning if there can be a hybrid approach of both mtls and plaintext traffic or if this mtls should be strict to the point that only mTLS traffic is accepted. **Identity Management and Certificates** is also offered by creating a unique identity in the form of a **SPIFFE**(Secure Production Identity Framework for Everyone) URI, which is tied to the service account in Kubernetes. Offers **Security policies** for defining authentication settings like **PeerAuthentication** and **RequestAuthentication** and for authorization settings such as **AuthorizationPolicy**. Finally, it also offers security among **ingress** and **egress** traffic, which also molds the behavior of the traffic that enters or gets out of the cluster.

Since this works out of the box like an abstraction over the traffic, it becomes clear that observability and monitoring over it would be feasible. What can be found here is **Metrics collection** , **distributed tracing** , **logging** , and **visualization capabilities** . Relatively to **Metrics collection** , they cover an amplified range of matters, covering HTTP and TCP traffic. This data is usually in time-series format, where Prometheus comes to play to be its widely used collector. This way, Prometheus usually collects the data, and Grafana presents it to someone. Some of the metrics that can be found are something like **istio_requests_total** for the total number of requests that a service received,**istio_request_duration_seconds** for measuring latency in seconds, **istio_request_size** for checking the size of the requests received,**istio_response_size** for checking the size of the requests sent and **istio_tcp_connection_opened_total** for the total number of TCP connections opened. For **distributed tracing** , there are integrations with

tracing tools such as **Jaeger** , **Zipkin** , and **OpenTelemetry** , which allow us to understand the existing flow of communications between services, helping in debugging, bottlenecks, and error detection. Also, within the **logging** realm, Istio offers a generation of logs for all the service requests and responses, providing detailed information about such, which is imperative for things like troubleshoot detection, auditing of service interactions, and paths and headers monitoring. For visualization capabilities, on the other hand, Istio also offers good options, such as the integration with **grafana** and **kiali** . For both, existing dashboards are already at the disposition of everyone, which means that little configuration must be done, though Grafana is more suited for all and Kali is more custom-made to service meshes, presenting more capabilities out of the box.

Service Discovery and **load balancing** are key components from Istio. The first helps services discover each other, and the second one balances the traffic efficiently within the mesh. This is important because in microservices environments the number of components is not always constant; thereby, there must be some kind of mechanism to make their increase or decrease automatic with little configuration involved. Relatively to the discovery, Istio first retrieves information about services and their associated endpoints. Secondly, all of the services and endpoints are stored in a specific service registry, maintaining this way up-to-date information about service endpoints, IP addresses, and port mappings. Thirdly, the service discovery information and traffic management rules are pushed to the envoy sidecars, enabling sidecars to perform with the most recent service state. Forthly, the sidecars use the service registry information to make intelligent routing decisions, like selecting the appropriate instance of a service based on the available endpoints and configured load balancing strategies. Finally, by the usage of the previous spoken **ServiceEntry**, the Istio adds other external services to it's registry, enabling them to be there like they are apart of the mesh. In the behavior of the load balancing, on the other hand, it gets configured by **DestinationRule** resources. Also, it has a lot of different strategies such as **Round Robin** which is a default and simply distributed its traffic evenly, **Random** which randomly selects a service to handle the request, **Weighted Least Request** which routes the traffic to the instance with the least number of active requests, **Ring Hash** which maps requests using hashes, **Consistent Hashing** which behavior is alike **Ring Hash** but for managing the requests from a given client to a concrete backend, **Least Connection** to redirect to the instance with fewer active connections, and **Passthrough** which forwards the traffic to the backend without any load balancing.

The key component of Istio is the sidecar container based on **envoy**. This is what is deployed along with each service within the mesh, offering security, traffic management, observability, and resilience for microservices while ensuring compliance with policies, traffic splitting, security enforcement, and telemetry collection. It is a high-performance edge and service proxy designed for cloud-native applications and has become the default data plane for Istio.

Istio is also doted with multiple features to increase reliability. These features are very well known in the distributed systems context and tend to help ensure that microservices can handle graceful failures, prevent cascading issues, and provide robustness in complex service interactions. These services are: **Circuit Breaking**, which is a pattern used to prevent a system from repeatedly making requests to the same failing system. When a given number

of requests fail, this mechanism blocks the service for a given period of time, enabling him to recover and avoid it's exhaustion; **Retries** are another mechanism, but it stands for retrying the request in case it fails before returning an error to the client, and **timeouts** for a maximum time that a service should wait for a response from another service. In case the request exceeds this timeout, the request gets terminated [96][97][98].

Finally, the last capability of istio is something related to it's flexibility. With istio there is the possibility to create single to multi cluster deployments and single to multi network models as well, making is feasible even for segregated environments.

3.5.3 Analytics

This chapter centers it's focus in Analysis. When there are ideas about a given system, usually people don't think right ahead on this matters. Despite the possibility of creating systems without such capabilities, it will become clear further in time that it will be painful and more like a burden if there are none, this is because, the more complex the system gets the more there is a need to auditing and this is only possible if there are mechanisms prepared for the analysis of the network. In other words, despite not directly mandatory, it is almost a obligation to make our infrastructure doted of such, because by having good design over the analysis, auditing becomes more easier and there is the possibility of knowing where to do better by spotting, for example, bottlenecks but also where in the system something is failing (debug). Also, there is a need of having tools that simulate real behavior from users so that the responsible knows the limits of the given System. With this in mind, in this section, it will be covered all of the tools that in conjunction are used to deliver this capabilities that were just spoken. Firstly there will be a discussion about **Caliper** which is a framework to benchmarking blockchain networks, such as **hyperledger fabric** in our research projects. Secondly, there is **prometheus** overview which is a time-series data supplier. Thirdly, there will be information about **cAdvisor** which retrieves measures from machines and finally there will be a quick representation of **grafana**, because altought not used in the final implementation, it remains crucial for benchmarking since it has more visualisation capabilities for result comparison, which makes it more feasible than the ones built inside of the project.

3.5.3.1- Caliper

In what respects the domain of **Benchmarking**. It is well familiar that it represents the **measuring** and **comparing of performance**. As the need of metrics arises during a project, there will be space for such concept and within this thesis this could not be different.

In prospect of the usage of such concept, within this project,there will be justification of results. Thereby, the need to use a tool such as **Caliper** and the requirement for further clarification of it, which will be stated in this category.

Caliper [99] is a open-source multi-function benchmarking tool. It is part of the hyperledger project and allows it's clients to evaluate various aspects of their blockchain implementations. This aspects could be **transaction throughput**,**transaction latency**,**resource utilization** and even **scalability**. Thought it's modular architecture, it is possible to implement a vast number of custom connectors, enable the developer to connect in various ways to

multiple blockchains. Also, there is by standard connectors for a lot of network tastes such as **Hyperledger Fabric**,**Ethereum**,**Hyperledger Besu** [100] and **FISCO BCOS** [101]. This flexbility allows those that have power over it to compare different solutions. Key features of it are: **Benchmarking Across Multiple Blockchain Platforms**,**Customizable Performance Tests**,**Detailed Performance Metrics**,**Modular and Extensible Architecture**,**Automated Benchmarking** and **Visualization and Reporting**.

Relatively to the key metrics in the hyperledger caliper, there is the **Throughput**, which is a measure of how many transactions a network can handle per unit time. **TPS (Transactions per second)**. **Latency**, which is the time taken for a transaction to be confirmed and added to the ledger after submission, **sucess rate**, representing the percentage of submitted transactions that were included, **transaction response time** is the time that a response takes to come from the network, including both validation and commit phases. **resource utilization** for measuring the usage of CPU, memory, disk I/O, and network usage across different nodes, **read/write operations** to measure the number of read and write operations performed during the execution of transactions; **commit time** to the time required for a transaction to be committed to the ledger after being validated; **block creation time** for the time it takes to create a block from validated transactions; **transaction ordering time** to measure the time taken to order transactions before they are batched into a block; and **failure analysis** to analyze the reasons for transaction failures like invalid input, lack of consensus, or even resource constraints.

Entering into the architecture of caliper, there are several concepts that could be spotted here. This key components are **benchmark module**, **adapter layer**, **monitor module**, **workload module**, **client and peer nodes**, and **reporting module**. The **Benchmark module** defines the overall testing workflow, being responsible for reading the user-defined benchmark configuration files, specifying workload settings, test duration, and the blockchain network to target. The file to configure all of this typically is the **benchmark.yaml**, including information such as the number of test rounds, transaction rate, and concurrency level. In the **adapter layer**, there is the abstraction of the communication with blockchain platforms, where each blockchain has it's own connector, and also there is the ability to create a custom-made connector as well, making it feasible to adjust things to the needs and even add new blockchains to benchmark. There is also the **Monitor module**, which tracks the system resource utilization during the benchmark test, capturing data such as CPU usage, memory consumption, disk I/0, and network activity, while also being able to integrate with Prometheus and Docker to collect these statistics, making the operator enabled to know performance analysis and bottleneck identification. **Workload module** is another component that is responsible for actually making sure that the workload state specified by the user is actually filled to come up with the scenarios. In other words, it's what controls rates and the number of transactions. **Client** on the other hand is what actually sends out the transactions to target the blockchain network peer nodes, such that they operate in parallel, simulating a realistic multi-client environment to stress-test the network. Finally, there is the **Reporting module**, responsible for aggregating the results for further analysis from the user side. It can generate these reports in various formats, such as JSON, HTML, and CSV. This report includes metrics like throughput, latency, success rate, and resource utilization.

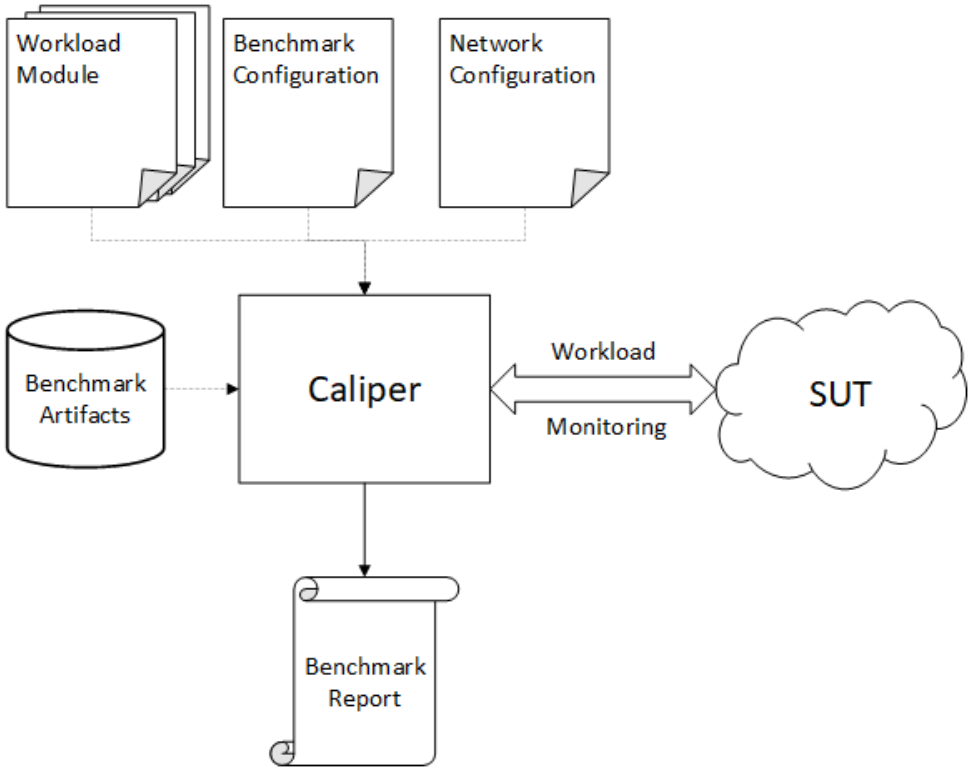

Figure 11: Caliper architecture

In order to come up with a benchmark, there are several workflow steps to be taken. Firstly, it should be defined as a benchmark configuration within the **benchmark.yaml** file, which includes network configurations, test scenarios, and workload parameters. Secondly, create the necessary configurations in the **network.yaml** and test it to see if it does work against the desired network. Thirdly, execute the workload module, which is the transactions that could be either of read type or write type. Fourthly, submit the transactions using the fabric adapter, connecting to the more suited node to ensure that transactions are validated, ordered, and committed to the ledger. Fifthly, the transactions get submitted and go through a validation phase where endorsement policies are checked. In case everything goes well, the transactions will be batched into blocks by the orderer and distributed among all the peers, giving a positive or negative response depending on whether it was successful or a failure. Sixth, the monitor module collects all of this data on system resource utilization, such as CPU, memory, disk I/O, and network usage in all participating nodes. Sept., there will be the reporting generation that the user must analyze. Finally, there is a result aggregation, and a final report is generated, which can be later used to take action over possible bottlenecks or misconfigurations.

3.5.3.2- Prometheus

Prometheus is an open-source **monitoring** and **alerting tool** designed for recording real-time metrics. At the essence of this thesis, there will be instances where **Prometheus** is mentioned. This is because it is very useful to gather data from various places at the same time, making it easier to grab the data at a single point of failure. In other terms, it is a tool based on **time-series** and it remounts to the collection of metrics from various systems and services, making it very suited for this kind of project. Also, it provides various useful features

such as **Time-Series Data Model**,**Pull-based model**,**PromQL**,**Alerting**,**Service Discovery**,**Multi-Dimensional Data**,**Efficient Storage** and **Dashboards and Visualization**.

More information about **Prometheus** will be delved into deeper in this category.

When crossing the architecture of Prometheus, it must be known that it has a lot of components in it, making it robust but at the same time reliable for aggregating sources of data. The key components of Prometheus are: **prometheus server**, **data storage and time-series database**, **service discovery and target scraping**, **promql query engine**, **alert manager**, **exporters**, **push gateway**, and **visualization tools**. As the core component, there is the **prometheus server**, which is responsible for collecting and storing time-series data, running queries, and generating alerts. It is capable of scrapping metrics from targets and storing locally data in a custom format for optimized high-speed writes and queries. After, there is the **data storage and time-series database**, which is where the data will be stored locally, combining in-memory with on-disk data. Blocks store data in intervals of two hours, and there can be set policies to establish how many times data can still remain in the storage, like 15 days as an example. **Service discovery and target scraping** also comes into play to detect several targets such as kubernetes, consul, AWS EC2, and others. These targets can all be configured in the **prometheus.yaml** for smaller or less dynamic environments, while scrap configurations also define how frequently the scrape should be made, what endpoints to scrape, and which labels should be placed to make queries easier. For querying, **PromQL Query Engine** must also be spoken, which is a powerful query language designed specifically for working with time-series data, allowing users to filter, aggregate, and make mathematical operations on it's metrics while still performing real-time data analysis, creating complex queries, and generating visualizations or tables. **Alert managers** is also another important concept to mention since it throws alerts based on rules that come from queries. Basically, if the given query gets a value that was settled as a limit, an alert must be released while having integrations as well with email, Slack, or pager duty to receive them. **Exporters** is also important because these are the targets that must be created in the services in order to actually retrieve data from them. This exporter can be implemented in multiple types of languages. For short-lived jobs, there is also the **push gateway** that can gather the data from components that run shortly and thereby cannot be scraped using the traditional method. Finally, there are the visualization tools. While Prometheus comes already with a very basic UI, Grafana's integration creates a more complete and flexible approach to analyzing this sort of data.

3.5.3.3- Cadvisor

As mentioned before, containerization is a concept for package applications. It creates lightweight,standalone and executable applications, while giving to them all the necessary dependencies to work in a portable,scalable and isolated way. By the effect of such, containers have gained an extreme importance in the current sphere of software engineering,making them of a big magnitude of importance to every single application that needs to be done nowadays.

With this in mind, and within the curious landscape of congregating measures, there is **Cadvisor** as a very useful tool that was built by Google. It has enormous advantages when it comes to the collection of data from insights about containers. By collecting and exposing

metrics such as **CPU**, **memory**, **network**, and **disk usage**, it helps operators to monitor **resource consumption** of containerized workloads. Also, it works like a glove when combined with **Docker**, making it a valuable resource for providing a complete observability of containers. It's features range from **Container Resource Monitoring**,**Per-Container Isolation**,**Historical Performance Analysis**,**Container Health Metrics**,**Integration with Monitoring Tools**,**Low Overhead** to **Kubernetes Integration**, making it imperative to be delved into deep terms in this section.

In terms of integration, it does integrate with various platforms and tools, which provide the previous mentioned features. The most well known platforms are **Kubernetes**, **prometheus** and **influxdb**. Also, it's use cases are enormous like performance analysis and bottleneck detection, resource allocation optimization and application behavior analysis.

Cadvisor, within the advantages, can automatically detect multiple containers while supporting multiple container runtimes like **Docker** and **Containerd**. Second, it is very light and has the capability of giving info in real-time, which means that it not only does not represent an overhead for the network but also projects real-time metrics for further usage. In third, it was an easy integration with **Kubernetes**. This is because cadvisor is already contained within the kubelet module, which is present in every single instance of a node within the cluster and kubernetes uses it as well for pod autoscaling (HPA) to automatically scale applications based on observed resource usage. As a third advantage, it is very easy to implement with the most used tools available on the market, like **kubernetes**, **prometheus**, and **influxdb**, as mentioned before. As a fourth advantage, there is also the versability and flexibility that it offers since it supports multiple container runtimes and also enables custom metrics that could be exposed inside of the proper containers. Fifth, it also has a built-in visualization and user interface UI, which, despite being very simple, could potentially be very interesting from the operator's point of view, making them empowered with fast troubleshooting and analysis. Also in sixth, this is the standard technology to make this monitoring possible, and it could potentially extend to more advanced monitoring solutions such as long-term storage, anomaly detection, and predictive analytics. In seventh, there is the cost-efficiency and resource optimization, meaning that it helps the administrator to make optimizations in the network by alerting bottlenecks and describing the communication behavior between the services. Finally, there is the community support and open-source nature as another advantage, since it creates a sense of active development and community support, thereby having active updates, bug fixes, and new functionalities, while also making it possible to have custom functionalities that could be added by the operator or by someone that wants to help inside of that community.

This technology is very important in the context of the project, since in one of the use cases there is it's extensive usage for collecting metrics and comparing solutions in a common way, evaluating this way the same aspects and data that multiple solutions may have.

3.5.3.4- Grafana

For showing data within the **Benchmarking** landscape, there is the honored mention of **Grafana**. This is a open-source platform for **monitoring**,**visualizing** and **analyzing** real-time data. Thus, alongside **Prometheus**, it's useful to gather data that was collected during tests of performance, but this will be covered better in a future discussion. With this in mind, it is valuable

to debate such tool into this segment.

When abording the data sources and integrations of Grafana, it should be known that it supports multiple data sources, making it easier to create visualizations, dashboards, and even alerts. The supported data sources are **prometheus**, which was covered already in the previous section; **influxDB**, which is another time-series database used for monitoring and alerting; **Elasticsearch**, which is a distributed search and analysis engine usually used for logs and unstructured data, making it worth when searching and visualizing logs and event data; **Graphite**, a monitoring tool used for storing and visualizing time-series data for analyzing system performance metrics; **AWS Cloud Watch**, for monitoring resources and applications that belong to AWS; **Google cloud monitoring**, providing monitoring, logging, and diagnostics for applications hosted on Google Cloud; **Azure monitor** for monitoring Azure cloud resources; **SQL Database**, allowing querying and visualization of relational data like those from MySQL, Postgresql, and MS SQL; **Loki** for aggregating logs, enabling log collection, search, and visualization in Grafana alongside metrics; Finally **OpenTSDB** for scalable and distributed time-series data. With all these integrations, Grafana stands firm as a universal platform for showing data, making it feasible for this project.

As it's core, Grafana supports multiple and different types of dashboards. A dashboard is a collection of multiple panels that together make a view, enabling users to visualize data from various sources. Panels in Grafana have 5 types, like **Graph Panel**, which is a default panel type used for KPI's and critical metrics; **Stat Panel**, to show a single value for scenarios such as KPI's and critical metrics; **Table Panel**, to display metrics in a tabular format, ideal for detailed views; **Heatmap Panel**, for visualizing high-density data, for showing distributions or patterns; **Logs Panel** for allowing logs from sources like Loki and Elasticsearch.

Grafana is important for the context of this project because in one of the use cases, there was the need to have a tool for displaying data, where this fit's like a glove.

3.5.4 Web Development

To build any kind of application, there is a need for standard suitable tools. Since the projects involve creating multiple microservices, there is a huge dispersion in terms of frameworks. This is because, some technologies are better for certain tasks and others are more suited for other kind of use cases. There are cases where within the project it can be spotted 2 to 3 different programming languages and with that also different frameworks. This is because some are better for certain **API's** (ex: Kubernetes API) and others are better for faster build and development in containerized environments (ex: quarkus) which is fine in one hand, if considered that microservices usually means heterogeneous services but it's a struggle in terms of complexity, because it requires a high skilled set of individuals within the team, all to handle that context switch. With this in mind, here resides the most important concepts, specially presented for the web development realm. There are plenty of other technologies/concepts but these are the ones that more satisfy the condition of understandability. Firstly there is React, to create UI's in a more reusable way. Secondly, Keycloak for managing permissions to communication between services, which is very important when it comes to fetching information which it will be the case. Thirdly, quarkus for serving some information within the project. Forthly graphql, which

is a protocol to serve **HTTP** request. Fifth **REST**, which is a alternative to graphql but more used in the web development due to it's **CRUD** capabilities and finaly there is gRPC which is more suitable for more micro and high performing tasks such as side container tasks as an example. Despite all frameworks and protocols differences, they are all essential to build some ideas that were gathered in this project, giving perspectives of robust and well design solutions where each has it's given task to serve.

3.5.4.1- React

If further examination around the questions under research is performed, particularly the second, then it may be expected that there is a UI evolved. Not as a obligation, but more like something that could eventually occur. In the case of this project, there will be mentions around creating one for management purposes of a infrastructure, but more will be revealed as the use case cases come closer. Thus, explanation of the given tool that achieves this must be taken.

React is a tool to build web applications, centered in the paradigm of reusing multiple components, components this, that are modular and can be used in multiple places of the application. Instead of using the traditional way of creating a HTML code for each page, with this technology there is the option of for example using a piece of code that represents a top bar and apply this same top bar in multiple pages, having this way to write less code and also make the application easier to develop. Also, it is easier to make components render in the desired way when data changes occur.

This tool is important for the project in hands because it enables the creation of menus for one of the use cases.

3.5.4.2- Keycloak

When it comes to management of access to services, **Keycloak** surges as a valuable asset. It is an open-source **identity and access management (IAM)** solution aimed at modern applications, services, and APIs. Offers **authentication**,**authorization** and **identity federation services**. In this section, there is a deep dive into microservices, and having a technology that could create means for authentication when dealing with a determined number of services that get exposed at a global level is important. With this in mind, let's dive into the main capabilities of such a valuable arsenal.

Like previously spoken, **Keycloak** is an **identity and access management tool** (IAM). Which means that it was processes to manage digital identities and control access to resources within an organization. To accomplish this, security policies are enforced, authentication and authorization are secured, and compliance is also expected because there is a management of who has access to what resources. The key functions of IAM are **authentication**, **authorization**, **user management**, **single sign-on (SSO)**, which stands for allowing users to log in only once and keep the access, and **federation** to authenticate users across multiple identify providers.

As a key security feature, **Keycloak** has **RBAC (Role-Based Access Control),** which enables administrators to define and manage user permissions based on roles. This mechanism enables that only authorized users have access to specific resources, providing a fine-grained approach to securing applications and services. To implement this, **keycloak** must use it's **Real Roles** and **Client Roles**, where there is the need to assign permissions over both global

(real) and application-specific levels (client).

Keycloak also supports **OAuth** 2.0 and **OpenID Connect** (OIDC), which are both frameworks used for authorization and authentication in modern applications. **OAuth** is an authorization framework that enables applications to obtain limited access to user accounts on an **HTTP** service. **OIDC**, on the other hand, is something that enables clients to verify the identity of the user and obtain profile basic information. In other terms, **OAuth** 2.0 is used for authorization, while **OIDC** is used for authentication[102] [103] [104].

Speaking more concretely about authentication security, it must be known that Keycloak supports **Multi-factor authentication (MFA)**, requiring users to provide additional verification methods beyond a username and a password. With such capabilities, the user is usually forced to combine multiple practices for authentication. As an example, there could be combinations such as password and pin, a one-time password generated by authentication and a code received via SMS, and even fingerprint and facial recognition biometrics; basically, the usages are endless. However, for keycloak the ones supported are **time-based one-time password (TOPTP)** which can be used with authenticators such as Google authenticator or Authy, **webauthn** which enables users to authenticate themselves with hardware, such as a yubikey, **SMS OTP** which is a code that gets's received in the phone, and **email OTP** which is receiving codes via email [105].

The importance of such in this project is uncovered further by the necessity of user management.

3.5.4.3- Quarkus

Quarkus is an open-source Java framework, designed specifically to **optimize** Java applications for **Kubernetes**, **containerized environments**, and **cloud native development**. It's usage is still gaining presence due to its premature environment, but it does show to be valuable when it comes to building and starting up containers, and that is the reason why it is called the cloud native Java solution. By working seamlessly with **GraalVM**, which is used to compile Java apps to native executables, it reduces the **start-up time** and **memory consumption**. Also, there is also included the **live reloading** feature, which, despite being useful because there is sometimes the need to change the application within the cluster, may be a bit worthless because of the existence of technologies such as **devspace** since it allows programming within the cluster and at the same time runs tests like it would be done in a local machine. But with this in mind, it must be said that this tool does improve productivity, and it was a powerful tool for the arsenal of this project [106] [107].

3.5.4.4- Graphql

Within the landscape of web development, there is always the urgency of adopting an approach for the implementation of APIs. This is of extreme importance and can be difficult due to all of the aspects that the given tasks may take. Ranging from a vast number of characteristics, the ones that should be more taken into account when choosing which methodology to use are **Data Model** (if it should be RPC-based, resource-based, or query-based), **data format**, **protocol of transportation**, **performance**, **if it should have real-time data supply**, and **typing**. With this in mind, within this section there will be discussed **GraphQL**.

Graphql is a mid-level, high-performance approach for designing API's. It is **query-based**, supports **JSON**, **has a single endpoint for making the queries**, **runs in http**, **supports real-time with subscriptions**, and **it is strongly typed**. It was invented by Google, and it is used usually for **complex queryies** and **dynamic data requirements** use cases like **Social media apps**, **dashboards**, and even **mobile apps**. The reason for exceeding in this area is because while in **REST**, there is the request of all parameters of a model, in this approach you **only request what you want from a model**, therefore with this approach there is better performance overall in the client side and more overhead in terms of CPU for only returning what was requested by query. As an example, let's think of a model **Person**, where the parameters are: **name** and **age**. With this approach and by using queries, there is the possibility to only request the name and leave the age behind. At first glance, it appears that this has not much relevance, but when it comes to thousands of records and full of relationships, this approach becomes meaningful, and if mobile development is considered, it becomes even more so because of the even more limited resources when compared to a computer. Thereby, the usage of such is indeed useful because, although the resources within mobile are becoming hugely better, there are still a lot of people that cannot afford to have good resources, which could potentially exclude them from using their favorite applications, be it either **Facebook**(the one that created this approach) or **X** [108][109][110].

3.5.4.5- Rest

When it comes to the most widely used approach for designing API's, **REST** is the way that is more present in every single organization. Though less performant in a lot of occasions, it is still the most standardized one, thereby the most used and more accepted. It is **resource-based**, **has multiple endpoints as a source of data**, **uses HTTP**, and is **typically loosely typed**. The best use cases for this type of way are **Simple CRUD operations** and **resource-centric API's** which, more concretely, correspond to **web services** and **public API's** applications [111].

3.5.4.6- gRPC

Delving more into **Microservices** tools, the **gRPC** approach can be seen as a valuable arsenal for that sort of applications. This is because this way of creating APIs is the most efficient because it's presented at a at a lower level, which can be a huge helper for extending networking communications between services. Coming from Google, this is an RPC implementation. The **g** stands for **google** and **RPC** for the remote procedure call. Usually seen as the modern implementation of **java RMI**, it works with **protocol buffers** which is more performant than **json**, it has a **single endpoint with method-based calls**, **uses http2**, it's **high-performant** and **lightweight**, supports **real-time share of data with bidirectional streaming** and is **strongly typed**. Also, the best use cases for this sort of approach are **high-performance**, **low-latency communication**, and **microservices**, with common applications being **microservices**, **real-time streaming**, **internal API's**, and **extension of traditional application communications with more modern ones**. This approach exceeds this project since it is very good for extending communication capabilities and is also the protocol used to build hyper ledger fabric [112][113].

4 Use cases

This section is directed to the use cases produced during this project. At first, a high-level theoretical conception was conceded, which originated a first-use case. This work was further presented in a congress and was the genesis block for the creation of the second, which was more practical, but it had as a starting point reaching a stage where the first could be implemented. In another terms, the first use case "**IPFS and Hyperledger Fabric: Integrity of Data in Healthcare**" works as an extension of a **hyperledger fabric** network due to the incrementing of **IPFS** and the second "**Hyperledger Fabric: Seeking standardization through designing for each type of organization**" is for the actual design of the blockchain network, making sense because of the natural proceeding of engineering where there is a high representation of something, and to reach that representation, there must be the conception of each element of it individually; therefore, after the final conception of the second, there it can be added to the first. Also, the second was the aim of scientific publication because of the results gathered, but that is something that will be shown as there is a movement further on this work. Also, despite only having two use cases by now, the first use case will most likely originate another use case, which would be related to the creation of private **IPFS** network and making benchmarking around it to see it's feasibility.

4.1 IPFS and Hyperledger Fabric: Integrity of Data in Healthcare

The first use case presented in this thesis is the one that was thought first. It works as an extension of the second use case, but it was thought first as a means of theoretical conception of something that could be feasible in the future to do, depending on the benchmarking of the results under the performance and usability of such. Despite not having a practical implementation that could tell us it's usability, it remains as the genesis block that gave the idea of coming up with the second use case; thereby, it will be important in future work to delve more into the capabilities of these ideas. Also, this same use case was presented in the prestigious congress "**5th International Congress on Blockchain and Applications in Guimarães, Portugal**," which gave a lot of insights about the current trends and implementations around private solutions.

Speaking about the project, it concerns the current traditional storage of data, trying to come up with a conceptual implementation of what could be a system that could keep track of file changes by leveraging both **IPFS** and **Hyperledger fabric**. Thus, the main idea was to first store the data in a private **IPFS** network and then store it's main representation in the **hyperledger fabric**, which would keep track of the existing files in the network and at the same time make sure that no changes were done all over these files. Also, to support all of this, it was conceded a probable network that could support such a network, which will be under research in the second use case. But this is something that will be covered further.

4.1.1 Objectives

Speaking about such a theoretical initial project, there is more concretly a set of objectives that were set upon the inception and concretization of such. This objectives are: **Theoretical**

Development of a Feasible Use Case, **Probable Integration of IPFS and Hyperledger Fabric**, **Foundation for Future Research and Implementation**, **Presentation and Academic Recognition**, and **Development of a Probable Network Architecture**. Within the **Theoretical Development of a Feasible Use Case**, this was one of the objectives because **DSR** was in mind; therefore, there must be a use case that could be actually created for later evaluation and gain of knowledge. The conception of such a network is feasible, but if it is efficient, it is something that will be covered in the future. **Probable Integration of IPFS and Hyperledger** was another objective. This is because both technologies have been gaining a lot of interest in the last couple of years, and understanding until each measure **IPFS** could be efficient in private environments would be very interesting for gaining insights in the scientific world and even more alongside the **hyperledger fabric** since most of the public implementations of **IPFS** rely on a blockchain, which is something that was thought to keep. **Foundation for Future Research and Implementation** was another objective, precisely because by inferring a theoretical conception, there is the need to delve into the concepts of both technologies and perceive the environment that is being targeted, therefore making a researcher capable of understanding what could be done next and in a better way. This idea has been proved to be true since the second use case came from this type of logic. **Presentation and Academic Recognition** was another thing in mind since a congress was about to come and there was plenty of curiosity around **IPFS** and its private usage, which works are very low in terms of quantity and quality. Finally, there was this objective of **Development of a Probable Network Architecture** that could address these ideas, which led to a microservices architecture that would be the target of the next use case, which will be covered more ahead.

4.1.2 Plan

A general plan was already covered in the **Methodologies and Tools** but, within this section, the intention is to create a more specific plan within the general one in order to give further comprehensive understanding of what was covered and in each time-frame. With this in mind, let's describe each phase of this use-case and understand more deeply how the result from this was obtained, giving a overview about the knowledge earned during all of that time.

1º Phase: Ipfs deep dive
In the first phase, since it was of the most interest, there was a deep dive of **IPFS**. Documentation was consumed, and this was the beginning of what would be the basis of this theoretical perception of the project.

As an implementation that was meant to become the file system of the future, there is **IPFS**, or **Interplanetary File System**. It presents itself as a decentralized file system that covers a vast decentralized network composed of peers that could be either private or public depending on its configuration. By the effect of such, files and images could be stored in a decentralized way, which, as a consequence, would create an identifier (or **CID**). This identifier, however, is stored within a **DHT(Distributed Hash Table)** [114], serving as a point that could later be used by anyone to retrieve a given file since this identifier is directly linked to the content of the file. This is because generating a **CID** requires hashing the file out, thereby

making this identifier a unique representation of the file where changes to it would create another file with a different hash, creating this way a deterministic methodology to give insights that this file was not changed, making it by extension immutable. Also, to retrieve this file, a user must communicate with a **Gateway**; however, this can also be either public or private. **Ipfs** can be seen as interplanetary due to the fact that it depends on the number of peers within the area and not in a single point as the internet. This means that you can potentially retrieve files on Mars; the only thing that must be feasible to achieve that is that the file that is being requested is within a peer that is reachable on that planet. Besides that, the concept is the same: requesting the file the first time may take a lot of time, but the second time will be much faster due to the caching system that each peer has, making the most nearby peer where the request was done proprietor of the requested file. This cache, on the other hand, has its limitations. If the file does not get marked, it will disappear with the time if no one requests it for a while, which means that if the intention is to not lose the file ever, there must be a marking from the peer on that precise file.

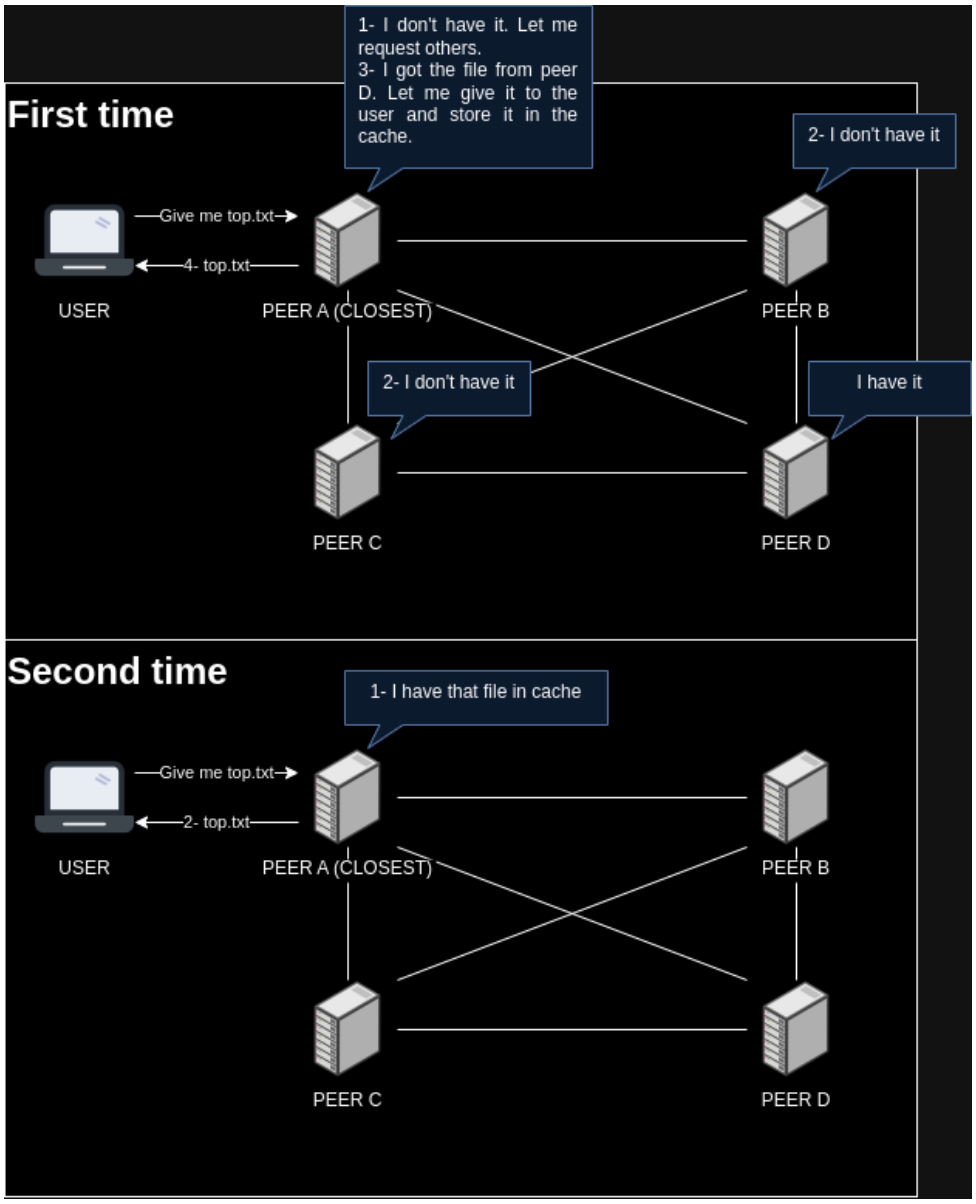

Figure 12: Ipfs simplified

2º Phase: Study of ipfs composition

In this phase, the time was spent almost around trying to understand concepts that were inherited from the **IPFS**. There were a lot of concepts, but a lot of them were focused on covering the **P2P** basics and how distributed systems started to have a body and a form. The main reason for calling this use case the starting building block for the second and future use cases to come is because the basics were all studied during this phase. However, a lot of general common knowledge about blockchains was gathered from public projects such as **Bitcoin** and **Ethereum** from an investor and technical curiosity perspective. With this in mind, in this section, there will be a basic overview of the main ideas from previous projects that were important for the conception of what is **IPFS** now.

Kademlia DHTS

Kademlia [115] is one of the ideas where **IPFS** got inspired. This algorithm is a decentralized

hash table where it stores information relative to the most nearby peers in the network. The value stored comes from generating a public key within a range of space keys that are related to a given region and hashing it right away to form a **peer ID**. When creating a new node, this peer will first connect to a bootstrap node, and apart from there, it will recursively find more and more nodes, updating always its nearby peers as the peer receives requests and serves them. This update is based on checking the **k-buckets** of each peer that is serving the request, having the knowledge of more peers, and thereby updating it's table faster without having to communicate with a bunch of peers individually. This distance is calculated using an **XOR** distance procedure, where the comparison between two binary strings is done, resulting in a binary string that contains more or less 0's depending on its similarities. Every single position of a given peer ID binary string representation is compared with the same position of the other peer ID, resulting in 0 in case both positions are equal and 1 in case both positions are different. Such that in case the binary string representation has a lot of 0's, it means that the id is very alike; otherwise, it's very different, and because these id's are based in space ranges, having more similarities means more proximity between two heterogeneous peers, making this a decisive factor to decide if the table of a given peer should be updated or not. In case it does, the following 2 behaviors may occur: The k-bucket is full, adding a new item because it has more similarities that a member of this list will drop the farthest one and add this new one; The k-bucket is not full; adding a new item because it has more similarities than a member of this list will simply add the new one; With this mechanism, there is a guarantee that the peer will try to grab a file from the nearest peer first, while by not having requests during a certain amount of time this peer will make a scan on its own to make sure that this hash table is still relevant. More ahead, there is a graphical representation of such behavior.

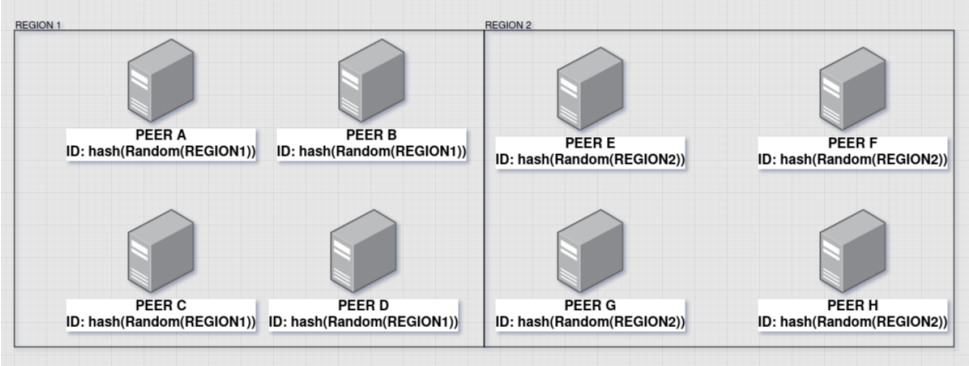

Figure 13: Kademlia: peer id generation

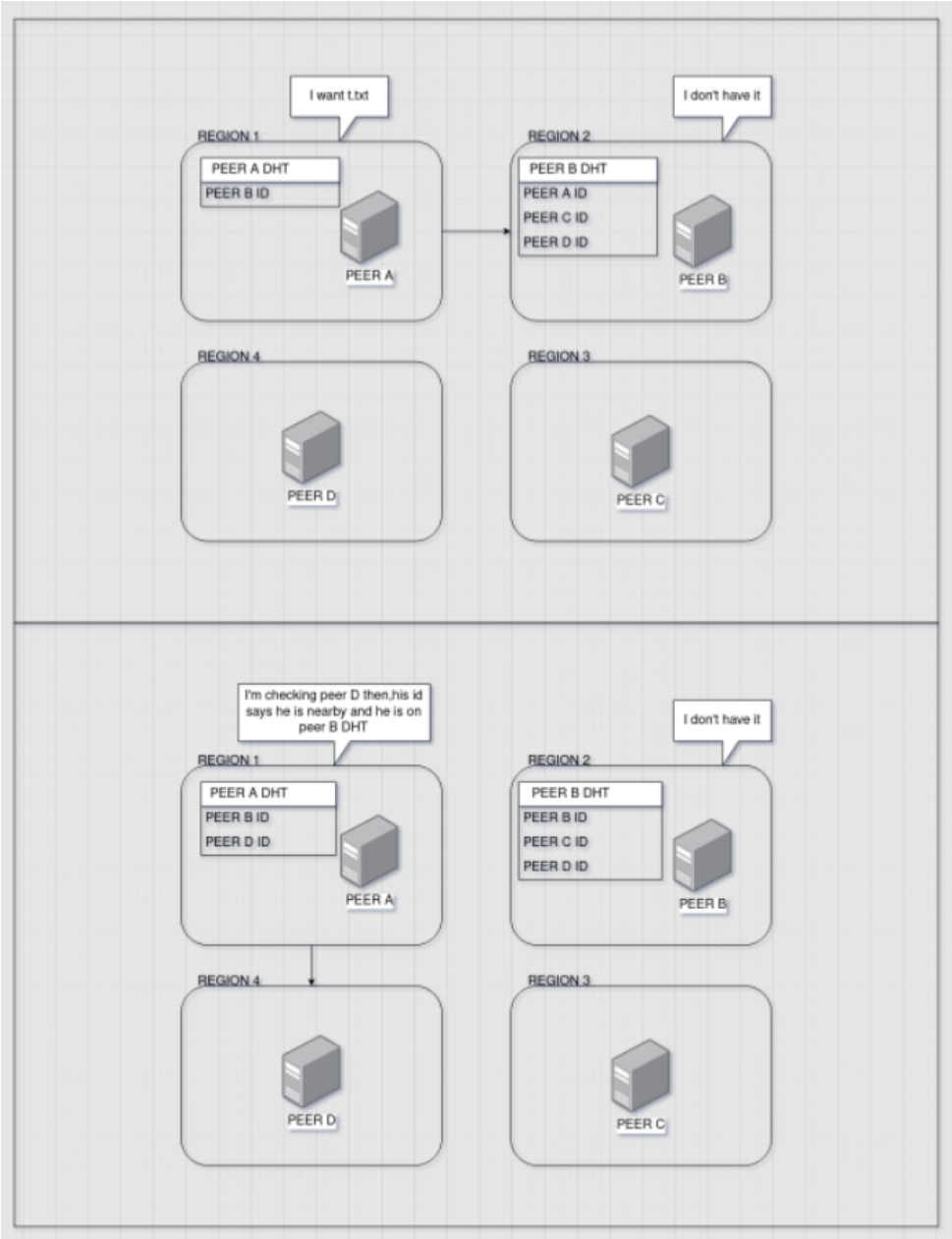

Figure 14: Kademlia: How it works for updating the k-buckets

For example, consider two node IDs `A` and `B`, where `A = 1001` and `B = 0110`. The XOR distance between `A` and `B` can be calculated as follows:

yaml Copy code

```
A = 1001
B = 0110
      ^^
      XOR
 = 1111
```

Figure 15: Kademlia: Example 1 of XOR

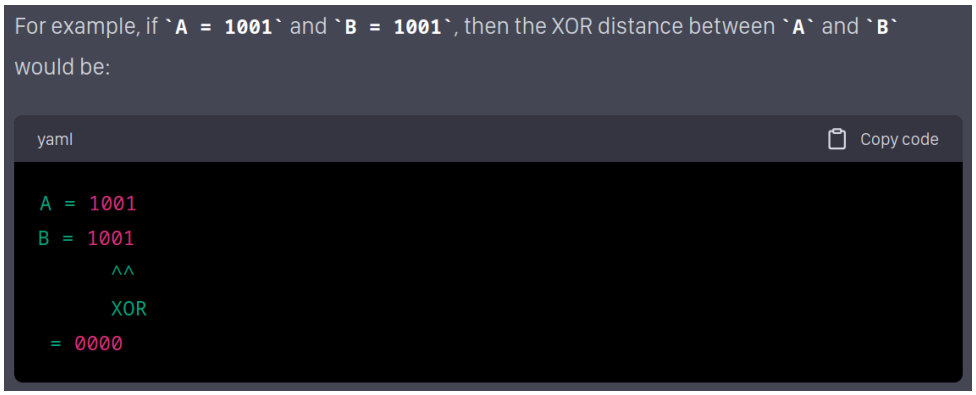

Figure 16: Kademlia: Example 2 of XOR

BitTorrent

Another facet of **IPFS** is the way it shares blocks of data all over the network. which was inspired in **BitTorrent** . **BitTorrent**, like **IPFS**, is a decentralized file system, though it does not call the data by an cid but instead the file name and location. It is meant to provide scalability and efficacy in terms of downloading files, because by storing the files in pieces, it makes it easier for a user to download it faster. The big difference between the two is that IPFS is meant to long-term data storage and addressing content immutability, while the **BitTorrent** is more related to temporary data distribution [116].

Git

Git is another instrument where **IPFS** got some ideas. It provides a version control system that enables the user to go back in time and save some massive changes and update content. In order to do this it uses blobs, but instead of lists, it was trees; However, the concept is almost the same; the difference is that **IPFS** has an **Merkle DAG** and **Git** has a **DAG**. This data structure differs because **Merkle Dags** stores the hash of the hash, which means that it has the hash of the file as the hash of the children's.In contrast, the **DAG** only hashes the file itself. Which means that instead of retrieving the hash, you need the whole content, which in contrast in **Merkle dags** [117] you only need the hash of the children's, making the verification of hashes faster and still reliable. Also, **Git** is built in Ruby script and IPFs is built in Golang [118].

SFS - Self Certified File System

This is an extension of protocol that is aimed at security and creating secure channels. Ipfs has his own implementation, but the way they are represented is alike. **Sfs** uses /sfs/¡Location¿:¡HostID¿, where the location is the server address and HostID = hash(publickey —— Location). This way you can verify the peer while using the address because it was the location associated with the hash plus the public key. In contrast, **IPFS** uses /ipfs/¡CID¿ [119].

Identities

This protocol refers to how **IPFs** identify their peers. This identification is executed like Kandemlia. An user can create a new peer ID whenever he wants to. Though it is recommendable that it stays the same, but users are free to change it periodically if they feel to for questions

of security. While connecting, the peers verify if the public key hashed corresponds to the peer ID; otherwise The connection is rejected.

Network

Ipfs can use any transport protocol, though the best suits are **WebRTC** and **uTP** or **SCTP**. Also, it uses **ICE NAT** transversal techniques; it checks the integrity of messages using hash checksum and checks authenticity of messages using HMAC with senders public key. To know which protocol of transport we are using, IPFS needs to receive a multiaddress, which comes with the protocols used and the way they are connecting [120] [121] [122] [123].

```
# an SCTP/IPv4 connection
/ip4/10.20.30.40/sctp/1234/

# an SCTP/IPv4 connection proxied over TCP/IPv4
/ip4/5.6.7.8/tcp/5678/ip4/1.2.3.4/sctp/1234/
```

Figure 17: IPFS: Example of multiaddress

Routing

In terms of routing, preoccupations can be spotted perfectly when speaking about Kademlia. Despite based on it, Ipfs uses their methodology in sum with **coral** in their implementation. According to their white paper, there's an interface to deal with this tech, and the routing can be swapped if precautions are considered to fit their rules to that referenced interface.

Blocks Exchange

When discussing the way blocks are exchanged in **IPFS**, it must be put into conscience that it is very based on the BitTorrent protocol. However, the main difference relies on the way the object is retrieved. **IPFS** uses the **BitSwap** protocol, which is their own implementation for managing the transfer of blocks between peers. Both protocols are similar in the sense that both involve exchanging pieces of data (blocks) between peers, but **Bitswap** does not impose strict limits on the number of blocks a peer can request or provide, making **IPFS** more flexible in terms of the amount or type of data passed. Also, both implement a marketplace, whereas **Bitswap** can request any kind of block of data that it requires without this being necessary for the original request, function this way in a supply and demand paradigm, making this exchange more dynamic and flexible system when compared to **BitTorrent's** straightforward block exchange. Additionally, **Bitswap** implements a credit system called **Bitswap Credit**, which incentives peers to contribute to the network by either storing and providing data or searching and finding data for other peers. Thus, even if a peer doesn't have requested data blocks, it can still help the network. A **balancing contributions and block sharing** system is also implemented. This is because the network expects that every component delivers value to the network, and in case it does not, other peers may not prioritize serving the requests of the giving peer. Besides this, there are other mechanisms that help on that, like the sharing of blocks of peers with high demand with peers with small demand, making a "bigger fish" share its

blocks with a "smaller" one, preventing "free-riders," and promoting the collaboration between all peers within the network environment [124].

Naming

Naming refers to the **IPNS** concept of **IPFS**. This is a **Interplanetary Naming System**, which can be used to create a point to **CID's**. This is very useful in situations where there can be multiple versions and there is the need to point to the most recent, which can be changed as there are new releases [125].

Objects

In respect to the handling of objects within **IPFS**, it is well known that it got inspired on **Git**. This is because the objects have their value represented by hashes using a **Merkle DAG**. This represents a tree, which creates a deep linking between multiple objects that share a given relationship. As an example, in the event of having a directory, the behavior of storing such structure is to recursively hash the children's all together until having a single father, which represents the directory that owns all of the files. This system proves to be useful because not only does it provide content addressing, tamper resistance, and non-deduplication (file stored twice), but it also makes life easier when it comes to assertions since every file can be asserted against the tree with little few comparisons due to only having to hash the children's to prove that the file that is supposed to be the father is the actual father. In addition, this resulting hash is the before spoken **CID**, which is an ID that a user can use to identify which file is desired, while the person in charge still has the capability to create a given endpoint that controls the versioning like mentioned before. However, the user, by maintaining this **CID**, is known for a fact that the file he wishes is the same file ever.

Files

To manage the merkle dag, there are multiple different data structures that can be used in **IPFS**, which are very alike **Git**: **Blobs**,**List**,**Trees** and **Commits**.

Firstly, a blob is what it is used to actually store the data of the file, while having no links inside of it since it is supposed to be an actual file and not a directory.

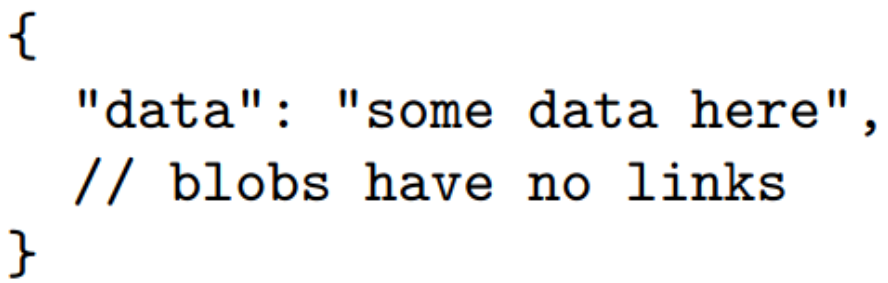

Figure 18: IPFS: blobs

Secondly, there is List which can have blobs and links inside, forming this way a array of objects that can either retain data or point to something that may contain data.

```
{
  "data": ["blob", "list", "blob"],
    // lists have an array of object types as data
  "links": [
    { "hash": "XLYkgq61DYaQ8NhkcqyU7rLcnSa7dSHQ16x",
      "size": 189458 },
    { "hash": "XLHBNmRQ5sJJrdMPuu48pzeyTtRo39tNDR5",
      "size": 19441 },
    { "hash": "XLWVQDqxo9Km9zLyquoC9gAP8CL1gWnHZ7z",
      "size": 5286 }
    // lists have no names in links
  ]
}
```

Figure 19: IPFS: lists

Thirdly, there is the tree which is the same thing as the list but the main difference is that it also contains name of files and can contain commits.

```
{
  "data": ["blob", "list", "blob"],
    // trees have an array of object types as data
  "links": [
    { "hash": "XLYkgq61DYaQ8NhkcqyU7rLcnSa7dSHQ16x",
      "name": "less", "size": 189458 },
    { "hash": "XLHBNmRQ5sJJrdMPuu48pzeyTtRo39tNDR5",
      "name": "script", "size": 19441 },
    { "hash": "XLWVQDqxo9Km9zLyquoC9gAP8CL1gWnHZ7z",
      "name": "template", "size": 5286 }
    // trees do have names
  ]
}
```

Figure 20: IPFS: tree

Lastly, there is the commit which represents a snapshot in the version history, representing changes that were made in a given file. In addition, it also links the author to the objects.

```
{
  "data": {
    "type": "tree",
    "date": "2014-09-20 12:44:06Z",
    "message": "This is a commit message."
  },
  "links": [
```


Figure 21: IPFS: commit

3º Phase: Practical Implementation

Reaching the third phase, after reviewing everything relative to **IPFS** and it's origin, a practical project was conceded just for creating a tangible way to obtain knowledge. This is important because relying only in documentation sometimes leaves some subjective understanding, which could result in bad insights. With comprehensive basic projects, there is more probability for this to not be the case. However, despite having practical insights this cannot be considered within the **DSR** methodology, precisely because there is no objective such as solving a real problem in sight.

The project was a decentralized per peer dropbox, which could be private or public depending of if you open the gateway to other peers or not. With such application, adding files on drop, add files from other peers using their **CID**, display the name of those files and it's information, download those files to our local machine, list all the connected peers and force an connection to an certain peer directly is possible. Pinning is not discussed, but it's something that could be done.

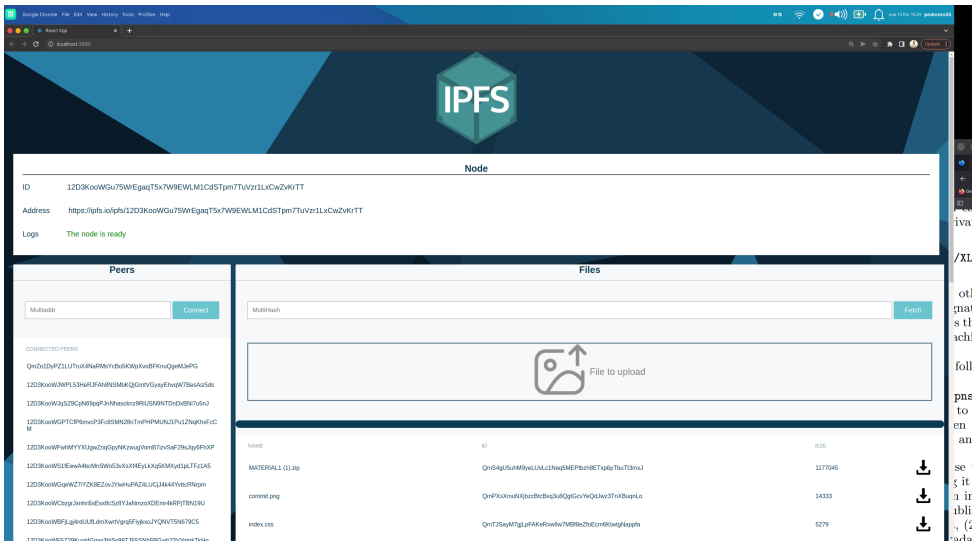

Figure 22: IPFS: Practical Implementation

4º Phase: Create a article regarding this project

At this phase, the creation of an article was in sight. The objective was to try to think of a possible real implementation that could leverage both **IPFS** and **hyperledger fabric**. These

ideas were the genesis block for the second use case. However, the feasibility of such in terms of performance is not confirmed because no benchmarking or works were released based on this idea, which is something that is in mind in the future work in this section. In addition, all of this must be intended to be private, which is possible by isolating the gateways for the peers only to work with a restricted collection of peers by leveraging their buckets. The target of such implementation was the healthcare environment.

The idea was simple but at the same time very intriguing. This is because the idea was to create a centralized point for managing nodes of each instance, while at the same time not enabling this person to actually mess with the data that is stored in each node. This is possible if strict policies are created in order to allow an admin to boot new instances and remove them from the stack. However, there must be rules that don't allow this single point of failure to change configurations or change sensitive data that will be present in each volume that an instance will be targeting. To achieve this, a layer diagram was constructed with the basic components that will be present in each node within an organization or different organizations. The way the cryptographic materials that identify each instance of the **hyperledger fabric** network will be disposed of depends on the setup that the organization wants. This requires a lot of thinking because nodes having the same origin of certificates or not could influence a lot how decisions are taken within the network, but since this is a theoretical conception, such considerations were not taken into account, and every node will work like it is from a different organization.

Figure 23: First use case: Layer Architecture

Besides the layer representation, schemas around how these components would be placed in each node were also done where this network would be built around a **Kubernetes** cluster and the disposition of resources would be the same among nodes, though supposedly it's not recommended that the master node has the same number of components that others have. The number of peers is variable, and there is only one instance of an **IFPS** node per node, but that's something that depends on the admin to decide. This is just a conception demonstration; the way things are done in the second use case is completely different because a lot of other aspects were considered, like the size of the network and the resources that the organization may have. Also, a simple example of a network was created where multiple peers inside of a given node are apart of different channels. In the provided schema, a peer of type admin could access both channel admin and group channel, where a peer destined to be apart of a group channel is only apart of the group channel. It should be kept in mind that when a peer is apart of 2 channels, it will have a ledger for each, but that's something that was not illustrated to

simplify the diagram. Also, having more than one peer per node is something that should be thought through well because each peer instance is very powerful, and in this representation, the idea was to have a single node for the orderers. Additionally, there could be multiple types of networks with different channels and permissions, but like I mentioned before, this was just a very simple example of the network capabilities. A possible interaction of a doctor with the network was also conceded.

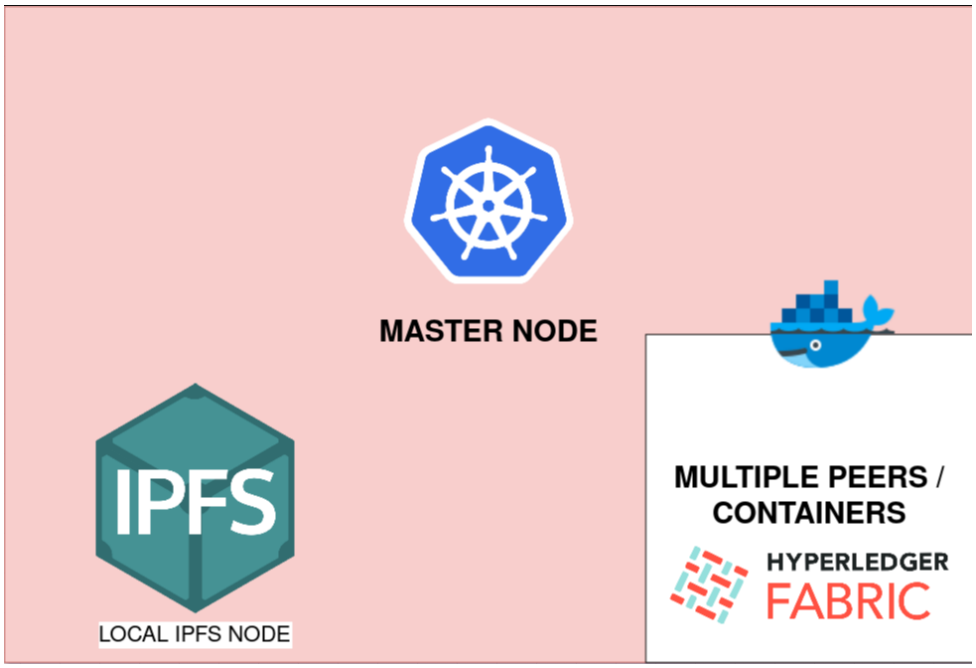

Figure 24: First use case: Master node composition

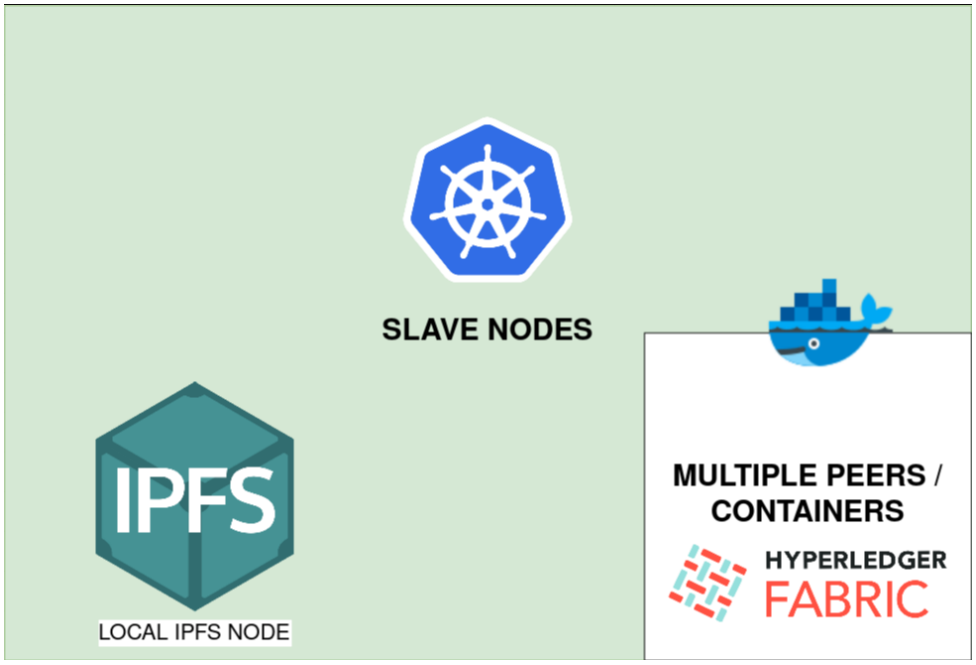

Figure 25: First use case: Slave node composition

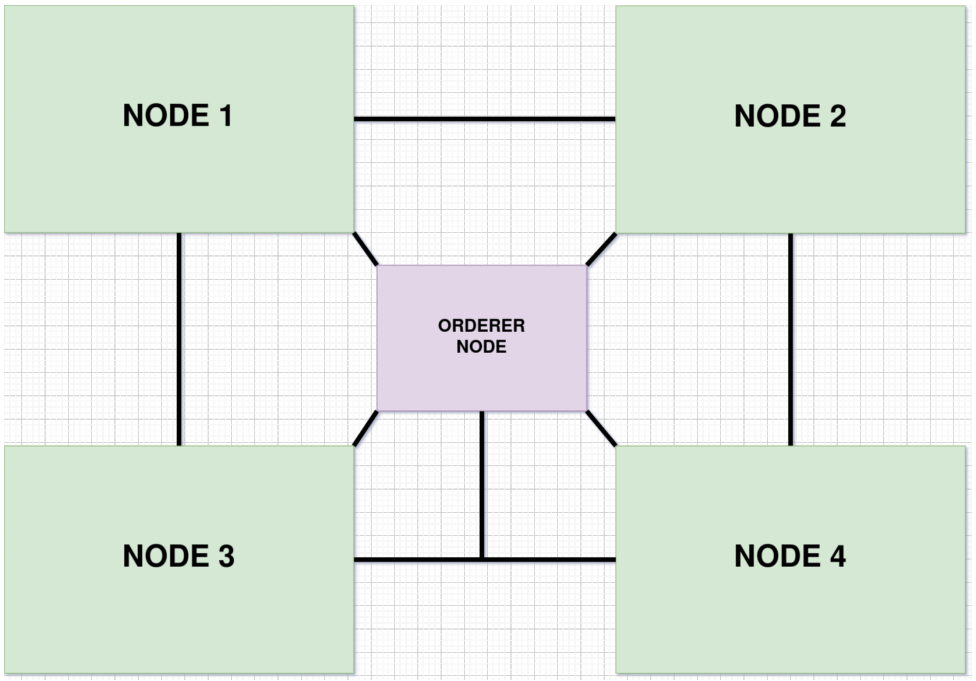

Figure 26: First use case: Global vision

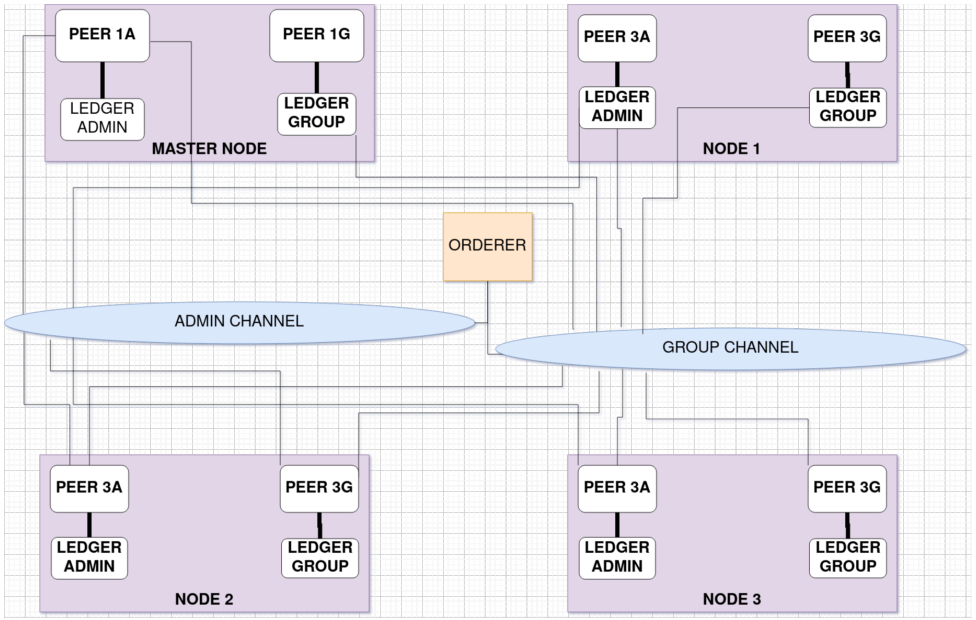

Figure 27: First use case: hyperledger fabric simple network example

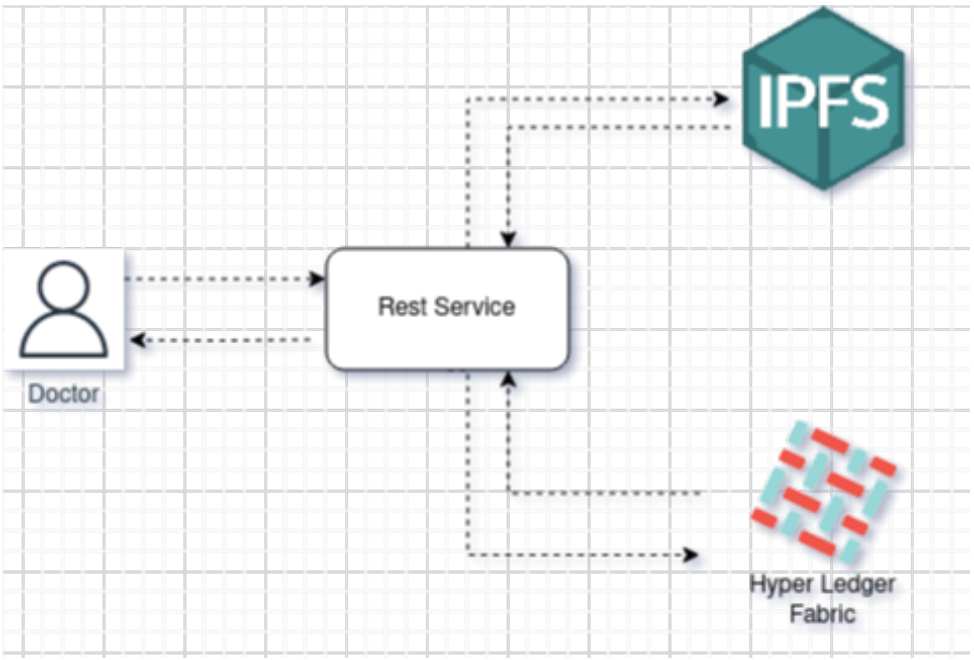

Figure 28: First use case: Workflow

5º Phase: Review the article regarding this project
In this phase a deep review to the article was done, corrections were taken and after everything appeared well the article was successfully accepted for the conference.

6º Phase: Present the Article
Within the final phase, the article was presented in the blockchain congress, giving more insights of what is being implemented currently in the private landscapes, contributing to more interesting ideas regarding future work around this matter.

	April	May	June	July
1º Phase: Ipfs deep dive	X			
2º Phase: Study of Ipfs composition	X	X		
3º Phase: Practical Implementation		X	X	
4º Phase: Create a article regarding this project		X	X	
5º Phase: Review the article regarding this project				X
6º Phase: Present the Article				X

Table 6: Monthly plan 2023: Ipfs use case

4.2 Approach

Relatively to the approach that was taken during this part of the work, like specified before, the **DSR** was not followed due to the fact that practical creations in the direction of solving a problem were not inferred, leaving as the only valid path to investigate, doing some theoretical conceptions around the subject and thinking after how this could actually be put into practice,

which was the main reason why the second use case occurred. In other terms, no framework was adopted, but a lot of insights were taken by own procedures.

4.3 Risk List

The following table refers to the expected risks that this first use case project may have:

Nº	Description	Probability(P)	Impact(I)	Seriety(P*I)	Mitigation Action	Verified
1	Unreachable deadlines	3	5	15	Planning better the next events and gain experience from previous deliveries;	No
2	Requirements of the project change	2	4	8	Make frequent communication with stakeholder;Check well the limitations imposed by the project;	No
3	Lack of Time	4	5	20	Implement only what is actually needed	No
4	Resources	4	5	20	There are no mitigation measures	No
5	It may not work as expected	2	5	10	Creating some practical iteractions; Search for known solutions;	No
6	Mistakes	1	5	5	Don't do tasks in automate mode; Focus more in the tasks;	No
7	Complexity of the project	2	3	6	Document Everything; Simplify everything;Join Communities;	Yes
8	Unknown validity of a IPFS private network	4	5	20	Create and benchmark a IPFS private network;	No

9	Unknown validity of a hlf network	4	5	20	Create and benchmark a hlf private network;	No

Table 7: First use case: Risk List

4.3.1 Discussion

As discussion, there will be the presence of a reflection about the bad sides and positives sides of the given project. In that behalf, **SWOT**(Strengths,Weaknesses, Opportu- nities, Threats) diagram was created:

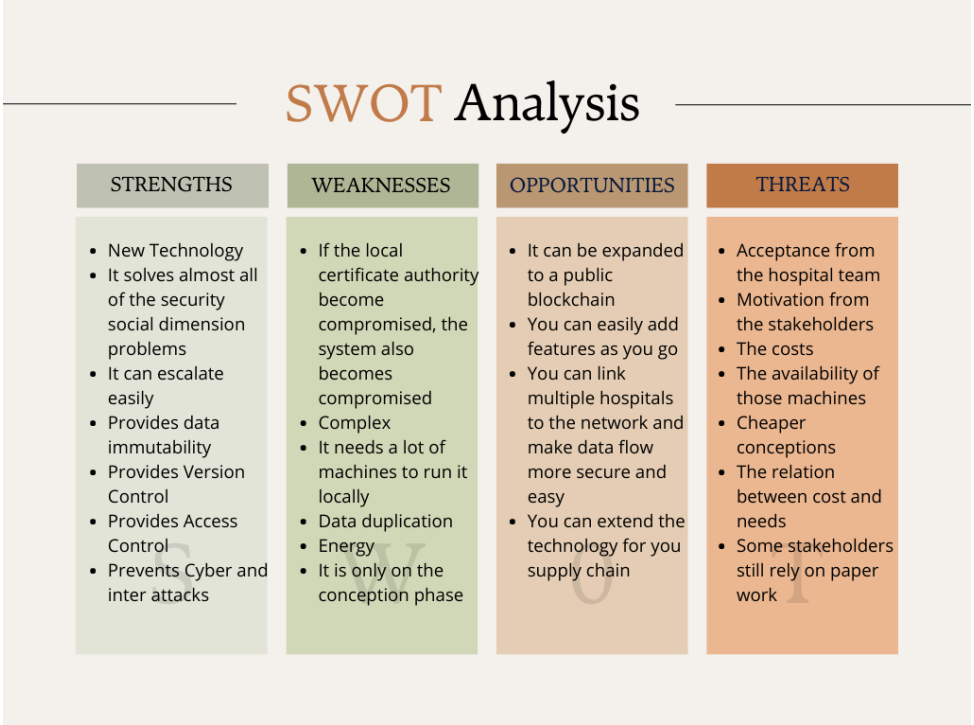

Figure 29: First use case: Swot

4.3.2 Future Work

This project works as an extension of the second use case, even though it was the one that originated it. The work is being conducted all around this project, and the future work will be conducted around the same idea, which is solving component by component of the previously shown architecture. Hyperledger fabric has been solved with the last work, and what is expected next is to do something alike that experience but now with **IPFS**. Also, possible improvements will be expected in delving into this, but that is something that will be better covered in this section. The expected work items for the future are the following:

Creation of a private IPFS network

One of the work items to be expected is the creation of a private **IPFS** network. This is

because the **Hyperledger fabric** has been uncovered already in the second use case, making it necessary, like said before, to uncover the remaining components, more concretely the **IPFS**, private network.

Benchmarking of the IPFS private network
Another work item would be to benchmark the **IPFS** private network to actually understand until each measure is efficient in private environments. In case it is not, maybe it would not be that clever to keep going with this idea, but it is still interesting to give insights of the following steps in case it does not come to be a fail.

Creation of a hyperledger fabric network that was IPFS as it's extension
In case the benchmarking over the private network goes right, there is also the intension of creating a conjunction of both **IPFS** and **Hyperledger fabric**, which was the actual main objective at first glance of the idea.

Benchmarking of the network that implements both IPFS and hyperledger fabric
In the realm of testing the feasibility of mixing such 2 powerful components, there is the benchmarking of the later mentioned network which could lead to a continuation or banishment of the idea in cause.

Creation of a web client
The most important component when it comes to this kind of implementation is the way users will interact with it. A client that is capable of both storing files and simply sending transactions will be needed.

4.3.3 Conclusion

In conclusion, despite not being implemented, this project seems to have a lot of work and possibilities, and it will be very interesting to see until each measure such a random idea could play out. It's feasibility is still something unknown, but constructing things and applying them into practice is one of the fundamental needs of engineering, and since this project works with 2 different approaches for data immutability, the healthcare environment could benefit in terms of security and traceability like it never did, which could potentially revolutionize the methodologies used to work with data within the healthcare landscape, but further research and work is still required to understand how this can impact it, making it something truly nutritional for emergent technology enthusiasts.

4.4 Hyperledger Fabric: Seeking standardization through designing for each type of organization

In the advent of the second use case, it should be known that it is concerned about having the best infrastructure that could be implemented within a healthcare institution. Here, by recurring to a **DSR** methodology, there is the gathering of both theoretical and practical knowledge about such. Requirements are reunited, networks with different typologies, sizes,

concepts and benchmarkings, there is comparison between solutions, and conclusions regarding this work are present with a given provence of success. Putting in another perspective, this is the network infrastructure that would be required in the first use case, which is the main reason why the first is associated as an extension. In this section, it is expected to present **Objectives**,followed **plan**,**approach**,**implementations**,**compare solutions**,**final architecture**,**discussion**,**future work** and **conclusion**, but this will be better covered as it moves forward.

4.4.1 Objectives

When speaking about the objectives in this use case, the main one is to answer the research question, "What kind of infrastructure design is necessary to support a blockchain solution in such a vast and complex environment as healthcare?". This question will be answered by creating means to an administrator becoming capable of managing a blockchain network using a single point of failure. This is necessary because having multiple nodes within our organization makes it not feasible to configure due to the fact that it would require connecting to each node to make the necessary changes, which with these ideas could be something easier to setup. This requires sacrificing a bit of security over usability, but that is something that could be faced by using really good authentication mechanisms and also allowing the administrator to only be able to add or remove elements from the nodes without changing the data and configurations that each element may have. Additionally, the administrator should be capable of adding members with new configurations, but this is something that can be discussed further.

4.4.2 Plan

A plan is always something critical when it comes to keeping track. This enables both personal and also professional goal achievement, involving setting objectives, determining strategies, and organizing resources to reach desired outcomes.

In the general plan, there was a highlighting of all the activities and steps in a broad way. However, like in the first use case, there is the need to go even deeper within that plan and understand what was done more specifically to the second use case. With this in mind, a lot of what has been done so far will be covered in this section, explaining not only what was done during that period but also when it was done. Additionally, it is important to understand that most of the activities resulted from a **DSR** perspective where this solution has been evolving smoothly.

1º Phase: Gather requirements

In the first instance, there was the gathering of requirements. This requirement could be of the typical two types: **functional** and **non-functional**.

Functional requirements are those that describe what a system should do, defining what behavior and specific functions a system should have. Those are usually derived from the user's needs and expectations and are essential for the software to perform as it is intended. As an example, there are things such as **user authentication**, **user authorization**, and **interaction with external systems or databases** [126].

Non-Functional requirements, on the other hand, are the description of how the system should perform rather than what it should do. Those are the requirements that ensure that the system has quality and overall performance while covering aspects such as usability, reliability, performance and security. As an example, there could be mentioned in the performance sphere the system **responding to user requests within 2 seconds**. For scalability the need of a **system handling up to 1 million users** and for security **data being encrypted during transmission and storage** [127].

For this project, since it is within a research realm, there were not many limitations in terms of requirements. In fact, those limitations were what was under research in some cases. However, in terms of **functional** requirements for admins (since the project is focused on the infrastructure), there were some set of requisites such as: **user authentication**,**user authorization**,**interaction with existing systems**,**should provide an on-premise infrastructure**,**the infrastructure should be easy to manage**,**data must be exchanged securely**,**the infrastructure should be easy to debug**,**the infrastructure should be easy to observe**,**the infrastructure should have a block explorer**, **the infrastructure should be scalable**, and **the infrastructure should be secure**. In addition, in terms of **non-functional** requirements, the ones that could be described at this point are: **usability should be user-friendly and accessible** and **there must be unauthorized access and integrity of data insurance**, rather than those that will be considered after the gathering of the benchmarkings to simply check if the system fits the needs of the organization.

2º Phase: Study the methodology framework

Since the project's very complex, having a **methodology framework** is a must, essential for ensuring structured, consistent, and repeatable processes. With this in mind and like mentioned before, this project will be faced using **DSR** practices, which are crucial for seeking the bridge gap between theorical and practical applications. Studying this framework was the main objective in this phase.

3º Phase: Interim Report

For the dissertation, an interim report was an obligation, which was something that had been developed along the way during this phase. Conceding this report is necessary because it was a way of responsible for analyzing, validating and approving the theme under research.

4º Phase: Reading all of hyperledger fabric and it's components theory

This phase was one of the most important within the lifecycle of the project. Articles about hyperledger fabric and documentation were meticulously read, gaining a lot of knowledge that could be later used for the creation of the first network.

Relatively to the documentation, the most important sections were the ones within the **key concepts** such as the **hyperledger fabric model**,**how fabric networks are structured**,**identity**,**MSP's**,**policies**,**peers**,**ledgers**,**ordering service**,**smart contracts and chaincode**,**fabric chaincode lifecycle**,**private data** and **security model**. In addition, there was a following of operational practical guides around how to setup each of the components of the

hyperledger fabric, like **Certificate Central Authorities**,**peer**,**orderer**,**chaincode**,**channel**, and **ledger**.

Within the **hyperledger fabric model**, there was a dive into the design of the framework, presenting the main concepts that are present that form the blockchain network, like the concepts of **asset**,**chaincode**,**ledger**,**privacy**,**security and membership services**, and the **consensus**.

How fabric networks are structure was another important topic because it explains how the blockchain is configured, taking as an example the **sample network** that they provide for practical insights, while showing in theoretical terms how to create the network, how certificate authorities operate, how nodes are joined into a channel, how the chaincode lifecycle operates, how to use the chaincode, how to join multiple components in multiple channels, how to create new channels, how to join components to the new channel, how to add an organization to an existing channel, and how to add existing components to newly joined channels.

Identity, another section of the key concepts, was a section to give important insights of how authentication is managed within the network. This was done through explaining the usage of an identity, X509 certificates, **PKI** (public key infrastructures), digital certificates, authentication, public keys, private keys, certificate authorities, root certificate central authorities, intermediate certificate central authorities, and **CRL** (certificate revocation lists).

After the identity section, the key features also provide more information about what exactly a **MSP** is, which is something that uses as its base the previous section: **MSP**'s leverage the concepts of identity to develop their activity, which is to prove identity to the rest of the network.

In the **policies** section, there is reference to the definition of what is a policy, why they are needed, how policies are implemented, how to write them in fabric, the chaincode lifecycle, and even how to override the policy definitions.

The **Peers** section exists for having a theoretical overview of the role of the peer within the **hyperledger fabric** network. Questions like the peer playing in chaincode terminology, ledgers, chaincode,transaction lifecycle, organizations, identity, and orderers are discussed.

When it comes to the **Ledger**, there can be introduced the concept of what is a ledger, what kinds of state there is in it, it's play in blockchain's, how does it come from transactions and blocks, it's database options, ledger namespaces, and ledger and channels.

On the **Ordering** section, there is everything that is needed to understand with respect to an orderer component: its configurations, its identity, the transaction flow, and consensus algorithms.

Another concept that it has is **Smart Contracts and Chaincode**, where there are theoretical assumptions about smart contracts, it's play in ledgers, what is endorsement, how transactions are validated, it's play in channels, the default channels that are always present in peers, and system chaincodes in more depth.

Fabric Chaincode Lifecycle is a section to define more in depth what a chaincode is and how to deploy, install, and upgrade.

To introduce the concept of **private data collections**, there is also the section **private data**. While introducing it, it also gives examples about how to work with it.

Finally, within the key concepts, there is also the section referring to the **security model**, mentioning how **hyperledger fabric** offers security within it's network and explaining extra capabilities. With relation to security identities, MSP's,Policies,Peers,orderers,TLS, and HSM (hardware security models) are mentioned components.

Moving to the practical guidelines, they were very helpful to construct a operational guide to boot a network from scratch which is important for later reference, since some configurations of **hyperledger** are more into the taste of the operator, thereby creating some fuzzy subjectiveness because concrete implementations were not presented and that was something that presented as a difficulty during the conception. However, this guidelines despite being very complete and useful remain with needs of restructuring for better understanding. Everything is created in a github repository using markdown files.

The first guideline was around the **CA's**, describing how to setup a **hyperledger fabric** own certificate central authority like creating the CA,explaining it's components,configuring it,explaining how to revoke certificates,how to register new certificates, how to enroll certificates and explaining how to create a chain of certificate central authorities.

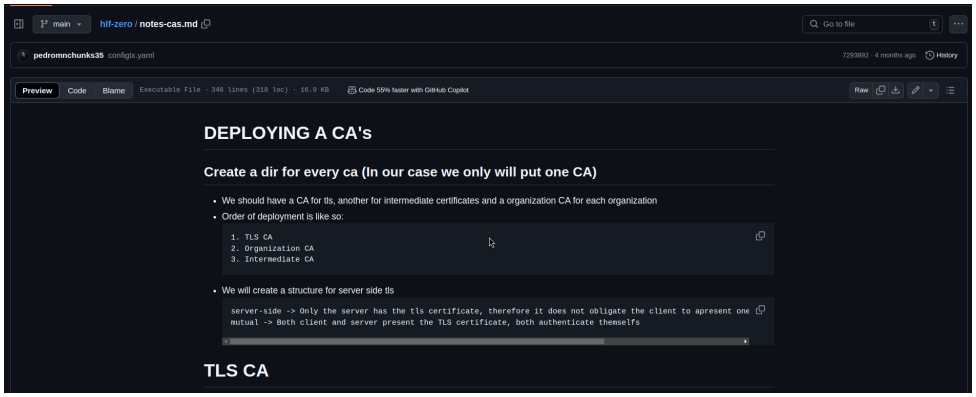

Figure 30: Second use case: ca's operational guide

The second guideline was more around the **orderers**, focused on introducing how to deploy a orderer. This is done by creating it's identities, creating a structure for the usage of each identity within the configuration file of the orderer and for later dependencies such as channel creation.

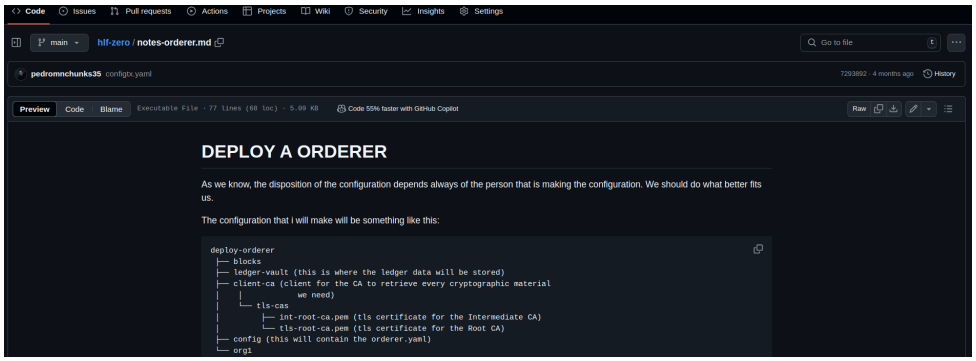

Figure 31: Second use case: orderer operational guide

The third guideline was directed for the **ledger**, which despite having multiple options of

databases to choose, the choosen one was actually the couchdb. This is because this ledger is currently the standard one. Also, it is important to mention that there was a need to start by the ledger because the peer requires the ledger as dependency to actually remain deployed, otherwise it will crash due to not having it.

Figure 32: Second use case: couchdb operational guide

Moving to the forth, this time the **peer** took precedence, which is normal since the last tutorial was it's dependency. Alike in the orderer, the focus was to create it's identities,structure and configuration. Both orderer and peer were deployed in debug mode for having more info about why something was not working at that time. This was crucial because, like mentioned before most of the operational guides had some subjectiveness in the structure: this disposition of the configurations was a bit at the taste of the operator in question under creation of deployment.

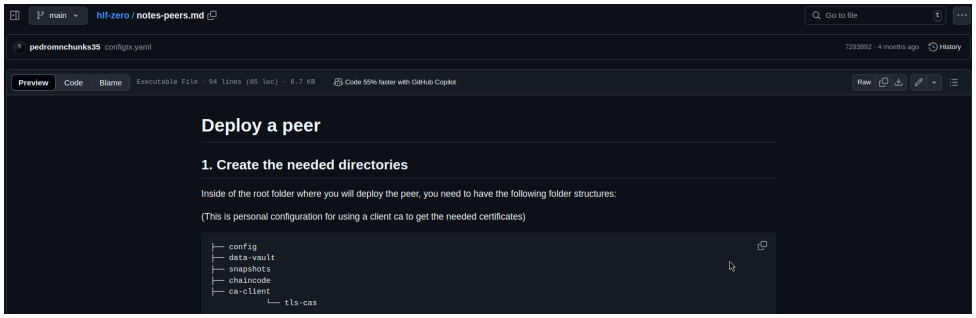

Figure 33: Second use case: peer operational guide

In the fifth, the focus was more into creating a configuration file for the creation of a channel. This was important for actually create means for orderers and peers know how they should communicate with each other. In the operational guide, the focus was into reviewing policies and understand how to actually setup a configuration file which is something complex that requires lots of planning. This step took a while since mistakes in the previous settled cryptographic identities could lead to errors.

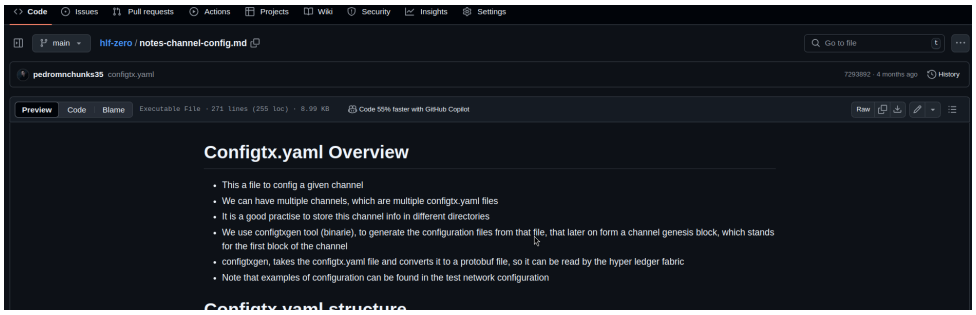

Figure 34: Second use case: channel operational guide

In the final operational guide, the **chaincode** was the most targeted matter. It explains how to create a channel and how to deploy the chaincode in it, showing various phases of the chaincode life-cycle.

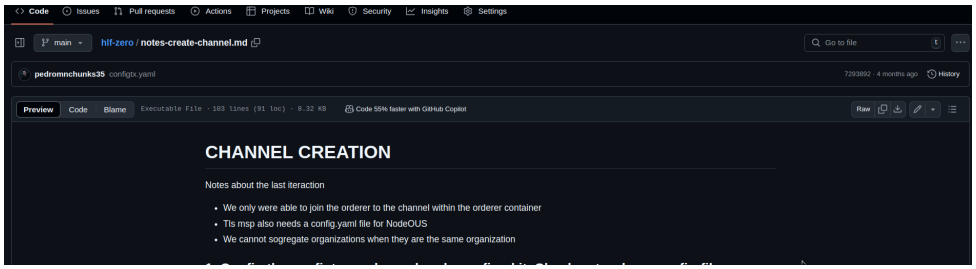

Figure 35: Second use case: chaincode operational guide

Leveraging all of this, knowledge for the creation of a prototype network was gathered, serving as the most important base for constructing everything that would be concluded during this project.

5º Phase: Creating the first network

In the phase 5, the knowledge that came from the previous phase came handy, enabling the creation of the first network prototype. Additionally, all of this was created within a personal machine by leveraging both the machine and 2 virtual machines which formed together the standardized network composed of 3 peers,1 orderer,1 CA,1 tls CA and 1 intermedium CA in the first machine, 1 orderer and 2 peers in the second machine and 1 orderer in the third machine. To simplify everything each machine was a different organization.

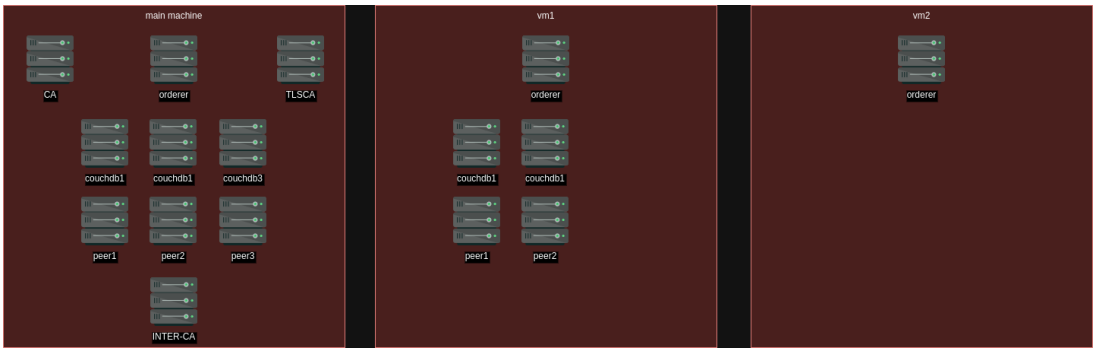

Figure 36: Second use case: first network

Additionally, it is important to mention that a very basic chaincode was created at this stage that served for later instances of our project, where benchmarking was necessary.

6º Phase: First installation in a hospital

After the first creation of a network, there was the intention to try to deploy a network in the hospital machines that were assigned to this project. Unfortunelly at the beginning there were only 2 machines available, which caused a modification of the previous network, giving origin to a default first network where the same number of components were spread in 2 machines, machine69 and machine70. The machine69 had 1 orderer,2 peers,1 CA, 1 CA for tls and 1 intermedium CA and the machine70 had 2 orderers and 2 peers. Aditionally, it should be known that all components from machine69 were cryptographically from a organization and in the machine70 cryptographically 1 orderer and 2 peers were from a second organization,while having 1 orderer from a third organization, much like the first network but without having a third machine.

Figure 37: Second use case: first hospital network

7º Phase: Creating a more robust network

Thinking in the next iteration, here there was an adaptation of an existing blockexplorer from

Hyperledger fabric for the prototype demonstration. Also, there was a refinement of the previous structure of the configuration files from the guidelines, resulting in a template. This template was done according to a set of scripts that were developed. One script was to generate all of the configurations for a given component according to its type (ledger, peer, and orderer), enabling deterministic creation of new components if needed; another script was in charge of putting these configurations in the desired machine; and the third script was to deploy the desired components while still maintaining previous configurations of components in the machine, enabling to reset the network to any structure of network that was specified in a file with a specific syntax, which was a must for further usage in benchmarking, where different networks were tested, always created from scratch.

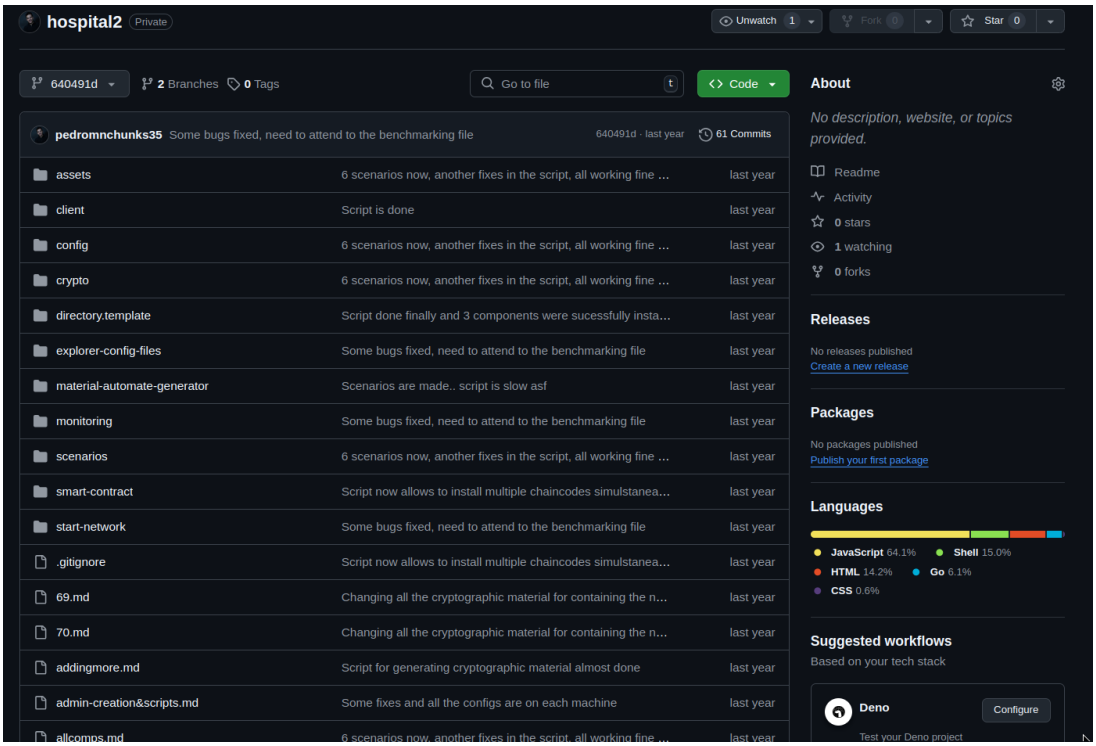

Figure 38: Second use case: hospital v2 repository

In the given picture, it can be spotted that there is a directory for templates (directory.template), a directory for scripts that generate new configurations and transfers them to a desired machine (material-automate-generator) and there is a directory for resetting a network making different networks out of the box (start-network).

8º Phase: Prototype demonstration

At this stage, there was the need to present a prototype of the network. Since the spectators were not technical, recursion to a block explorer was necessary, such that until an own implementation surged, a legacy version was used and the presentation went smoothly.

9º Phase: Interim report delivery

During this phase the report that started before was delivered, such that some fixes were needed before being actually accepted by the responsibles.

10º Phase: Dissertation development

Within the 10th phase realm, the dissertation started to exist and got incremented in terms of content as the time was progressing, making sure that the schedules were being fulfilled and the project was running accordingly to the expectations.

11º Phase: Studying Benchmarking

This was the phase where **caliper**,**cadvisor**,**prometheus** and **grafana** were under inspection. **Cadvisor** was used to expose time series data relative to performance metrics of the machine, **caliper** is the actual benchmark performer, **prometheus** is where the data from cadvisor will be gathered to be used in **grafana** which creates a visualization.

12º Phase: Making a benchmark locally

In this phase,a whole dashboard for visualizing and extracting data was created with grafana and a benchmarking was done in a local network. This local network was the one that was created in the first instance and this benchmark was only to make sure that everything was working as expected.

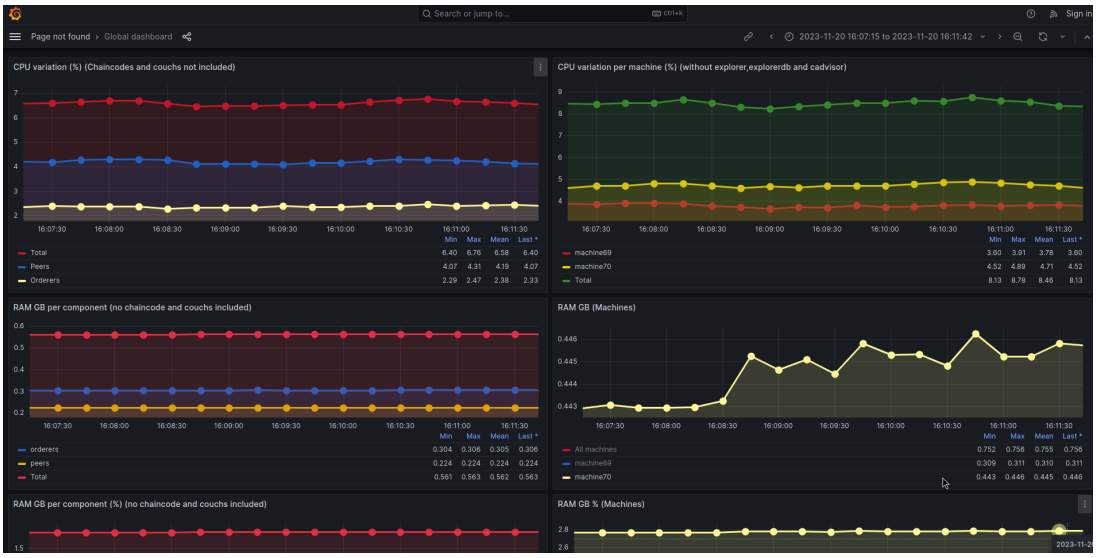

Figure 39: Second use case: first benchmark

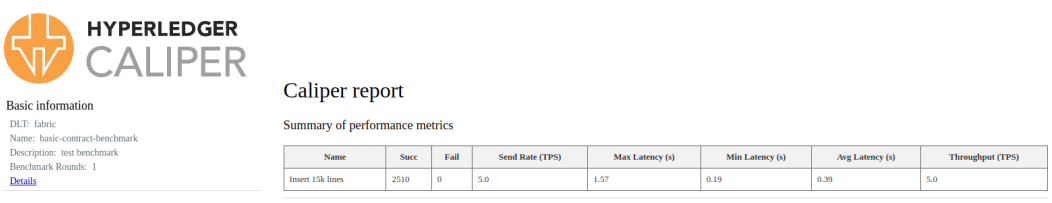

Figure 40: Second use case: first benchmark (part 2)

13º Phase: Benchmarking the first network

After having everything tested in a local environment, a real benchmark over the first hospital network was conducted. This time, the data was actually extracted as CSV. This is because it becomes easier to create graphics that are more eager to be interpreted by paper, while the one's

created are better for dynamic analysis which is fine but not for scientific work. The benchmarks were yield 10 times to make sure that it was not a one time occasion.

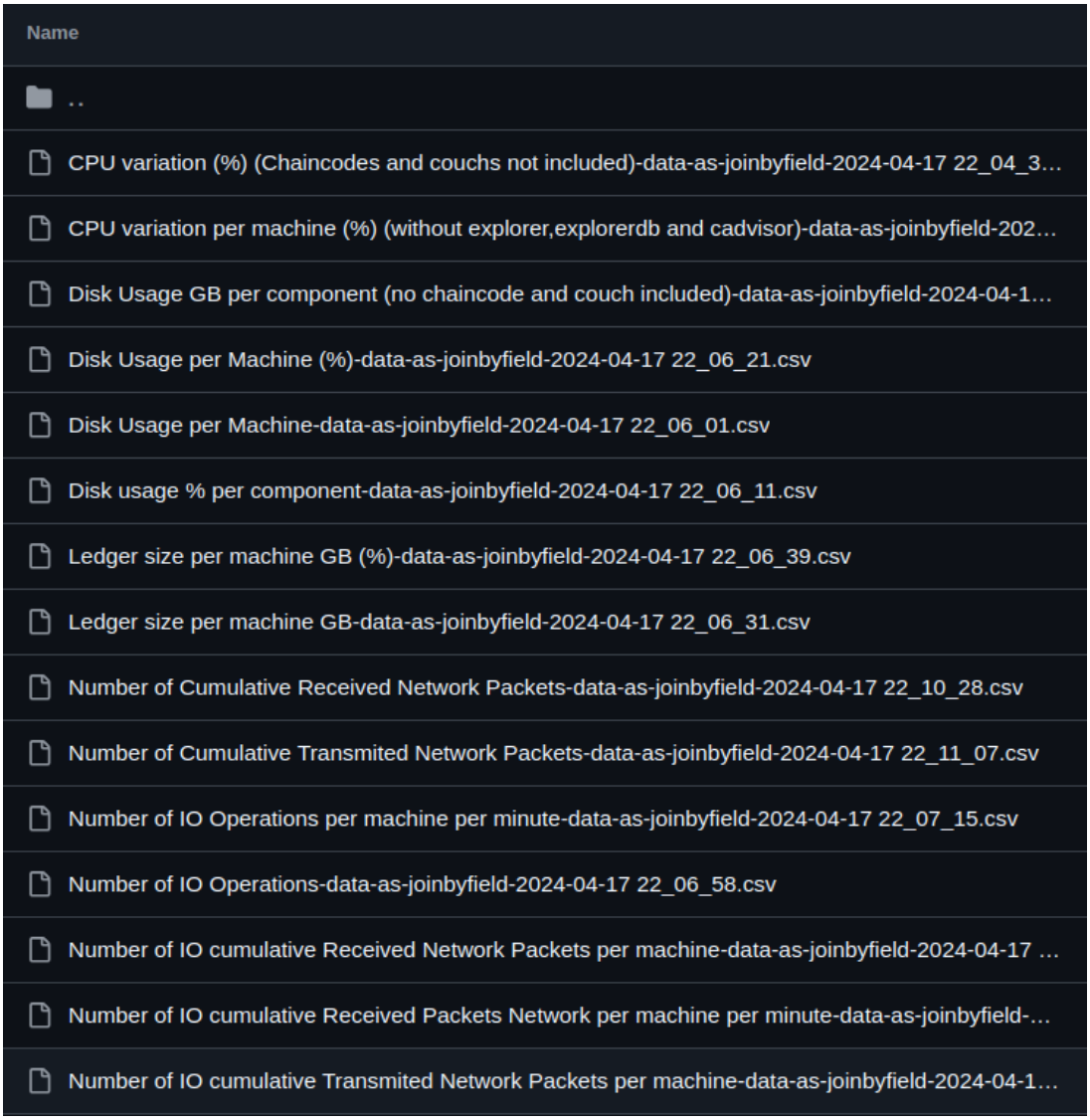

Figure 41: Second use case: first benchmark hospital

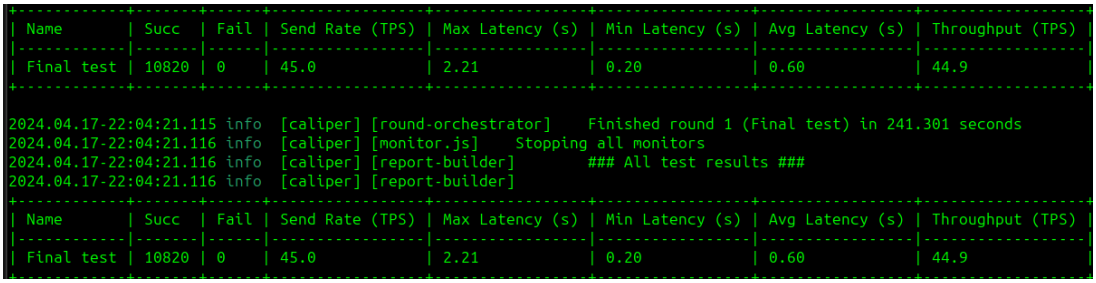

Figure 42: Second use case: first benchmark hospital (part 2)

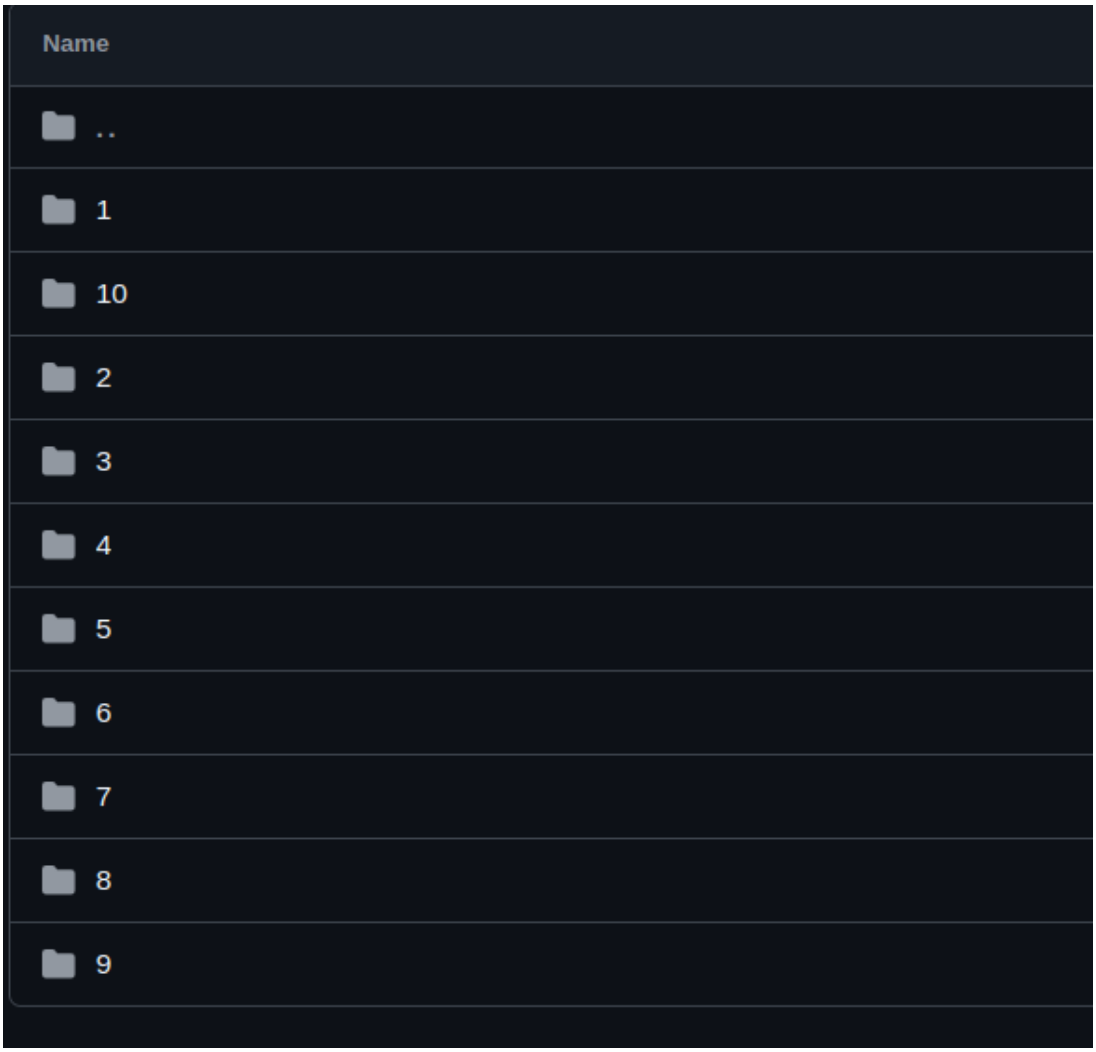

Figure 43: Second use case: first benchmark hospital (part 3)

14º Phase: Studying Kubernetes

After benchmarking a normal bare metal network, there was a deep reflection about the ease to use such infrastructure. Between configuring everything and putting everything in every single machine and managing every single instance within a healthcare organization, it became obvious that it would create burden for a administrator that wanted to spoil it's machines with multiple components.

With this in mind, within this phase there was a deep dive into kubernetes. Very powerful as it is, it imposed a very huge challenge in terms of knowledge as the **hyperledger fabric**, where the documentation was very complete and extensive, not speaking about the tutorials that were done in order to understand the basis of such tool. In addition, this became even more clear with the fact that the infrastructure had the need to be implemented on premise, where everything must be installed from scratch, different from the cloud where everything is already installed and ready to use.

Besides **kubernetes**, other technologies were observed for the on-premise imposition. This technologies were **metallb**,**calico** and **kubeadm**. **Metallb** was required to offer load balancing capabilities, **calico** was a network plugin for pods to communicate with each other and

kubeadm was to deploy and join nodes to a cluster.

Relatively to operational guides and abstracts of what was covered during this interval, they were placed in github repositories where information was both by photo and markdown files making this information accessible whenever needed.

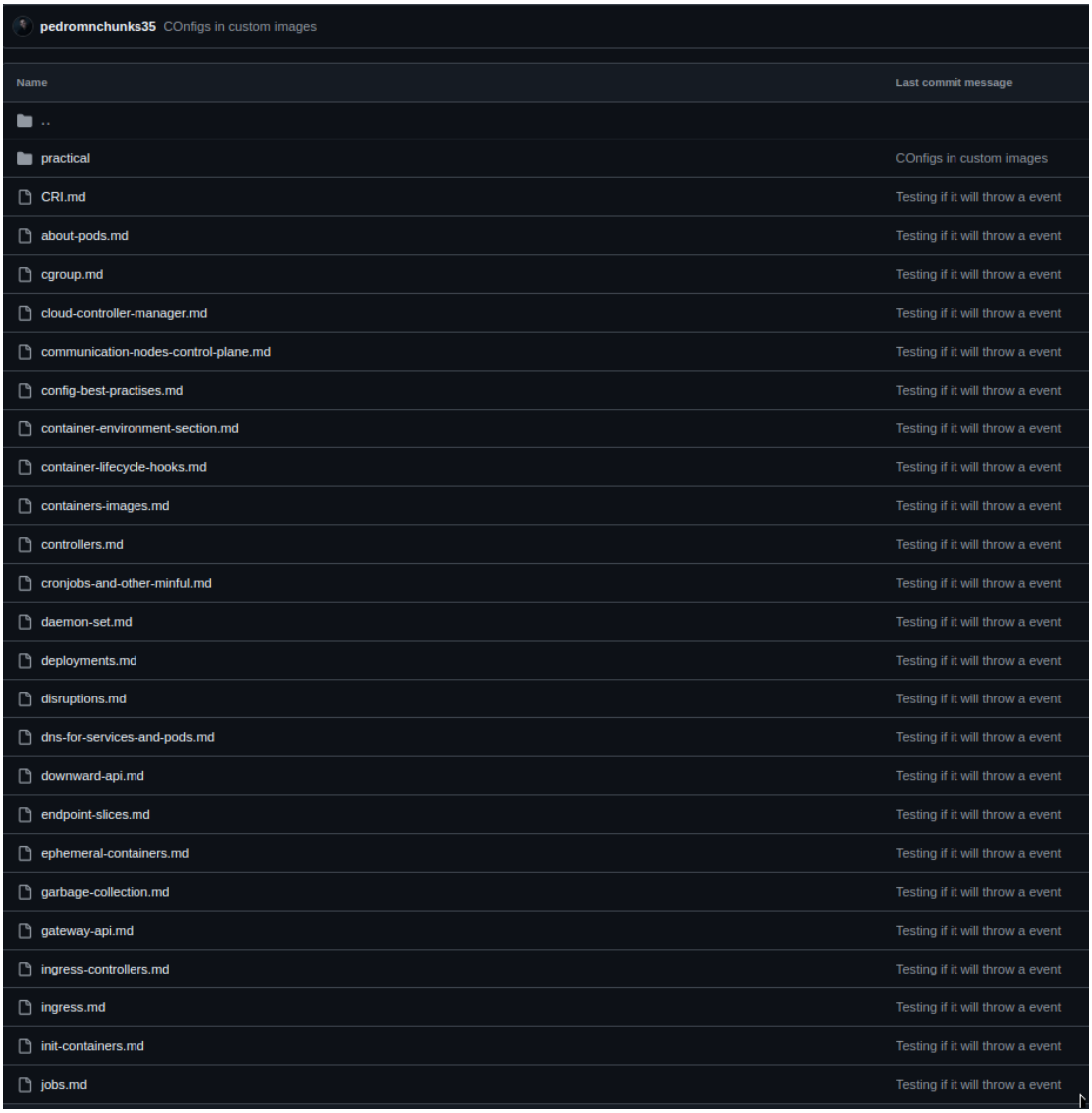

Figure 44: Second use case: kubernetes study

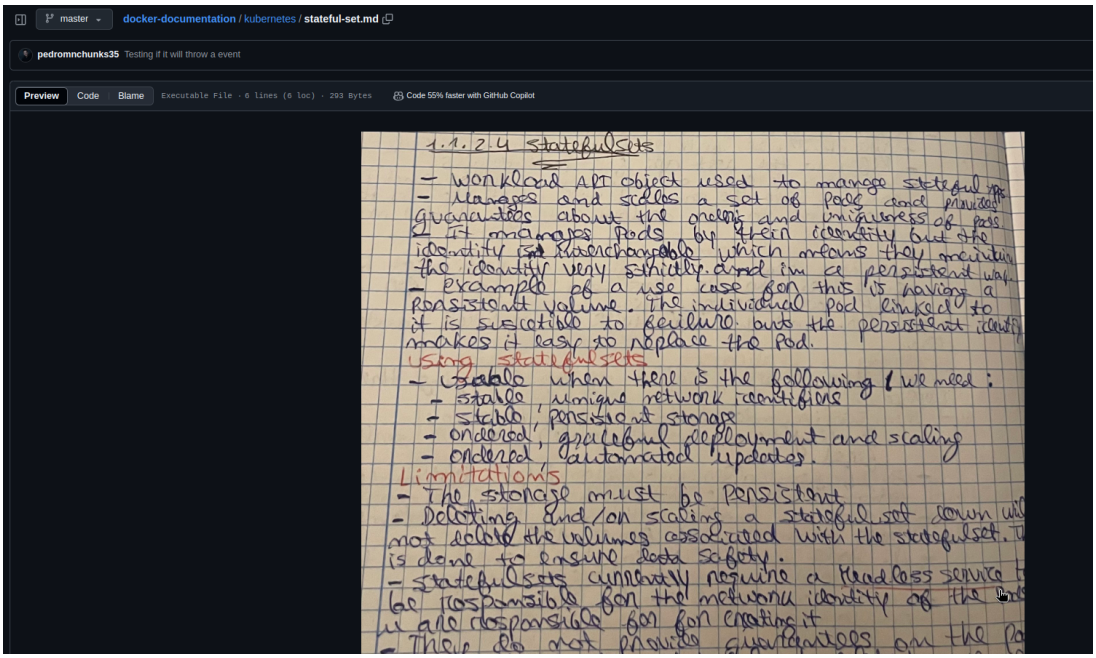

Figure 45: Second use case: kubernetes study (handwriten)

15º Phase: Implementing a blockchain network with kubernetes

In this phase, a **hyperledger fabric** network was built under **kubernetes**. To achieve this, this network was deployed firstly locally with the main machine as the master node and 1 VM as it's slave node. The network that was deployed there was relatively small compared to the first that was deployed in the local environment and no load balancer was implemented in this inception because the objective was to simply put a network in such environment. Additionally, it should be known that in **hyperledger fabric**, deploying a chaincode in bare metal is completely different from implementing it in a **kubernetes** environment: In a bare metal environment the installation of the peer can be done directly in the peer, while in the case of kubernetes chaincode must be installed as a service that can be shared by multiple peers. The reason for this has to do with the control over the container runtime. There are operational guidelines regarding this first implementation.

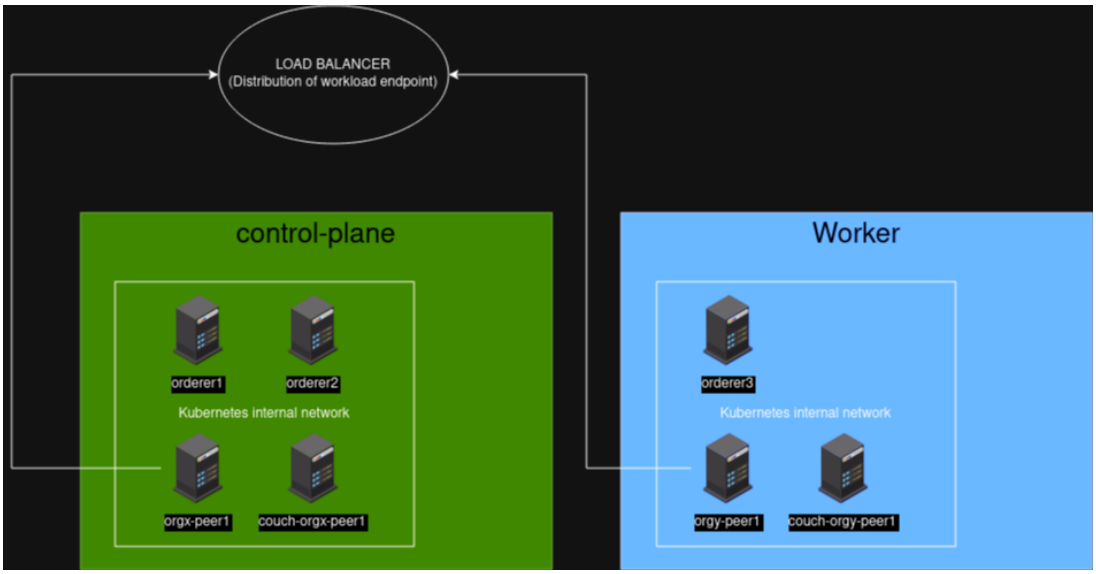

Figure 46: Second use case: local kubernetes hlf network

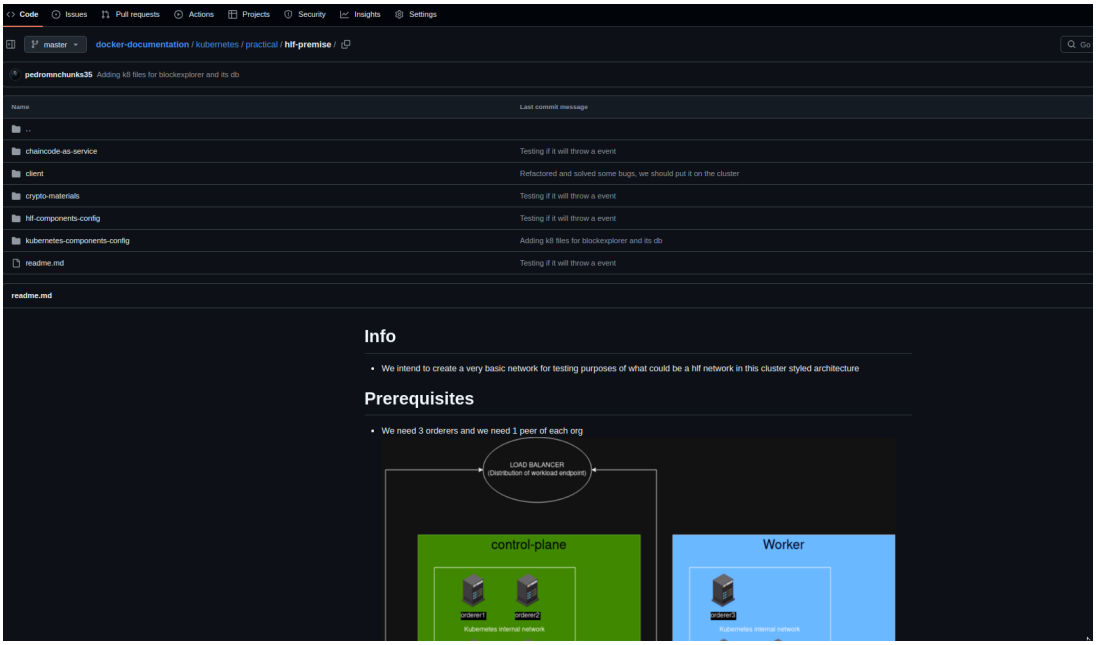

Figure 47: Second use case: local kubernetes hlf network (part 2)

It should be noted that despite a reference to a load balancer in the network structure image, there is no real load balancer but rather a nodePort which also does load balancing but in a limited way since that is not it's main purpose.

16º Phase: Creating a more robust network for a kubernetes environment

After accomplishing a working **kubernetes** network, 2 things were added to provide more support to the network and a third thing was added just for testing. Firstly, a real load balancer that supports on-bare metal kubernetes was deployed which was the previous mentioned **metallb**. In second, Automation mechanisms were also added to contribute to reseting and managing the network more effectively but with it's aim in adding in the future a UI that could give an

administrator the power to manage it's infrastructure. Lastly, a service mesh was implemented for testing purposes for knowing how could such practise give more grained control over the network by controlling in which cases traffic is allowed, while providing monitoring features that are very useful for an admin.

Focusing more in the second point, this was a innovative idea that came from a **Sidecar** container Architectural pattern. This was the case, since every component of the network has a side-container to extend it's communication features, enabling to upload files and also to run commands destined to the binaries exactly like the admin would be inside of the container. During this phase, a web UI prototype was also used to test this features and this way of communicating with every service was also widely used to automate network booting alongside with scripts which would also automate multiple network schemes to the network, just like it was done in the on bare-metal approach but in a even more easier to use methodology, since with this there is the opportunity to use a general purpose programming language such as golang or javascript.

During this conception, another structure of network was implemented, presenting 3 machines instead of only 2, where there was a control plane with a single point of failure graphql service and 1 peer with a side container and a orderer with a side container in each machine forming all together a cluster locally where an administrator could run commands easily.

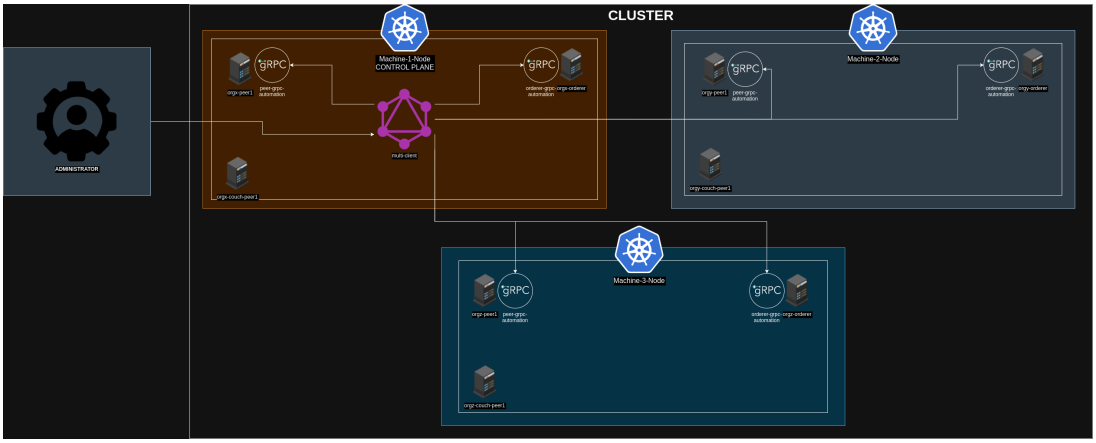

Figure 48: Second use case: kubernetes automation creation

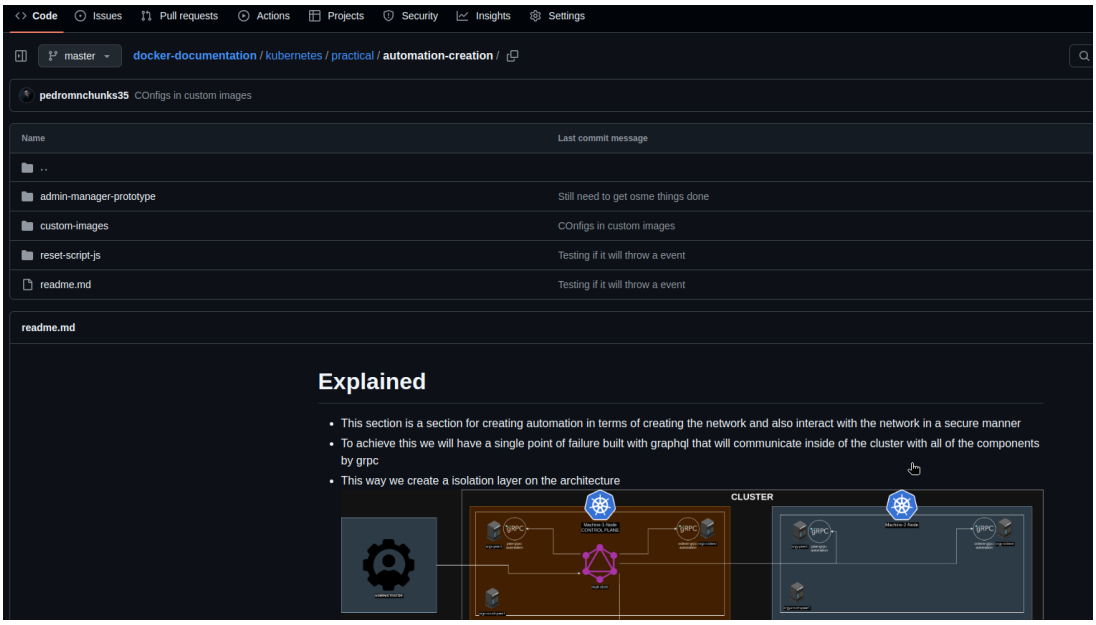

Figure 49: Second use case: kubernetes automation creation repository

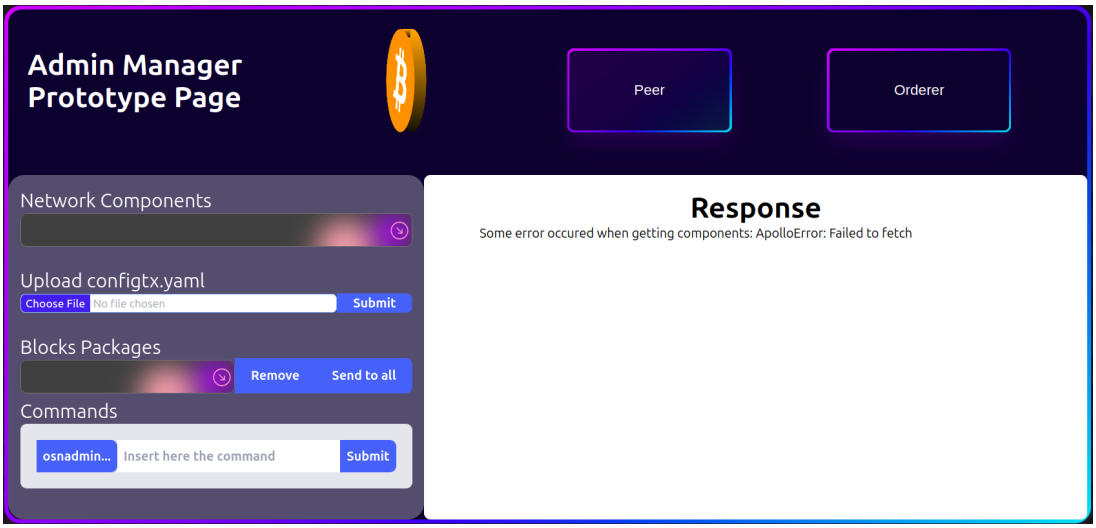

Figure 50: Second use case: kubernetes automation creation orderer prototype

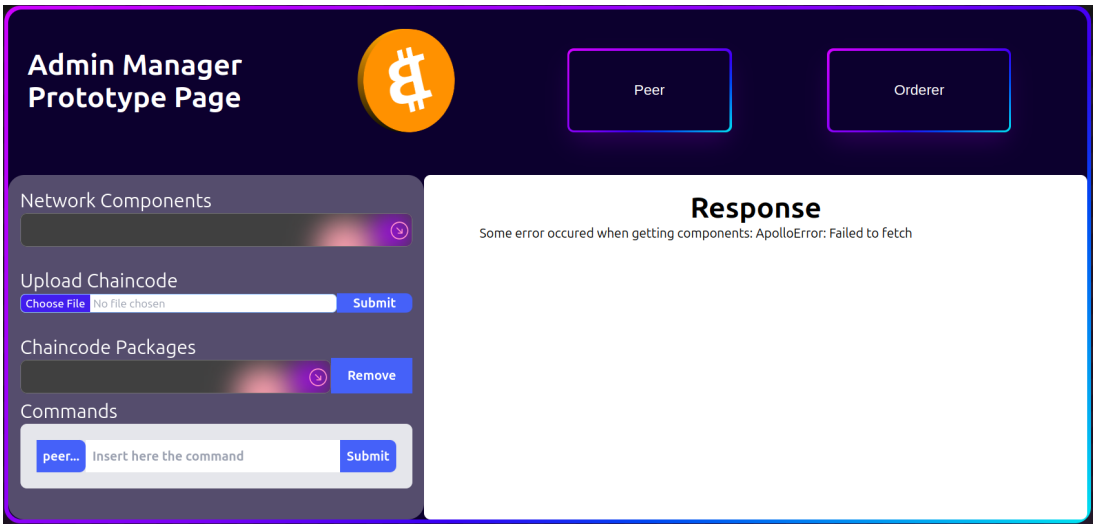

Figure 51: Second use case: kubernetes automation creation peer prototype

17º Phase: Second installation in a hospital

In this phase, fortunately there was a release of an extra machine for this work which was very pertinent for what was about to come and it was even more suitable because the same network structure was already implemented in the local network. With this in mind, the same network with the same sidecar container and metallb functionalities was implemented successfully in the hospital environment, while effectively handle the segregation with **non-kubernetes** implementation by deploy a chaincode-as-service instance.

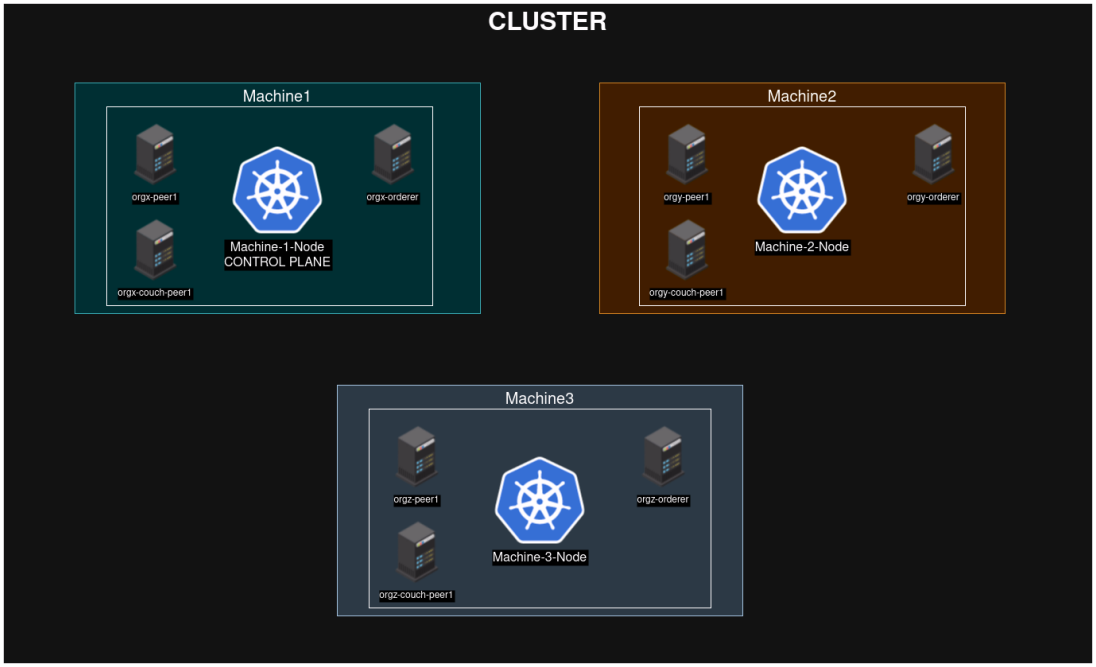

Figure 52: Second use case: kubernetes first hlf hospital network

18º Phase: Creation of multiple networks

As mentioned before, the bigger the network in terms of number of different identities, the bigger the network burden. With this in mind, within this phase knowing the limitations of the number

of components that could be handled by 2 machines was putted into cause. At this time, the third machine was on maintenance, which causes to only test this overhead in 2 machines instead of 3. However, to see until each measure this could impose a threat in a small set of resources it was created 6 scenarios of networks which are the following:

The default scenario, was the first scenario that was implemented in the first hospital network when there was only 2 machines: first organization had 1 orderer and 2 peers and it was located in the first machine, the second organization had 1 orderer and 3 peers and was located in the second machine and the third organization had 1 orderer and it was located in the second machine as well.

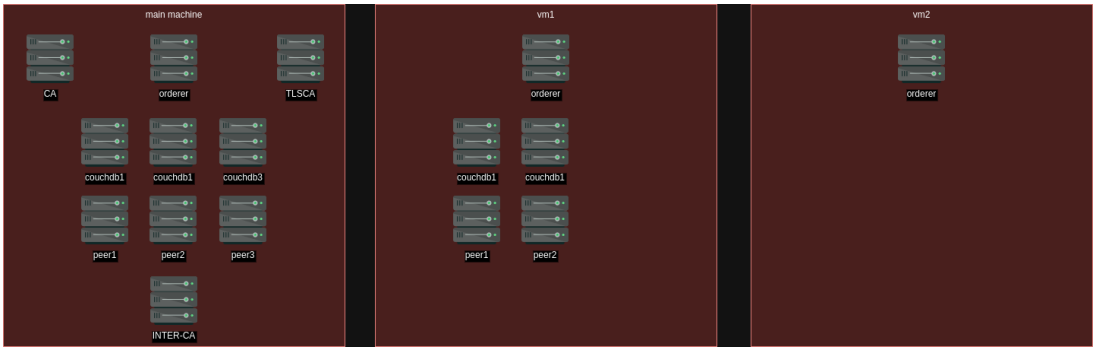

Figure 53: Second use case: multiple networks default scenario

In the first scenario, the first machine had one organization with 1 orderer and 1 peer and the second machine had 2 organization where the first had 1 orderer and 1 peer and the second had 1 orderer.

Figure 54: Second use case: multiple networks scenario 1

Within the second scenario, in the first machine there were 3 organizations: The first one had 1 orderer, the second one had 1 orderer and the third had 1 orderer and 2 peers. Also, in the second machine there were 2 organizations: The first one had 1 orderer and 3 peers and the second one had 1 orderer.

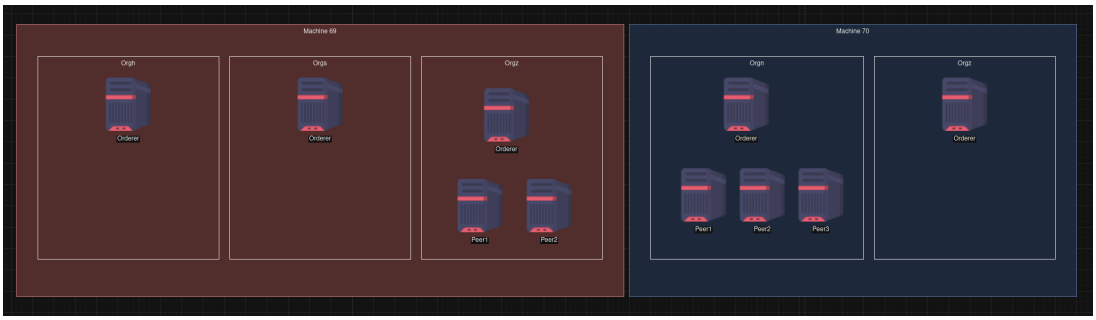

Figure 55: Second use case: multiple networks scenario 2

In the third scenario, in the first machine there were also 3 organizations: the first had 1 orderer, the second had 1 orderer and 3 peers while the third had 1 orderer and 2 peers. In the second machine, there were 2 organizations: The first had 1 orderer and 3 peers and the second had 1 orderer and 3 peers.

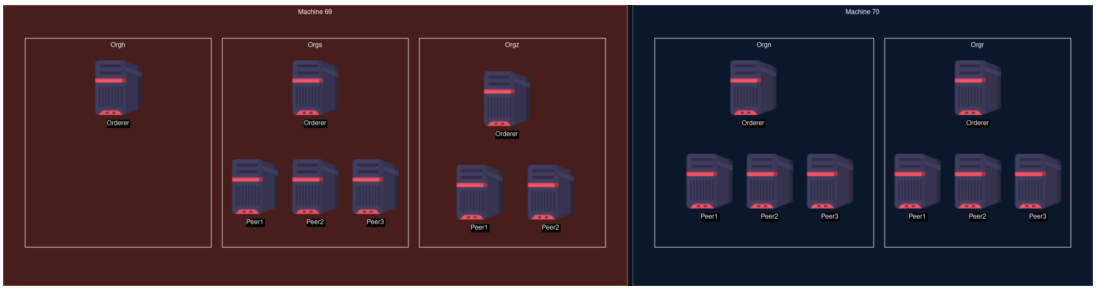

Figure 56: Second use case: multiple networks scenario 3

In the forth scenario, in the first machine there were 4 organizations: the first had 1 orderer and 2 peers, the second 1 orderer and 3 peers, the third 1 orderer and 2 peers and the forth had 1 orderer. In the second machine, there were 3 organizations: the first had 1 orderer and 3 peers, the second had 1 orderer and 3 peers and the third had 1 orderer and 2 peers.

Figure 57: Second use case: multiple networks scenario 4

In the fifth scenario, in the first machine there were 4 organizations: the first one had 1

orderer and 3 peers, the second had 1 orderer and 3 peers, the third had 1 orderer and 3 peers and the forth had 1 orderer and 3 peers. In the second machine there were 3 organizations: the first had 1 orderer and 3 peers, the second had 1 orderer and 3 peers and the third had 1 orderer and 3 peers.

Figure 58: Second use case: multiple networks scenario 5

Despite designing so many networks, and having in consideration that there was the removal of the kubernetes for the sake of performance unfortunately after the scenario 3 it was not possible to conduct a full benchmarking. What was noticed is that it was exceeding in remarkable ways already in terms of CPU in the scenario 3 and it needed more resources to reach the scenario 4, which makes sense since the resources were not the most powerful ones but it was interesting still to see how many components it could handle.

19º Phase: Benchmarking the network

Within the realm of the 19º phase, benchmarkings were conducted. The benchmarking that was done before was refactored because of some errors in the dashboard and also because now there was the existence of a third machine which would make the tests even more interesting. By the effect of such, benchmarkins were conducted within the scope of the last network that got implemented in the hospital. In the case of the **kubernetes** implementation, there were 2 types of benchmarkings conducted: one with load balancing and another without load balancing. On another hand, in respect to the **non-kubernetes** implementation, only the normal test was considered. Because both **kubernetes** and **non-kubernetes** had the same structure of network, the data could be compared effectively. Aditionally, **jenkins** was used to automate this since it was used previously in a continuous integration perspective to come up with the side containers present in each of the components like mentioned brefore. With this, benchmarkings were totally automated both for bare metal but also for **kubernetes** which speed up even more this process and gathered all of the tests in no time.

By the effect of this, 3 architectures were under test: one architecture which only relies on **docker**, a second architecture that leverages **kubernetes** without load-balancing and a third which leverages **kubernetes** with a load-balancing.

Docker

Speaking about the considered network of **Docker**, this is composed by 3 machines with 3 different organizations. Each organization had the same amount of components like 1 peer and 1 orderer. However, since this is a network fully composed by docker containers the way the chaincode is deployed is accordinally to the default way, where each peer has it's chaincode attached to it like a side container. There are no life savers when it comes to configuring such network, which means that adding or removing components requires more work altought it has less burden when compared to kubernetes.

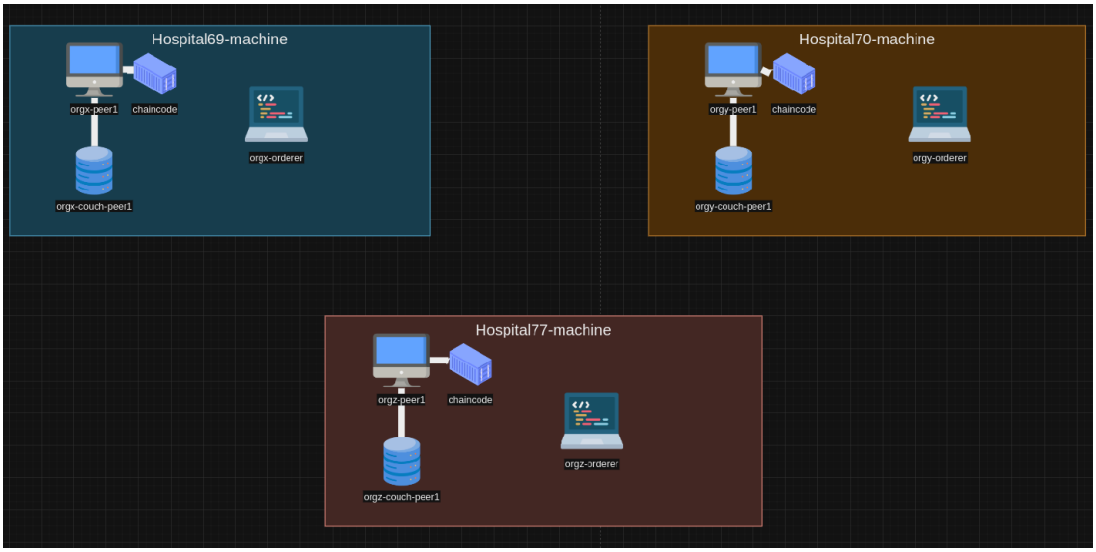

Figure 59: Second use case: Docker only Architecture

Kubernetes

Reaching the **Kubernetes** considered network, there is a network with 3 different machines and 3 different peers. Each organization, like in the previous architecture, had 1 peer and 1 orderer, where the most significant changes are related to the fact that each main component had one side container attached that served as a proxy for managing configurations within the components. This approach compared to the docker one makes the management of the network easier, since there was the possibility to configure each component from the same unique point of failure (multi-client). However, this is supposed to be less performant because there is the burden of the **kubernetes**, causing a higher need of resources.

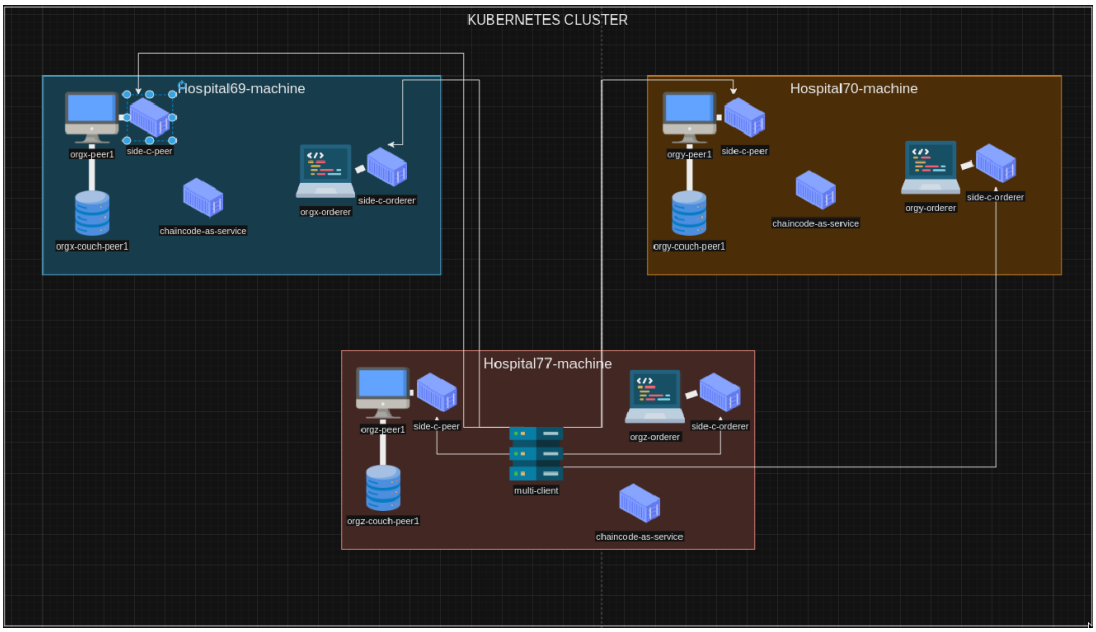

Figure 60: Second use case: Kubernetes Archtiecture

Kubernetes with load balancing

When it comes to the last Architecture **Kubernetes with load balancing**, there is the same architecture as the previous **Kubernetes** one, where the difference resides in the fact that there is a addition of a load-balancer, where the load gets evenly spread among the participants of the network.

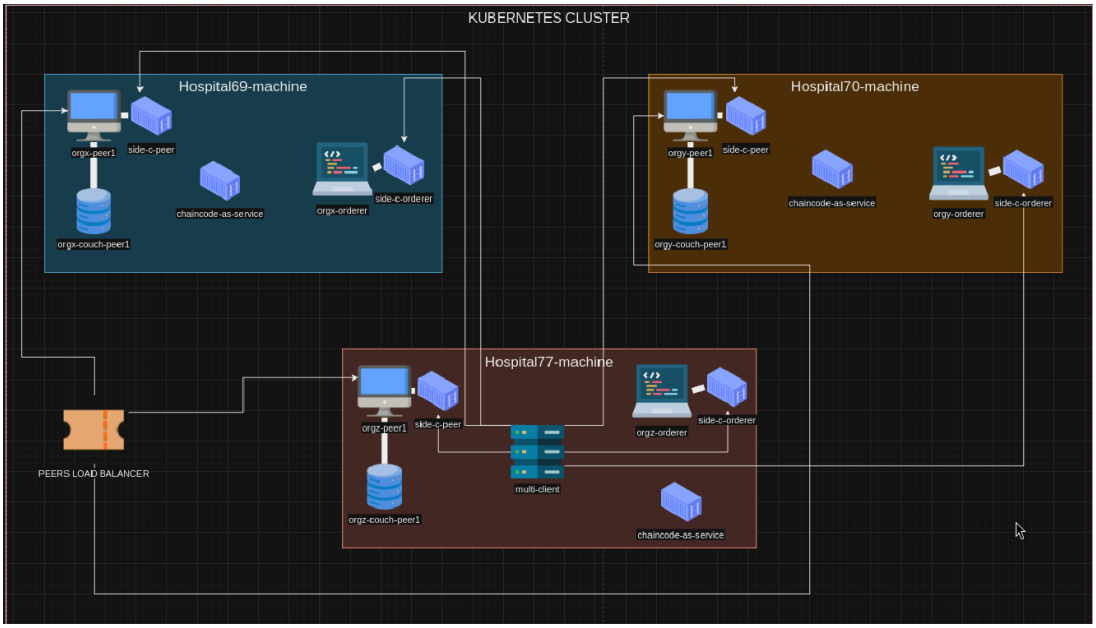

Figure 61: Second use case: Kubernetes with Load Balancer Architecture

Evaluation

The following image shows the results of the analysed metrics regarding the architecture

that uses **Docker**, architecture 1. In the Architecture 1 image presented below it is possible to observe the results of the metrics analysed for architecture 1. Starting the analysis with the CPU variation, this shows a sharp initial growth, followed by oscillations and a slight drop before growing again until the end where it reaches the peak with a value of 242.0. Disk usage on this architecture starts at a low level and shows steady growth until it peaks at 0.08 GB. After this, there is a small reduction before growing again at the end of the period. Analysing the metric, use of the ledger grows gradually over time, with some fluctuations until reaching a maximum value of 0.01 at the end of the period. Finally, analysing the use of RAM, it shows several fluctuations. In the graph it is possible to observe specific variations, but the trend is for growth over time, reaching its maximum peak at the end of the period with a value of 1.66 GB.

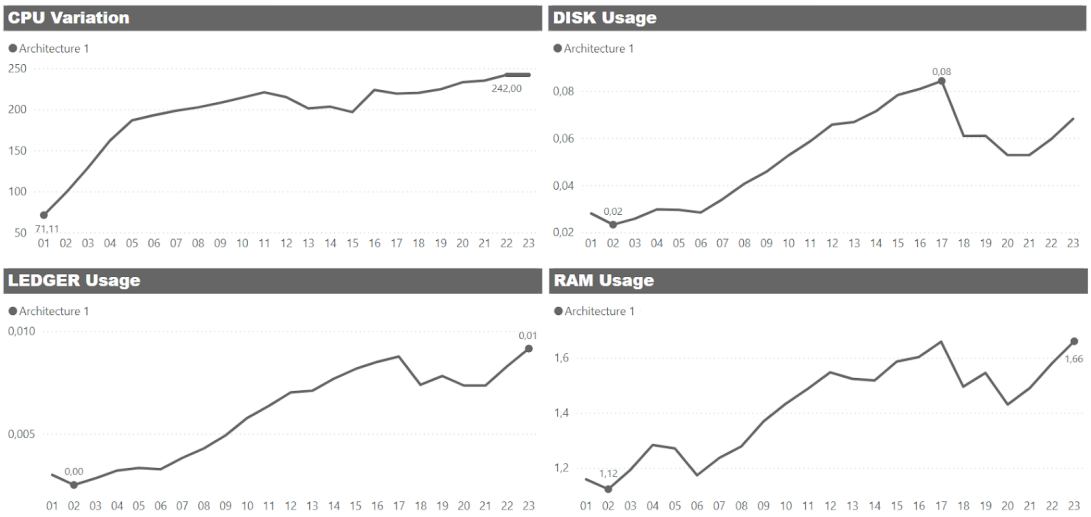

Figure 62: Second use case: Docker Architecture 1 Evaluation

The following image shows the results of the metrics analysed in both architectures. The analysis is carried out comparatively between both because they both use **Kubernetes**. They differ from each other, while architecture 2 uses **Kubernetes**, architecture 3 uses **Kubernetes** with load balancing. In this way, a comparative analysis is carried out between them.

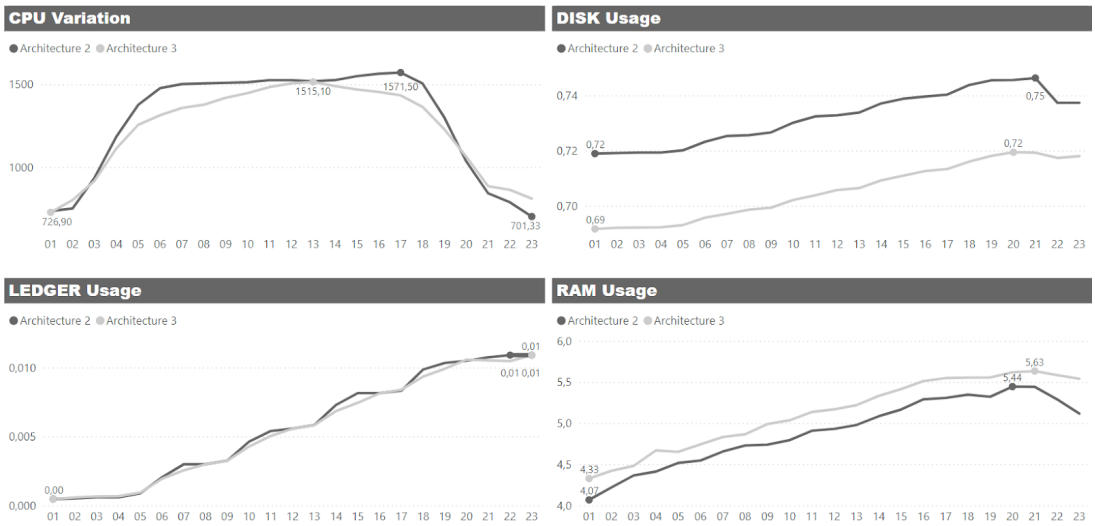

Figure 63: Second use case: Kubernetes Architecture 1 and 2 Evaluation

In the image related to the Architecture 1 and 2 above, it is possible to observe that architecture 2 shows a sharp increase in CPU variation, reaching higher peaks, with a maximum value of 1571.50 while architecture 3 presents a maximum peak of 1515.10. Analysing disk usage, both grow consistently, however, architecture 2 has a higher disk usage with a peak of 0.75 GB when compared to architecture 3 which has a peak of 0.72 GB. At the end of the analysis period, both showed a decline and subsequently stabilised. The use of the ledger in both architectures is practically identical throughout the analysis. Both follow a similar growth trajectory, with a maximum peak of 0.01 GB. Finally, in relation to the RAM usage metric, both grow consistently, however, architecture 3 has a slightly higher RAM usage than architecture 2 throughout the analysis period, reaching its peak at 5.63 GB.

20º Phase: Solutions comparison

On this phase comparison between the previous benchmarks was done, reaching a consensus about what must be sacrificed in order to choose one over another.

21º Phase: Creation of extra services

In the context of the phase 21, a set of extra services was created. This services are needed for the purpose of logging,monitoring and security. This was all done with continous integration pipelines and by leveraging technologies that enable cloud native applications inside of a **kubernetes** cluster, so the context could be the same for testing purposes.

The first service created was a own implementation of a block explorer. This proved to be very hard because, despite the gathering of data being something relatively easy to have by listening to the peer events, it was serialized in **protobuf** which means that the molding of the data should be uncovered to see in a human readable format, something that is not that easy because it is not explicitly documented and despite having the models knowing which slice of protobuf serialized data corresponds to one of the numerous existing models was something unfeasible, pretty much like a puzzle without instructions and the picture to know where which piece would fit. However, this came to a success and this extra service proved to be very useful

in future iterations.

The second service created was a **quarkus** instance, conceded for making queries to the database where the block explorer was storing everything and this way serving a future client.

The third service created was a C++ web service for **prometheus** data. This is important because it is a good practise to not expose all of the **prometheus** data to the exterior, therefore the need to only expose those services that were actually in need and because the functionalities are very limited, using a high performing programming language was something that seamed better, specially because if the functionality is simple there are less concerns about eventual errors that lower level languages give, which are greater when the project complexity evolves.

The forth and final service that was created as a keycloak server, which was there to manage users. By logging in, a key was generated and could be used to authenticate a admin for him to interact with the previous mentioned services like **quarkus**,**cpp** and the **graphql** single point of failure that communicates with the components side containers.

22º Phase: Creation of a web client application

After the creation of extra services, something that could be used to communicate effectively with the infrastructure to supply an admin with management features is required. This management features were divided into **block explorer**,**general network**,**peer config** and **orderer config**.

Block explorer

This first main component was the Block Explorer, to monitor and manage network transactions. It collects and stores data through a custom implementation that captures all relevant details from the network. This data is securely stored in a database, granting easy access and analysis. Data is later processed by a backend architecture that makes it available to the administrative web interface. This system allows administrators to perform a number of administrative functions over most parts of the blockchain. These include viewing the network activities by channel, the performance of the network in considerable details, and the inspection of blocks, transactions, and other related network objects. It also enables the supervision and administration of the components, as well as the channels they form part of, which are necessary for the network to operate.

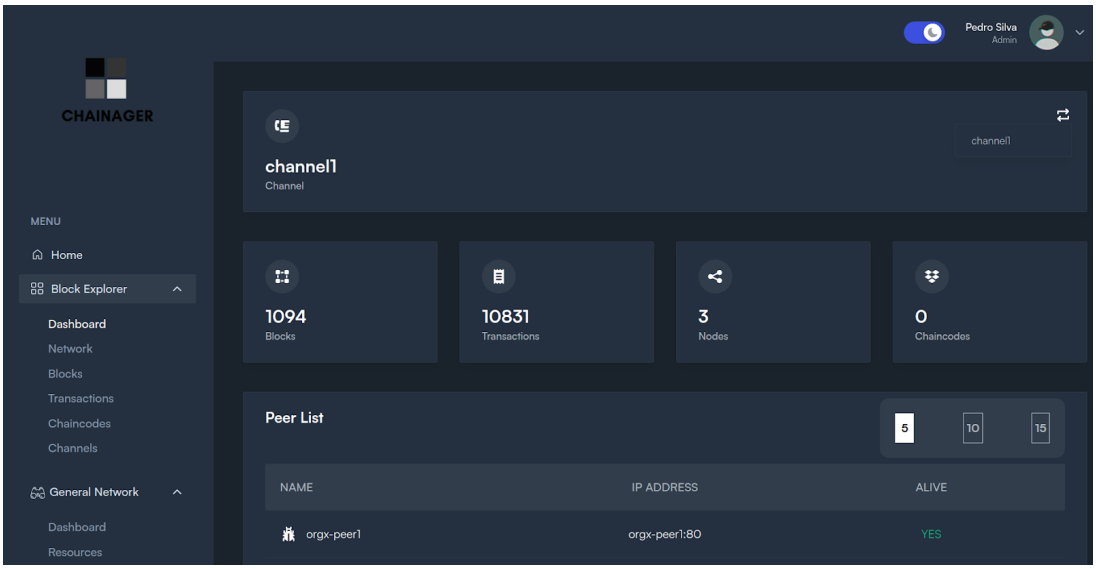

Figure 64: Second use case: admin web client block-explorer dashboard

In the dashboard menu, users can access detailed metrics, including the number of blocks, transactions, nodes, chaincodes, and peers within the system. Additionally, the dashboard provides insights into the percentage of transactions categorized by organization, allowing for a clear understanding of activity distribution. A timeline-ordered block list is also available, offering a chronological view of block creation and updates. All of this information is meticulously organized by channel, enabling users to easily navigate and analyze data specific to each channel.

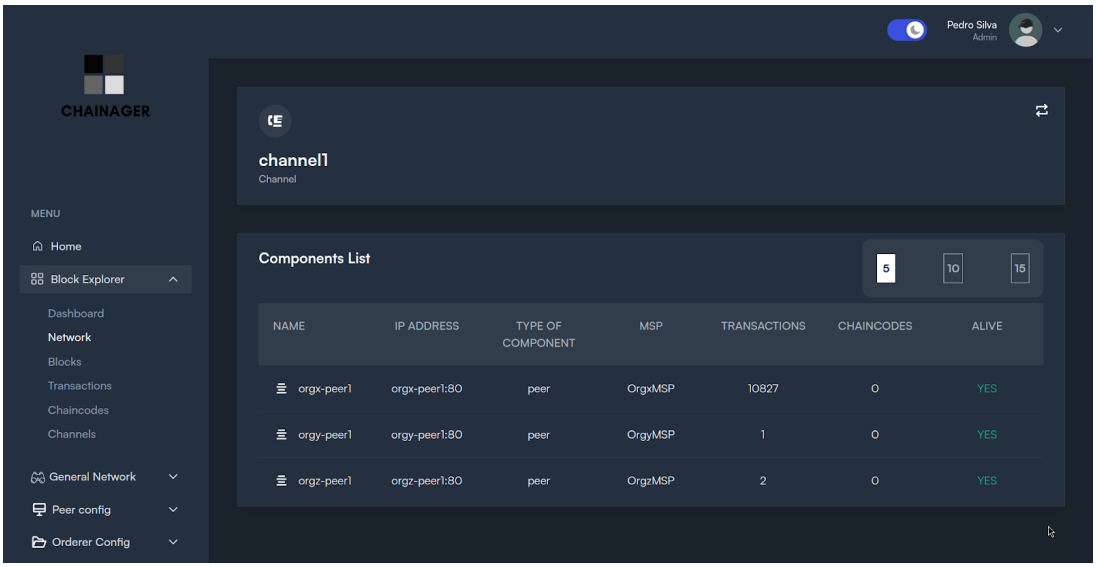

Figure 65: Second use case: admin web client block-explorer network

In the network section, users can access a comprehensive list of nodes currently participating in the channel. This list is similar to the one found in the dashboard but offers more detailed information for each peer. For instance, it includes the peer's IP address, type, Membership Service Provider (MSP), the number of transactions it has processed, the number of chaincodes it supports, and its current status (whether it is active or not). As with the dashboard, all of this data is organized by channel, allowing users to focus on specific channels as needed.

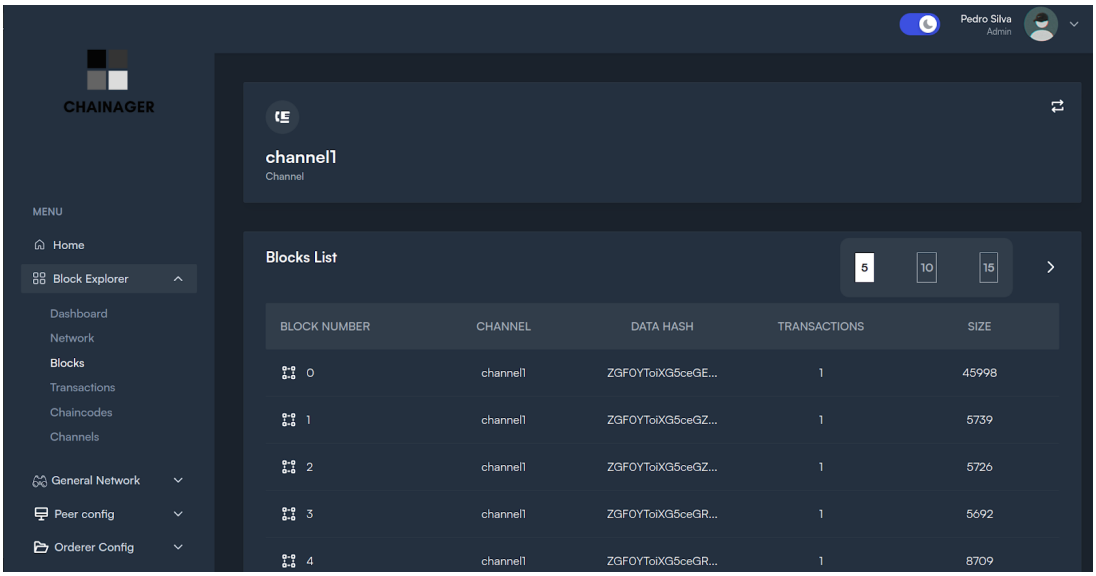

Figure 66: Second use case: admin web client block-explorer blocks

In the blocks section, detailed information about each block is provided, including the channel it belongs to (which is obvious since it is possible to select the channel to inspect), its data hash, the number of transactions it contains, and its size within the ledger.

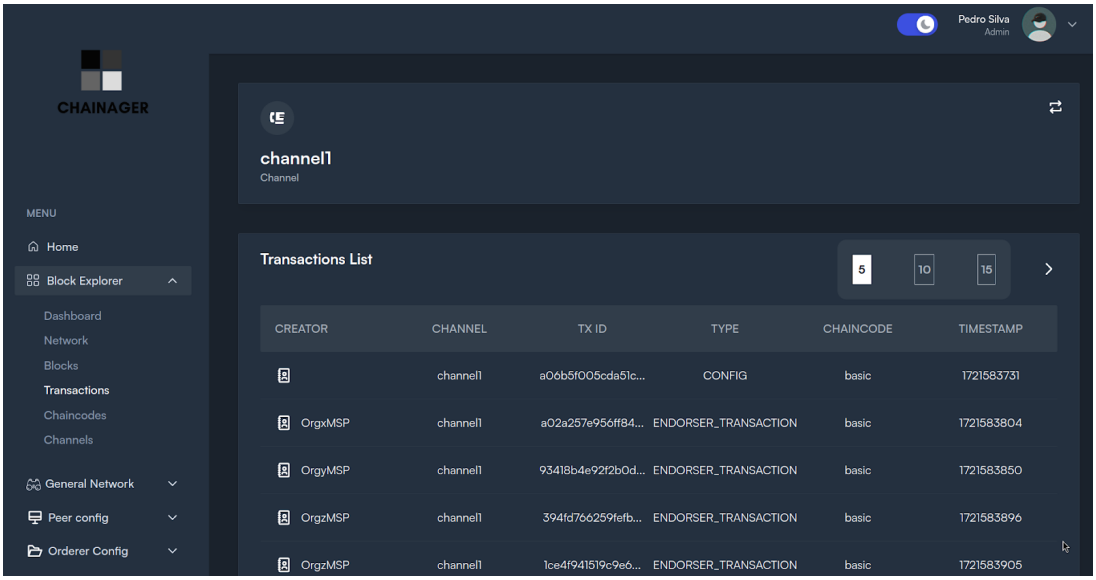

Figure 67: Second use case: admin web client block-explorer transactions

In the Transactions section, users can access a comprehensive overview of all transactions, each accompanied by detailed information organized by channel. This includes the creator of the transaction, the name of the channel in which the transaction occurred, the transaction ID (TX ID) for unique identification, the type of transaction performed, the name of the chaincode invoked, and the precise timestamp when the transaction was executed. This detailed breakdown, organized per channel, allows users to thoroughly analyze and track each transaction, ensuring complete transparency and traceability within the system.

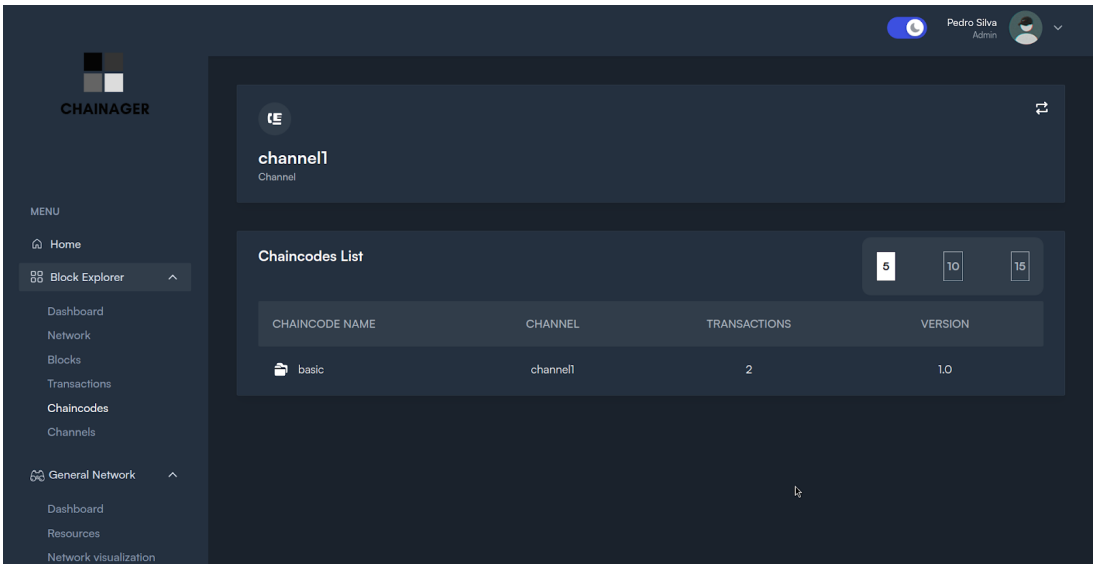

Figure 68: Second use case: admin web client block-explorer chaincodes

In the Chaincodes section, users can view detailed information related to the chaincodes within a given channel, including the chaincode name, associated channel, number of transactions, and the version of each chaincode.

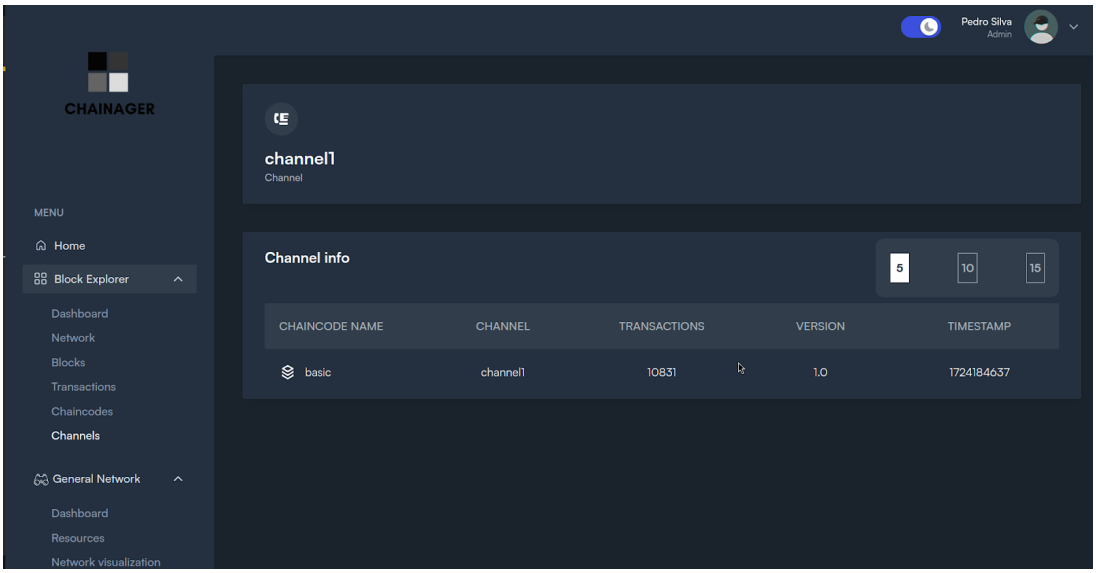

Figure 69: Second use case: admin web client block-explorer channels

In the Channels section, users can access detailed information about each channel, including the chaincode name, associated channel, the number of transactions per chaincode, the version, and the timestamp.

General Network

This part analyzes the resource utilization of the network. It provides monitoring of several performance metrics: CPU, RAM, disk usage, ledger activity, I/O operations, and network performance. Also, it presents the resources within each cluster, a graphical representation of

the network's structure, and allows for an examination of their configurations and how they are organized within each component's directory.

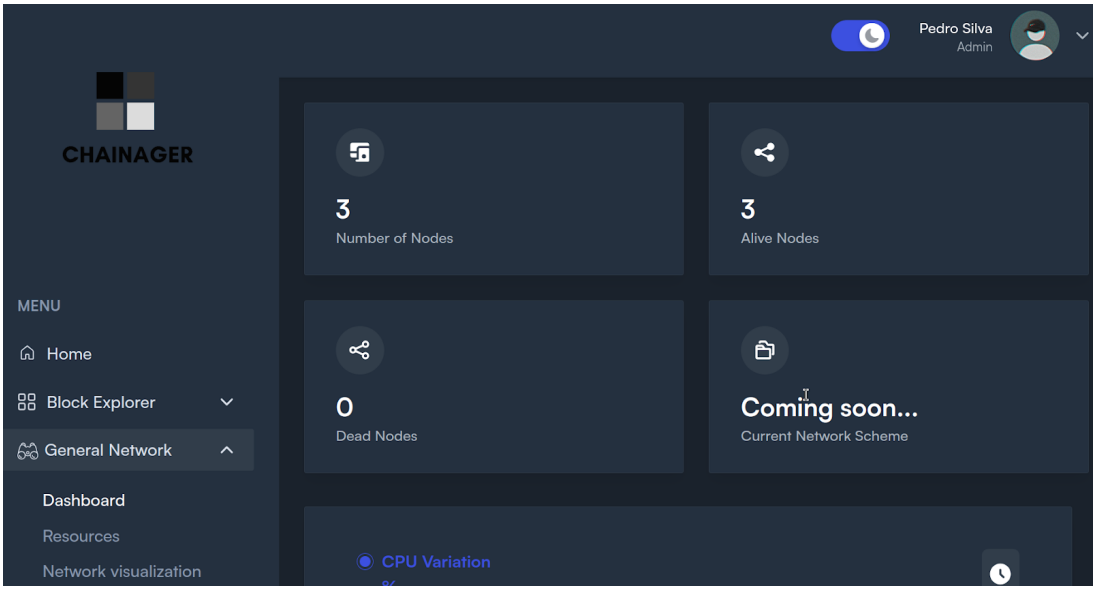

Figure 70: Second use case: admin web client general network dashboard

In the "Dashboard" section it is possible to have a glance of all the network like: checking number of nodes, alive nodes, dead nodes, CPU variation%, disk usage per machine GB, disk usage machine%, RAM per machine GB, RAM per machine%. Relatively to the network scheme, that's a probable future feature of changing between clusters because this is cluster-oriented.

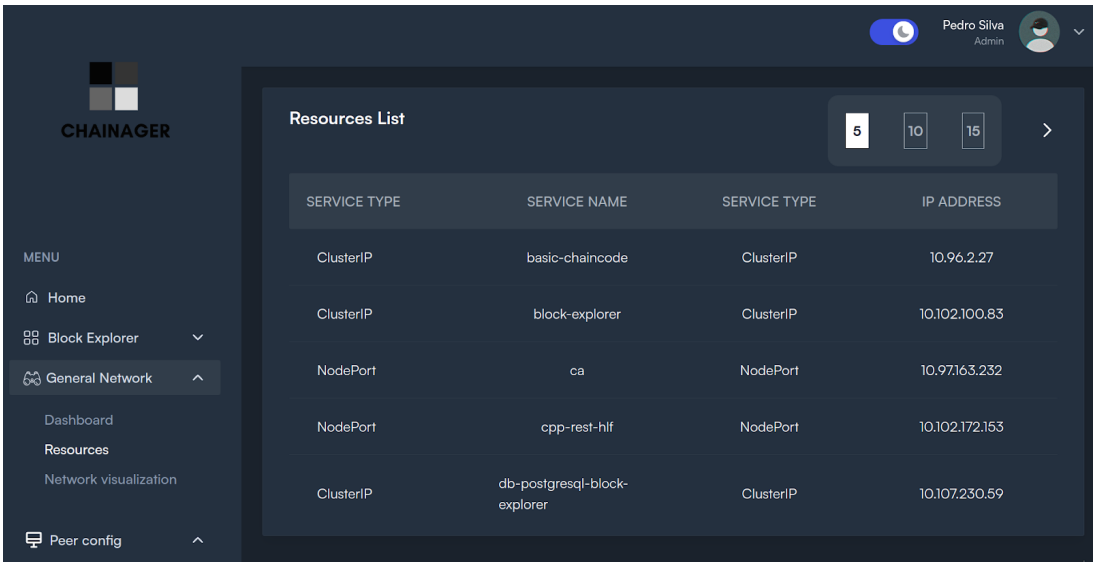

Figure 71: Second use case: admin web client general network resources

The "Resources" section provides a list of the services residing in that cluster, each corresponding to the DNS name of that service.

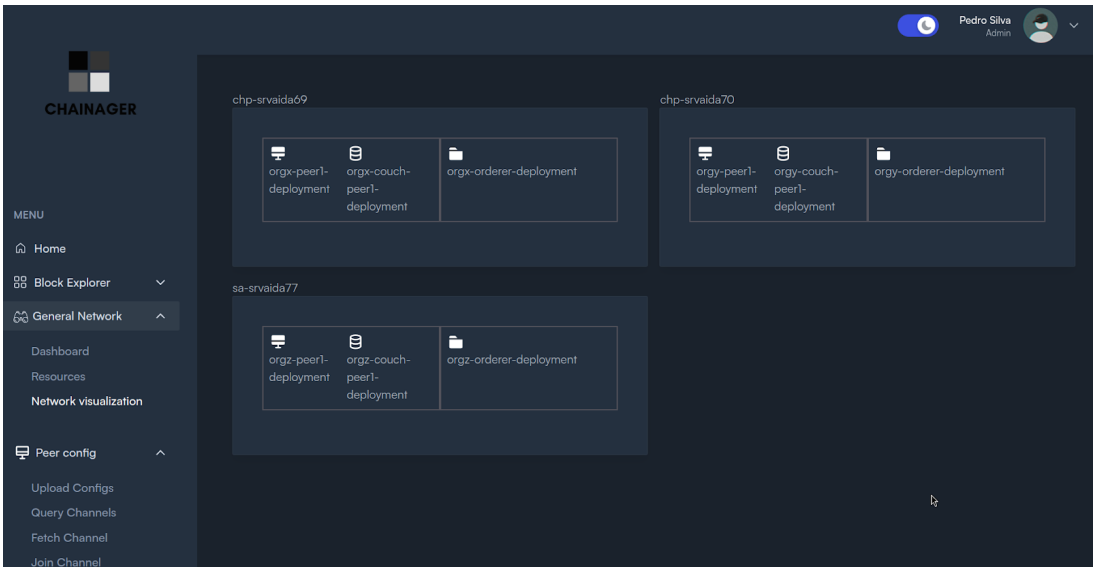

Figure 72: Second use case: admin web client general network resources

In the "Network Visualization" section, a cluster-oriented representation of the current components is provided. This view is cluster-oriented because it displays resources based on what exists in the cluster rather than on a specific network channel. If it were based on a network channel, the visualization could include components outside of the cluster.

Peer Config

This section deals with peer management and provides a comprehensive user interface for performing all standard peer operations, like: uploading configurations, querying, fetching and joining channels, installing, querying, aproving and committing chaincode, checking chaincode approvals, invoking chaincode for testing, and executing custom commands for more advanced operations.

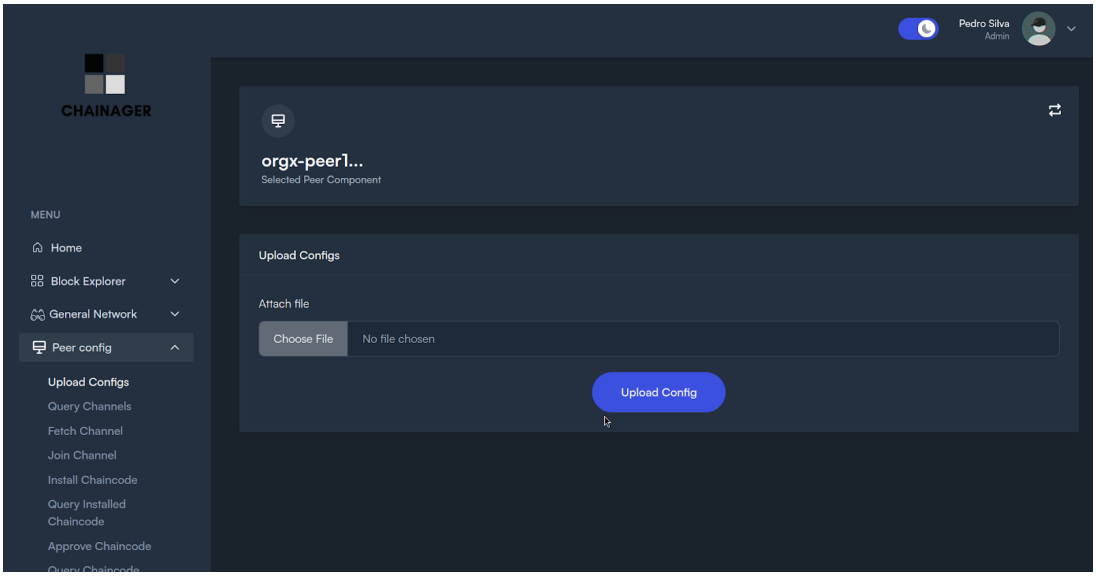

Figure 73: Second use case: admin web client peer config upload of configurations

This section allows for the upload of pair definitions, as well as the chaincode package for

installation. To do this, a pair is selected to proceed with this process.

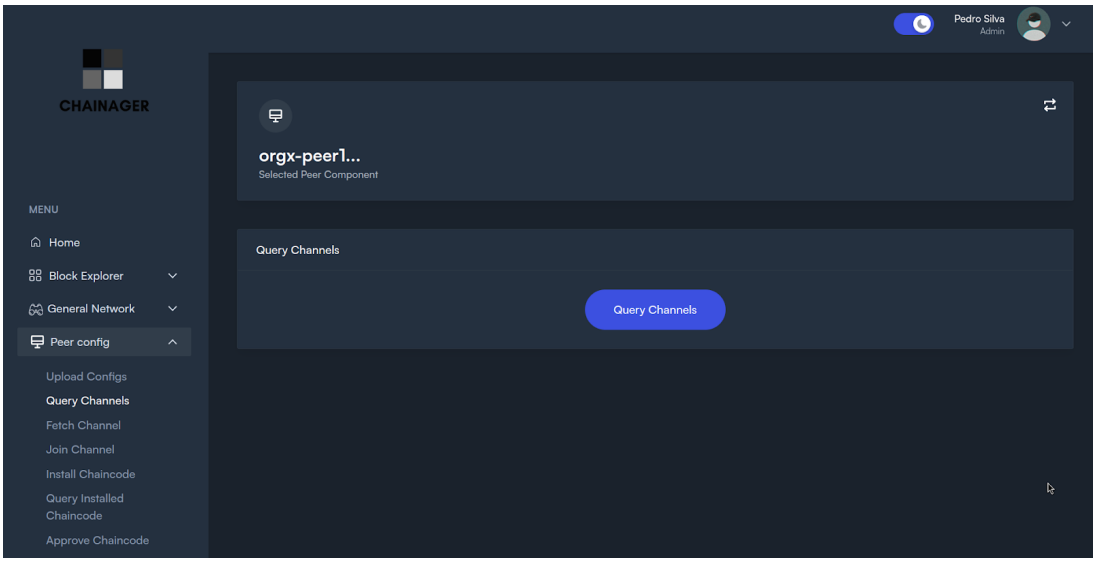

Figure 74: Second use case: admin web client peer config query of channels

This section makes it possible to consult the channels to which a particular peer is connected, and that is why to get to this point it is necessary to select a particular peer.

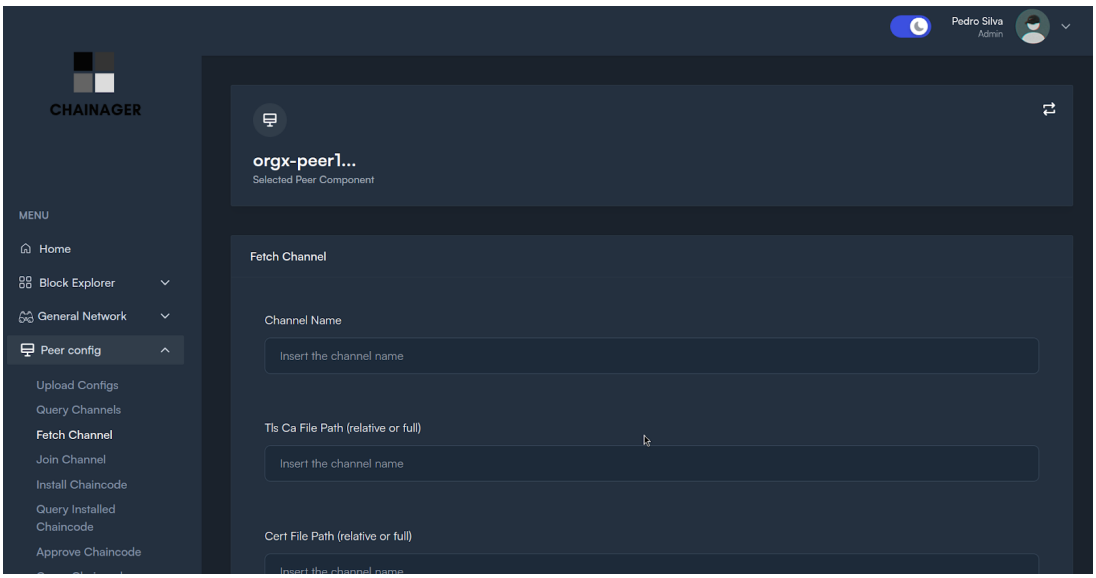

Figure 75: Second use case: admin web client peer config fetch a channel

In this section, searching for an order channel is possible. To achieve this, the channel name, TLS CA file path, certificate file path, key file path, and IP address of the requestor from which it fetches the channel are provided.

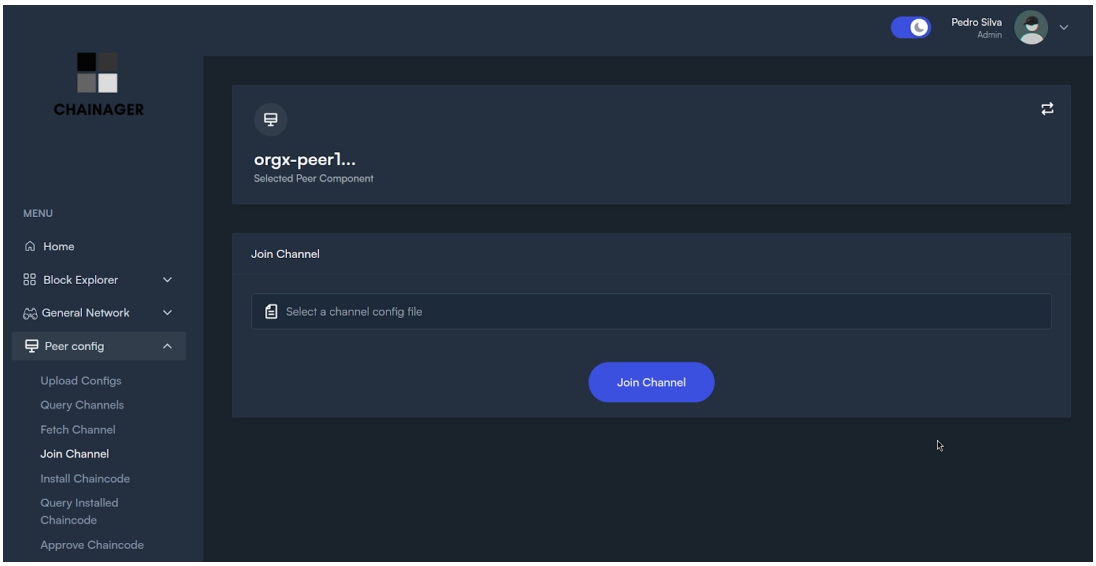

Figure 76: Second use case: admin web client peer config join channel

In the "Join Channel" section, joining a channel is enabled by selecting the correct searched channel on a given peer.

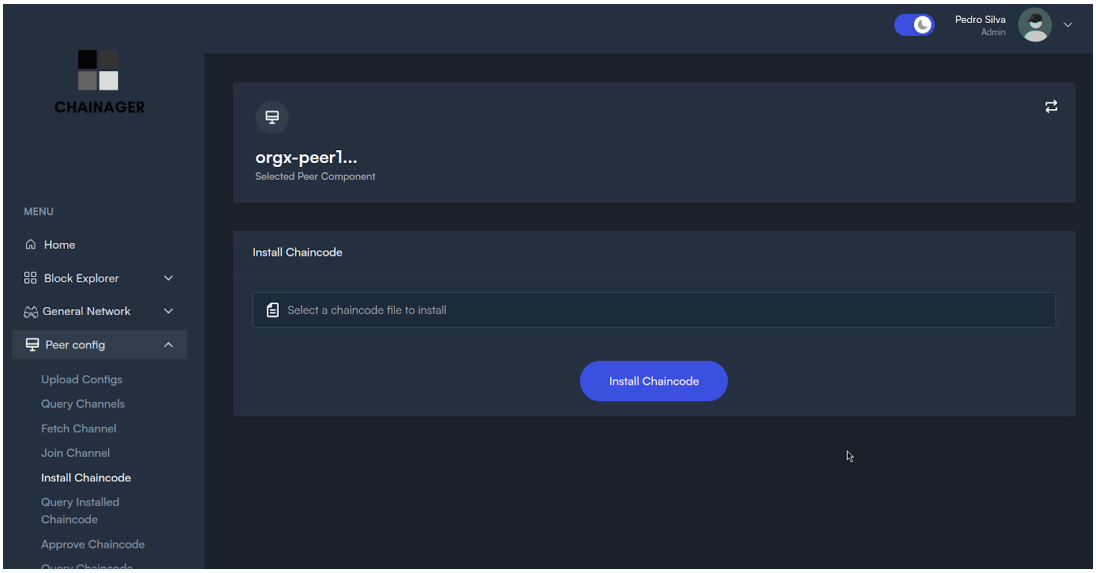

Figure 77: Second use case: admin web client peer config install chaincode

In this "Install Chaincode" section, access to files uploaded from the "Upload Configs" section is provided, making it easy to select the desired chaincode.tar.gz file for installation.

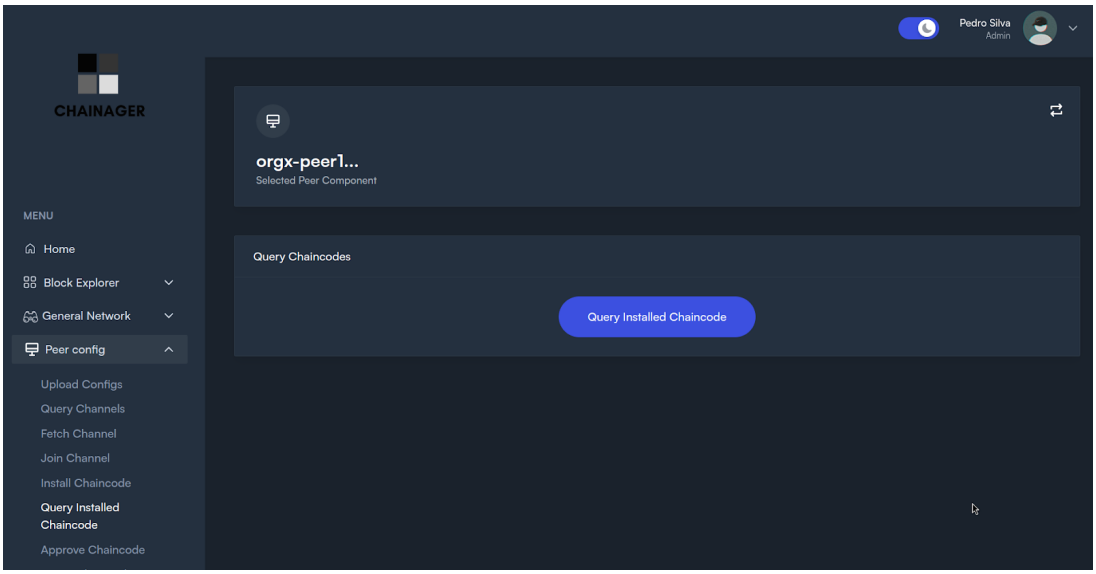

Figure 78: Second use case: admin web client peer config query chaincodes

The "Consult Installed Chaincode" section provides access to the chaincodes that are already installed.

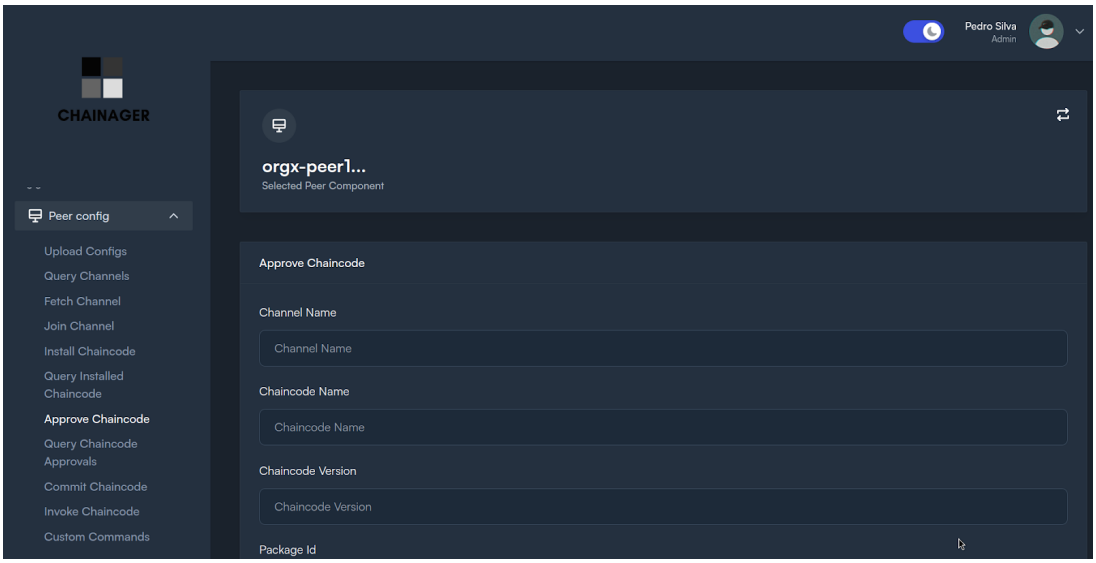

Figure 79: Second use case: admin web client peer config approve chaincode

The "Approve Chaincode" section provides all the information needed to approve a chaincode, including the channel name, chaincode name, chaincode version, package ID, sequence, TLS CA certificate path, CA certificate path, private key path, and orderer IP address.

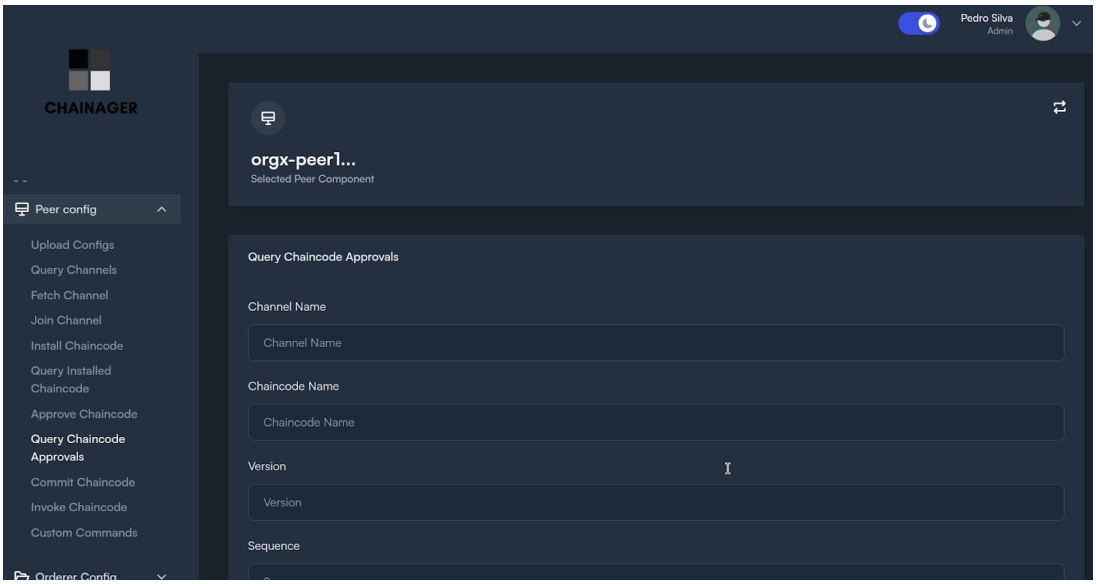

Figure 80: Second use case: admin web client peer config query chaincode approvals

The "Querying chaincode approvals" section provides a form with all the fields needed to query chaincode approvals, including the channel name, chaincode name, version, sequence, TLS CA certificate path, CA certificate path, and private key path.

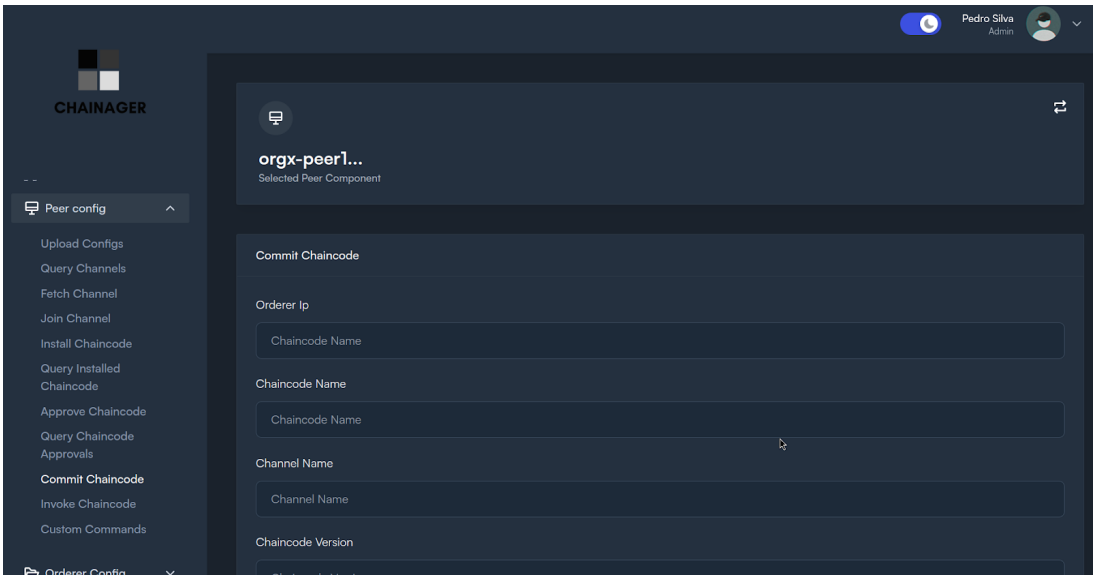

Figure 81: Second use case: admin web client peer config commit chaincode

The "Commit Chaincode" section provides all the fields required to commit a chaincode to a given peer and channel, including the orderer IP, chaincode name, channel name, chaincode version, required peer signatures, chaincode sequence, and the TLS CA file for connecting to the peer.

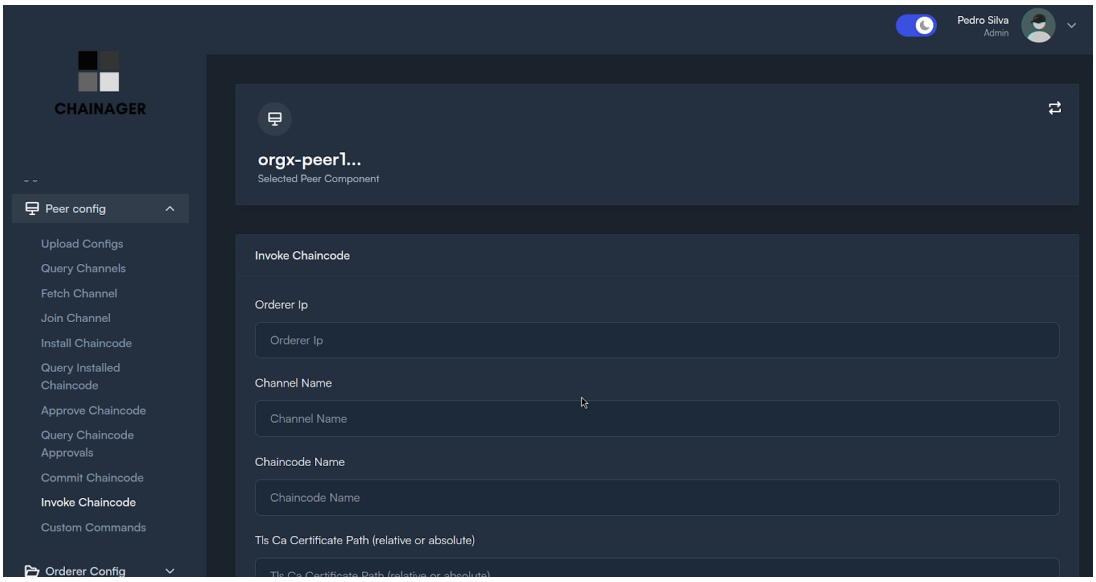

Figure 82: Second use case: admin web client peer config invoke chaincode

In the "Invoke Chaincode" section, I have all the necessary information to invoke a chaincode, including the orderer IP, channel name, chaincode name, TLS CA certificate path, certificate path, private key path, function name, and arguments separated by commas.

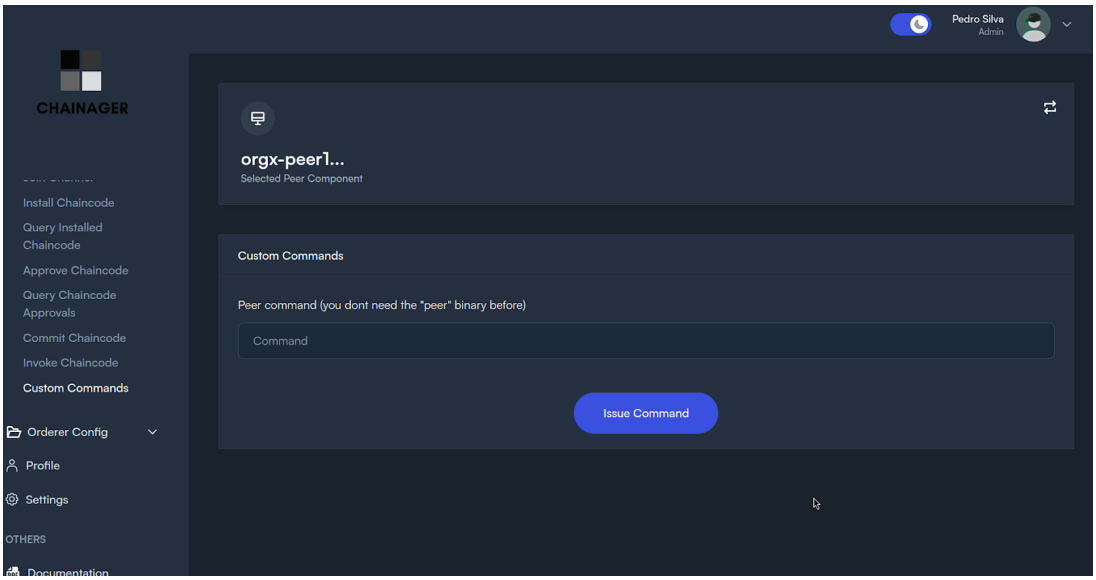

Figure 83: Second use case: admin web client peer config custom commands

In the "Custom Commands" section, I have all the necessary tools to create a custom command with a peer, which can be valuable in unexpected situations.

Orderer Config

This section focuses on the management of the orderer and provides a comprehensive set of tools for efficiently handling all regular duties. Users can create new channels, join existing ones, and execute custom commands for more complex configurations.

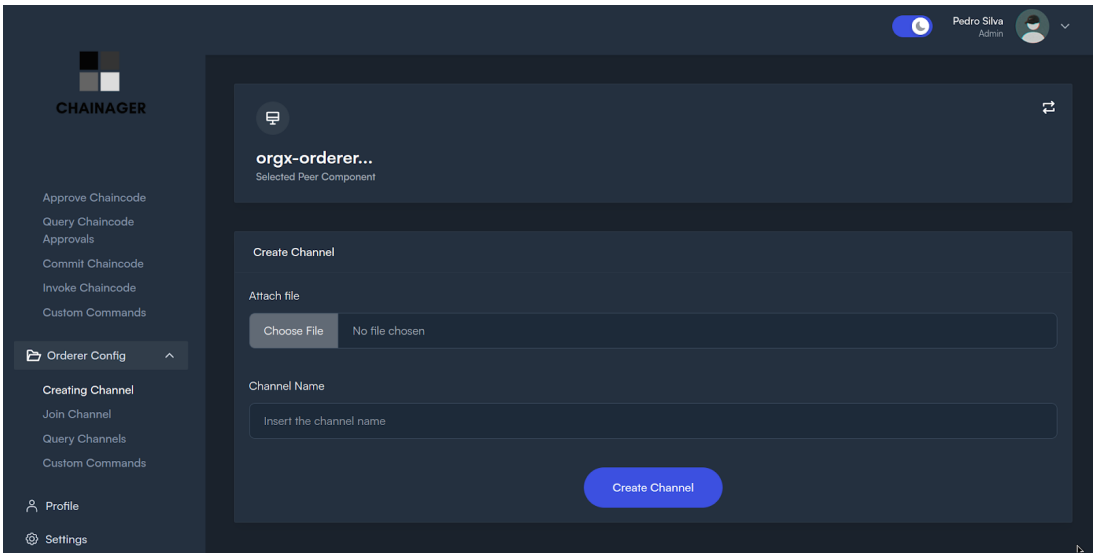

Figure 84: Second use case: admin web client orderer config creating a channel

This section is designed to create a channel for a selected orderer. To do this, an order is selected, a channel configuration file (configtx.yaml) is attached, the channel is named, and finally the request is sent.

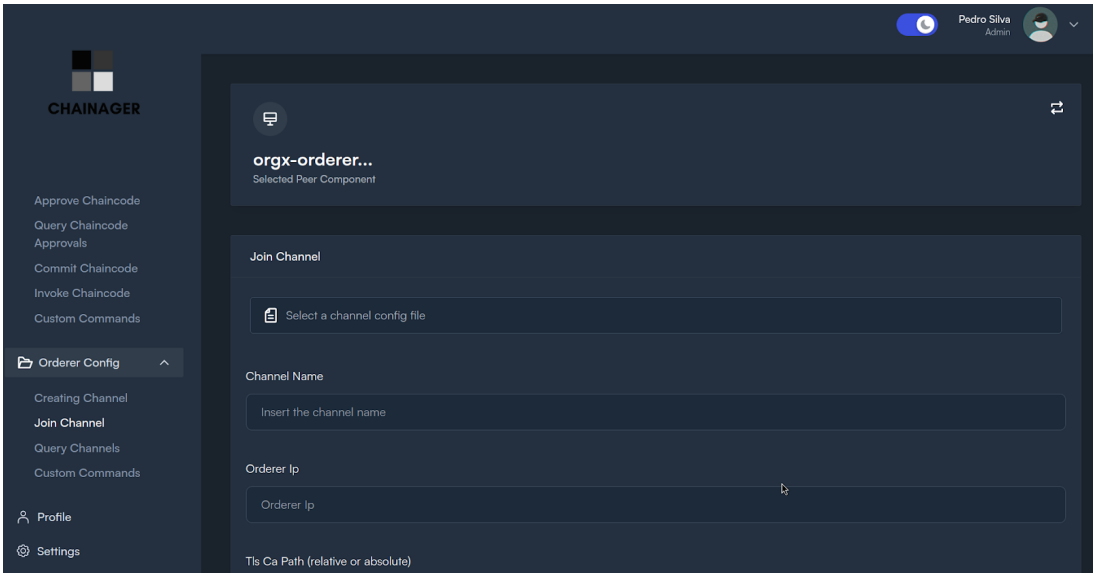

Figure 85: Second use case: admin web client orderer config join a channel

The "Join Channel" section is designed to facilitate joining a channel. Here, all necessary components are displayed. These include the channel junction file, channel name, requestor IP, TLS CA path, client certificate, and client private key.

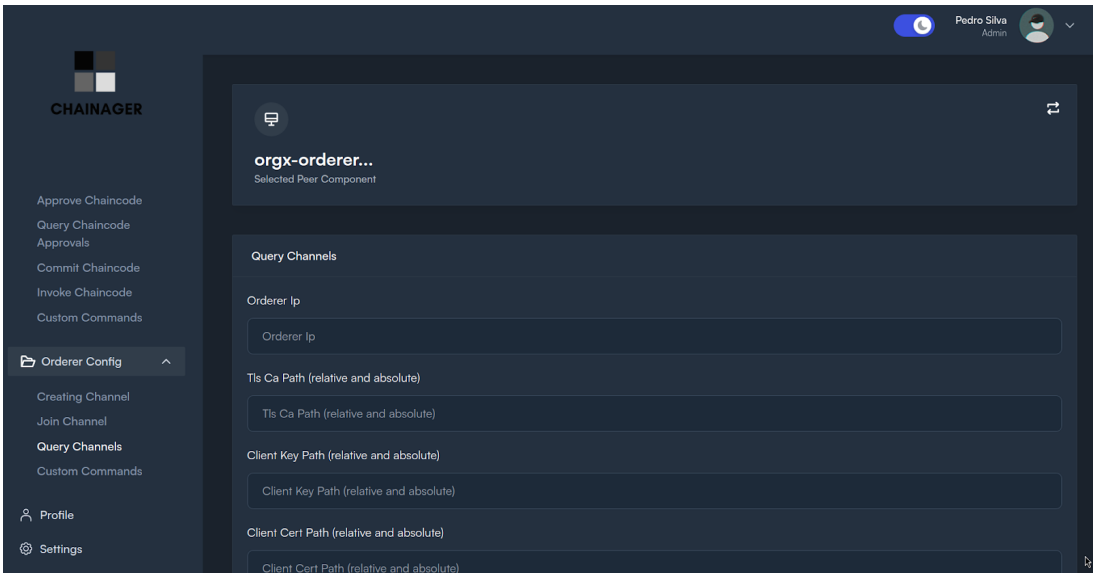

Figure 86: Second use case: admin web client orderer config query channels

This section is for querying the channels associated with a given order. To perform the query, the request IP, TLS CA path, client key path, and client certificate path must be provided.

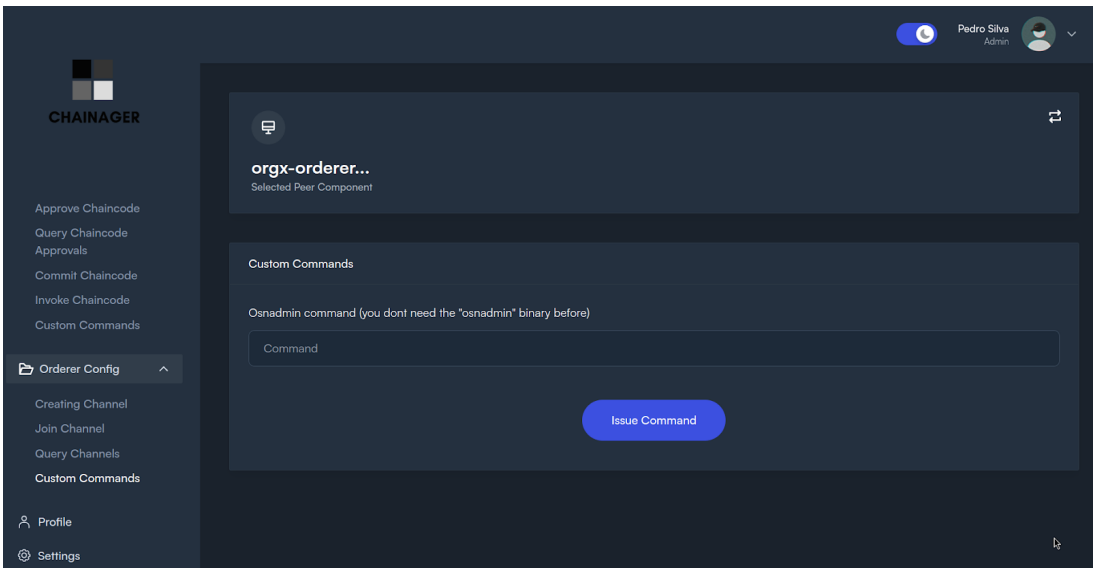

Figure 87: Second use case: admin web client orderer config custom commands

This is the custom commands section. Custom commands are executed in the order component if there is a need for a personalized interaction with it.

Final Architecture

After having the web client application, it could be said that the final architecture was achieved. There may be some improvements in the future but the most of it was already accomplished.

In this final architecture, it was implemented with the same setup as the one verified in the Kubernetes with load balancer. Here, the architecture is presented with additional components that are essential to a production environment. With this in mind, besides the core components, there are also: prometheus (for fetching the data from all the cadvisors and also

components of the network), cadvisor (measures collector for prometheus), cpp-rest-service (to wrap the data from prometheus and serve it to the admin web server; it is a good practice to not expose entirely the prometheus API), keycloak (for users management), blockexplorerlistener (a developed implementation of a blockexplorer listener to grab the data from events and store it on a database), postgresql-blockexplorer (database for the blockexplorer listener), and quarkus rest for serving this blockexplorer data to the admin web server. Note that the multi-client now serves the admin web server.

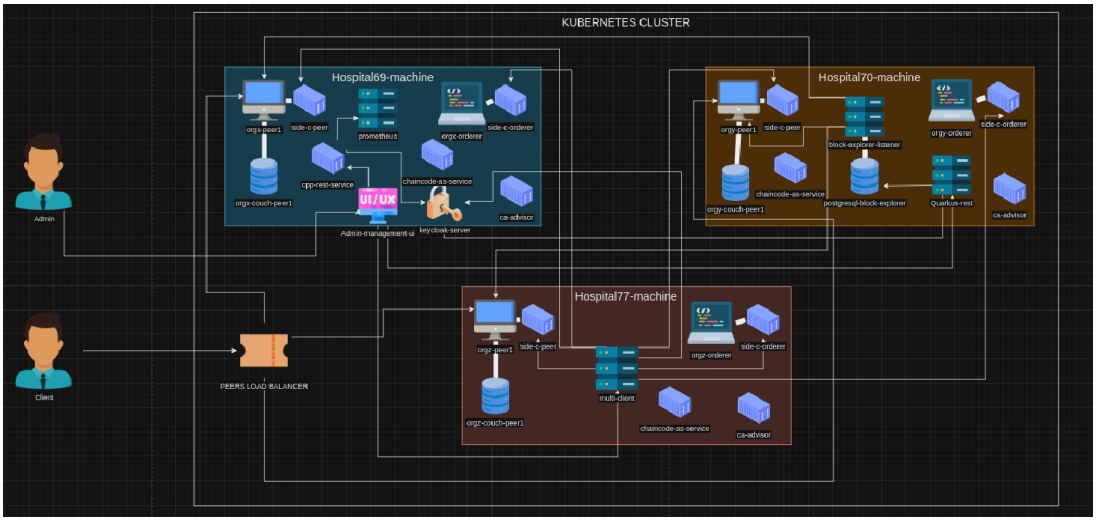

Figure 88: Second use case: final architecture

23º Phase: Article development

For the 23º Phase there was the support of this project for a existing article. This article had in mind the extraordinary Q1 journal IEEE, where the second use case findings could give a significant presence.

24º Phase: Article Review

In this period, there was a review and improvement of the article in hands, to make sure that everything was done accordingly with the quality that this kind of journals require. This phase could be replicated after the delivery, since the article could be rejected for further acquaintance in case something isn't well covered.

25º Phase: Article Delivery

In this period, the delivery of the article is putted into practise and further response with feedback would be expected.

26º Phase: Dissertation Review

In the realm of this phase, there will be conducted a dissertation review. This is of extreme magnitude because it ensures quality,validity and academic integrity of the work. This represents the culmination of all the efforts that were done until this point, contributing to the body of knowledge of the target field.

27º Phase: Dissertation Delivery

At this period, a significant milestone in the academic journey is accomplished. This symbolizes the completion of a important step in the career of the researcher. Preparation,submission guidelines and the presentation of such work is prepared but despite the fact that most thesis end their work with this delivery, for this project that is not possible since there is still a lot of work to do.

28º Phase: Dissertation Presentation

The final step in the calendar is the presentation of the dissertation. This is the oral defense or formal presentation required to showcase the research, defending the findings and also to engage with an academic audience. A successfully presentation is meant to only only demonstrate the understanding of a topic but also to show the ability to communicate complex ideas clearly and with confidence.

	July	**August**	**September**	**October**	**November**	**December**
1ºPhase	X					
2ºPhase	X					
3ºPhase		X	X	X	X	
4ºPhase	X	X	X			
5ºPhase		X	X			
6ºPhase			X			
7ºPhase				X		
8ºPhase				X		
9ºPhase				X		
11ºPhase					X	
12ºPhase					X	
13ºPhase					X	X
14ºPhase						X

Table 8: Monthly plan 2023: Second use case (part 1)

	January	**February**	**March**	**April**	**May**	**June**	**July**	**August**	**September**
9ºPhase			X						
10ºPhase			X	X	X	X	X	X	X
14ºPhase	X	X							
15ºPhase	X	X	X						
16ºPhase			X	X					
17ºPhase				X	X				
18ºPhase					X	X			
19ºPhase							X		
20ºPhase							X	X	X
21ºPhase							X	X	X

22ºPhase									X
23ºPhase							X	X	X

Table 9: Monthly plan 2024: Second use case (part 2)

	October	**November**	**December**
10ºPhase	X	X	
22ºPhase	X	X	X
23ºPhase	X		
24ºPhase	X		
25ºPhase	X		
26ºPhase		X	X

Table 10: Monthly plan 2024: Second use case (part 3)

	January	**February**	**March**	**April**
22ºPhase	X	X		
26ºPhase	X			
27ºPhase		X		
28ºPhase				X

Table 11: Monthly plan 2025: Second use case (part 4)

4.4.3 Approach

Relatively to the approach adopted during this use case 2 project, there must be known that every single phase that was represented during the previous mentioned plan was mostly subjected to **DSR**. This is because, everytime a given outcome was not reached, a given refined was expected, creating the loop that was already expressed before. Also, a given problem is trying to be solved within this context which is the problem of managing such complex infrastructure, giving wings to multiple iteractions which generated both theorical but also practical insights that are very valuable for the research realm. Aditionally, during the project the best good practises and innovative processes were considered ranged from testing and creating cloud native applications to devops practises such as continuous integration.

4.5 Risk List

Nº	**Description**	**Probability(P)**	**Impact(I)**	**Seriety(P*I)**	**Mitigation Action**	**Verified**

1	Unreachable deadlines	3	5	15	Planning better the next events and gain experience from previous deliveries;	Yes
2	Requirements of the project change	2	4	8	Make frequent communication with stakeholder;Check well the limitations imposed by the project;	No
3	Lack of Time	4	5	20	Implement only what is actually needed	Yes
4	Resources	4	5	20	There are no mitigation measures	Yes
5	Incorrect understanding of data	2	5	10	Spend more time analyzing the data; Make sure that the data is relevant for the object of study;	Yes
6	Mistakes	1	5	5	Don't do tasks in automate mode; Focus more in the tasks;	Yes
7	Security implications	1	5	5	Follow standards; Try to learn more as the time passes;	No
8	Complexity of the project	2	3	6	Document Everything; Simplify everything;Join Communities;	Yes
9	Unknown Errors	2	4	8	Join Communities;Arrange more debug mechanisms;	Yes

10	Acceptance	4	5	20	Make the best solution possible;Make clear the need of the solution;	No
11	Maintain the solutions	4	5	20	Document everything; Pay attention to updates;	No
12	Infrastructure is more oriented to larger organizations	4	5	20	Think in how to target smaller organizations;	No
13	It was never implemented in cloud environments	4	5	20	Make a operational guide of how to implement in cloud environments;	No
14	Not knowing when to stop iteractions	4	5	20	Understand in each point the refinations are less relevant;	No

Table 12: Risk List: Second use case

4.5.1 Discussion

Throughout this section, the different architectures will be discussed accordinally to previous testing results, considering them in terms of scalability and sustainability. Also, a **SWOT** diagram will be presented, to understand the strengths,weaknesses,opportunities and respective threats.

Architectures discussion

The following table represents the main sides to take into accordance when it comes to choose between the kubernetes and the docker architecture.

Characteristic	Docker	Kubernetes
More performance for smaller environments	X	
Less abstract and easier to learn	X	
Incorporated load balancer		X
Resource management		X
Robust API for managing resources		X
High availability		X

Built-in mechanism for horizontal scaling		X
Higher resource overhead		X

Table 13: Second use case: Characteristics comparasion between docker and kubernetes

While **Kubernetes** does come with a steeper learning curve and a higher resource overhead, these complexities are justified in larger, more complex environments. In this case, The smaller environment favors Docker for its performance. and ease of use. However, as the project expands and the environment scales, **Kubernetes** will likely become the more beneficial option due to its advanced features and ability to handle larger workloads. Regarding the comparison between Architecture 2 and 3, even though the implementation of a load balancer had only residual benefits in some of the metrics and that its Implementation may introduce complexity to the development of the network, it provides several advantages that, in the long term, can be very beneficial. It can ensure better scalability, reliability, resource allocation, and performance, contributing for the solution sustainability over time. In summary, **Docker** has proven to be the more performant solution for the current, smaller environment. Yet, this project relies on **Kubernetes** for specific aspects, such as the UI tool, due to its powerful API and management capabilities. As growth is anticipated, the robust management and high availability offered by Kubernetes by Deploying a load balancer will become increasingly valuable. making it the preferred choice in larger, more complex deployments.

SWOT analysis

The following figure dictates a **SWOT** analysis for the second use case project.

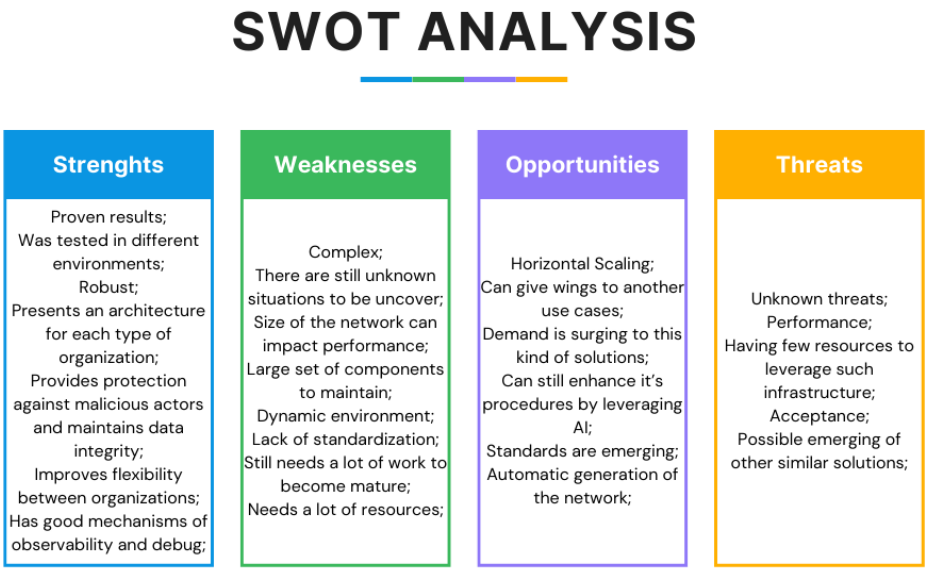

Figure 89: Use case 2: SWOT analysis

4.5.2 Future Work

Since blockchain networks are not a basic matter and managing multiple components in that realm scalates to even more complex situations, future work regarding what has been covered so far is more than needed. The work that remains to be done ranges from observing further details about the infrastructure chosen architecture and processes refining to improving existing solutions that have been constructed so far. There is no perfect system, and thinking forward to collate possible issues is more than a primary key. However, additions to the current implementation are out of question since the more is introduced, the more features must be supported, thereby quality can be sacrificed.

By the effect of such, in this section, the most critical future work in mind will be covered and explained in a brief way.

Horizontal Scaling in a HyperLedger fabric network

Horizontal scaling, or scale-out, is a critical optimization of the system to manage more workload—notably within containerized environments—into one node, while horizontal scaling will increase the number of nodes to share a load over more than one machine. This means that in the case of **Hyperledger Fabric** (HLF), which is a commonly used distributed ledger platform for blockchain-based solutions, there is a potential application of horizontal scaling. It would be quite interesting, to say the least. In the case of **Hyperledger Fabric** (HLF), since a network often involves the co-existence of multiple organizations, which are usually

referred to as consortiums, the potential key issue of effective performance optimization may lie in the ability to scale horizontally when the size and complexity of this kind of network grows. Horizontal scaling will allow the number of instances in the system to bend with real-time traffic demands, or surges in computational work—things that normally accompany every organization in an HLF network. Every organization within an HLF network operates its own set of peers including services for ordering and many other components. In this regard, we will predict for subsequent work how much horizontal scaling can be done with an HLF network. We will look especially at when it can be pushed forward—when no diminishing returns from the system's architecture meet a bottleneck. This will comprise the evaluation of different factors like latency, throughput, and resource usage under different scaling scenarios. We also want to verify if the application of horizontal scaling in such networks offers real advantages for large consortiums, mainly those operating with a high volume of transactions or in need of high fault tolerance. This is significant potential for research, as it could bring insight that is useful to organizations running large-scale **HLF** networks. With this, architectural teams would then have the knowledge of whether horizontal scaling really yields measurable improvement in performance and resilience that is satisfying to their operational needs. Finally, findings emerging from this work may help consortiums understand the trade-offs involved and make a decision on whether horizontal scaling could really be a feasible and beneficial approach for the considered blockchain projects.

Automating Network Component Generation

This will focus on automating the generation of network components, such as peers and orderers, directly in a user interface (UI). This automation will be possible by using scripts and a standard template to the creation process, reducing the need for manual configuration which results in a reducing of errors. By integrating this directly in the the UI, we can significantly improve operational efficiency by enabling rapid deployment and management of HLF components. This approach will allow network administrators to create new components with ease, which will give them more time to plan how exactly they want their network to be because creating it by themselves creates a huge overhead due to the number of configurations that can result in unexpected behavior, which may cause a very huge loss of time.

Creating Recovery Plans for HLF Components

Because the **HLF** is a complex infrastructure, developing recovery plans for components is of much importance. With this in mind, it is planned to actually create a set of detailed plans for given situations that outline the steps necessary to quickly identify, contain, and recover from failures in these components. These recovery plans will be essential to know exactly what to do when some unknown behavior happens, which is something that we should give special attention to due to the importance of the probable use case. By implementing these recovery strategies, organizations can feel confident to use our solution for their **HLF** networks and know for sure that the system remains resilient to a vast number of situations, preserving data integrity and service continuity. In case some new case occurs, we have also plans to create a community that communicates faced challenges, that later one could be added to this recovery plans because this work must be continuous.

Improving Documentation Practices

Documentation often lacks clarity and accessibility, which can lead to misunderstandings and inefficiencies while working with the given product. Because we have our own documentation for this project, it is important to refactor it in order to create well-formatted documentation that is easy to use and clearly explains how to complete certain tasks that may be required depending on the objective of the administrator. Improving the documentation is something that we see as very important, as it will be the basis for current and future administrators and programmers to take advantage of the created features in our project, making it easier to work with. It also helps with onboarding new team members and ensuring consistent practices across the organization. The point of access to our documentation is still under work, but it is expected to be available in our UI.

Improving the current UI

Despite being very composed, our current UI must be even more improved. There are features to be added and others that must have further modification, as for now it is still in a beta version. With this in mind, we want to: Add a menu for documentation, which we already discussed in the section before, and develop a mechanism that allows administrators to select and manage multiple clusters simultaneously in order to address the increasing complexity of network infrastructures. This multi-cluster management capability will be a good ally for managing multiple clusters, which may be a big ally for bigger organizations, improving scalability and reducing administrative burden. In addition to what was previously mentioned, a new menu option will be introduced precisely to provide a tree model view of the component structure. This feature is meant to understand the current position of given artifacts needed for certain operations within the network (certificates, config blocks, etc.).Finally, the development of a geopositioning map that aims to display the physical location of network components. This feature will allow administrators to view the geographic distribution of components, which can be something valuable for physical interaction with components of the cluster (like a node, for example). Although not as relevant as the other ideas discussed before, it remains a very good feature for this project.

Developing Prototypes for Enhanced Security Protocols

Because security is so important, especially in the context of service authentication within the network, there is a need to strengthen the procedures to this effect. With this in mind, there is future work that must involve the development of prototypes for security protocols. The idea is to implement optional extra steps for authentication that secure even more our platform. This is what will enable those services that collect block explorer data, manage peer connections, and manipulate Prometheus data servers. These new protocols will focus on strengthening the security of those services, ensuring that they are protected against unauthorized access and potential threats. There are ideas that got discussed already about this matter, like, for example, creating a structure that does not allow access if one of the steps gets compromised. Also something that will evolve more and more with the time.

Gathering Regulatory Information on Private Ledgers

As private ledgers are being developed, remaining informed about the regulatory landscape will be very crucial. It therefore follows that there is future work related to collecting and analyzing information about regulations related to private ledgers. Such regulations make it easier to shape the network architecture for it to meet the legal impositions that exist around different countries or even use common standards to have broad acceptance of the platform. This work is also incremental, especially in view of the emergent nature of the matter discussed, where more and more standards come into play.

4.5.3 Conclusion

To summarize, in terms of infrastructure architectures, three implementations were developed during the project: one using **Docker**, the other using Kubernetes, and a third one using **Kubernetes** alongside a load balancer. This implementation evolved as we developed further in time, aiming to firstly create the most basic network (docker), secondly to present a more robust architecture that could also be used to manage better an infrastructure that could evolve in number of components with time (kubernetes), and lastly to an infrastructure where we could balance traffic within the same channel contained in a certain number of components (kubernetes with load balancing). Docker is suitable for small businesses due to its greater simplicity and ease of use, although there is more overhead around setting how components must communicate, while Kubernetes is better for treating components as simple resources designed for larger, more complex environments, more scalable, more incremental, and easier to maintain architectures. We also benefit from features provided by the Kubernetes API that we can use to obtain information about concrete resources that we can use to create analysis tools like the ones mentioned in our project. With **Kubernetes**, regardless of the number of components we have, we will be able to use this analysis tool without further configurations than the resource itself. The **Kubernetes** with load balancer (K8 with LB) approach was chosen. This makes sense because of all of the characteristics mentioned so far that are very suited for an infrastructure for the healthcare industry—a sector that demands a large, resilient infrastructure and advanced features to manage and monitor the network for its administrators, aligning with our primary goal.

Speaking about the platform itself, the system developed provides a robust and comprehensive solution for monitoring and managing the **HLF** network. This is only possible due to the selected solution. The system guarantees continuous access and analysis through its design architecture, where various critical statistics are consolidated. This system not only replicates but also improves on the functionality of previous solutions, allowing administrators to perform a wide range of tasks with greater efficiency and precision all in one single place, reducing errors as common network tasks are successfully automated and monitoring is provided. More concretely, the system provides more than a basic network management. It offers information about resource utilization, peer management, and orderer operations. It also provides a user-friendly interface to easily handle complex tasks such as monitoring network activity, managing resources, or adjusting settings. This is a solution that offers everything that an administrator needs to control its network.

5 Discussion

Despite there is already a discussion regarding every single use case, it is always important to create a general overview of discussion over the research, discarding specific projects or implementations. With this in mind there will be a discussion pointing out what were the results obtained through this research, a **SWOT** analysis for referring it's major aspects in a more abstract way and which research questions were answered and how they got answered. A general discussion is important, otherwise only the use cases will be considered and not the full extension of actually having the ideas of both conjured to a more general conclusion, focusing in general perspectives but also in the existent linkage between all of the factors and what is actually under research.

Putting all together

In respect to what has been covered so far, within the discussion realm, it was noticed that for the first use case discussion was not that specific. This is because results were not taken into account due to the fact that it was more theorical-oriented, which means that no results were dragged to compare and delve into. However, since the second use case's a base for the theoretical scheme of the first use case, providing it's blockchain infrastructure to store file hash representations becomes feasible to test the overall idea. As such, future work about gathering both dimensions will be covered in the appropriated section, giving the necessary insights to discuss until each measure using both ideas can be beneficial where all the mentioned architectures can come to play, and even the management of an **IPFS** node can be something that could potentially be managed by the same tool as the one covered in the **kubernetes** architecture. Regarding this aspect, it should be known that until reaching that level of robustness, a lot of work remains to be done, and overcomplicating the existing infrastructure is not a requirement. Thinking forward to see if such hypotheses are useful, minimal insights could be taken into account in case the previous tasks take too long, but that's something that should have its own place and time to succeed. Additionally, a lot more ideas and intriguing questions may surge, which makes this work even more interesting due to the fact that multiple probable scenarios may occur.

SWOT analysis

In the following figure, a **SWOT** analysis will be conducted having both use cases combined. At this stage, this analysis will be done in a more general way. However, for covering in more depth aspects presented here, the suggestion remains to go to the before-mentioned use cases and see more concrete aspects that otherwise will be covered here at a more high level:

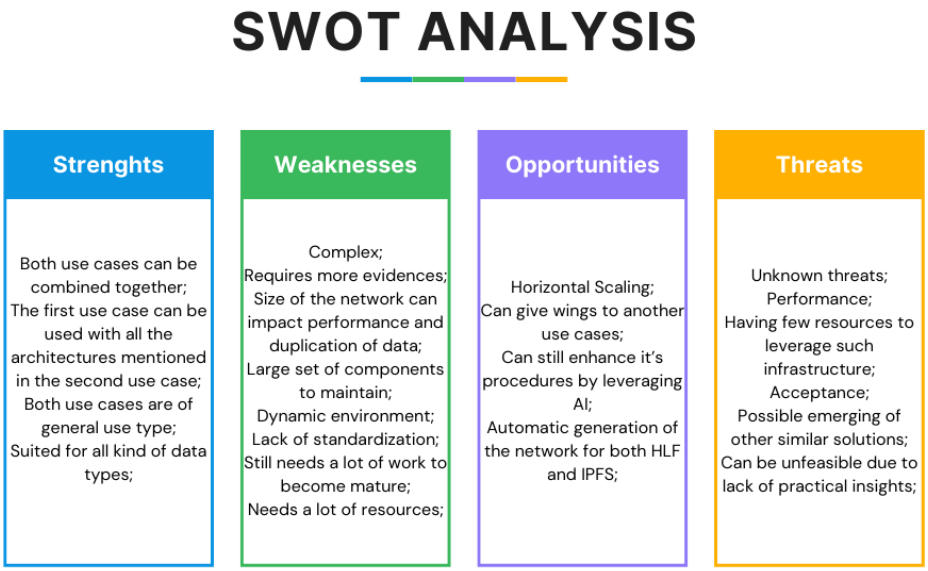

Figure 90: Discussion: SWOT analysis

Research Questions

Coming across answering how the research questions "How can a private blockchain network be utilized to improve data integrity, security, flexibility, and collaboration within the healthcare ecosystem?" and "What kind of infrastructure design is necessary to support a blockchain solution in such a vast and complex environment as healthcare?" were answered, it should be known that the first one got answered in the state of the art, covering all the major aspects of the blockchain,it its types,it its influence in the healthcare sector, benefits, implementation requirements, current implementation challenges and issues, and its implementations. The second answer, however, was answered in more depth in the second use case, presenting a solution that could be leveraged to both smaller and larger environments, focused more in an on-premise environment, specially designed for healthcare environments, though not feasible in any use case.

6 Conclusions

In the realm of abstracting what has been concluded so far, it must be known that this thesis has been exploring the application of blockchain technology, addressing challenges such as data security, data integrity, and interoperability within the healthcare sector.

This is very important due to the fact that healthcare data systems face an increase in pressure due to privacy concerns, regulatory demands, and the need for secure, real-time data sharing, where blockchain emerges as a powerful allie. This is done by introducing two main

use cases: integrating **Hyperledger Fabric** with **IPFS** for secure file storage and optimizing blockchain infrastructure with modern tools, providing to the research a comprehensive view of how blockchain capabilities can actually support the combat of such challenges.

The first use case combined **Hyperledger Fabric** along side **IPFS**, offering significant improvements in data immutability and security, where it ensures that sensitive healthcare data can be stored securely by leveraging the distributed methodology of both technologies, while making sure that easy retrieval and verification using each data hash representation. In contrast to traditional centralized databases, this innovative blockchain approach can reduce vulnerabilities to cyberattacks, enhance auditability, and even forbid malicious usage of patient data. Basically, with both technologies combined, this approach could support the critical need for data integrity in healthcare, ensuring that any kind of record remains unaltered and accurate across time within the healthcare sphere. This use case was only theorically covered, while the next has actual findings.

The second use case, however, is an infrastructure that the first use case needs and is the focus of this work. Within this category, various configurations of blockchain network architectures were tested, giving insights about performance, scalability, and operability. By the effect of such, views about the customization and deployment of blockchain networks were put into practice, where organizational needs like resource allocation, data control, and security requirements were major concerns. Deploying blockchain with centralized tools such as **Kubernetes** demonstrated how automation over network management, enabling of fault tolerance, and enabling of scalability is possible, something that remains essential for handling high transaction volumes in large networks such as healthcare institutions. The results from the benchmarking suggested that blockchain, when implemented correctly, can achieve the necessary requirements for normal operation within an organization.

Furthermore, this research contributes to ongoing discussions around the transformative potential of blockchain in healthcare, especially achieved by presenting it as a secure, flexible, highly customizable, and interoperable system, highlighting the capability of such to support secure data exchanges across diverse healthcare entities, including hospitals, insurance companies, and regulatory bodies. With this in mind, insights about how promising this could be to achieve collaboration among stakeholders are shown, while ensuring that there is compliance with privacy standards like **GDPR** and **HIPAA**.

As these technologies get more mature, the insights of such studies could provide healthcare institutions with a foundation for further consideration in their implementations, securing data sharing, patient identity verification, and even pharmaceutical supply chain management. This work definitely underlines blockchain potential to offer healthcare stakeholders a reliable platform for managing sensitive data, thereby promoting trust, transparency, and integrity across the sector.

In summary, this work can demonstrate how blockchain technology, with its cryptographic security and decentralized design, achieves the addressing of cubersome challenges in modern healthcare data management. Leveraging the analysis of both scenarios, this study affirms the value of blockchain in building a resilient infrastructure, setting the stage for further iteration over a lot of more use cases, and aiming to create an environment where blockchain can thrive.

7 References

References

Kalaycı, İ. (2023). Anatomy of cryptocurrencies. *Uluslararası Sosyal Bilimler Akademi Dergisi*, (13), 412–427. https://doi.org/10.47994/usbad.1385275

Amarta, C. C., & Latifah, F. (2023). The influence of understanding financial literacy and community readiness on the use of central bank digital currency (cbdc). *Jurnal Ekonomi Syariah Indonesia (JESI)*, *13*(1), 45–53. https://doi.org/10.21927/jesi.2023.13(1).45-53

Hasselgren, A., Kralevska, K., Gligoroski, D., Pedersen, S. A., & Faxvaag, A. (2020). Blockchain in healthcare and health sciences—a scoping review. *International Journal of Medical Informatics*, *134*, 104040. https://doi.org/10.1016/j.ijmedinf.2019.104040

Alhadhrami, Z., Alghfeli, S., Alghfeli, M., Abedlla, J. A., & Shuaib, K. (2017). Introducing blockchains for healthcare. *2017 International Conference on Electrical and Computing Technologies and Applications (ICECTA)*, *2018-Janua*, 1–4. https://doi.org/10.1109/ICECTA.2017.8252043

Lin, I. C., & Liao, T. C. (2017). A survey of blockchain security issues and challenges. *International Journal of Network Security*, *19*(5), 653–659. https://doi.org/10.6633/IJNS.201709.19(5).01

Dinh, T. T. A., Liu, R., Zhang, M., Chen, G., Ooi, B. C., & Wang, J. (2018). Untangling blockchain: A data processing view of blockchain systems. *IEEE Transactions on Knowledge and Data Engineering*, *30*(7), 1366–1385. https://doi.org/10.1109/TKDE.2017.2781227

Hölbl, M., Kompara, M., Kamišalić, A., & Zlatolas, L. N. (2018). A systematic review of the use of blockchain in healthcare. *Symmetry*, *10*(10), 470. https://doi.org/10.3390/sym10100470

Prokofieva, M., & Miah, S. J. (2019). Blockchain in healthcare. *Australasian Journal of Information Systems*, *23*, 1–22. https://doi.org/10.3127/ajis.v23i0.2203

Kuo, T. T., Kim, H. E., & Ohno-Machado, L. (2017). Blockchain distributed ledger technologies for biomedical and health care applications. *Journal of the American Medical Informatics Association*, *24*(6), 1211–1220. https://doi.org/10.1093/jamia/ocx068

McGhin, T., Choo, K. K. R., Liu, C. Z., & He, D. (2019). Blockchain in healthcare applications: Research challenges and opportunities. *Journal of Network and Computer Applications*, *135*(February), 62–75. https://doi.org/10.1016/j.jnca.2019.02.027

Kumar, T., Ramani, V., Ahmad, I., Braeken, A., Harjula, E., & Ylianttila, M. (2018). Blockchain utilization in healthcare: Key requirements and challenges. *2018 IEEE 20th International Conference on e-Health Networking, Applications and Services (HealthCom)*, 1–7. https://doi.org/10.1109/HealthCom.2018.8531136

Azaria, A., Ekblaw, A., Vieira, T., & Lippman, A. (2016). Medrec: Using blockchain for medical data access and permission management. *2016 2nd International Conference on Open and Big Data (OBD)*, 25–30.

Bell, L., Buchanan, W. J., Cameron, J., & Lo, O. (2018). Applications of blockchain within healthcare. *Blockchain in Healthcare Today*, *1*, 1–7. https://doi.org/10.30953/bhty.v1.8

Velmovitsky, P. E., Bublitz, F. M., Fadrique, L. X., & Morita, P. P. (2021). Blockchain applications in health care and public health: Increased transparency. *JMIR Medical Informatics*, *9*(6). https://doi.org/10.2196/20713

About us - healthverity [Accessed: Jan. 25, 2024]. (n.d.). https://healthverity.com/about-us/

About - healthwizz [Accessed: Jan. 25, 2024]. (n.d.).

Bamakan, S. M. H., Moghaddam, S. G., & Manshadi, S. D. (2021). Blockchain-enabled pharmaceutical cold chain: Applications, key challenges, and future trends. *Journal of Cleaner Production*, *302*, 127021. https://doi.org/10.1016/j.jclepro.2021.127021

Shivom - crunchbase company profile & funding [Accessed: Jan. 25, 2024]. (n.d.). https://www.crunchbase.com/organization/project-shivom

Ghadge, A., Bourlakis, M., Kamble, S., & Seuring, S. (2023). Blockchain implementation in pharmaceutical supply chains: A review and conceptual framework. *International Journal of Production Research*, *61*(19), 6633–6651. https://doi.org/10.1080/00207543.2022.2125595

Chronicled [Accessed: Jan. 25, 2024]. (n.d.). https://www.chronicled.com/

Ozercan, H. I., Ileri, A. M., Ayday, E., & Alkan, C. (2018). Realizing the potential of blockchain technologies in genomics. *Genome Research*, *28*(9), 1255–1263. https://doi.org/10.1101/gr.207464.116

Abdallah, M. M. (2022). Hands-on permissioned blockchain platforms. *Proceedings of the Federated Africa and Middle East Conference on Software Engineering*, 86. https://doi.org/10.1145/3531056.3542760

Cash, M., & Bassiouni, M. (2018). Two-tier permission-ed and permission-less blockchain for secure data sharing. *2018 IEEE International Conference on Smart Cloud (SmartCloud)*, 138–144. https://doi.org/10.1109/SmartCloud.2018.00031

Bakos, Y., & Halaburda, H. (2021). Tradeoffs in permissioned vs permissionless blockchains: Trust and performance. *SSRN Electronic Journal*. https://doi.org/10.2139/SSRN.3789425

Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., & Polk, T. (2008, May). Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. https://doi.org/10.17487/RFC5280

Chandramouli, R. (2019, March). *Security strategies for microservices-based application systems* (tech. rep.). National Institute of Standards and Technology (NIST). https://doi.org/10.6028/NIST.SP.800-204

Lennon, J. (2009). *Beginning couchdb*. Apress. https://doi.org/10.1007/978-1-4302-7236-6

Lamport, L. (1998). The part-time parliament. *ACM Transactions on Computer Systems (TOCS)*, *16*(2), 133–169. https://doi.org/10.1145/279227.279229

Lamport, L. (2001). Paxos made simple. *Proceedings of the 21st Annual ACM Symposium on Principles of Distributed Computing (PODC)*, 55–58. https://doi.org/10.1145/383962.384045

Corbett, J. C., Dean, J., Epstein, M., Fikes, A., Frost, C., Furman, J. J., Ghemawat, S., Gubarev, A., Heiser, C., Hochschild, P., Hsieh, W., Kanthak, S., Kogan, E., Li, H., Lloyd, A., Melnik, S., MWAO, D., Nagle, D., Quinlan, S., ... Wang, R. (2013). Spanner: Google's globally-distributed database. *Proceedings of the 10th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, 251–264. https://www.usenix.org/system/files/conference/osdi12/osdi12-final-16.pdf

Hunt, P., Konar, M., Junqueira, F. P., & Reed, B. (2010). Zookeeper: Wait-free coordination for internet-scale systems. *Proceedings of the 2010 USENIX Annual Technical Conference (USENIX ATC)*, 145–158. https://www.usenix.org/conference/usenix-atc-10/zookeeper-wait-free-coordination-internet-scale-systems

Sivasubramanian, S. (2012). Amazon dynamodb: A seamlessly scalable non-relational database service. *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*, 729–730. https://doi.org/10.1145/2213836.2213945

Nakamoto, S. (2008). Bitcoin: A peer-to-peer electronic cash system [Accessed: 2024-09-27].

Vukolić, M. (2016). The quest for scalable blockchain fabric: Proof-of-work vs. bft replication. *International Workshop on Open Problems in Network Security*, 112–125. https://doi.org/10.1007/978-3-319-39028-4_9

Pandey, S., Kamarajugadda, S. K., & Mangal, A. (2018). Hybrid consensus algorithms for permissioned blockchains: A proof-of-work and bft approach. *International Journal of*

Computer Science and Information Security (IJCSIS), *16*(4), 43–51. https://sites.google.com/site/ijcsis/vol-16-no-4-apr-2018

Wang, G., Koshy, J., Subramanian, S., Paramasivam, K., Zadeh, M., Narkhede, N., Rao, J., Kreps, J., & Stein, J. (2015). Building a replicated logging system with apache kafka. *Proc. VLDB Endow.*, *8*(12), 1654–1655. https://doi.org/10.14778/2824032.2824063

Ongaro, D., & Ousterhout, J. (2014). In search of an understandable consensus algorithm. *2014 USENIX Annual Technical Conference (USENIX ATC 14)*, 305–319. https://www.usenix.org/conference/atc14/technical-sessions/presentation/ongaro

Nalawala, H. S., Shah, J., Agrawal, S., & Oza, P. (2022). A comprehensive study of "etcd"—an open-source distributed key-value store with relevant distributed databases. *Emerging Technologies for Computing, Communication and Smart Cities*, *875*, 481–489. https://doi.org/10.1007/978-981-19-0284-0_35

HashiCorp. (2024). Consul documentation [Accessed: 2024-09-27]. https://www.consul.io/docs

Castro, M., & Liskov, B. (1999). Practical byzantine fault tolerance. *Proceedings of the Third Symposium on Operating Systems Design and Implementation (OSDI)*, 173–186. https://pmg.csail.mit.edu/papers/osdi99.pdf

Congress, U. (1996). *Health insurance portability and accountability act of 1996* [Accessed: 2024-09-27]. U.S. Department of Health and Human Services (HHS). https://www.hhs.gov/hipaa/for-professionals/index.html

Parliament, E., & of the European Union, C. (2016). *General data protection regulation (gdpr) (eu) 2016/679* [Accessed: 2024-09-27]. https://eur-lex.europa.eu/eli/reg/2016/679/oj

Brotsis, S., Kolokotronis, N., Limniotis, K., Bendiab, G., & Shiaeles, S. (2020). On the security and privacy of hyperledger fabric: Challenges and open issues. *2020 IEEE World Congress on Services (SERVICES)*, 197–204.

Author(s). (2020). Supporting private data on hyperledger fabric with secure multiparty computation. *Proceedings of the IEEE International Conference on Blockchain and Distributed Systems Security*. https://doi.org/10.1109/XXXXXXX

Dame, K. R. V., Bergmann, T. B., Aichouri, M., & Pantoja, M. (2022). A comparative study of consensus algorithms for distributed systems. In I. Gitler, C. J. B. Hernández, & E. Meneses (Eds.), *High performance computing (carla 2021)* (pp. 103–117, Vol. 1540). Springer, Cham. https://doi.org/10.1007/978-3-031-04209-6_9

Kustov, V., Beksaev, N., & Ravi, R. (2023). Sharding in the blockchain or divide and conquer. *2023 16th International Conference on Advanced Technologies, Systems and Services in Telecommunications (TELSIKS)*, 223–227. https://doi.org/10.1109/TELSIKS57806.2023.10316070

K, D., & Mohan, M. (2022). Sidechain: A scalable blockchain. *2022 International Conference on Applied Artificial Intelligence and Computing (ICAAIC)*, 1337–1342. https://doi.org/10.1109/ICAAIC53929.2022.9793041

Thakkar, P., Nathan, S., & Viswanathan, B. (2018). Performance benchmarking and optimizing hyperledger fabric blockchain platform. *2018 IEEE 26th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)*, 264–276. https://api.semanticscholar.org/CorpusID:44088292

Bettio, M., Bruse, F., Franke, A., Jakoby, T., & Schärf, D. (2019). Hyperledger fabric as a blockchain framework in the financial industry. In V. Liermann & C. Stegmann (Eds.), *The impact of digital transformation and fintech on the finance professional* (pp. 29–44). Springer International Publishing. https://doi.org/10.1007/978-3-030-23719-6_3

Kumar S., N., & Dakshayini, M. (2020). Secure sharing of health data using hyperledger fabric based on blockchain technology. *2020 International Conference on Mainstreaming Block Chain Implementation (ICOMBI)*, 1–5. https://doi.org/10.23919/ICOMBI48604.2020.9203442

Cai, L., Li, Q., & Liang, X. (2022). Hyperledger fabric application case studies in detail. In *Advanced blockchain technology*. Springer, Singapore. https://doi.org/10.1007/978-981-19-3596-1_9

Tan, E., Mahula, S., & Crompvoets, J. (2022). Blockchain governance in the public sector: A conceptual framework for public management. *Government Information Quarterly*, *39*(1), 101625. https://doi.org/https://doi.org/10.1016/j.giq.2021.101625

Jena, S. K., Kumar, B., Mohanty, B., Singhal, A., & Barik, R. C. (2024). An advanced blockchain-based hyperledger fabric solution for tracing fraudulent claims in the healthcare industry. *Decision Analytics Journal*, *10*, 100411. https://doi.org/https://doi.org/10.1016/j.dajour.2024.100411

Benet, J. (2014). Ipfs - content addressed, versioned, p2p file system [https://arxiv.org/abs/1407.3561]. *arXiv preprint arXiv:1407.3561*.

Stoica, I., Morris, R., Karger, D., Kaashoek, M. F., & Balakrishnan, H. (2001). Chord: A scalable peer-to-peer lookup protocol for internet applications [https://dl.acm.org/doi/10.1145/383059.383071]. *Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*, 149–160.

Haryanto, T., Ramli, K., & Pramudianto, A. D. (2023). Data availability in decentralized data storage using four-node interplanetary file system. *Jurnal Teknik Informatika (Jutif)*, *4*(3), 639–645. https://doi.org/10.52436/1.jutif.2023.4.3.1030

Zeng, R., You, J., Li, Y., & Han, R. (2022). An icn-based ipfs high-availability architecture. *Future Internet*, *14*(5). https://doi.org/10.3390/fi14050122

Wang, Q., Gong, X., Nguyen, G. T. K., Houmansadr, A., & Borisov, N. (2012). Censorspoofer: Asymmetric communication with ip spoofing for censorship-resistant web browsing. https://arxiv.org/abs/1203.1673

Benet, J., Dalrymple, D., Greco, N., Spannos, J., Angell, I., Miyazono, E., Scott, W., Chen, S., & Stern, A. (2017). Filecoin: A decentralized storage network. *arXiv preprint*, 1–16. https://filecoin.io/filecoin.pdf

Sakız, B., & Gencer, A. (2021). Blockchain beyond cryptocurrency: Non-fungible tokens [Available at: https://www.avekon.org/papers/2527.pdf]. *Conference on Economics and Finance Research (CEFR)*. https://doi.org/10.36880/c13.02527

Team, O. (2024). Opensea: The largest nft marketplace [Accessed: 2024-10-01].

Crockford, D. (2014, March). The javascript object notation (json) data interchange format [RFC 7159, Internet Engineering Task Force (IETF)]. https://tools.ietf.org/html/rfc7159

Boghosian, R. D., & Motz, R. (2023). D-seli: Proposal for a distributed educational blockchain network with ipfs. *Revista Tecnológica de Investigación e Innovación*, *4*(3), 13973. https://doi.org/10.55267/rtic/13973

Mahmoud, N., Aly, A., & Abdelkader, H. (2022). Enhancing blockchain-based ride-sharing services using ipfs. *Information Systems and Web Applications*, *5*(2), 200135. https://doi.org/10.1016/j.iswa.2022.200135

Davies, J., & Pagani, A. (2022). Ipv4 and ipv6 for blockchain networks: A comparative analysis. *Proceedings of the IEEE International Conference on Blockchain (iGET)*. https://doi.org/10.1109/iGETblockchain56591.2022.10087175

Team, I. (2015). Ipfs and ipns: Interplanetary file system and name system [Accessed: 2024-09-30]. https://docs.ipfs.io/concepts/ipns/

Huang, B., Zheng, H., Qu, X., & Xie, T. (2022). Consortium blockchain distributed storage protocol. *2022 14th International Conference on Communications, Signal Processing, and Networks (ICCSN)*. https://doi.org/10.1109/iccsn55126.2022.9817570

Jayabalan, J., & Jeyanthi, N. (2022). Scalable blockchain model using off-chain ipfs storage for healthcare data security and privacy. *Journal of Parallel and Distributed Computing*, *168*, 148–160. https://doi.org/10.1016/j.jpdc.2022.03.009

Mehbodniya, A., Neware, R., Vyas, S., Kumar, M., Ngulube, P., & Ray, S. (2021). Blockchain and ipfs integrated framework in bilevel fog-cloud network for security and privacy of iomt devices. *Computational and Mathematical Methods in Medicine*. https://doi.org/10.1155/2021/7727685

Tong, W., Yang, L., Li, Z., Jin, X., & Tan, L. (2024). Enhancing security and flexibility in the industrial internet of things: Blockchain-based data sharing and privacy protection. *Sensors*, *24*(3), 1035. https://doi.org/10.3390/s24031035

Li, Y., Yu, Y., & Wang, X. (2022). Three-tier storage framework based on tbchain and ipfs for protecting iot security and privacy. *ACM Transactions on Internet Technology (TOIT)*, *23*(2), 54–70. https://doi.org/10.1145/3549910

Briggs, D. C., & Domingue, B. (2013). The gains from vertical scaling. *Journal of Educational and Behavioral Statistics*, *38*(6), 551–576.

Yeom, Y.-J., Kim, T., Park, D.-H., & Kim, S. (2020). Horizontal pod autoscaling in kubernetes for elastic container orchestration. *Sensors*, *20*(16), 4621. https://doi.org/10.3390/s20164621

Rossi, F., Nardelli, M., & Cardellini, V. (2019). Horizontal and vertical scaling of container-based applications using reinforcement learning. *2019 IEEE 12th International Conference on Cloud Computing (CLOUD)*, 329–338.

Mohan, P., Jambhale, T., Sharma, L., Koul, S., & Koul, S. (2020). Load balancing using docker and kubernetes: A comparative study. *International Journal of Recent Technology and Engineering (IJRTE)*, *8*(6), 3938–3942. https://doi.org/10.35940/ijrte.b3938.079220

Ul Haque, M., Kholoosi, M. M., & Babar, M. A. (2022). Kgsecconfig: A knowledge graph based approach for secured container orchestrator configuration. *2022 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER)*, 420–431. https://doi.org/10.1109/SANER53432.2022.00057

Callaghan, B. (1999). *Nfs illustrated*. Addison-Wesley.

Curtis, J. A., & Eisty, N. U. (2024). The kubernetes security landscape: Ai-driven insights from developer discussions. *arXiv preprint arXiv:2409.04647*. https://arxiv.org/abs/2409.04647

Bose, D. B., Rahman, A., & Shamim, M. S. I. (2021). 'under-reported' security defects in kubernetes manifests. *2021 3rd International Conference on Inventive Research in Computing Applications (ICIRCA)*, 32–37. https://doi.org/10.1109/EnCyCriS52570.2021.00009

Bhardwaj, A. K., Dutta, P., & Chintale, P. (2024). Ai-powered anomaly detection for kubernetes security: A systematic approach to identifying threats. *British Journal of Machine Learning*, *2024*, 14. https://doi.org/10.58496/bjml/2024/014

Zerouali, A., Opdebeeck, R., & De Roover, C. (2023). Helm charts for kubernetes applications: Evolution, outdatedness and security risks. *2023 IEEE/ACM 20th International Conference on Mining Software Repositories (MSR)*, 523–527. https://doi.org/10.1109/MSR59073.2023.00078

Saxena, V., Saxena, D., & Singh, U. P. (2023). Security enhancement using image verification method to secure docker containers. *Proceedings of the 4th International Conference on Information Management & Machine Intelligence*. https://doi.org/10.1145/3590837.3590879

Yang, N., Chen, C., Yuan, T., Wang, Y., Gu, X., & Yang, D. (2022). Security hardening solution for docker container. *2022 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC)*, 252–257. https://doi.org/10.1109/CyberC55534.2022.00049

Wang, C., Zhu, S., & Li, W. (2023). Docker-based security protection system for container applications. In X. Cai & B. H. bin Ahmad (Eds.), *International conference on computer network security and software engineering (cnsse 2023)* (p. 1271406, Vol. 12714). SPIE. https://doi.org/10.1117/12.2683171

Balabanian, F., & Henriques, M. (2019). Tocker: Framework para a segurança de containers docker. *Anais Estendidos do XIX Simpósio Brasileiro de Segurança da Informação e de Sistemas Computacionais*, 145–154. https://doi.org/10.5753/sbseg_estendido.2019.14016

Chang, Y.-S., Lee, Y.-K., Juang, T., & Yen, J.-S. (2013). Cost evaluation on building and operating cloud platform. *Journal of Grid and High Performance Computing (JGHPC)*, *5*(2), 18–33. https://doi.org/10.4018/jghpc.2013040103

Gandhi, R., Liu, H. H., Hu, Y. C., Lu, G., Padhye, J., Yuan, L., & Zhang, M. (2014). Duet: Cloud scale load balancing with hardware and software. *Proceedings of the 2014 ACM Conference on SIGCOMM*, 27–38. https://doi.org/10.1145/2619239.2626317

Younes, O. (2018). Modeling and performance analysis of a new secure address resolution protocol. *International Journal of Communication Systems*, *31*(2). https://doi.org/10.1002/dac.3433

Zamany, M. A. Z. (2022). Optimalisasi routing menggunakan satu autonomous system number (asn) border gateway protocol (bgp). *Jurnal Ekonomi dan Informatika*, *2*(1). https://doi.org/10.56486/jeis.vol2no1.158

Musril, H. A. (2017). Simulasi interkoneksi antara autonomous system (as) menggunakan border gateway protocol (bgp). *Jurnal Informasi dan Teknologi Jaringan*, *2*(1). https://doi.org/10.30743/INFOTEKJAR.V2I1.151

Kanai, K., Tsuda, T., Nakazato, H., & Katto, J. (2022). Information-centric service mesh for autonomous in-network computing. *Proceedings of the 9th ACM Conference on Information-Centric Networking*, 159–161. https://doi.org/10.1145/3517212.3559481

Li, M., Lu, W., Lin, H., Wu, J., Zhang, Y., & Yan, G. (2023). Flatproxy: A dpu-centric service mesh architecture for hyperscale cloud-native application. *arXiv preprint*. https://doi.org/10.48550/arXiv.2312.01297

Sharma, R., & Singh, A. (2019). *Getting started with istio service mesh: Manage microservices in kubernetes*. Apress. https://doi.org/10.1007/978-1-4842-5458-5

Zhu, X., She, G., Xue, B., Zhang, Y., Zhang, Y., Zou, X. K., Duan, X., He, P., Krishnamurthy, A., Lentz, M., Zhuo, D., & Mahajan, R. (2023). Dissecting overheads of service mesh sidecars. *Proceedings of the 2023 ACM Symposium on Cloud Computing*, 142–157. https://doi.org/10.1145/3620678.3624652

Hussain, F., Li, W., Noye, B., Sharieh, S., & Ferworn, A. (2019). Intelligent service mesh framework for api security and management. *2019 IEEE 10th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*, 0735–0742. https://doi.org/10.1109/IEMCON.2019.8936216

Miraj, M., & Fajar, A. (n.d.). Model-based resilience pattern analysis for fault tolerance in reactive microservice. *International Journal of Advanced Computer Science and Applications*.

Sedghpour, M. R. S., Klein, C., & Tordsson, J. (2022). An empirical study of service mesh traffic management policies for microservices. *ACM Journal*. https://doi.org/10.1145/3489525.3511686

Sedghpour, M. R. S., Garlan, D., Schmerl, B., Klein, C., & Tordsson, J. (2023). Breaking the vicious circle: Self-adaptive microservice circuit breaking and retry. *Proceedings of the IEEE International Conference on Cloud Engineering (IC2E)*. https://doi.org/10.1109/IC2E59103.2023.00012

Hyperledger Foundation. (2023a). *Hyperledger caliper benchmarking tool* [Available at: https://github.com/hyperledger/caliper]. Hyperledger Foundation.

Hyperledger Foundation. (2023b). *Hyperledger besu documentation* [Available at: https://besu.hyperledger.org/]. Hyperledger Foundation.

FISCO BCOS Project Team. (2023). *Fisco bcos documentation* [Available at: https://fisco-bcos-documentation.readthedocs.io/].

Hammer-Lahav, E., Hardt, D., Richer, J., Bradley, J., Jones, M. B., & Mills, B. (2012). *The oauth 2.0 authorization framework* [RFC 6749]. Internet Engineering Task Force (IETF). https://tools.ietf.org/html/rfc6749

Sakimura, N., Bradley, J., Jones, M. B., de Medeiros, B., & Mortimore, C. (2014). *Openid connect core 1.0 incorporating errata set 1* [Final Specification]. OpenID Foundation. https://openid.net/specs/openid-connect-core-1_0.html

Jones, M. B., Bradley, J., & Sakimura, N. (2012). Json web token (jwt) [RFC 7519]. *Proceedings of the Internet Engineering Task Force (IETF)*. https://tools.ietf.org/html/rfc7519

Boonkrong, S. (2021). Multi-factor authentication. In *Authentication and access control* (pp. 35–50). Apress, Berkeley, CA. https://doi.org/10.1007/978-1-4842-6570-3_6

Koleoso, T. (2020). *Beginning quarkus framework: Build cloud-native enterprise java applications and microservices*. Apress. https://doi.org/10.1007/978-1-4842-6032-6

Lamouchi, N. (2021). *Pro java microservices with quarkus and kubernetes: A hands-on guide*. Apress. https://doi.org/10.1007/978-1-4842-7170-4

Hartig, O., & Pérez, J. (2018). Semantics and complexity of graphql. *Proceedings of the World Wide Web Conference (WWW)*, 1155–1164. https://doi.org/10.1145/3178876.3186014

Katamreddy, S. P. R., & Upadhyayula, S. S. (2023). Graphql with spring boot. In *Beginning spring boot 3* (pp. 325–339). Apress. https://doi.org/10.1007/978-1-4842-8792-7_15

Kim, Y. W., Consens, M. P., & Hartig, O. (2019). An empirical analysis of graphql api schemas in open code repositories and package registries. *Proceedings of the 13th Alberto Mendelzon International Workshop on Foundations of Data Management (AMW)*. https://doi.org/10.5281/zenodo.3352419

Giretti, A. (2023). *Coding clean, reliable, and safe rest apis with asp.net core 8: Develop robust minimal apis with .net 8*. Apress. https://doi.org/10.1007/978-1-4842-9979-1

Giretti, A. (2022a). Understanding the grpc specification. In *Beginning grpc with asp.net core 6: Build applications using asp.net core razor pages, angular, and best practices in .net 6* (pp. 85–102). Apress. https://doi.org/10.1007/978-1-4842-8008-9_3

Giretti, A. (2022b). *Beginning grpc with asp.net core 6: Build applications using asp.net core razor pages, angular, and best practices in .net 6*. Apress. https://doi.org/10.1007/978-1-4842-8008-9

Rowstron, A., & Druschel, P. (2001). Pastry: Scalable, decentralized object location and routing for large-scale peer-to-peer systems. *Middleware*, *2218*, 329–350.

Maymounkov, P., & Mazieres, D. (2002). Kademlia: A peer-to-peer information system based on the xor metric. *Proceedings of the 1st International Workshop on Peer-to-Peer Systems*, 53–65.

Cohen, B. (2003). Bittorrent: A peer-to-peer file sharing protocol. *Proceedings of the 2003 International Workshop on Peer-to-Peer Systems*, 1–5.

Merkle, R. (1979). Protocols for public key cryptosystems. *IEEE Symposium on Security and Privacy*, 122–134.

Torvalds, L., & Hamano, J. (2005). Git: A version control system [Accessed: 2024-10-29]. https://git-scm.com

Baker, M., & et al. (1997). Self-certifying file systems. *Proceedings of the 1997 USENIX Annual Technical Conference*, 69–84.

WebRTC Working Group. (2011). Webrtc: Real-time communication between browsers [Accessed: 2024-10-29]. https://webrtc.org

Rudberg, R., et al. (2011). Utp: A transport protocol for p2p applications. *Proceedings of the 2011 ACM SIGCOMM Workshop on Networked Systems for Developing Regions*.

Floyd, S., & et al. (2000). The stream control transmission protocol. *IEEE/ACM Transactions on Networking*, *8*(5), 745–757.

Rosenberg, J., & et al. (2020, July). *Interactive connectivity establishment (ice)* [RFC 5245]. https://tools.ietf.org/html/rfc5245

Benet, J. (2015a). Bitswap: The block exchange protocol [Accessed: 2024-10-29]. *InterPlanetary Networking Concept Document*. https://github.com/ipfs/specs/tree/master/bitswap

Benet, J. (2015b). Ipns: The interplanetary naming system [Accessed: 2024-10-29]. *InterPlanetary Networking Concept Document*. https://github.com/ipfs/specs/tree/master/ipns

Cai, A. (2012). The research of software requirements analysis of core needs. *Proceedings of SPIE*, *8499*, 8499X. https://doi.org/10.1117/12.968556

Chung, L., & do Prado Leite, J. C. S. (2009). On non-functional requirements in software engineering. In *Conceptual modeling: Foundations and applications* (pp. 363–379). https://doi.org/10.1007/978-3-642-02463-4_19

8 Appendix I - Abstract of Submitted Articles

8.1- IPFS and Hyperledger Fabric: Integrity of Data in Healthcare

Abstract: Since the Information of Things (IoT) arrival, one of the main prob- lems we encounter daily is data breaches and data integrity. Now more than ever, the expertise needed to develop an attack is decreasing. Developers are creating software that does the same thing as an expert, requiring less knowledge from the attacker. Also, ill-intended professionals within the institutions can compromise and access information without authorization. Healthcare Information Technologies must be aware of and have a proactive approach to this problem within each sector. With this in mind, researchers and developers must propose and study solutions and architectures to store and query sensitive files and information more securely. When we think about security, there are more dimensions to consider other than the technology itself, but it does remove some constraints. This article presents an architecture that relies on Blockchain through HyperLedger Fabric and Interplanetary File System (IPFS) to securely host sensitive documents such as contracts within Healthcare.

Keywords: Blockchain;Healthcare Industry;Data Integrity;Ipfs;Hyper Ledger Fabric;Kubernetes;Security Blockchain;Web3;Linux Foundation;

8.2- Scalable and Sustainable Blockchain: Architecting Infrastructure and Developing a Platform for Efficient Management and Exploration

Abstract: The application of blockchain technology in the health and healthcare sector is considered disruptive, thus requiring trust, integrity, and value of data. In addition, this document analyses the potential of permissioned blockchain, more specifically Hyperledger Fabric, to improve the security, integrity, and efficiency of healthcare systems. It examines ways in which this can be done in a manner that has both scalability and sustainability prospects in the long-term. This, by reaching the optimal compromise between efficiency of resources and its ease of use, management and upscaling. The ability to be as lightweight as possible, while being able to quickly expand the infrastructure and seamlessly integrate new functionalities, maintaining operational efficiency. The research developed involves a review of the literature, the development of three different blockchain implementations/architectures where the practical assessment of performance metrics is performed. Finally, a blockchain management platform is presented. This was developed to ensure long-term usability and maintenance of blockchain solutions in the healthcare industry. This work aims to advance the application of blockchain in healthcare, addressing both immediate and long-term needs for security and efficiency.

Keywords: Blockchain; Blockchain in Healthcare; Healthcare Systems; Hyperledger Fabric, Kubernetes;