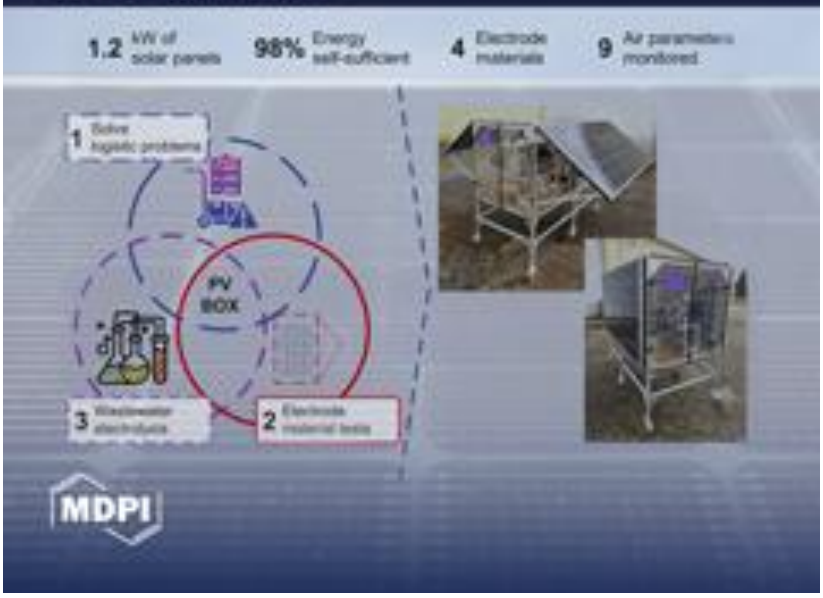
technologies

Impact Factor 4.2
CiteScore 6.7

ISSN 2227-7080

Portable Solar-Powered Open-Source Laboratory for Wastewater Electrolysis Tests

Volume 13 · Issue 2 February 2025



MDPI

Outlook

[Technologies] Manuscript ID: technologies-3387410 - Promotion Tips

From liping.yu@mdpi.com <liping.yu@mdpi.com>
on behalf of
technologies@mdpi.com <technologies@mdpi.com>

Date Sat 2/1/2025 10:02 AM

To MINANGO NEGRETE JUAN CARLOS <juancarlos.minango@ister.edu.ec>; henry.carvajal@udla.edu.ec <henry.carvajal@udla.edu.ec>; Marcelo Zambrano <marcelo.zambrano@ister.edu.ec>; nathaly.orozco@udla.edu.ec <nathaly.orozco@udla.edu.ec>; frapecar@upvnet.upv.es <frapecar@upvnet.upv.es>

Cc technologies@mdpi.com <technologies@mdpi.com>; daisy.dai@mdpi.com <daisy.dai@mdpi.com>

Dear Authors,

Congratulations! Your manuscript has been published open access in Technologies.

Thank you for choosing our journal. Your article is now publicly and freely available on the journal website. As part of the post-publication process, we encourage you to promote and share your work to increase its visibility, impact, and citations.

Please find the Publication Certificate and a banner for promoting your manuscript here (please note, this is only available to corresponding authors when logged into their account):
https://susy.mdpi.com/user/manuscripts/review_info/087d473f521239020e71921b4fe187e6

To promote the discoverability, views, and downloads of your manuscript you can promote your manuscript in the following ways:

(1) Social Media
- Share your article on various social networks, such as Researchgate, Sciprofiles Facebook, LinkedIn, Mendeley, and Twitter by clicking the share button on the webpage of your article. Please do not forget to tag us, @MDPIOpenAccess so that we can re-tweet your work to further increase its visibility.
- Send a short text (up to 200 characters) to the MDPI assistant editor who was your contact for the publication of your manuscript to post about your manuscript on the journal's Twitter account.
- Write a blog post to explain the meaning and possible outcomes of your Research.
- Ask your institution or society to post your paper on their social media accounts and to include a story about your paper in their newsletters.

(2) Link Share

- Share the article link directly with colleagues and peers in your field.
- Add a link to your article in your email signature.
- Update your personal and institutional websites by adding the title of your article and a link to it.

(3) Academic Research-sharing Platforms
- Set up your profile on academic research-sharing platforms, such as SciProfiles, ResearchGate, Academia.edu, or Google Scholar, and add a summary of your article.
- Register an ORCID author identifier and add the article information to your profile.
- Deposit your article in repositories (such as those run by your university) to make your research more discoverable.

(4) Conferences
- Showcase your publication at conferences in the form of a presentation or a poster. You can find conferences hosted by MDPI to participate in at Sciforum.net.

For additional tips to promote your article please see here:
https://www.mdpi.com/authors/promoting

Please keep up with any journal updates and news via our twitter account (https://twitter.com/Technologies_OA)

Find out more about MDPI on our website: https://www.mdpi.com/about.

Please keep in touch!

Kind regards,
Technologies Editorial Office
Postfach, CH-4020 Basel, Switzerland
Office: Grosspeteranlage 5, CH-4052 Basel
Tel. +41 61 683 77 34 (office)
E-mail: technologies@mdpi.com
https://www.mdpi.com/journal/technologies/

---

technologies

MDPI

Article

Distributed Ledger Technology in Healthcare: Enhancing Governance and Performance in a Decentralized Ecosystem

Juan Minango [1], Henry Carvajal Mora [2], Marcelo Zambrano [1], Nathaly Orozco Garzón [2,*] and Francisco Pérez [3,4]

[1] Departamento de Investigación, Instituto Tecnológico Superior Rumiñahui, Sangolquí 171103, Ecuador; juancarlos.minango@ister.edu.ec (J.M.); marcelo.zambrano@ister.edu.ec (M.Z.)
[2] Faculty of Engineering and Applied Sciences, Telecommunications Engineering, ETEL Research Group, Universidad de Las Américas (UDLA), Quito 170503, Ecuador; henry.carvajal@udla.edu.ec
[3] SATRD Lab, Universitat Politècnica de València (UPV), 46022 Valencia, Spain; frapecar@upvnet.upv.es or fperez@favit.es
[4] FAVIT, 46013 Valencia, Spain
* Correspondence: nathaly.orozco@udla.edu.ec

Abstract: This paper evaluates the technical feasibility of Distributed Ledger Technology (DLT) within the healthcare ecosystem, with a focus on the use of Corda DLT to enhance governance and performance in a decentralized ecosystem, ensuring data integrity, security, and trustworthiness. Key attributes examined include the guarantee of data integrity, ensuring that transmitted data remain unaltered; authenticity through the implementation of digital signatures and certificates; confidentiality achieved via secure peer-to-peer communication accessible only to authorized parties; and traceability and auditing mechanisms that enable tracking of information changes and accountability. To validate these features, a Corda Distributed Application (CorDapp) was developed to manage the core logic of the healthcare ecosystem. The CorDapp was deployed across nodes and executed within the Corda network. Its performance was assessed using metrics such as throughput, latency, CPU usage, and memory consumption in both local and cloud network environments. Results demonstrate the feasibility of using Corda DLT technology in healthcare, effectively addressing critical requirements such as integrity, authenticity, confidentiality, traceability, and auditing while maintaining satisfactory performance across diverse deployment scenarios.

Keywords: distributed ledger technology; blockchain; peer-to-peer; throughput; decentralized networks

Academic Editor: Luc de Witte

Received: 8 December 2024
Revised: 3 January 2025
Accepted: 5 January 2025
Published: 1 February 2025

Citation: Minango, J.; Carvajal Mora, H.; Zambrano, M.; Orozco Garzón, N.; Pérez, F. Distributed Ledger Technology in Healthcare: Enhancing Governance and Performance in a Decentralized Ecosystem. *Technologies* **2025**, *13*, 58. https://doi.org/10.3390/technologies13020058

1. Introduction

The concept of blockchain was first introduced in a 2008 white paper by the pseudonymous person or group known as "Satoshi Nakamoto" [1]. In that paper, Nakamoto described a decentralized electronic cash system called Bitcoin, which used a blockchain to record and verify transactions. The first blockchain-based cryptocurrency, Bitcoin, was launched in 2009. It used a decentralized network of computers to maintain a shared ledger of transactions, with the blockchain serving as a secure and transparent record of all transactions on the network. Since the launch of Bitcoin, blockchain technology has evolved and been adapted for various applications beyond cryptocurrency.

Blockchain has the potential to revolutionize a wide range of industries, including finance, healthcare, and supply chain management, by enhancing security, efficiency, and transparency [2–4]. Moreover, its social impact is equally significant, as it can foster

trust and transparency in untrusted environments and promote a sense of community, as highlighted in a study on blockchain's role in social good [5]. Since its introduction, blockchain technology has experienced rapid innovation and adoption, leading to diverse networks such as public, private, and consortium blockchains, each suited for specific use cases [6,7].

Notably, the immutability and security features of blockchain make it particularly suitable for applications in interconnected systems like the Internet of Things (IoT). Recent advancements, such as multi-layer blockchain architectures, have demonstrated their utility in secure IoT data management [8]. These developments underscore blockchain's adaptability and its potential to address contemporary challenges in various domains.

Blockchain and Distributed Ledger Technology (DLT) are terms frequently used interchangeably, but it is important to clarify the distinction between them in this context. While blockchain is often considered a specific type of DLT, it can also refer to a collection of chained blocks that might not necessarily fall under the broader DLT category—especially in the case of private blockchains [9]. In contrast, DLT is a more general term that encompasses any technology enabling a decentralized and distributed ledger, including blockchain but also other technologies like directed acyclic graphs (DAGs) and distributed hash tables (DHTs) [10].

However, for the purposes of this paper, the focus is not on the precise technical classification of these systems but on their shared capabilities. Both blockchain and other forms of DLT serve the broader goal of providing decentralized, secure, and transparent systems for managing data, for example in the healthcare sector. The distinction between blockchain and DLT is not critical to the paper's main argument, which emphasizes the technology's capacity to enhance governance, security, and data integrity in a decentralized system.

Corda is an open-source DLT platform designed specifically for financial services. It was developed by the Corda Consortium, a group of leading financial institutions and technology companies, to create a more efficient and secure way of managing financial transactions and automating complex business processes [11]. Corda is built on the principles of privacy, security, and interoperability. It uses a DLT architecture to enable the secure and transparent exchange of data and assets between participants or nodes on the network. Corda uses smart contracts to automate complex business processes and to ensure that transactions are executed consistently and predictably. Further, Corda is designed to be highly scalable and can handle a high volume of transactions without sacrificing performance or reliability [12]. Although the theoretical foundation of Corda DLT has advanced significantly, research on its feasibility within the healthcare ecosystem remains limited [13].

Existing studies point out certain limitations [14–16], such as challenges in ensuring information confidentiality and the overall maturity of blockchain systems tailored to healthcare. While many blockchain technologies prioritize data immutability and transparency, healthcare systems require more sophisticated mechanisms to address regulatory and ethical obligations [17]. For instance, laws such as the General Data Protection Regulation (GDPR) in the European Union [18], the Health Insurance Portability and Accountability Act (HIPAA) in the United States [19], and Brazil's Lei Geral de Proteção de Dados (LGPD) [20] impose stringent requirements for data confidentiality, controlled access, and the right to erase personal information. To address these demands, Corda offers a permissioned blockchain architecture that restricts access to transaction data, enabling compliance with such regulations. Unlike public blockchains like Ethereum [21] or permissioned alternatives such as Hyperledger Fabric [22], Corda incorporates unique features like notary services and point-to-point data sharing, enhancing both privacy and scalability.

Corda's design makes it particularly suitable for managing sensitive healthcare information. Compared to Hyperledger Fabric [23], which relies heavily on channels to ensure confidentiality, Corda avoids the scalability issues that arise in complex ecosystems by limiting data propagation strictly to relevant parties. Additionally, Corda's flexibility in creating custom smart contracts and its compatibility with existing healthcare systems provide a distinct advantage in adapting to diverse regulatory frameworks across different regions. This adaptability ensures Corda can meet the specific legal and operational requirements of the healthcare sector more effectively than other blockchain solutions.

Thus, this article addresses this gap by conducting a comprehensive study, evaluation, and analysis of Corda's technological feasibility for implementation in the healthcare sector. Moreover, this work addresses aspects related to the architectural structure of an application based on Corda DLT, as well as a programming paradigm. As a result, a Corda distributed application (CorDapp (CorDapps are applications that run on the Corda platform. CorDapps are built using the Corda programming languages (Java and Kotlin) and can be used to manage and transfer assets, automate complex business processes, and interact with other Corda nodes. Further, CorDapps are used to implement the logic and business logic of the Corda network and can be customized to meet the specific needs of different service applications and use cases.)) is developed for the healthcare area where aspects such as integrity, authenticity, confidentiality, traceability, and auditability through obtaining performance metrics are evaluated and analyzed. This CorDapp is designed to facilitate the secure exchange of sensitive medical information among key stakeholders, including doctors, patients, caregivers, and family members. By improving communication, it aims to reduce misunderstandings and ensure that all parties are accurately informed. The application creates formal records of procedures, events, and occurrences, thereby enhancing treatment adherence and continuity of care. It also centralizes the storage of medical data, generating comprehensive reports in one location, which can lead to more accurate diagnoses and better overall patient care. This is particularly beneficial when multiple specialties are involved, allowing doctors to spend more time directly interacting with patients. By integrating all components of the healthcare ecosystem, the Corda application streamlines the management of the medical care cycle, reducing the time required for care, diagnosis, and treatment.

Contributions

This work contributes to the integration of Distributed Ledger Technology (DLT) within the healthcare ecosystem in the following ways:

- **Development of a healthcare-specific CorDapp:** A Corda Distributed Application (CorDapp) was designed and developed specifically for managing healthcare information. This CorDapp addresses the specific needs and challenges of the healthcare sector.
- **Comprehensive evaluation of CorDapp attributes:** the effectiveness of the CorDapp in ensuring data integrity, authenticity, confidentiality, traceability, and auditability was analyzed.
- **Performance evaluation:** a detailed performance evaluation of the CorDapp was conducted, evaluating metrics such throughput, latency, CPU usage, and memory consumption in both local and cloud network environments.
- **Enhanced communication and data handling:** the CorDapp facilitates secure exchanges of sensitive medical information among stakeholders, which improves communication and mitigates misunderstandings within the healthcare ecosystem.

- **Improved patient care:** by centralizing and consolidating medical data storage and generating comprehensive reports, the CorDapp contributes to better treatment adherence, continuity of care, and diagnostic precision.
- **Architectural and programming insights:** insights into the architectural design and programming paradigm of the CorDapp offer valuable information for similar applications in different sectors.

The remainder of this work is organized as follows: in Section 2, a state-of-the-art review of blockchain applications in healthcare is presented. Next, Section 3 covers the fundamental concepts of Corda. Section 4 details the structure and development of the CorDapp within the healthcare ecosystem. Section 5 describes the testing methodology, while Section 6 analyzes the performance test results of the Corda network. Finally, Section 7 concludes the paper.

2. Literature Review

This section provides a comprehensive review of the existing literature, highlighting key developments and research trends relevant to this work.

Initiatives for transmitting and storing medical information are varied and have been extensively studied. One common approach is the centralization of medical data within non-distributed systems, where security must be concentrated at a single point to manage and protect all medical information transmitted across the network [24,25]. There are both public and private initiatives that seek to achieve this objective [26]. However, the user or patient has no guarantees regarding the security of their medical information, and in most cases, it is difficult to prove its origin [27].

Some initiatives have sought to use DLT [28–30] and blockchain [31–34] technologies to carry out communication between health companies and the government. Other initiatives already have greater maturity on the subject. Thus, in 2019, Deloitte published a framework for the use of blockchain (see [35]), which can be used in the health market to solve problems related to the per capita cost of medical care, improve the health of the population, and offer a better experience for the patient [36,37]. However, within the blockchain ecosystem, these models are increasingly considered outdated and face significant challenges concerning information confidentiality [38]. Additionally, many of these initiatives are still in the early stages of experimentation, highlighting the need for a more thorough assessment of their maturity and effectiveness.

On the other hand, the development of integration protocols for medical systems has progressed, with HL7 [39] emerging as the most commercially valuable medical protocol today. HL7 defines standards and domains for use in healthcare communication, providing a robust solution for interoperability challenges. However, despite its effectiveness in addressing these issues, HL7 alone does not fully ensure the confidentiality of the medical data being transmitted.

Finally, blockchain and DLT have the potential to revolutionize the healthcare industry by improving the security, efficiency, and transparency of healthcare-related processes. As follows, some initiatives to use these technologies are described that could be used in the healthcare industry:

1. **Medical records' management**: Blockchain and DLT can be used to securely store and manage electronic medical records (EMRs). The decentralized nature and cryptographic security of these technologies make them well suited for storing sensitive medical data.
2. **Clinical trial management**: blockchain and DLT can be used to track and verify the progress of clinical trials, ensuring that data are accurately recorded and that the trials are conducted ethically.

3. **Supply chain management**: blockchain and DLT can be used to track the movement of pharmaceuticals and other medical supplies through the supply chain, ensuring that they are not tampered with and that they are delivered to the intended recipient.
4. **Medical research**: blockchain and DLT can be used to securely store and share data from medical research, enabling researchers to collaborate more effectively and accelerate the pace of discovery.
5. **Telemedicine**: blockchain and DLT can be used to securely store and access medical data for telemedicine consultations, enabling healthcare providers to deliver care remotely.

Despite the progress in utilizing blockchain and DLT technologies in healthcare, significant gaps remain in the existing literature. Current models and protocols, while addressing interoperability and some security concerns, often fail to fully ensure the confidentiality of medical data. Additionally, many initiatives are still in their experimental stages, with limited evaluation of their technological maturity and real-world applicability. This is where our work contributes by not only evaluating the technological feasibility of Corda DLT within the healthcare sector but also by addressing specific architectural and programming aspects through the development of a CorDapp.

3. Corda Basic Concepts

This section explores the fundamental components of Corda, including its network architecture, states, transactions, smart contracts, flows, and notaries.

As a private distributed ledger technology (DLT), Corda incorporates a workflow messaging network, facilitating the creation of decentralized applications referred to as CorDapps [40]. These CorDapps are developed for specific purposes to incorporate the functionalities into the DLT system. In general, blockchain is just one type of DLT.

Corda's main goal was to help regulated financial institutions with interoperability processes [40]. Corda also provides dependable scalability, state consistency, transaction privacy, and workflow adaptability, making it suitable for various business applications, including trade finance, supply chain management, healthcare, capital markets, insurance, telecommunications, and governance [41]. Corda's architecture follows a "need-to-know" principle. For instance, if node A and node B execute a transaction, node C remains unaware. The Corda notary validates the transaction, and if approved, an immutable record is created in the separate databases of nodes A and B, reflecting Corda's decentralized database structure [40], so in this way, the immutable record of the transaction carried out is accessible only to the nodes involved, that is, nodes A and B.

3.1. State

In Corda, a state represents a unit of shared information on the ledger. A state consists of data and a set of rules governing the data, which are known as a smart contract. In general, states are used to track the ownership and transfer of assets on the Corda ledger. For instance, if we want to track the ownership of a financial asset, we can create a state that represents the asset and the current owner. When the ownership of the asset changes, we can create a new state that reflects the new ownership.

Transactions in Corda create and update states. When a transaction is proposed, the nodes on the network verify that it is valid and, if it is, update the ledger to reflect the changes specified in the transaction.

States are stored in a node's vault, which is a database of states that the node is aware of. The vault is used to track the history of states and to enable the node to access the current state of an asset. Further, states are identified by a unique identifier called a linear identifier, which is assigned by the smart contract when the state is created. The linear

identifier is used to track the history of a state and to ensure that a state is not created or updated multiple times.

States can be shared with other nodes on the network by including them as inputs or outputs in a transaction. When a state is included as an input in a transaction, it is consumed and can no longer be used in future transactions. On the other hand, when a state is included as an output in a transaction, it is created and becomes available for use in future transactions.

3.2. Transaction

A transaction is a unit of work that updates the ledger. It consists of one or more inputs (states that are being consumed or modified by the transaction), outputs (new states that are being created by the transaction), and commands (specifying the actions that are being taken by the transaction, such as issuing a new asset or transferring an existing asset).

Transactions are proposed by nodes on the Corda network and are verified by the other nodes on the network before they are committed to the ledger. When a transaction is proposed, the nodes on the network verify that the transaction is valid, which includes checking that the input states are valid and have not been modified and that the output states are valid and comply with the rules specified in the smart contract. Thus, if the transaction is valid, the nodes on the network update the ledger to reflect the changes specified in the transaction, otherwise, if the transaction is not valid, it is rejected, and the ledger is not updated.

Transactions in Corda are signed by the parties involved to ensure their authenticity and prevent tampering. The signatures verify the parties' identities and ensure that the transaction has not been modified. Further, transactions in Corda are recorded on the ledger in an append-only manner, which means that once a transaction is committed to the ledger, it cannot be modified or deleted. This ensures the integrity and immutability of the ledger.

3.3. Smart Contract

In Corda, a smart contract is a set of rules that governs the data in a state. A smart contract defines the conditions that must be satisfied for a state to be valid and the actions that can be taken with the state. For example, consider a smart contract that governs the ownership and transfer of a financial asset. The smart contract might define a rule that the asset can only be transferred to a new owner if the current owner has signed the transaction. Thus, the smart contract would implement this rule by checking that the signature of the current owner is present in the transaction.

Further, smart contracts are used to enforce the rules of the ledger and ensure the integrity of the data on the ledger. When a transaction is proposed, the nodes on the network verify that it is valid and complies with the rules specified in the smart contract. If the transaction is not valid, it is rejected, and the ledger is not updated.

3.4. Flow

A flow is a piece of code that represents a business process or communication between two or more nodes on the network. Flows coordinate the execution of a task or a series of tasks on the Corda ledger. For instance, imagine a flow that coordinates the transfer of a financial asset between two nodes. This flow could involve steps such as verifying the validity of the asset, requesting the current owner's signature, creating a transaction to transfer the asset, and sending the transaction to the new owner for validation.

Further, flows are used to automate the execution of business processes and to coordinate the communication between nodes on the network. They are an important mechanism for automating the execution of tasks on the Corda ledger and for enabling the nodes

on the network to collaborate and achieve a common goal. Finally, flows are executed asynchronously and can be invoked by other flows or by external clients.

3.5. Node

A node, also referred to as a party in Corda, is an entity on the network that maintains a vault, which is essentially a database where states or data are stored. When a transaction is created, it is not immediately shared with all nodes on the network. Instead, it is first validated and signed by the relevant parties involved in the transaction. After this validation, the transaction is shared with other nodes only if they are participants in that transaction. This is a key distinction from traditional blockchain networks, where all transactions are broadcast to every node in the network.

This p2p model of data sharing distinguishes Corda from traditional blockchains. In Corda, transactions are not immediately broadcast across the entire network; instead, they are shared only with the nodes that have a direct stake or involvement in the transaction. Each node maintains a vault, which stores the states relevant to the transactions in which it is involved. This selective sharing of transaction data, combined with the vault's role in securely storing and managing state data, ensures greater privacy, scalability, and efficiency. These features are crucial for use cases like finance, healthcare, and enterprise applications. The ledger in Corda is distributed across nodes, but each node only stores and shares data that are directly relevant to its participants, preserving confidentiality and improving performance.

3.6. Notary

In Corda, a notary is a node on the network that provides a trusted service to ensure the uniqueness of states on the ledger. The notary is responsible for preventing double-spending of assets and ensuring the ledger's integrity. Thus, when a transaction is proposed, the notary checks the inputs of the transaction to ensure that they are unique and have not been used in a previous transaction. If the inputs are unique, the notary signs the transaction to confirm their uniqueness, allowing the transaction to be committed to the ledger.

Moreover, the notary acts as a gatekeeper for the ledger, ensuring that only valid and unique transactions are committed to it. This helps prevent fraud and maintain the integrity and consistency of the data on the ledger.

A critical aspect of Corda's notary system is that it ensures that states are unique and not double-spent across transactions. Unlike public blockchains, where the entire ledger is accessible to all nodes, Corda follows a more privacy-preserving approach. In this model, transaction facts (or states) are shared only among the nodes directly involved in the transaction. This selective data sharing ensures confidentiality, while the notary still performs its role of validating and securing the ledger's integrity.

4. Design and Development of CorDapp in the Health Ecosystem

This section explores how Corda DLT technology can address the challenges of managing and transferring medical records, proposing a common platform for secure and efficient data sharing across healthcare entities.

In the healthcare ecosystem, medical record systems vary widely, encompassing both paper-based and electronic formats, each with its own standards for data representation and sharing [25]. However, this vast amount of critical medical information is often scattered across multiple entities, including hospitals, clinics, doctors' offices, laboratories, pharmacies, and patients themselves. Unfortunately, this information is frequently inaccessible when it is most needed, leading to significant costs and, in some cases, even loss of life. Consequently, the portability and interoperability of medical data present major challenges,

as data transfers between different entities or stakeholders often fail. Additionally, medical data are frequently lost by both patients and healthcare providers.

It is often necessary to transfer the medical data of a patient from one hospital or clinic to another for better care, so the question arises as to how medical data can be transmitted safely, with the patient's permission, without having to regenerate them and without the risk of losing information? To answer this question, the use of Corda DLT technology in the healthcare ecosystem is proposed in this work since Corda is a private DLT that allows transactions only between the participating nodes, which guarantees the privacy of the information.

In this way, the present proposal is to create a common platform based on Corda DLT where hospitals, clinics, offices, laboratories, pharmacies, and research institutes can upload and share patient data, always with their authorization. Thus, in this proof-of-concept proposal for the use of Corda, a network composed of the following participants is considered:

- Hospital A;
- Hospital B;
- Hospital C.

Each participant has their own Corda node hosted in their environment, be it local or in the cloud. These nodes are integrated into the Corda network, which allows the transfer of medical data between the participating nodes; in this way, relevant information can be shared between the nodes quickly without loss of information and duplication. In addition, by inherence, the Corda network provides identity and service verification to ensure that participating nodes can operate on the network in complete safety [40]. Figure 1 shows the proposed Corda network with the participating nodes.

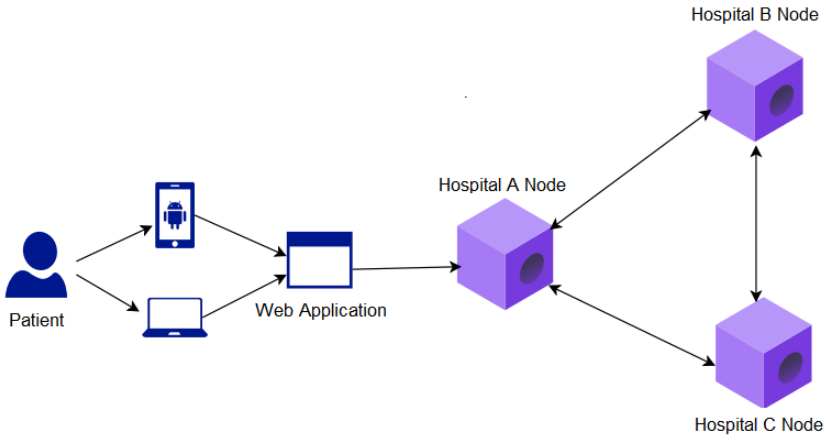


Figure 1. Proposed Corda network architecture.

During the development of CorDapps to be executed on each node within the Corda network, the design of the flows is the initial step. To illustrate this process, consider the following example, which demonstrates how a patient's flow can be established to enable communication between two or more nodes within the Corda network:

1. Patient A records their medical history at Hospital A.
2. Hospital A registers Patient A's information on the distributed ledger through its node.
3. Each time Patient A has an appointment with a doctor at Hospital A, the medical data are updated on the ledger.
4. If a doctor at Hospital A identifies a serious issue and recommends that Patient A undergo tests at the laboratory of Hospital B and later consult a specialist at Hospital C,

Hospital A, Hospital B's Laboratory, and Hospital C now have shared information, allowing them to access and update Patient A's medical history on the ledger.

5. Patient A can be directly transferred to Hospital C, where they can receive immediate care since all of their medical records are accessible through the decentralized ledger network.

Once the flow of a given patient has been defined in the Corda network, the state `MedicalRecordsState` is generated, which represents the medical history of a patient and is shared among the participating nodes of the network. The variables used in the `MedicalRecordsState` are the following:

- `patientEMR` represents the unique identifier of the patient within the node.
- `patientName` represents the name of the patient.
- `patientMother` represents the name of the patient's mother.
- `patientIdentifier` represents the patient's identity document.
- `patientData` represents the patient's medical data, for example, information about a disease.
- `receiverHospital` It is the participating node of the network that will receive the patient's information.
- `requestHospital` It is the participating node of the network that requests patient information.

Once the `MedicalRecordsState` state has been defined, the CorDapp includes the flows defined in the following subsections.

4.1. Medical Record Creation Flow

The medical record is created by the flow named `FillMedicalRecords`, which, when executed by the `receiverHospital`, creates new states of type `MedicalRecordsState` for each new patient with the variables defined previously. Thus, each node is able to save the medical history in its respective database and share it with another node through a request.

4.2. Medical Record Sharing Flow

It is represented by the flow `RequestPatientRecords`, and it is always executed by the `requestHospital` node, which requests the medical history of a given patient. Once the `RequestPatientRecords` is executed, there is a response flow named `ResponderPatientRecords`, within which the following two scenarios are possible:

1. A successful response results in the patient's medical history being shared with the`requestHospital` node. Consequently, both `receiverHospital` and `requestHospital` have the same version of the MedicalRecordsState, stored in their respective databases.
2. An unsuccessful response where an exception is thrown indicating the prohibition of sharing the state `MedicalRecordsState` of the patient.

For nodes participating in a transaction to save a patient's medical record, represented by the `MedicalRecordsState`, in their databases during the execution of flows, certain business rules must be met. These rules are essential to validate whether the consumption and sharing of the `MedicalRecordsState` are appropriate and make sense within the context of the transaction. As previously mentioned, in Corda, smart contracts govern the consumption or transition of states and represent sharing validation rules that must be executed by participating nodes [41]. In this way, the smart contract named `MedicalRecordsContract`, which governs the transition of the state `MedicalRecordsState` inside a flow, was developed. Note that the execution of the rules of the `MedicalRecordsContract` must be deterministic so that the participating nodes can reach the same conclusion.

The smart contract `MedicalRecordsContract` is composed of the definition of two commands that represent the intention of creating the `MedicalRecordsState` given by `FillMedicalRecords` and the sharing of `MedicalRecordsState` among the nodes when the `RequestMedicalRecords` flow is executed. Thus, these two commands are described below.

4.3. Create Command

Represented by `Commands.Create` and linked to the `FillMedicalRecords` flow, this process validates the creation of the `MedicalRecordsState` for the first time. As a result, a completely new `MedicalRecordsState` is generated for each patient who records their medical history for the first time. This process is illustrated in Figure 2.

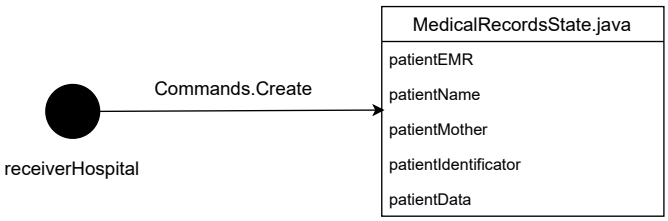


Figure 2. Create command.

4.4. Requisition Command

Represented by `Commands.Request` and associated both to flows. `RequestPatientRecords` and `ResponderPatientRecords`, this command allows one to validate whether both `receiverHospital` and `requestHospital` agree to share the state named as `MedicalRecordsState` for which both nodes must sign the transaction through their respective public keys. Once this validation is complete, the `Medical Records State` can be shared and stored in the database of the `receiverHospital` and `requestHospital` nodes, respectively. A diagram of this command is shown in Figure 3.

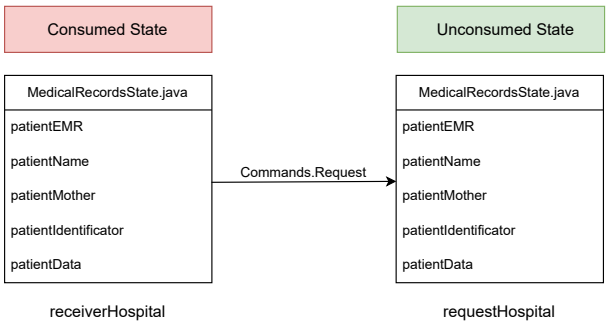


Figure 3. Requisition command.

4.5. Design and Technical Implementation

This section outlines the technical design of the proposed CorDapp, focusing on its flow logic, cryptographic mechanisms, and state transitions. The design incorporates robust cryptographic techniques such as hash functions and digital signatures to ensure secure, deterministic state transitions validated by all participating nodes in the network.

Corda leverages standard cryptographic functions to guarantee data integrity, authenticity, and security across its distributed ledger. The following cryptographic elements are crucial for the secure operation of the proposed CorDapp:

- **Hash function (SHA-256)**: Corda uses the SHA-256 hash function to compute a fixed-size digest of the transaction payload. This ensures that any modification to the data can be detected by comparing the resulting hash with the original, providing a reliable mechanism to verify data integrity.

- **Digital signature (ECDSA)**: Corda employs the Elliptic Curve Digital Signature Algorithm (ECDSA) to ensure authenticity and non-repudiation for transactions. ECDSA is a widely adopted cryptographic algorithm that uses elliptic curve cryptography (ECC), offering high security with relatively small key sizes.
- **Verification**: Using the public key of the sender, Corda verifies the authenticity of the transaction by checking the digital signature. This guarantees that the transaction was indeed authorized by the entity holding the corresponding private key and that the data have not been tampered with.

These cryptographic mechanisms are fundamental to the design of our CorDapp, enabling the following:

- **Deterministic state updates**: ensuring consistency across the ledger by guaranteeing that all state transitions are deterministic, meaning that every participant will reach the same outcome for a given set of inputs.
- **Privacy**: leveraging confidential identities and encrypted data, Corda ensures that only the participants involved in a transaction can view its details, while maintaining privacy and confidentiality.
- **Security**: using cryptographic signatures and hash functions ensures the integrity and authenticity of transactions, preventing unauthorized modifications and providing non-repudiation.

The `FillMedicalRecords` flow is designed to update or create a `MedicalRecordsState` for a given patient. It retrieves an existing record if available, updates it with new medical data, or creates a new record if none exists. After validation through the associated contract, the state is stored in the node's vault, and the transaction is completed successfully. The detailed pseudocode for this flow is presented in Algorithm 1.

Algorithm 1 FillMedicalRecords Flow

Input: P (Patient Information), M (Medical Data)
Output: Updated or newly created `MedicalRecordsState`
Retrieve the existing `MedicalRecordsState` S for patient P
if S exists **then**
    Update S with M
else
    Create a new `MedicalRecordsState` S with patient P and medical data M
end if
Validate S via business rules using `MedicalRecordsContract`
if validation fails **then**
    Abort the transaction and return failure
end if
Store S in the node's vault
Return success

Similarly, the `RequestPatientRecords and ResponderPatientRecords` flows facilitate the exchange of medical records between two nodes, ensuring the authenticity and integrity of the data. The flow includes steps for the requesting node to generate a digital signature for the request and for the responder node to verify the signature, validate the request, and then share the `MedicalRecordsState`. This flow ensures secure sharing of sensitive medical information. The pseudocode for this flow is provided in Algorithm 2.

Algorithm 2 RequestPatientRecords and ResponderPatientRecords Flows

Input: R_q (Request Details), pk_R (Public Key), sk_R (Private Key)
Output: Success or failure
RequestPatientRecords (Requesting Node):
Retrieve the existing `MedicalRecordsState` S for patient P based on R_q
if S not exist **then** Abort the transaction and return failure
else
    Compute the hash $H(R_q) = \text{Hash}(R_q)$ using SHA-256
    Generate a digital signature using the private key sk_RSignature $= \text{Sign}_{sk_R}(H(R_q))$ via ECDSA function
    Validate S via business rules using `MedicalRecordsContract`
    Send the tuple $\langle R_q, H(R_q), \text{Signature} \rangle$ and share S to the responder node
end if
ResponderPatientRecords (Responder Node):
Verify the hash $H(R_q)$ signature using the public key pk_R with ECDSA
Validate S using the `MedicalRecordsContract` and the notary node to prevent double-spending.
if Verification fails **then**
    Reject the request and return failure
end if
Store the S in the responder node
Update the vaults on both nodes
Return success

4.5.1. Workflow Representation and Technical Overview

This subsection illustrates the workflow of the proposed solution through two key flows: `FillMedicalRecords` and `RequestPatientRecords`. Using Corda Designated Language (CDL) block diagrams, these workflows are presented step by step, demonstrating the technical implementation and practical applicability of the proposed CorDapp.

FillMedicalRecords Flow

The `FillMedicalRecords` flow manages the creation or update of medical records for a given patient. The process begins when Hospital A initiates the flow for a specific patient, such as Patient A. The workflow proceeds as follows (see Figure 4):

- **Retrieve** `MedicalRecordsState`**:** Hospital A queries its vault to check whether a `MedicalRecordsState` already exists for Patient A.
- **Create or update state:**
  - If no state exists, a new `MedicalRecordsState` is created and validated using the `MedicalRecordsContract`.
  - If a state exists, it is updated with new Patient A's information.
- **Validation:** the `MedicalRecordsContract` ensures adherence to predefined rules governing state transitions.
- **Transaction finalization:**
  - If validation fails, the transaction is reverted.
  - If successful, the `MedicalRecordsState` is stored in Hospital A's vault as a new unspent transaction output (UTXO).
- **Flow completion:** The flow guarantees atomicity, ensuring the transaction is either entirely successful or entirely reverted.

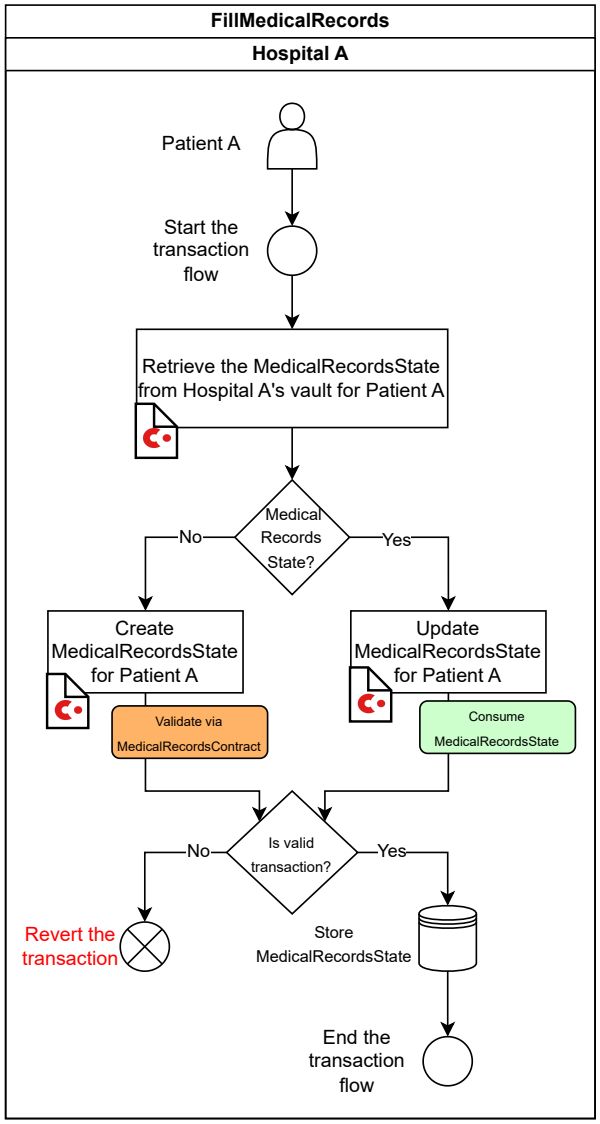


Figure 4. Workflow of the FillMedicalRecords Flow.

RequestPatientRecords and ResponderPatientRecords Flows

These flows facilitate the secure sharing of medical records between two nodes, such as Hospital A and Hospital C. The process is split into two flows: `RequestPatientRecords Flow` (Hospital A) and `ResponderPatientRecords Flow` (Hospital C) (see Figure 5).

- **Hospital A:**
  - Initiates the `RequestPatientRecords` flow to retrieve the `MedicalRecordsState` for Patient A from its vault.
  - Validates the state using the `MedicalRecordsContract`.
  - If valid, shares the `MedicalRecordsState` with Hospital C. If not, the transaction is reverted.
- **Hospital C:**
  - Initiates the `ResponderPatientRecords` flow upon receiving the `MedicalRecords State` from Hospital A.
  - Validates the state using the `MedicalRecordsContract` and ensures double-spend protection through the notary service.
  - If validation succeeds, the `MedicalRecordsState` is stored in Hospital C's vault as a shared fact. Otherwise, the transaction is reverted.

- **Finalization:** Both subflows conclude with the successful recording of the shared `MedicalRecordsState` in Hospital C's vault, maintaining data integrity, privacy, and atomicity.

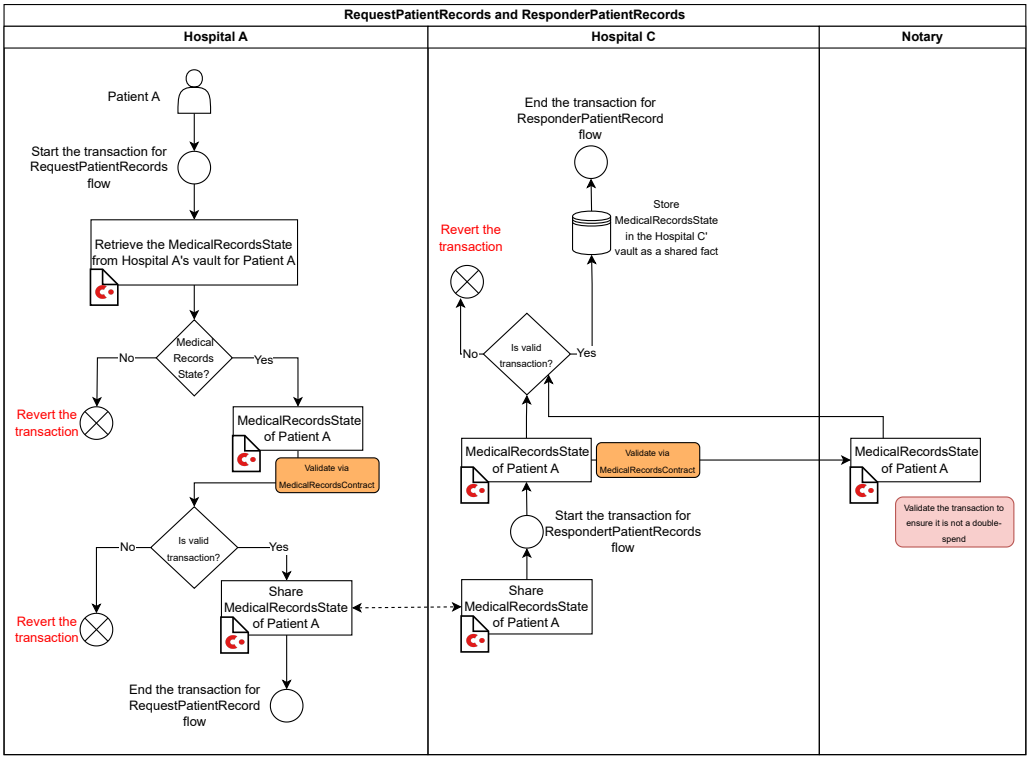


Figure 5. Workflow of the RequestPatientRecords and ResponderPatientRecords flows.

4.6. Web Server

There are three ways to connect to Corda nodes, which are through their *shells* with Secure Socket Shell (SSH), through a Remote Procedure Call (RPC) client, or a web server. In general, the Corda template on which the proposed CorDapp was based and created includes a spring web server and is the one that allows one to connect to the node via RPC.

In this way, seven application programming interfaces (API) were developed in Swagger (Swagger is a set of tools for developing APIs), and they are shown in Figure 6, where it can be seen that three APIs refer to queries (GET) of general information of the nodes and four APIs refer to the medical history within which the query APIs (GET) and the APIs of publication and requisition (POST) of medical records are shown.

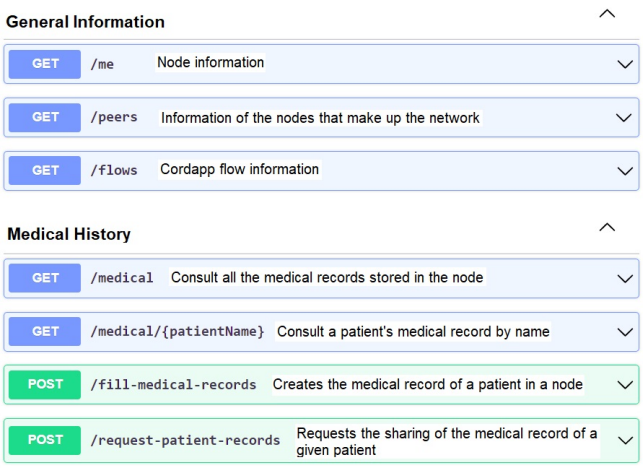
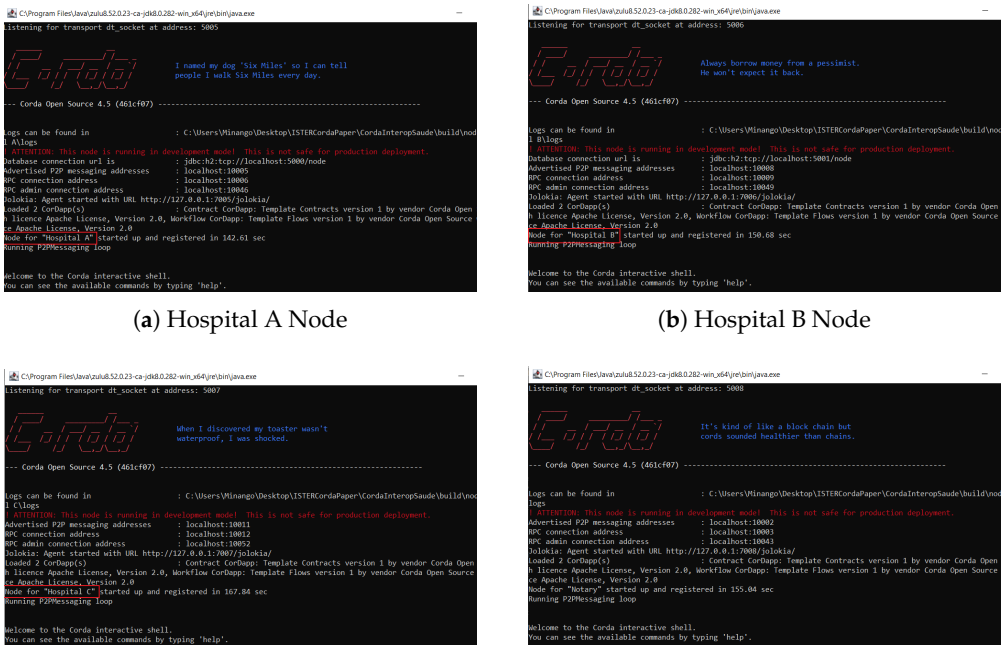


Figure 6. APIs developed in Swagger.

4.7. Running the CorDapp on the Corda Network

In Figure 7, four terminal windows corresponding to the nodes of Hospital A, Hospital B, Hospital C, and the notary are shown. These terminals run on a Java Virtual Machine (JVM), as Corda is built on the JVM, which it uses to execute code and manage the runtime environment. Additionally, the name assigned to each node in the Corda network is highlighted within the red box of each terminal.



(**a**) Hospital A Node (**b**) Hospital B Node

(**c**) Hospital C Node (**d**) Notary

Figure 7. Corda nodes executed by the CorDapp developed.

Once each node has started, as seen in Figure 7, by considering the node corresponding to Hospital A, the `FillMedicalRecords` flow can be executed, which creates the `MedicalRecordsState` state with the parameters passed in the call to the flow, as can be seen in the red box of Figure 8. In case the call to the `FillMedicalRecords` flow is successful, that is, the `MedicalRecordsState` state was created, the ledger reports as a response the hash of the transaction resulting from the `FillMedicalRecords` flow, as can be observed in the green box in Figure 8.

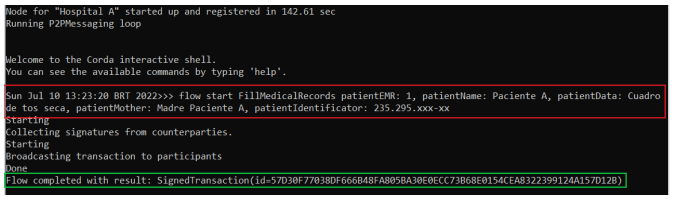


Figure 8. Hospital A node running the FillMedicalRecords flow.

To verify that the `MedicalRecordsState` status is already saved in Hospital A's node, we can perform a query on its database with the command in the red box in Figure 8 and obtain as a result the state `MedicalRecordsState` already created and represented in the green box of Figure 9.

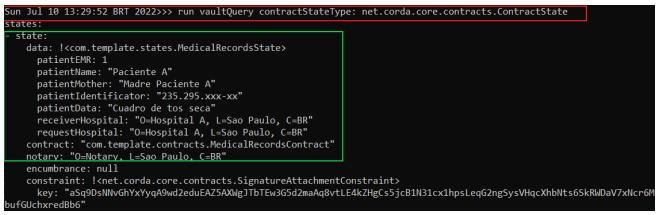


Figure 9. Hospital A node performing a query on its database.

Once Patient A's medical record, represented by the `MedicalRecordsState`, is saved on Hospital A's node, it is ready for retrieval. Assuming that Hospital C requests the `MedicalRecordsState`, the `RequestPatientRecords` flow is executed on Hospital C's node, passing as parameters the `requestHospital` together with the name of Patient A. This call can be seen in the red box of Figure 10, resulting in the corresponding hash of said transaction shown in the green box of the same figure, which indicates the successful sharing of the `MedicalRecordsState` state. In other words, both Hospital A and Hospital C nodes have in their databases the `MedicalRecordsState` without any immutability having existed.

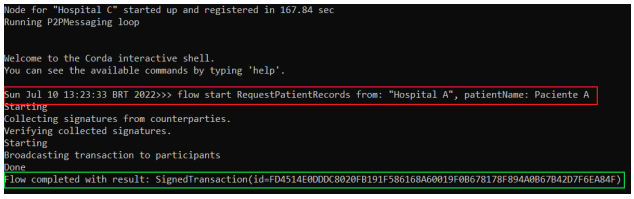


Figure 10. Hospital C node running the RequestPatientRecords flow.

To verify that both databases of Hospital A node and Hospital C node contain the same copy of `MedicalRecordsState`, a query was made on their database or vault. The result of this query can be viewed in the green box of Figures 11 and 12 corresponding to Hospital A and Hospital C's nodes, respectively.

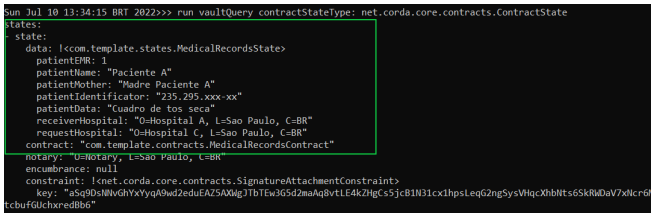


Figure 11. Hospital A node performing a query on its database.

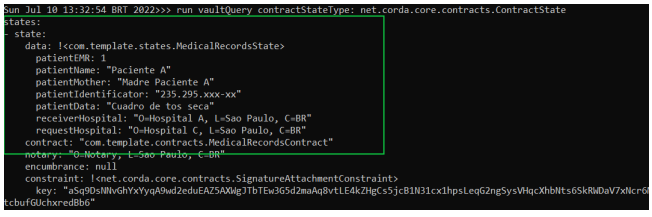


Figure 12. Hospital C node performing a query on its database.

Once each node is composed of a vault, its database schema consists of two main sets of tables, which are crucial for tracking both state information and metadata about transactions, nodes, and network operations:

1. **Vault tables**: The first set of tables, known as *vault tables*, are responsible for storing the states within the Corda node. These tables track the state data, such as the `MedicalRecordsState` for each patient, and their table names are prefixed with `VAULT`.

Each node maintains its own copy of the vault, ensuring that the data are locally available and can be queried as needed. This design provides both data availability and privacy, as each node is in control of its own data.

2. **Transaction metadata tables**: The second set of tables tracks metadata related to transactions, nodes, the network, and state machines. These tables are pre-configured with the `NODE` identifier and include information about transaction history and state transitions that occur within the network. These tables help in monitoring and auditing network activity, ensuring transparency and integrity across the system.

In addition to querying the *vault tables*, another way to verify the accuracy and integrity of the `MedicalRecordsState` is by examining the `NODE_TRANSACTIONS` table within the vault. This table tracks all completed transactions for the node and provides a detailed history of the transactions processed by the node. By inspecting this table, we can observe the full lifecycle of transactions related to state changes.

As shown in Figure 13, the `NODE_TRANSACTIONS` table for Hospital A's node records two specific transactions:

- One transaction for the `FillMedicalRecords` flow, which registers the `MedicalRecordsState` for Patient A (as illustrated in Figure 11).
- Another transaction for the `RequestPatientRecords` flow, where Hospital A's node shares the `MedicalRecordsState` of Patient A with Hospital C's node.

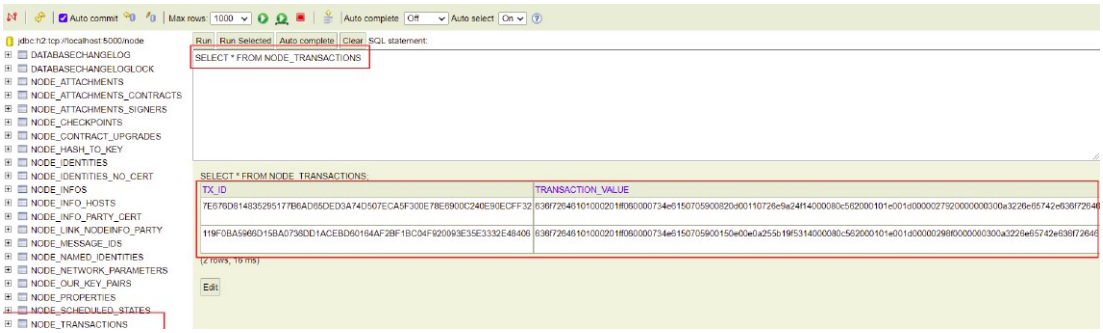


Figure 13. Query to the NODE_TRANSACTIONS table in the vault of Hospital A's node.

In this table, we find the transaction ID (`TX_ID`) and the serialized and encoded `TransactionState` object in the `TRANSACTION_VALUE` column, which encapsulates the `MedicalRecordsState`. The `TransactionState` objects are serialized using the AMQP (Advanced Message Queuing Protocol) serialization technique. AMQP serialization provides efficient binary serialization and deserialization, ensuring that complex objects, such as the `MedicalRecordsState`, can be encoded into a compact binary format and then decoded when needed, while maintaining compatibility across different platforms and Corda nodes. This allows us to verify the transaction and the associated state at any point in the process by decoding the serialized `TransactionState`.

5. Testing Methodology

This section describes the performance metrics used to evaluate the proposed CorDapp, which were throughput, response time, latency, error rate, CPU, and memory utilization.

5.1. Throughput

Throughput in Corda refers to the number of transactions that can be processed by the platform in a given period. The throughput of a Corda network depends on several factors, including the hardware and software configuration of the nodes, the size and complexity

of the transactions, and the number of nodes participating in the network. Thus, in Corda, throughput is calculated in transactions per second (TPS) and is given by

$$\text{TPS} = \frac{\text{number of transactions}}{\text{total time}}, \tag{1}$$

where the total time is calculated from the sending of the first to the last transaction, including all the intervals between transactions.

Transactions are equivalent to flow calls; typically, one or more transactions are recorded by one or more nodes that make up the Corda network. In other words, TPS are defined as the amount of a certain flow completed per second. The following two types of TPS can be defined:

- Number of TPS in the network as a whole where more than one node is involved (TPS Network).
- Number of TPS where a single node is involved (TPS Node).

5.2. Response Time

Response time in Corda refers to the time it takes for a node to process a request and return a response. Thus, it corresponds to the time elapsed between the start ($\mathcal{S}$) and the end ($\mathcal{E}$) of a transaction and is given by

$$\text{tr} = \mathcal{E} - \mathcal{S} \quad [\text{s}] \tag{2}$$

5.3. Latency

The time required to execute a task that can be consensus or communication is measured in seconds.

5.4. Error Rate

The error rate is the number of transactions not accepted by the system, which is given in terms of percentage.

5.5. CPU Utilization

It refers to the amount of CPU usage during a transaction, which may involve queries in the database of each node, creating or updating states, building the transaction, checking the rules (smart contracts) that must be followed during the transaction, signing the transaction, verifying the signatures of the corresponding nodes, sharing the transaction with other nodes involved in the transaction, and saving the transaction in the databases of the communicating nodes.

5.6. Memory Usage

Represents the amount of memory used during a transaction based on the number of flows, the number of RPC call connections, and so on.

5.7. Performance Analysis Tool

The JMeter tool, an open-source testing application that analyzes and measures software performance, was used to evaluate the performance metrics of the proposed CorDapp. JMeter is entirely developed in Java and allows load and stress tests to be carried out, to which CorDapp was submitted. Thus, JMeter can be used to test the performance and reliability of the Corda network. JMeter is designed to simulate many users making requests to a system and can be used to measure the system's response time and throughput under different load conditions.

5.7.1. Load Tests

This test models the expected use of CorDapp through simulations of simultaneous access from several users or patients to CorDapp flows. Load tests can evaluate the performance and reliability of the Corda network under different load conditions. Further, load tests in Corda can be used to identify bottlenecks and optimize the network's performance.

5.7.2. Stress Tests

Once the nodes have a maximum load capacity, they respond slowly and cause errors when the load exceeds the limit. The purpose of stress tests is to find the maximum load that the node can support.

5.8. Definition of Performance Tests

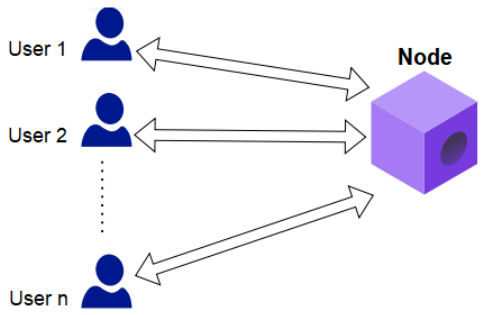
Two types of requests were utilized in the load and stress tests conducted during the CorDapp performance evaluation, presented in Figure 14 and described below.

5.8.1. Simultaneous Requests from Multiple Users

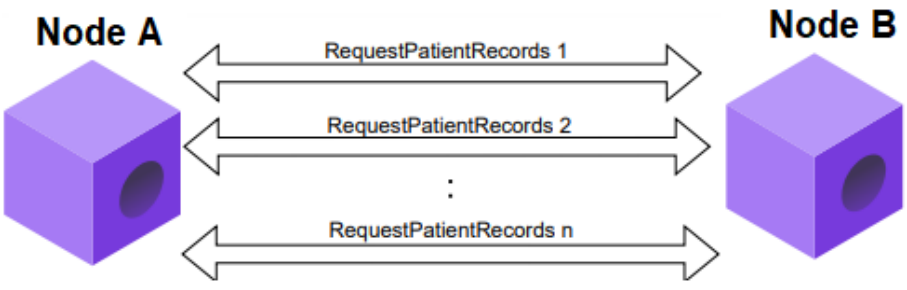
In this situation, several users simultaneously executed the `FillMedicalRecords` flow, that is, several patients registered their medical history simultaneously in a given node as shown in Figure 14a.

5.8.2. Simultaneous Requests Between Nodes

It was performed in the communication among nodes, when one requested information from other nodes for a certain period. Thus, the flow `RequestPatientRecords` was executed several times, that is, the medical history of a given patient stored in Hospital A's node was requested several times by Hospital B's node as seen in Figure 14b.



(**a**) Requests from several users simultaneously



(**b**) Simultaneous requests between nodes.

Figure 14. Considered requests.

6. Performance Results

This section presents the evaluation of the results. Two scenarios were considered for collecting CorDapp performance metrics: one within a local network and one in the cloud. These scenarios are briefly described below.

6.1. Local Network

It is known that the performance of a computer is limited by its processing power. However, running a Corda network locally allowed us to compare and discover some connections between parameters. The Corda network and CorDapp were tested locally on a computer with the characteristics described in Table 1.

6.2. Cloud Network (Cluster)

A cluster consists of interconnected servers where the computational power of a cluster is highly dependent on the number of computers it contains. The cluster used for benchmarking consisted of three fully connected nodes located on the Google Cloud Computing platform (Google Cloud Computing refers to the virtual space through which several tasks that previously required hardware or software can be performed and that now use the Google cloud as the only way to access, store, and manage data) (GCP) with the characteristics presented in Table 1.

Table 1. Characteristics of the local and cloud networks.

	Local Network	**GPC Network**
OS	Windows 10 Pro 64-bit	Ubuntu 20.04.3 LTS
Processor	Intel i7-5600U 2.60 GHz	Intel E-2288G 3.7 GHz
Core	4 CPUs	4 CPU
Memory	8 GB	8 GB
Type	-	E2 general purpose

6.3. Throughput

Two throughput scenarios were evaluated: the throughput that a node could achieve (TPS node) during the execution of the flow `FillMedicalRecords` and the throughput of the network as a whole (TPS network) through the `RequestPatientRecords` flow.

Figure 15 shows the comparison between the local network and the cloud network (GCP) in terms of throughput for different numbers of users in the execution of the flow `FillMedicalRecords`, that is, the TPS node. From this figure, it is evident that the local network achieved higher throughput. However, this comparison was somewhat unfair since the local network operated on the same computer, making it a hypothetical scenario.

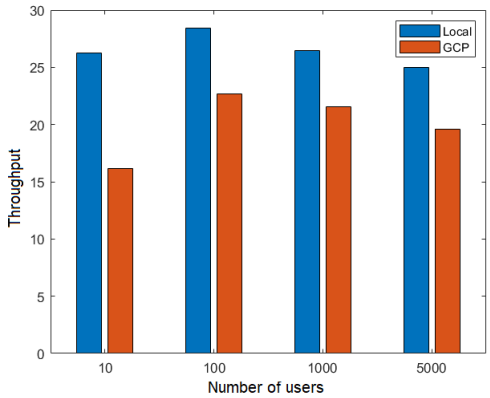


Figure 15. Throughput depending on the number of users for the TPS node.

On the other hand, Figure 16 shows the throughput for different numbers of users in the execution of the flow `RequestPatientRecords`, the TPS network. Compared with Figure 15, we observed a reduction in throughput once more than one node participated in the transaction in the TPS network.

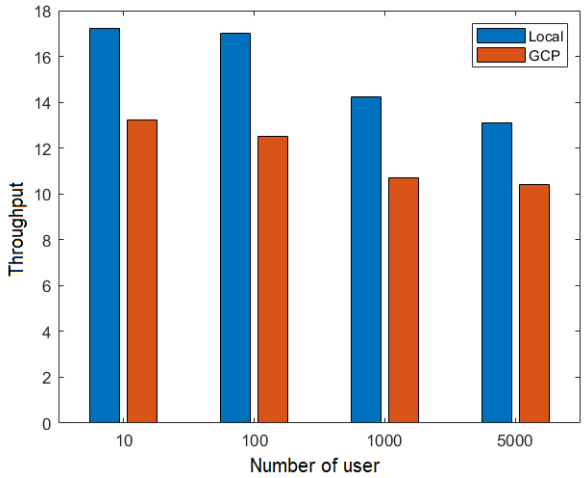


Figure 16. Throughput depending on the number of users for the TPS network.

Table 2 summarizes the average throughput of the two scenarios, the TPS node and TPS network, previously evaluated.

Table 2. Average throughput.

	TPS Node	**TPS Network**
Local network	26.52	15.39
GCP network	20	11.71

6.4. Latency

Figures 17 and 18 show the latency as a function of the number of users obtained in the execution of the flows `FillMedicalRecords` and `RequestPatientRecords`, respectively, both on the local network and in the cloud (GCP). These figures show that latency increased as more simultaneous users were added to the network. Additionally, latency was higher in the `RequestPatientRecords` scenario (Figure 18), as it involved more than one participating node.

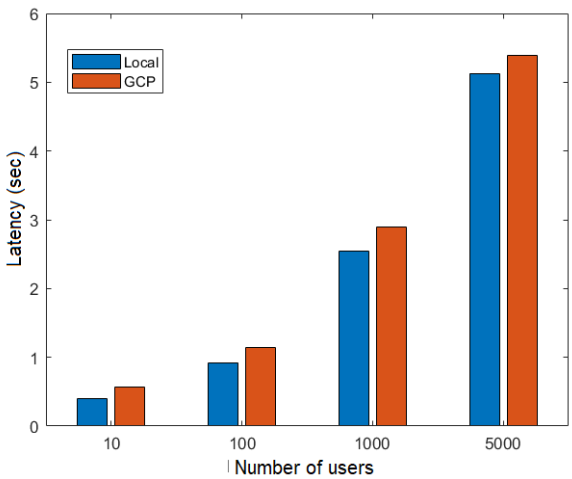


Figure 17. Latency versus number of users for the FillMedicalRecords flow.

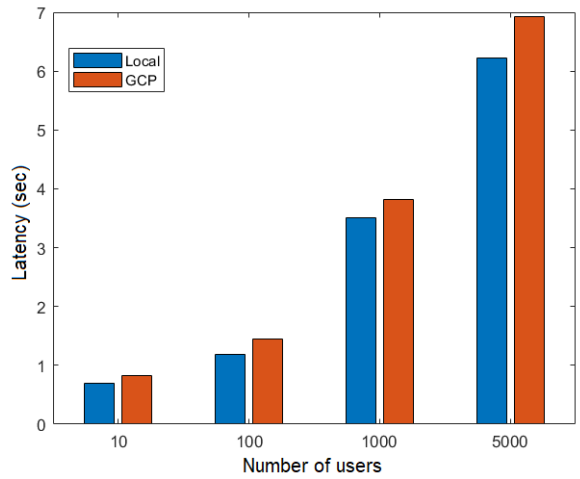


Figure 18. Latency versus number of users for the RequestPatientRecords flow.

Table 3 summarizes the average latency during the execution of `FillMedicalRecords` and `RequestPatientRecords`.

Table 3. Average latency.

	FillMedicalRecords	**RequestPatientRecords**
Local network	2.251	2.905
GCP network	2.498	3.253

6.5. Response Time

Figures 19 and 20 present the response time depending on the number of users for the `FillMedicalRecords` and `RequestPatientRecords` flows, respectively. The response time samples in both figures are centered around their mean values, with only a few outliers. This suggests that the response times followed a Gaussian distribution.

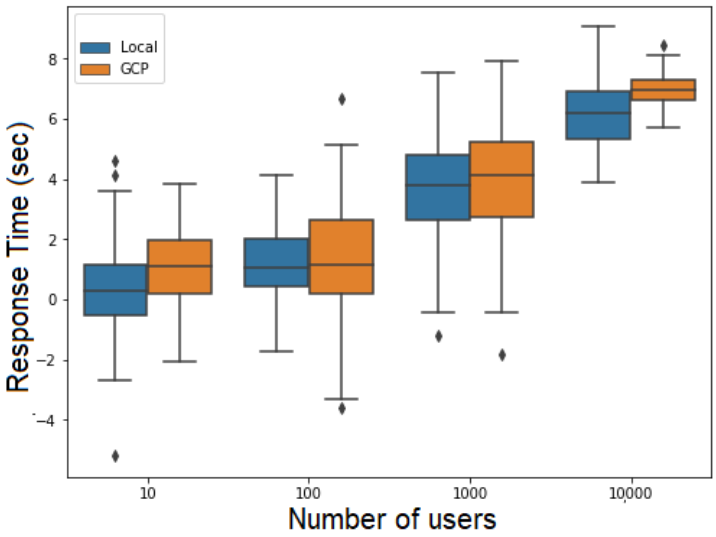


Figure 19. Response time versus number of users for the RequestPatientRecords.

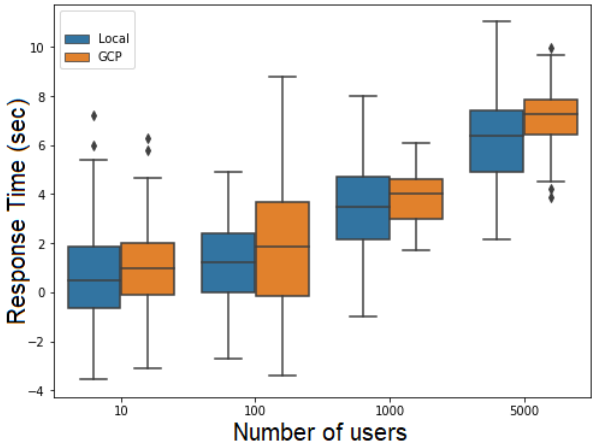


Figure 20. Response time versus number of users for the RequestPatientRecords.

6.6. Error Rate

The error rate was previously defined as the percentage of failed transactions/total number of transactions. This metric was obtained through stress tests. Table 4 presents the average error rate obtained during the execution of the stress tests for the `FillMedicalRecords` and `RequestPatientRecords` flows. From this table, it can be concluded that the GCP network had a higher transaction error rate in both flows. Also, it should be noted that the execution of the `RequestPatientRecords` flow was the one that presented a higher error rate compared to the flow `FillMedicalRecords`.

Table 4. Error rate.

	FillMedicalRecords	RequestPatientRecords
Local network	5.71%	9.52%
GCP network	5.82%	10.23%

The following tables (Tables 5 and 6) summarize the performance metrics for the `FillMedicalRecords` and `RequestPatientRecords` flows, comparing the results observed on both the local network and the GCP network. These metrics include transaction throughput (TPS), latency, and error rates, providing an overview of the system's performance under different network conditions.

Table 5. Performance metrics for FillMedicalRecords.

Metric	Local Network	GCP Network
TPS Node	26.52	20.00
TPS Network	15.39	11.71
Latency (s)	2.251	2.498
Error Rate	5.71%	5.82%

Table 6. Performance metrics for RequestPatientRecords.

Metric	Local Network	GCP Network
TPS Node	15.39	11.71
TPS Network	11.71	9.52
Latency (s)	2.905	3.253
Error Rate	9.52%	10.23%

6.7. CPU and Memory Utilization Results

Figures 21 and 22 show the percentage of CPU and memory usage, respectively, based on the number of users during the joint execution of the flows `FillMedicalRecords` and `RequestPatientRecords`, that is, the use of CPU and memory was evaluated considering the stages of recording a patient's medical history in a node and then requesting its delivery to another node participating in the Corda network.

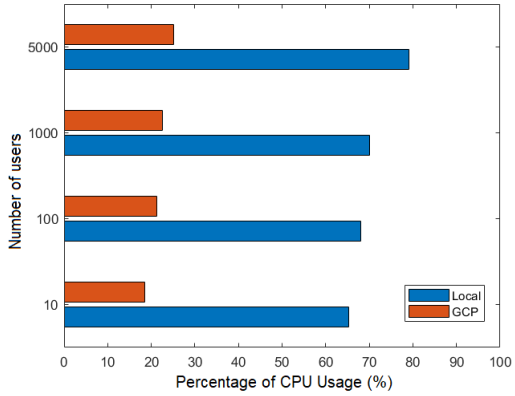


Figure 21. Number of users versus percentage of CPU usage.

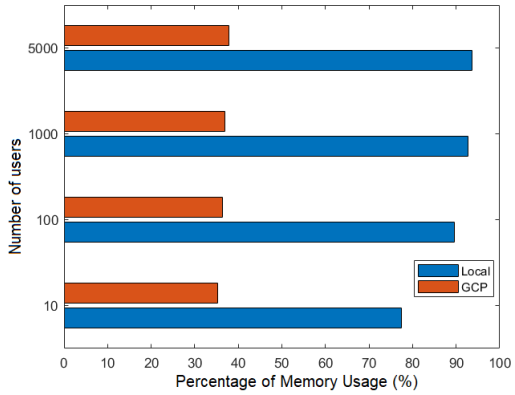


Figure 22. Number of users versus percentage of memory usage.

Figures 21 and 22 illustrate the advantages of using the GCP cloud network. In the GCP scenario, each participating node operates on a separate virtual machine within the cloud, whereas, in the local network, all nodes run on the same machine. This leads to resource saturation, resulting in higher CPU and memory consumption on the local network.

7. Conclusions and Future Works

7.1. Conclusions

In this work, we successfully structured, developed, and tested a Corda network and its corresponding CorDapp as a proof of concept for implementation within the healthcare ecosystem. Our work demonstrates the unique advantages of Corda's architecture, particularly in enabling only the nodes involved in a transaction to share, store, and track data while registering and retrieving a patient's medical history. This ensures high privacy and security, which is crucial in medical data management.

Moreover, adopting Corda within the healthcare ecosystem introduces several interesting benefits: it guarantees the immutability of medical data transmission, eliminates the need for reconciliation processes between multiple nodes, and provides real-time visibility during data sharing. These features are critical for ensuring that medical information is handled with the utmost integrity and efficiency.

Through rigorous evaluation of performance metrics, our research confirms that Corda DLT technology is viable and highly effective for medical information transactions. The technology ensures data remain intact, traceable, auditable, confidential, authentic, and interoperable—essential qualities in healthcare.

Finally, the most tangible outcome of this research is the successful construction of the CorDapp, explicitly designed for the secure exchange of health data between medical institutions with patient consent. We validated its functionality in local and cloud environments, underscoring its practical application and scalability in real-world healthcare settings.

7.2. Future Works

In the future, our work can be extended in different aspects:

1. **Specialized healthcare functionalities:** Expanding the CorDapp to add new functionalities to specific healthcare needs. For example, incorporating tracking of patient medication can significantly enhance the coordination among healthcare providers, ensuring that treatments are administered accurately.
2. **Interoperability through HL7 standards:** Integrating Health Level 7 (HL7) standards into the CorDapp to facilitate interoperability across different healthcare systems. This would enable the seamless exchange of critical medical information, thereby improving communication and collaboration between healthcare institutions.
3. **Predictive analytics with machine learning:** Enhancing the CorDapp's capabilities to support predictive analytics by using machine learning algorithms. This approach would allow for the analysis of network data to identify trends and patterns in patient care, leading to timely interventions and improved healthcare outcomes.

Author Contributions: Conceptualization, J.M. and N.O.G., with a focus on applying blockchain principles within the healthcare ecosystem; methodology, J.M. and M.Z., specifically in designing the CorDapp structure; software, J.M., responsible for developing the CorDapp using the Corda framework; validation, H.C.M., J.M. and M.Z., who validated the CorDapp performance in both local and cloud environments; formal analysis, H.C.M. and N.O.G., focusing on metrics like throughput, latency, CPU, and memory usage; investigation, H.C.M. and M.Z., who conducted extensive testing on the CorDapp; resources, F.P.; data curation, H.C.M. ; writing—original draft preparation, J.M., H.C.M. and N.O.G., focusing on blockchain application for healthcare data integrity and confidentiality; writing—review and editing, F.P.; visualization, N.O.G.; supervision, F.P., overseeing the project's blockchain and healthcare objectives; project administration, F.P.; funding acquisition, H.C.M. and N.O.G. All authors have read and agreed to the published version of the manuscript.

Funding: This work received the support of Universidad de Las Américas (UDLA), Ecuador, as part of the research project ERT.NOG.23.13.01.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data are contained within the article.

Acknowledgments: The authors would like to thank the technical team at the Tecnológico Universitario Rumiñahui for their support in setting up the network infrastructure and testing environments used in this study. We also extend our gratitude to the ETEL Research Group at Universidad de Las Américas (UDLA) for their collaboration in reviewing the methodology and analyzing the results. Finally, we thank the SATRD Lab at the Universitat Politècnica de València (UPV) for providing access to advanced computing resources essential for developing this research.

Conflicts of Interest: Author Francisco Pérez was employed by the company FAVIT. The remaining authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

References

1. Nakamoto, S. Bitcoin: A Peer-to-Peer Electronic Cash System 2009. Online Resource. Available online: https://bitcoin.org/bitcoin.pdf (accessed on 20 December 2024).
2. Chen, W.; Xu, Z.; Shi, S.; Zhao, Y.; Zhao, J. A Survey of Blockchain Applications in Different Domains. In Proceedings of the 2018 International Conference on Blockchain Technology and Application, New York, NY, USA, 25–30 June 2018; ICBTA 2018; pp. 17–21 . [CrossRef]
3. Alghamdi, T.A.; Khalid, R.; Javaid, N. A Survey of Blockchain Based Systems: Scalability Issues and Solutions, Applications and Future Challenges. *IEEE Access* **2024**, *12*, 79626–79651. [CrossRef]
4. Ren, K.; Ho, N.M.; Loghin, D.; Nguyen, T.T.; Ooi, B.C.; Ta, Q.T.; Zhu, F. Interoperability in Blockchain: A Survey. *IEEE Trans. Knowl. Data Eng.* **2023**, *35*, 12750–12769. [CrossRef]
5. Fiore, M.; Mongiello, M. History of Blockchain Technology and its Impact in Social Good. In Proceedings of the 2023 8th IEEE History of Electrotechnology Conference (HISTELCON), Florence, Italy, 7–9 September 2023; pp. 36–38. [CrossRef]
6. Forte, P.; Romano, D.; Schmid, G. Beyond Bitcoin—Part I: A critical look at blockchain-based systems. *Cryptography* **2017**, *1*, 15 . [CrossRef]
7. Zantalis, F.; Koulouras, G.; Karabetsos, S. Blockchain Technology: A Framework for Endless Applications. *IEEE Consum. Electron. Mag.* **2024**, *13*, 61–71. [CrossRef]
8. Spadavecchia, G.; Fiore, M.; Mongiello, M.; De Venuto, D. A Novel Approach for Fast and Secure Data Transmission using Blockchain and IoT. In Proceedings of the 2024 13th Mediterranean Conference on Embedded Computing (MECO), Budva, Montenegro, 11–14 June 2024; pp. 1–4. [CrossRef]
9. Halpin, H.; Piekarska, M. Introduction to Security and Privacy on the Blockchain. In Proceedings of the 2017 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW), Paris, France, 29–30 April 2017; pp. 1–3. [CrossRef]
10. Nguyen, H.N.; Pham, H.A.; Huynh-Tuong, N.; Nguyen, D.H. Leveraging Blockchain to Enhance Digital Transformation in Small and Medium Enterprises: Challenges and a Proposed Framework. *IEEE Access* **2024**, *12*, 74961–74978. [CrossRef]
11. Irvin, A.; Kiral, I. Designing for Privacy and Confidentiality on Distributed Ledgers for Enterprise (Industry Track). *arXiv* **2019**, arXiv:1912.02924.
12. Opare, E.A.; Kim, K. A Compendium of Practices for Central Bank Digital Currencies for Multinational Financial Infrastructures. *IEEE Access* **2020**, *8*, 110810–110847. [CrossRef]
13. Panwar, A.; Bhatnagar, V. Distributed Ledger Technology (DLT): The Beginning of a Technological Revolution for Blockchain. In Proceedings of the 2nd International Conference on Data, Engineering and Applications (IDEA), Bhopal, India, 28–29 February 2020; pp. 1–5. [CrossRef]
14. Kasyapa, M.; Vanmathi, C. Blockchain integration in healthcare: A comprehensive investigation of use cases, performance issues, and mitigation strategies. *Front. Digit. Health* **2024**, *6*, 1359858. [CrossRef] [PubMed]
15. Corte-Real, A.; Nunes, T.; da Cunha, P.R. Reflections about Blockchain in Health Data Sharing: Navigating a Disruptive Technology. *Int. J. Environ. Res. Public Health* **2024**, *21*, 230. [CrossRef] [PubMed]
16. Corte-Real, A.; Nunes, T.; Santos, C.; Cunha, P.R. Blockchain technology and universal health coverage: Health data space in global migration. *J. Forensic Leg. Med.* **2022**, *89*, 102370. [CrossRef] [PubMed]
17. United Nations. Department of Economic and Social Affairs Sustainable Development. SDGs2030. 2024. Available online: https://sdgs.un.org/goals (accessed on 27 December 2024).
18. General Data Protection Regulation (GDPR): Article 6(1) and Recital 40. 2024. Available online: https://gdpr-info.eu/art-6-gdpr/ (accessed on 27 December 2024).
19. Health Insurance Portability and Accountability Act (HIPAA). 2024. Available online: https://www.hhs.gov/hipaa/index.html (accessed on 27 December 2024).
20. Presidência da República Secretaria-Geral LEI Nº 13.709, DE 14 DE AGOSTO DE 2018 Lei Geral de Proteção de Dados (LGPD). 2024. Available online: http://www.planalto.gov.br/ccivil_03/_ato2015-2018/2018/lei/L13709compilado.htm (accessed on 27 December 2024).
21. Mole, J.S.S.; Shaji, R.S. Ethereum blockchain for electronic health records: Securing and streamlining patient management. *Front. Med.* **2024**, *11*, 1434474. [CrossRef] [PubMed]
22. Joy, S.; Neethu, P.S.; Patil, K.A.; Cheema, A.P.; Lal, M.K. Hyperledger Fabric as a Secure Blockchain Solution for Healthcare 4.0 Framework. In Proceedings of the 2023 International Conference on Advances in Computation, Communication and Information Technology (ICAICCIT), Faridabad, India, 23–24 November 2023; pp. 530–536. [CrossRef]
23. Jena, S.K.; Kumar, B.; Mohanty, B.; Singhal, A.; Barik, R.C. An advanced blockchain-based hyperledger fabric solution for tracing fraudulent claims in the healthcare industry. *Decis. Anal. J.* **2024**, *10*, 100411. [CrossRef]
24. Mani, V.; Kavitha, C.; Band, S.S.; Mosavi, A.; Hollins, P.; Palanisamy, S. A Recommendation System Based on AI for Storing Block Data in the Electronic Health Repository. *Front. Public Health* **2022**, *9*, 831404. [CrossRef] [PubMed]

25. Mubashar, A.; Asghar, K.; Javed, A.R.; Rizwan, M.; Srivastava, G.; Gadekallu, T.R.; Wang, D.; Shabbir, M. Storage and Proximity Management for Centralized Personal Health Records Using an IPFS-Based Optimization Algorithm. *J. Circuits Syst. Comput.* **2022**, *31*, 2250010. [CrossRef]
26. Brisimi, T.S.; Chen, R.; Mela, T.; Olshevsky, A.; Paschalidis, I.C.; Shi, W. Federated learning of predictive models from federated Electronic Health Records. *Int. J. Med. Inform.* **2018**, *112*, 59–67. [CrossRef] [PubMed]
27. Kumar, S.; Bharti, A.K.; Amin, R. Decentralized secure storage of medical records using Blockchain and IPFS: A comparative analysis with future directions. *Secur. Priv.* **2021**, *4*, e162. [CrossRef]
28. Lu, S.; Zhang, Y.; Wang, Y. Decentralized Federated Learning for Electronic Health Records. In Proceedings of the 2020 54th Annual Conference on Information Sciences and Systems (CISS), Princeton, NJ, USA, 18–20 March 2020; pp. 1–5. [CrossRef]
29. Bellaj, B.; Ouaddah, A.; Bertin, E.; Crespi, N.; Mezrioui, A. Drawing the Boundaries Between Blockchain and Blockchain-Like Systems: A Comprehensive Survey on Distributed Ledger Technologies. *Proc. IEEE* **2024**, *112*, 247–299. [CrossRef]
30. Rifat Hossain, M.; Nirob, F.A.; Islam, A.; Rakin, T.M.; Al-Amin, M. A Comprehensive Analysis of Blockchain Technology and Consensus Protocols Across Multilayered Framework. *IEEE Access* **2024**, *12*, 63087–63129. [CrossRef]
31. Hasselgren, A.; Kralevska, K.; Gligoroski, D.; Pedersen, S.A.; Faxvaag, A. Blockchain in healthcare and health sciences—A scoping review. *Int. J. Med. Inform.* **2020**, *134*, 104040. [CrossRef] [PubMed]
32. Zghaibeh, M.; Farooq, U.; Hasan, N.U.; Baig, I. SHealth: A Blockchain-Based Health System With Smart Contracts Capabilities. *IEEE Access* **2020**, *8*, 70030–70043. [CrossRef]
33. Tripathi, G.; Ahad, M.A.; Paiva, S. S2HS-A blockchain based approach for smart healthcare system. *Healthcare* **2020**, *8*, 100391. [CrossRef]
34. Hölbl, M.; Kompara, M.; Kamišalić, A.; Nemec Zlatolas, L. A Systematic Review of the Use of Blockchain in Healthcare. *Symmetry* **2018**, *10*, 470. [CrossRef]
35. Blockchain: Opportunities for Health Care. Available online: https://www.colleaga.org/sites/default/files/4-37-hhs_blockchain_challenge_deloitte_consulting_llp.pdf (accessed on 30 December 2023).
36. Chukwu, E.; Garg, L. A Systematic Review of Blockchain in Healthcare: Frameworks, Prototypes, and Implementations. *IEEE Access* **2020**, *8*, 21196–21214. [CrossRef]
37. Yazdinejad, A.; Srivastava, G.; Parizi, R.M.; Dehghantanha, A.; Choo, K.K.R.; Aledhari, M. Decentralized Authentication of Distributed Patients in Hospital Networks Using Blockchain. *IEEE J. Biomed. Health Inform.* **2020**, *24*, 2146–2156. [CrossRef] [PubMed]
38. Sheela, K.; Priya, C. Blockchain-based security & privacy for biomedical and healthcare information exchange systems. In *Materials Today: Proceedings*; Elsevier: Amsterdam, The Netherlands, 2023; Volume 81, pp. 641–645.
39. AlQudah, A.A.; Al-Emran, M.; Shaalan, K. Medical data integration using HL7 standards for patient's early identification. *PLoS ONE* **2022**, *16*, e0262067. [CrossRef] [PubMed]
40. Hearn, M. Corda: A Distributed Ledger. In *Technical Report, R3*; 2016. Online resource. Available online: https://www.corda.net/content/corda-technical-whitepaper.pdf (accessed on 25 November 2024).
41. Sheikh, J. *Mastering Corda: Blockchain for Java Developers*; O'Reilly: Springfield, MO, USA, 2020.