Blockchain-based Rental Documentation Management with Audit Support

[1,2]João F. Santos, [2]Miguel P. Correia, [1]Tiago R. Dias
[1]Unlockit.io, Portugal [2]INESC-ID, Instituto Superior Técnico, Universidade de Lisboa
joao.santos@unlockit.io, tiago.dias@unlockit.io, miguel.p.correia@tecnico.ulisboa.pt

***Abstract*—In the rental market, document management is a critical process to ensure the accuracy of financial transactions and regulatory compliance. In Portugal and many other countries, challenges include the complexity of legislation, particularly GDPR compliance, lack of transparency, and bureaucratic inefficiencies. With this in mind, a solution based on Hyperledger Fabric, a blockchain platform, is presented for the implementation of a document management system for the rental process that supports auditing. This system oversees the rental process. We present the system, a prototype, and an experimental evaluation.**

I. Introduction

Blockchain technology provides decentralisation to sectors where the single point of failure is the norm. Developers and entrepreneurs continue to explore new use cases for this new cutting-edge technology. Cryptocurrencies, such as Bitcoin, are an excellent illustration of this, since their primary purpose is to provide an alternative payment system that is decentralised but otherwise functions similarly to existing solutions, creating a more transparent and efficient way to conduct financial transactions. However, centralised organisations such as banks can also look at blockchain technology as an innovation with the ability to improve efficiency and transparency. There are still many use cases where blockchain can enter and improve the field, such as Supply Chain & Logistics, Finance, and Property & Real Estate, given that this is a new technology undergoing improvement and development.

Today, the real estate market is inefficient and slow in responding to consumer needs. In order to acquire or lease a new home, numerous bureaucratic steps must be followed to ensure adherence to legal standards; consequently, these processes are lengthy. The processes that slow down such procedures are, among others, risk verification, regulatory compliance, and fraud investigation.

Real estate audits and due diligence processes require manual validation of documentation. An audit is necessary to investigate information about properties and their owners whose objective is to form an independent opinion on financial and legal statements. During the process, all the financial, accounting and tax aspects of the property must be analysed. This process seeks to catch any possible fraud that the buyer, seller, or renter may be committing, for instance, debts that somehow may be linked to the property itself or even fake property reviews and forgers, who are armed with false documentation, to impersonate owners, sellers, or even attorneys. High intermediary or brokerage fees, the acquisition and verification of pertinent information from legal sources, fluctuating transaction prices, opacity of property rights, and tax fluctuation are other relevant issues within this entire procedure in real estate [1]. Due to its complexity and difficulty in regulation, it can be difficult to create innovative solutions to solve this real estate audit problem.

Blockchain shows a promising future for the enhancement and development of new tools in this area. However, it is imperative to recognise that these technologies are not bullet-proof. As a result, in addition to the benefits mentioned earlier, the space may suffer from any drawbacks these technologies may have. In the future, auditors can use these techniques to make the audit process more efficient and provide a better and more sophisticated service to anyone seeking their services.

This paper presents the Blockchain-based Documentation Management (BDM) system. BDM is a blockchain-based document management system for the rental process that supports auditing. This system oversees the rental process, which consists of three phases: the application for a property by the prospective tenant through the upload of necessary documents, acceptance/rejection by the landlord of various received applications, and the creation of a report by the system, which only an auditor can request and view. The system smart contract records metadata associated with the documents (hash, owner) and coordinates requests for file access by landlords to prospective tenants. Therefore, the system creates immutable and traceable records of the entire process and serves as a foundation for audits. After the landlord verifies the files and accepts the rental proposal, any authorised auditor can request a report for a property by accessing the records through the final report, which includes all events that occurred during the process. We present the system, a prototype, and an experimental evaluation.

II. Background

A blockchain is a distributed, durable and append-only ledger that contains records organised in blocks [2]. Blocks store valid transactions as a record book page. A blockchain is a distributed ledger technology (DLT) in which the ledger is not controlled by a central authority. This technology can be used to ensure safe transactions, reduce compliance costs, and simplify data processing. When a block is filled with transactions (block $t + 1$), it is closed and linked to the previous block (block t), forming a chain of data, the blockchain. Following that newly added block, all additional

data is collected into a new block, which is then added to the chain once it is complete. The transaction validation process ensures that transactions and blocks on the blockchain are verified [3].

1) Smart Contracts: Smart contracts, an innovation in blockchain technology, change the way agreements are made and enforced [4]. These self-executing digital agreements operate autonomously within blockchain networks, automating and verifying contract terms. They are coded to execute actions based on predefined conditions, be they simple criteria or complex events. Crucially, they interact with the blockchain data and modify it as needed.

These contracts are integral to decentralised blockchain networks and leverage low-level programming languages such as Ethereum bytecodes. They find applications in various domains, offering transparency, automation, and security. By eliminating intermediaries and reducing manual intervention, they improve efficiency and accuracy, making them valuable for supply chain management, finance, and real estate.

2) Permissionless Blockchains: In permissionless blockchains [2], there are no restrictions on joining the network. Anyone can participate in the consensus algorithm [5] and validate transactions. A user generates a personal address on a permissionless blockchain and interacts with the network by sending transactions to other users or assisting the network in validating transactions. If a user helps with block validation, a reward is earned for validating the new incoming blocks, so this type of blockchain receives more support from the community.

3) Permissioned Blockchains: A permissioned blockchain restricts access to authorised users and is often chosen for enhanced security. Blockchain administrators manage user authorisations, ensuring that only authorised individuals can interact with them. This approach is popular among organisations that prioritise data confidentiality and privacy. Permissioned blockchains are widely adopted, especially in corporate settings.

4) Permissioned Blockchain Frameworks: Various permissioned blockchain frameworks, such as Canton [6], Hyperledger Fabric [7], and R3 Corda [8], support programmable transactions, allowing entities managing the blockchain to define business rules and logic.

Hyperledger Fabric, part of the Hyperledger project under the Linux Foundation, is a permissioned distributed ledger platform. It emphasises modular architecture and adaptability, supporting various consensus algorithms. The unique feature of Fabric is its use of channels for segregated communication paths, ensuring data privacy. Customisable endorsement policies streamline transaction agreement, focusing on scalability. Identity management and access control are maintained through the Membership Service Provider (MSP) framework, ensuring accountability and transparency. Transactions in Fabric undergo a lifecycle, including proposal, endorsement, block distribution, consensus, and ledger update.

R3 Corda is an enterprise-grade distributed ledger platform that focusses on privacy, scalability, and interoperability. It emphasises shared ledgers, where only the parties involved access the transaction data for confidentiality. Fine-grained permissions determine data access. The Corda Flow Framework allows direct communication and negotiation, similar to real-world agreements. Smart contracts, called states and contracts, govern shared facts and transaction rules. Pluggable consensus models enable the choice of the most suitable algorithm for security and performance.

Canton is an innovative blockchain technology that prioritises efficiency, scalability, and practicality. It employs the Proof-of-Stakeholder (PoSH) consensus mechanism, considering participants' roles for influence, ensuring decentralisation and efficiency. Canton's architecture focusses on scalability using infrastructure nodes that efficiently process and verify transactions without accessing data content directly. Secure communication within defined trust zones optimises network efficiency and supports high-throughput use cases. By combining PoSH with an efficiency-oriented design, Canton offers an enterprise-ready platform for blockchain implementation. More details can be found in [6].

III. Audit Process in Real Estate

An audit is a meticulous process to verify the degree to which an organisation complies with various requirements, which can be regional, national, or international in nature [1]. These requirements can vary significantly based on the organisation's location, e.g., in terms of tax obligations. Real estate audits involve examining financial records, transaction procedures, and document quality. Financial scrutiny involves a detailed review of all money flows in a fiscal year, ensuring the accuracy of financial records. In addition, real estate audits focus on ensuring that transactions comply with local real estate laws. This includes property transactions, leases, contracts, and agreements. The quality of record keeping is crucial, as it demonstrates an organisation's commitment to regulatory standards.

A. Audit Categories

Audits can be divided into three main categories: internal, external, and governmental. Internal audits are often conducted by employees of the company. However, the business can also choose to contract out this service. In external audits, for the audit process to be impartial, external auditors, unlike internal auditors, must be able to operate on their own and provide an unqualified opinion. The last is that government audits are performed to verify that financial statements have been made appropriately and that a company's taxable income has not been distorted [9]. Audits are essential for business continuity for several reasons [10]:

- *Increase operational efficiency:* Find control recommendations to increase the efficacy and efficiency of processes by regularly assessing and monitoring them.
- *Evaluate risks and protects assets:* Assist in keeping track of any environmental alterations documented, as well as ensuring that any risks discovered are mitigated.

- *Assess organisational controls:* Enhance the organisation's control environment by analysing effectiveness and efficiency.
- *Ensure legal compliance:* Applicable laws and regulations are followed by conducting internal audits on a regular basis.

Even with a promising digital transformation in the audit process, many obstacles still exist. Auditors need to acquire the skills to undergo this digital transformation and are not ready to approach a more automated audit workflow, creating a significant obstacle in this regime change.

An interesting case is that of *SmartAudit*, a company that provides cloud-based audit services. Tasks such as lead scheduling, financial statement preparation, and report writing are automated, and the progress of this audit can be seen in real time [11]. When new clients are accepted, they must upload their data to their cloud-based infrastructure. Afterward, a new plan is set to audit all available files, in compliance with international standards. In the end, a report is generated.

B. Blockchain-based Smart Audit

A more reliable and secure environment for audits can be obtained by combining blockchain with the smart audit techniques mentioned above. Due to the sufficiency, relevance, and dependability requirements for audit evidence, blockchain technology is suitable for use in conjunction with intelligent auditing approaches. The integrity of the data provided by blockchain increases the trustworthiness of the audit evidence. The information flow steps are the following:

1) *Data Production and Control:* Data is collected using smart sensors, IoT, and other technologies. To find anomalies and useful information, a number of tests and analytics are performed using intelligent audit modules.
2) *Data Storage:* Metadata used to assess the integrity of the bulk data is maintained in a selected blockchain.
3) *Smart Contract Data Manipulation:* Smart contracts enforce, without human intervention, the proper operation of intelligent audit modules.
4) *Data Auditing:* The data stored in the blockchain will then be used to perform an audit of those data with the help of tools such as intelligent process automation, natural language processing, and machine learning.

C. Related Work

Several research efforts have explored the application of blockchain technology in various industries, including real estate. Here are some key findings:

Kang et al. [12] proposed a solution using blockchain, a peer-to-peer network, and the Interplanetary File System (IPFS) to improve file storage and sharing. This approach improves decentralisation, scalability, and data consistency. It combines blockchain, P2P networks, and IPFS to create a secure and efficient system for storing and sharing files.

Wouda et al. [13] discussed the potential of blockchain to streamline commercial real estate asset transactions, particularly in the Netherlands. They highlighted the challenges faced in these transactions, such as high costs and lack of transparency, and proposed the use of blockchain to address these issues. The aim is to create a transparent and efficient infrastructure for real estate transactions.

Bharimalla et al. [14] presented a solution for an Electronic Health Record System using blockchain, natural language processing (NLP), and machine learning. They implemented a permissioned blockchain, Hyperledger Fabric, to manage access to electronic health records. NLP and OCR technology was used to digitise medical records on paper, which were then standardised and stored on the blockchain.

JusticeChain is a proposal for an auditing solution for a critical Portuguese government application [15]. External oracles provide audit logs to JusticeChain, which are processed and recorded on a permissioned blockchain, Hyperledger Fabric. Auditors can access these logs only with the consensus of most auditors, ensuring the integrity of the audit process.

IV. BDM Architecture and Design

Blockchain-based Documentation Management (BDM) is a system designed to enable users to efficiently manage their real estate document sharing and transactions securely. Within this architecture, users exercise control over their document management through a set of defined processes, leveraging the immutable characteristic of a blockchain for enhanced data security and transparency. BDM incorporates two fundamental approaches to streamline document management: proactive consent and consent on request. In proactive consent, users have the ability to proactively grant or withhold consent to share their documents with relevant parties. This consent is encapsulated in permissions, which are subsequently recorded on the blockchain.

A. Solution Overview

BDM involves three primary user roles, auditor, tenant, and landlord, each with their own actions and responsibilities. Figure 1 presents each one of the actions for each role of the participant.

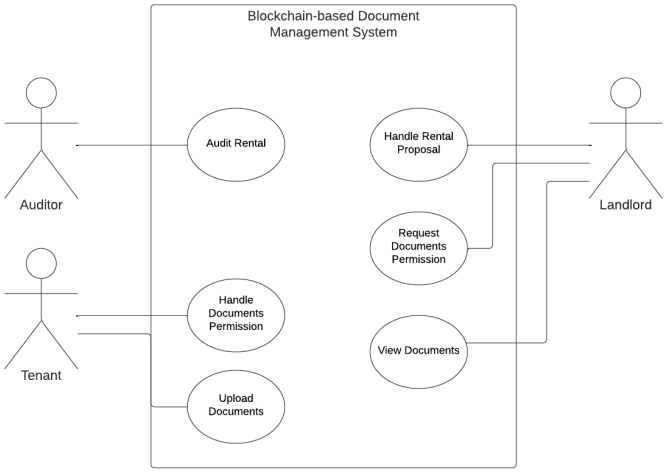

Fig. 1: BDM use case diagram

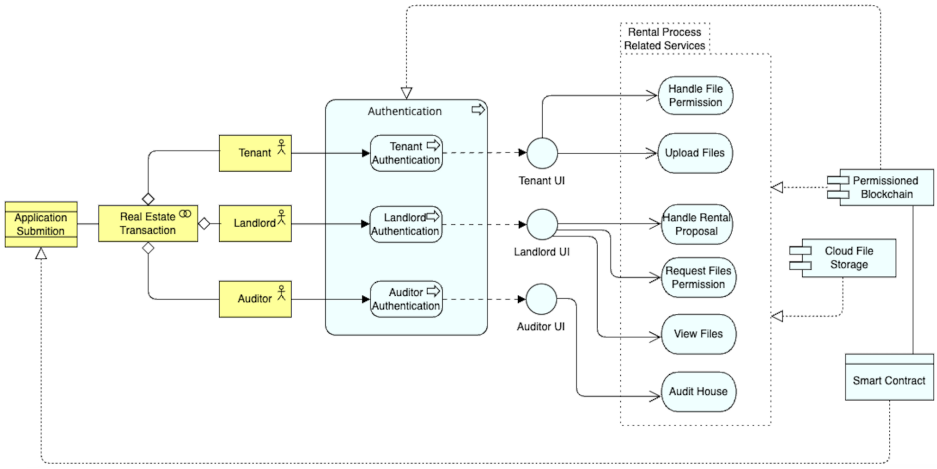

Fig. 2: BDM architecture Archimate diagram

- *Tenant:* The tenant can upload documents to submit a rental application or can handle the permissions to access the uploaded documents. The documents and their metadata are stored securely for audit purposes.
- *Landlord:* The landlord is an individual or organization who owns or manages a property, such as a house or apartment, and rents or leases it to tenants in exchange for a periodic payment. He can handle a rental proposal, either by declining or accepting it. He can also request to see the original uploaded documents. This permission must be granted by the owner of the document.
- *Auditor:* The auditor can audit any house he chooses and has access to. The BDM will generate a report with each interaction that happened in that house for the auditor to analyse.

B. System Architecture

The application architecture relies on multiple ecosystem components and a permissioned blockchain with a dedicated event register to support its core functionality. The architecture is represented in Figure 2.

The architecture comprises all the components required to make the BDM system work. The authentication module is a Certificate Authority (CA) that handles identities within a decentralised network. Every tenant that is authenticated on the platform must accept the terms and conditions so that, in the future, the system generates a final report with the data generated on the system. All related real estate rental services, document cloud storage, permissioned blockchain, and its smart contract are linked together to create an intelligent tool capable of auditing a house.

- *Authentication* is the entry point in the application. It handles the identities of users using a CA.
- *Document Cloud Storage* is the component where uploaded documents will be stored and accessed.
- *Permissioned Blockchain* is the blockchain network. This solution will contain two organisations, the tenant organisation and the landlord/auditor organisation. Smart contracts save document-related information and handle authorisation to access documents stored in the document storage. The blockchain itself serves as the register, tracking every event, including permission grants and document uploads.
- *Smart Contract* represents the core of this architecture, where specific techniques are applied to the document for each step of the process, defining a proper workflow for each task in the Rental Process Related Services.
- *Rental Process Related Services* are the methods established inside the smart contract that can be called depending on the role of the authenticated user.

An important concept for this architecture is the notion of a hash function [16]. It is a mathematical algorithm that takes an input and produces a fixed-length string of characters, known as the hash value or digest. It is designed to be a one-way process, which means that it should be computationally infeasible to reverse the hash value to retrieve the original input. The three main properties of a good hash function are:

- *Deterministic:* the same input always produces the same hash value.
- *Fast computation:* it is efficient to compute the hash for any given input.
- *Collision Resistance* it is extremely unlikely for two different inputs to produce the same hash value.

The metadata of the uploaded documents will be stored in the smart contracts, and the document itself will be stored in the document cloud storage component. We did not choose to store the documents in IPFS due to the private nature of the information present in the documents. A Ricardian contract is a digital contract that combines a legal contract with a machine-readable contract, often used in blockchain technology to automate and verify contract terms [17]. They allow the inclusion of legal language and privacy terms within the smart contract. This makes it possible to specify and enforce GDPR-related obligations, such as data protection, consent, and especially the right to be forgotten, directly in

the contract code. Here are the steps to implement a Ricardian contract with a smart contract using a document hash:

1) Create a legal document with the terms and conditions.
2) Compute the hash of the document.
3) Embed the hash in the smart contract.
4) Implement logic to validate the document's hash in the smart contract.
5) Deploy the smart contract on a blockchain.
6) Maintain an audit trail of interactions on the blockchain.

C. System Processes

The system processes outline the core processes within the application and highlight key interactions among tenants, landlords, and auditors. The three main processes covered include tenant document uploads, document authorisation, and the audit process. These processes are essential components of the application and encompass user onboarding, permission management, and regulatory compliance verification.

1) Tenant Upload Documents Process: Figure 3 represents the first interaction with the application. A user with a tenant role registers and is authenticated on the platform. Subsequently, the user must accept the terms and conditions so that the system generates data for future audit requests. Then he proceeds to choose a house to rent and submits its documents. The metadata of the documents is then uploaded to the blockchain, and the original documents are uploaded to the Google document storage. The next steps, when the tenant and the landlord interact to handle the permissions on the uploaded documents, are discussed in Section IV-C2.

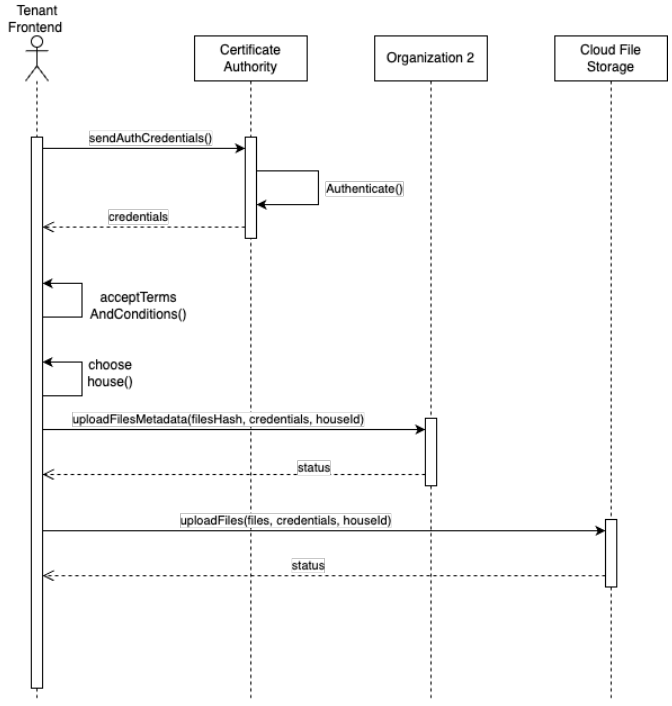

Fig. 3: Tenant documents upload process sequence diagram

2) Document Authorisation Process: Figure 4 illustrates the interaction following the submission of a tenant's proposal. The landlord, authenticated on the platform, selects a house with pending applications. He requests permission to view the original proposal documents, deploying a notification to the tenant. The tenant can either accept or decline the request. Upon acceptance, the landlord retrieves the original documents from Google Cloud storage. After reviewing documents from multiple applications, the landlord selects the best proposal, updating the proposal status for the tenant organisation.

For document integrity verification, the system retrieves the document's hash from the blockchain and compares it to the hash generated from the stored document. A match confirms the integrity of the document, while a mismatch indicates corruption. The landlord can then proceed to accept or deny the proposal application.

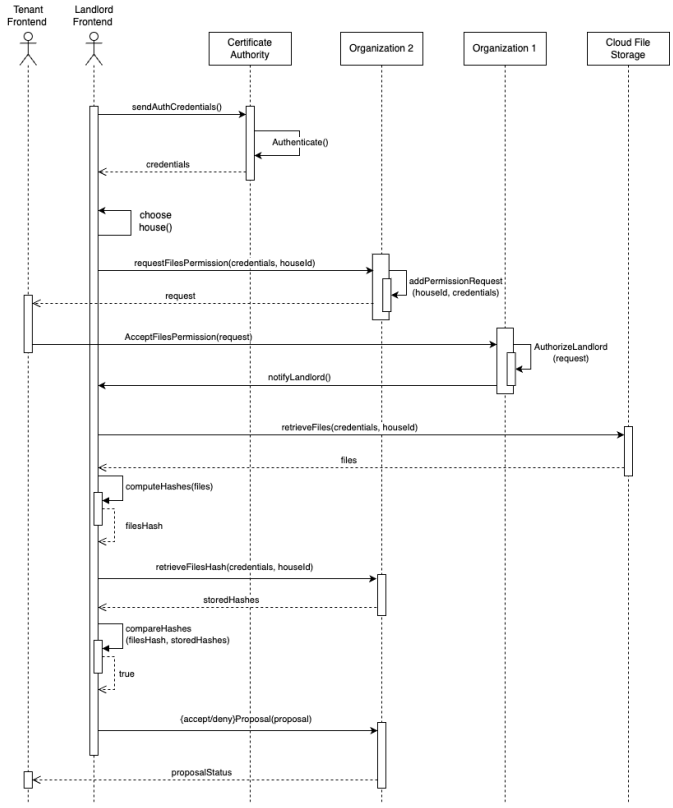

Fig. 4: Document authorisation process sequence diagram

3) Auditing Process: The final interaction occurs after the sequence of Figure 4. Now that transactions have occurred, a verified auditor can request a chosen house to audit and verify that everything is according to the regulations. The sequence starts with the authentication of the auditor (cf. Figure 5). A house is chosen and a request is made to retrieve the metadata of the documents. The frontend then generates a final report for the auditor to download.

V. BDM Implementation

In implementing this specific use case, an approach has been developed that takes advantage of the features of the Hyper-

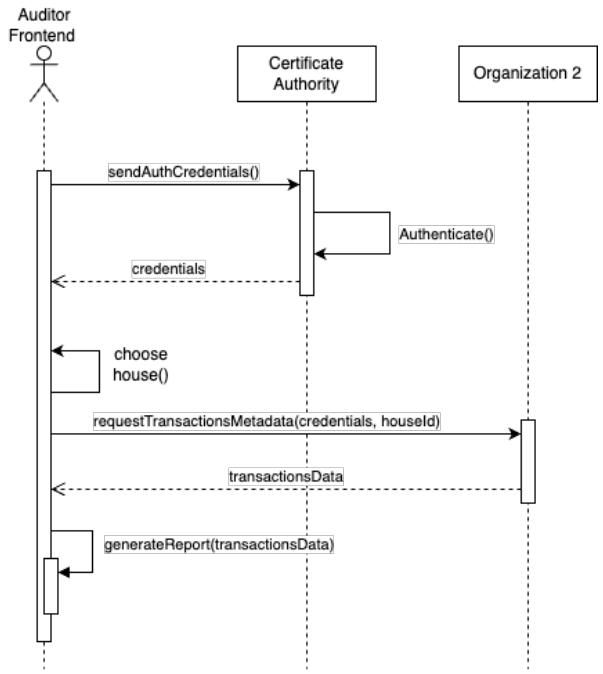

Fig. 5: Auditing process sequence diagram

ledger Fabric framework to seamlessly integrate blockchain technology. This implementation is further supported by the creation of a frontend application using React, a JavaScript framework. This Introduction lays the foundation for a thorough examination of the implemented approach and its essential components.

A. Blockchain Architecture

The network comprises distinct components: certificate authorities, organisations, an orderer, and channels, each with a defined role in the overall structure. These elements are explained next, and the blockchain infrastructure can be seen in Figure 6.

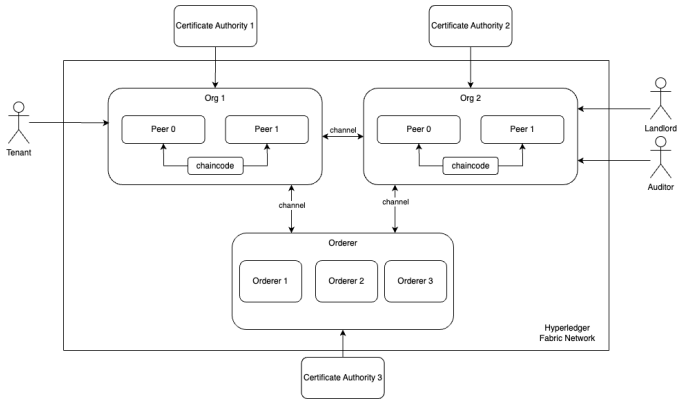

Fig. 6: Hyperledger Fabric implemented network architecture

- *Organization:* Organisations in this context are entities that define the participants in the network. Each organisation typically has its own set of peer nodes, a CA, and administrative control over its members.
- *Certificate Authority:* The CA issues digital certificates to network participants. These certificates contain cryptographic keys and are used to verify the identity of the nodes and maintain the integrity and confidentiality of transactions (Section V-B).
- *Chaincode:* Chaincode or Smart Contract, is a piece of code that defines the rules and logic for transactions on the blockchain. It is installed on peer nodes and can be invoked to modify the ledger state. Before chaincode can be used, it must go through an approval and commitment process. This involves an endorsement policy, where peers validate and approve the chaincode, and a commitment to the channel ledger. This ensures that all organisations agree on the legitimacy of the code.
- *Peer Nodes:* Peer nodes are individual instances within an organisation that maintain a copy of the ledger. They execute chaincode transactions, validate transactions, and endorse them before they are added to the blockchain. Having two peer nodes per organisation ensures redundancy and high availability.
- *Orderer Nodes:* Orderer nodes are responsible for maintaining the order of transactions on the blockchain. They validate transactions, create blocks, and ensure consensus among network participants. The consensus algorithm can be crash-fault-tolerant or byzantine-fault-tolerant. Three orderer nodes enhance fault tolerance and maintain the integrity of the ledger.
- *Channel:* A channel is a private communication layer in the blockchain network that allows the segregation of transaction data. It restricts access to specific organisations, ensuring that only authorised participants can view and transact on this channel.
- *Network APIs:* Application Programming Interfaces provide an interface for external applications to interact with the blockchain network. In this case, two APIs are deployed, each tailored to a specific organisation, allowing authorised users to send transactions and retrieve data from the blockchain.

In this architecture, tenants connect to Organisation 1 to input data into the ecosystem, while landlords and auditors connect to Organisation 2. They use this connection to view uploaded files or generate reports based on information extracted from blockchain transactions. Every transaction is signed by its author, and therefore non-repudiation is granted.

B. Authentication

Within the context of user identity management in Hyperledger Fabric, there exists a structured process to enable secure participation in the network. This process involves user registration, during which individuals provide vital information such as their username, password, and role. After successful registration, users are equipped with cryptographic credentials, namely an X.509 certificate and a private key. These credentials establish their secure digital identity within the network. The user identity is securely stored in a wallet, protecting cryptographic keys and certificates from unauthorised access.

When users intend to log in, the verification of their identity takes place through the CA, using the certificate and private key stored in the wallet for authentication. This meticulous process ensures that only authorised users, possessing valid credentials, gain access to the blockchain network, thus ensuring the security and reliability of interactions.

- *Registration with the CA:* The code includes a registration process that allows new users to join the Hyperledger Fabric network securely. When a user wishes to register, they provide essential information, such as a username, password, and role. The code first checks if the user identity already exists within the CA. If not found, it proceeds with the registration. During registration, the user enrolment ID and secret, often chosen by the user during signup, are used. These credentials are crucial to authenticating the user within the network.
- *Enrolment and Identity Creation:* Following successful registration, the code initiates the enrolment process. This step involves obtaining cryptographic credentials for the user, namely, an X.509 certificate and a private key. These credentials serve as the user digital identity within the Hyperledger Fabric network. The enrolment process ensures that the user identity is securely generated and linked to the CA. This identity creation process is an integral part of ensuring secure and authenticated interactions with the blockchain network.
- *Storage in the Wallet:* Once the user identity is generated and enroled with the CA, it is securely stored in a wallet. The wallet acts as a secure repository for user identities. It ensures that cryptographic keys and certificates are protected from unauthorised access. Users can conveniently access their identities from the wallet for subsequent interactions with the network. This secure storage mechanism is vital to maintaining the confidentiality and integrity of user credentials.
- *Log in with the CA and Wallet:* When a user wants to log in, the code checks the CA to verify the user identity. If the identity is found, the user X.509 certificate and private key stored in the wallet are used for authentication. This login process ensures that only authorised users with valid credentials can access the blockchain network. It also provides a secure and convenient way for users to participate in blockchain transactions and queries.

The authentication phase provides security by generating and storing user identities, ensuring that only authorised users can interact with the Hyperledger Fabric network. These processes are essential to maintain the integrity and confidentiality of blockchain transactions and user data.

C. Cloud File Storage

Google Cloud File Storage was selected for file storage and retrieval based on personal experience with the technology. A Google Service Account is essential for secure and automated access to Google Cloud services, enabling the application to interact with Google resources without user passwords. This is crucial for data processing, server-to-server communication, and integrating the application frontend with Google Cloud services. The main storage features include:

1) Store Files:

- *Upload Files:* upload files to Google Cloud File Storage using Google Service Account credentials through the Web interface.
- *Organize Files:* organise files into folders, all managed by the Google Service Account, ensuring a well-maintained and structured storage system.
- *Permission settings:* precise control over access permissions, granting read-only, read-write, or customised access to specific users or groups through the Google Service Account.

2) Retrieve Files:

- *Access Anywhere:* retrieve stored files from Google Cloud Storage using the Google Service Account within the Web interface.
- *Search and Retrieve:* locate files within the React application by employing keywords or parameters in the search function.
- *Permission settings:* enforce strict access control during file retrieval, allowing only authorised users authenticated by the Google Service Account to view or modify files.

D. Smart Contract Implementation

The application's core functions are divided into write and read functions, enabling users to interact with documents, houses, proposals, and access requests. These functions are crucial to creating, managing, and retrieving historical data, ensuring effective and secure user interactions.

1) Implemented Functions:

- *createHouse:* Allows users to create a new house associated with a landlord, subject to necessary permissions and checks.
- *createProposal:* Enables tenants to create rental proposals for houses and landlords, following permission and existence checks.
- *denyProposal:* Permits landlords to reject tenant rental proposals for specific houses, after verifying permissions and existence.
- *acceptProposal:* Allows landlords to accept tenant rental proposals, subject to checks and permissions.
- *getRequestsForTenant:* Retrieves access requests made by tenants after ensuring caller existence and permissions.
- *createDocument:* Tenants create documents related to rented houses, provided they have the necessary permissions and meet house-related criteria.
- *requestAccess:* Tenants request access to specific documents from landlords, subject to various checks.
- *acceptAccess:* Tenants grant access to landlords for specific documents, ensuring permissions and existence.
- *denyAccess:* Tenants deny access to landlords for specific documents, following checks and permissions.
- *getDocument:* Allows users to retrieve document details with proper access rights and after confirming document existence and permissions.

- *getProposalsForLandlord:* Retrieves rental proposals made to a specific landlord, provided that the landlord exists and has the necessary permissions.
- *getHistoricData:* Offers comprehensive historical data, including document metadata history, proposal history, and access request history. It checks various conditions for each aspect to ensure that data retrieval is valid and secure.

E. User Interfaces

For various types of users interacting with the system, a frontend has been designed featuring three separate interfaces: one for tenants, another for landlords, and a third for auditors. These interfaces provide a simple way for tenants to apply for properties and manage them, landlords to manage their properties and tenant applications, and auditors to review every transaction for a certain house rental. The frontend has been developed in React version v18.2.0, a JavaScript framework chosen for its flexibility and performance.

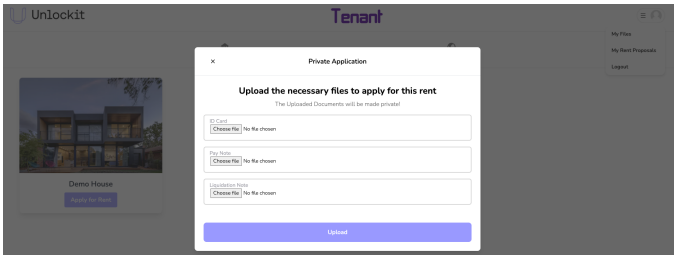

Fig. 7: Tenant document upload view

Figure 7 displays the Tenant Document Upload View, where tenants can securely upload the necessary documents for their rental application. This figure illustrates the document management capabilities of the interface, which streamlines the process for tenants while ensuring the secure storage of important documentation.

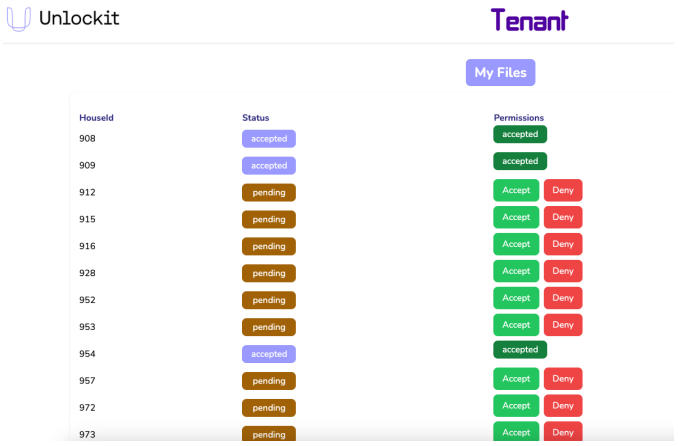

Fig. 8: Tenant document authorisation view

The Tenant Documents Authorisation View, as shown in Figure 8, where possible tenants manage and authorise access to their uploaded documents. This figure hints at features that allow tenants to grant access to these documents to landlords or other authorised parties, maintaining control over their information.

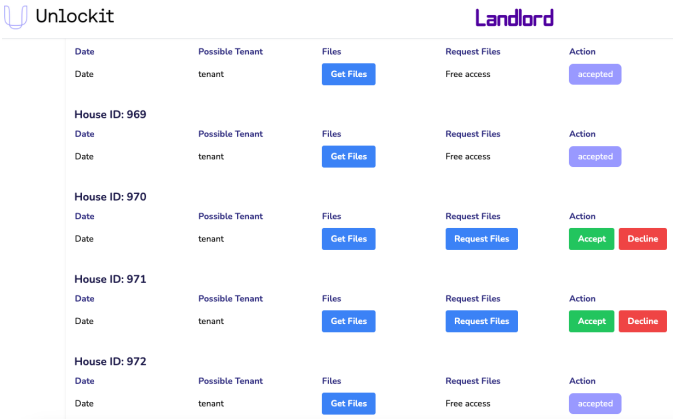

Fig. 9: Landlord rental proposals view

Figure 9 presents the Landlord Rental Proposals View, which showcases the interface designed for landlords. This interface likely enables landlords to review tenant applications and manage rental proposals efficiently. Provides a comprehensive overview of tenant applications, helping landlords in their decision-making process.

F. Audit Report

The blockchain system's audit report meticulously records all transactions related to a specific house, offering a unique capability to track the rental process. This transparency and precision simplify future audits. The built-in Hyperledger Fabric function *getHistoryByKey* is used to achieve this functionality. It provides the complete history of an object using a key representing a house, its landlord, and tenants. Accessing the history of the *house* object reveals the entire transaction lifecycle of that specific rental.

Authorised auditors can obtain detailed records of the rental process, from application to approval. Figure 10 presents a simple audit report for a rental application.

VI. Evaluation

A. Methodology

Apache JMeter is a versatile tool for testing server-based applications. It provides features like result trees and aggregate reports to analyse performance. The result trees show details of the execution of the HTTP request, which aids in the identification of the issue. Aggregate reports compile metrics across multiple test runs, revealing insights into response times, throughput, and errors.

B. Experimental Setup

Figure 11 presents the Hyperledger Fabric network infrastructure, hosted within a locally deployed Kubernetes cluster. This deployment relies on KinD (Kubernetes in Docker), a tool that facilitates the creation and management of Kubernetes clusters using Docker containers as nodes. The process is

Audit Report

Audit Report Creation Date: Sun Oct 22 2023 20:05:38 GMT+0100 (Western European Summer Time)

File request permissions log

landlord retrived the file with category: 2 and owner: tenant in house with Id: 181 at 2023-07-23T11:04:58.277Z

landlord got accepted by tenant to access file with category: 2 in house with Id: 181 at 2023-07-23T11:04:26.705Z

landlord request to access tenant file with category: 2 in house with Id: 181 at 2023-07-23T11:04:01.992Z

landlord got accepted by tenant to access file with category: 1 in house with Id: 181 at 2023-09-28T11:31:24.214Z

landlord request to access tenant file with category: 1 in house with Id: 181 at 2023-09-28T11:07:56.626Z

landlord retrived the file with category: 1 and owner: tenant in house with Id: 181 at 2023-07-23T11:04:58.290Z

landlord got accepted by tenant to access file with category: 1 in house with Id: 181 at 2023-07-23T11:04:26.688Z

landlord request to access tenant file with category: 1 in house with Id: 181 at 2023-07-23T11:04:01.984Z

landlord retrived the file with category: 3 and owner: tenant in house with Id: 181 at 2023-07-23T11:04:58.285Z

landlord got accepted by tenant to access file with category: 3 in house with Id: 181 at 2023-07-23T11:04:26.697Z

landlord request to access tenant file with category: 3 in house with Id: 181 at 2023-07-23T11:04:01.998Z

Fig. 10: Audit report example

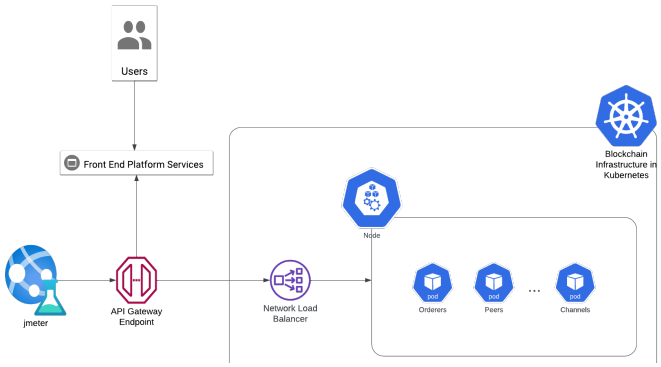

Fig. 11: BDM Kubernetes diagram

streamlined through the use of the HLF Operator, a Kubernetes operator designed to simplify the deployment and management of Hyperledger Fabric networks within Kubernetes clusters.

C. Experimental Results

Evaluation of system performance provided critical insight into how the system behaves under varying loads of concurrent requests. Eight functions were examined, with request loads ranging from 50 to 1000 concurrent requests. A consistent trend emerged: As the number of concurrent requests increased, the system latency increased, leading to decreased throughput. This well-documented inverse relationship between latency and throughput was observed. Figures 12 and 13 present specific findings related to individual functions, revealing their strengths and vulnerabilities with increasing concurrent requests. These results serve as a basis for optimising the system for real-world scenarios with dynamic workloads.

As expected, the results demonstrate that as concurrent requests increase, system latency increases while throughput decreases. The increase in latency with higher concurrency

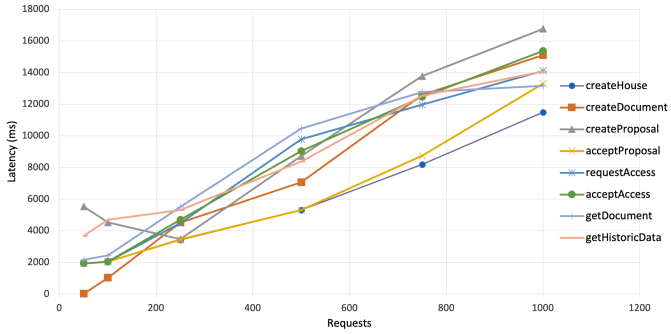

Fig. 12: Latency for each implemented function

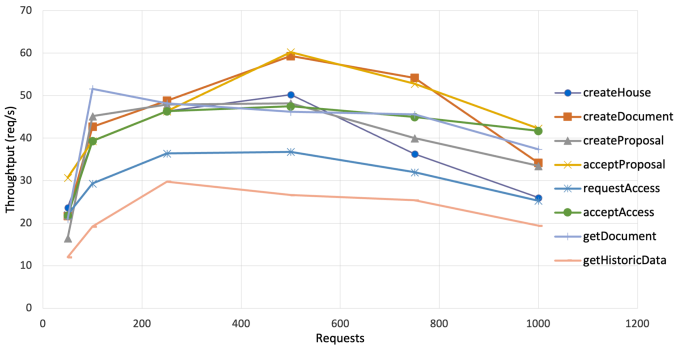

Fig. 13: Throughput for each implemented function

can be attributed to the finite resources of the system being spread among multiple requests, resulting in delays in processing, leading to higher response times. Concurrently, reduced throughput indicates that the system handles fewer requests per second as the levels of concurrency increase. This is due to the longer time required to process each request, which limits the overall capacity of the system. At the latency level, the functions constitute more or less the same latencies per number of requests, with the createProposal function being the slowest one. The getHistoricData function is the one with the lowest throughput, as expected, since it must carry for each historic transaction its content, resulting in the usage of more bandwidth per request. On the other hand, acceptAccess and acceptProposal are the ones with the highest throughput for 1000 requests, since they require less data. At high request levels, the system encounters issues like denying or dropping connections, impacting the above results, leading to request exceptions. These exceptions cause some requests to terminate prematurely, resulting in response times shorter than anticipated. For a complete overview of system malfunction across functions and concurrency levels, Table I is provided.

TABLE I: Error percentage per function for different numbers of requests

# requests	createHouse	createDocument	createProposal	acceptProposal	requestAccess	acceptAccess	getDocument	getHistoricData
50	0	0	0	0	0	0	0	0
100	0	0	0	0	0	0	0	0
250	0	0	0	0	0	0	0	0
500	0	0	0	0	0	0	0	0
750	0.15	78.53	0	0	0.6	0.6	0	2.3
1000	2	80	0	0	5	4	2	12

Most functions begin to encounter errors after around 750

requests. The *createDocument* function has a notably higher error rate compared to other functions because it handles the storage of document metadata, resulting in larger data transactions per request. In contrast, the functions *createProposal* and *acceptProposal* have no errors, as they are simpler and involve smaller data transactions. *createHouse*, *requestAccess*, *acceptAccess*, and *getDocument* functions exhibit low error levels even at 1000 requests. However, *getHistoricData* has a 12% error rate of 1000 requests due to larger data responses, occasionally causing connection drops.

D. Current Process vs Smart Audit Process

The BDM with audit support system represents an advance over current manual processes. In the current system, auditors spend a substantial amount of time and effort manually collecting, validating, and reconciling data from various sources. This is a laborious and time-consuming task. However, with the introduction of blockchain, these tasks are performed on top of verified data, freeing auditors from repetitive work and enabling them to focus on more strategic aspects of auditing.

Furthermore, the use of blockchain technology ensures the accuracy and reliability of the data. In the manual system, the risk of human error is a constant concern that requires additional effort to verify the data. In contrast, the blockchain immutable ledger guarantees the integrity of information, reducing the need for extensive error checking and increasing efficiency.

Collaboration and transparency are greatly improved by the blockchain-based platform. Current collaboration methods with stakeholders can be malicious and lack transparency. However, the system offers a secure platform for all parties to contribute and access data transparently with the necessary authorisation, facilitating seamless communication and improving transparency.

In addition, the system supports security and compliance efforts. Ensuring data security and compliance can be challenging in the current system. However, the cryptographic security features of the blockchain and the audit trail capabilities improve data security and compliance, reducing potential legal and financial risks.

Finally, the system results in significant time and cost savings. Manual processes are resource-intensive and can lead to high costs. Automation, real-time data access, and improved accuracy result in significant time and cost savings for auditors, allowing them to allocate resources more efficiently and strategically.

VII. Conclusion

This document introduces a solution to streamline the house rental process and audit reporting in the real estate market, with the aim of enhancing transparency and efficiency for tenants, landlords, and auditors. The solution leverages blockchain technology for added security and efficiency. Key functions in the system include creating and accepting rental proposals, accessing and verifying documents, and maintaining the integrity of uploaded files through hashing. Performance evaluation of the system involved measuring latency, throughput, and errors. The results indicate the system's capability to handle up to 500 concurrent transactions without errors. In addition, the document discusses how this technology can make auditors more efficient by automating repetitive tasks, comparing it with existing processes.

Acknowledgements

This work was financially supported by Project Blockchain.PT – Decentralize Portugal with Blockchain Agenda, (Project no 51), WP 6, Call no 02/C05-i01.01/2022, funded by the Portuguese Recovery and Resilience Program (PPR), The Portuguese Republic and The European Union (EU) under the framework of Next Generation EU Program. This work was also supported by national funds through Fundação para a Ciência e Tecnologia (FCT) with reference UIDB/50021/2020 (INESC-ID).

References

[1] A. Kilgore, R. Radich, and G. Harrison, "The relative importance of audit quality attributes," *Australian Accounting Review*, vol. 21, no. 3, pp. 253–265, 2011.

[2] M. E. Peck, "Blockchains: How they work and why they'll change the world," *IEEE Spectrum*, vol. 54, no. 10, pp. 26–35, 2017.

[3] A. Narayanan, J. Bonneau, E. Felten, A. Miller, and S. Goldfeder, *Bitcoin and Cryptocurrency Technologies: A Comprehensive Introduction*. Princeton University Press, 2016.

[4] Z. Zheng, S. Xie, H.-N. Dai, W. Chen, X. Chen, J. Weng, and M. Imran, "An overview on smart contracts: Challenges, advances and platforms," *Future Generation Computer Systems*, vol. 105, pp. 475–491, 2020.

[5] M. Correia, "From byzantine consensus to blockchain consensus," in *Essentials of Blockchain Technology*. CRC Press, 2019, ch. 3.

[6] Digital Asset Canton Team, "Canton: A Daml based ledger interoperability protocol," https://www.canton.io/publications/canton-whitepaper.pdf.

[7] E. Androulaki, A. Barger, V. Bortnikov, C. Cachin, K. Christidis, A. De Caro, D. Enyeart, C. Ferris, G. Laventman, Y. Manevich *et al.*, "Hyperledger Fabric: a distributed operating system for permissioned blockchains," in *Proceedings of the 13th ACM EuroSys Conference*, 2018.

[8] R3 Corda Team, "R3 corda," https://www.r3.com/, accessed: 09.10.2022.

[9] CFI Team, "An objective examination and evaluation of a company's financial statements," https://corporatefinanceinstitute.com/resources/knowledge/accounting/what-is-an-audit/.

[10] S. Harvey, "5 reasons why an internal audit is important," https://kirkpatrickprice.com/blog/5-reasons-why-internal-audit-is-important/.

[11] Smart Audit Team, "SmartAudit - audit workflow software," https://smartaudit.co/, accessed: 23.07.2024.

[12] P. Kang, W. Yang, and J. Zheng, "Blockchain private file storage-sharing method based on IPFS," *Sensors*, vol. 22, no. 14, p. 5100, 2022.

[13] H. P. Wouda and R. Opdenakker, "Blockchain technology in commercial real estate transactions," *Journal of property investment & Finance*, vol. 37, no. 6, pp. 570–579, 2019.

[14] P. K. Bharimalla, H. Choudhury, S. Parida, D. K. Mallick, and S. R. Dash, "A blockchain and NLP based electronic health record system: Indian subcontinent context," *Informatica*, vol. 45, no. 4, 2021.

[15] R. Belchior, M. Correia, and A. Vasconcelos, "Towards secure, decentralized, and automatic audits with blockchain," *In Proceedings of the European Conference on Information Systems (ECIS), June 2020*, 2020.

[16] B. Preneel, "Cryptographic hash functions," *European Transactions on Telecommunications*, vol. 5, no. 4, pp. 431–448, 1994.

[17] I. Grigg, "The ricardian contract," in *Proceedings 1st IEEE International Workshop on Electronic Contracting*, 2004, pp. 25–31.