TÉCNICO LISBOA

Blockchain Based Residential Smart Rent

André da Silva Proença

Thesis to obtain the Master of Science Degree in

Computer Science and Engineering

Supervisor(s): Prof. Miguel Nuno Dias Alves Pupo Correia
Tiago Rogério Romeira Dias

Examination Committee

Chairperson: Prof. Manuel Cabido Lopes
Supervisor: Prof. Miguel Nuno Dias Alves Pupo Correia
Member of the Committee: Prof. Miguel Filipe Leitão Pardal

December 2023

Dedicated to my parents and to the people who genuinely care about me.

Acknowledgments

As I am on the verge of turning the page of an important chapter in my life, I would like to express a few words of appreciation in this document, since it represents the culmination of my long and arduous academic journey. I dedicate these words to people who really made a difference in my life, well-being and positively influenced my personality and attitude.

First and foremost, I warmly thank my dear, loving parents, who always smiled at my victories, always rectified my mistakes and always dared to dream with me. I thank them from the depth of my heart, for all their affection and support throughout my journey. A word of appreciation also to my brother who never failed me and always showed me his unconditional support.

I would like to express my gratitude to three entirely different people who were the main support for this thesis, and without them this work would not have been possible.

Firstly, I would like to thank Prof. Miguel Nuno Pupo Correia, who I consider to be an outstanding, respectful and supportive professor. For his guidance, patience and for sharing his vast knowledge of this new world of the blockchain technology, I thank him.

Secondly, I would like to thank Tiago Dias, the driving force behind the whole idea of this thesis. The person I believe to be a real estate visionary, a person who is highly motivated to achieve his goals and an entrepreneur with a ton of drive and ambition. I appreciate his support and vision, upon which the structure of this work was based.

And thirdly, I would like to thank João Santos, a university colleague thirsty for knowledge about the decentralized world, and whom, I consider to be someone with principles and values similar to mine which I truly respect. I thank him for his endless support and advice.

Finally, I would like to pay my tribute to the Instituto Superior Técnico for being a second home and for providing me with the conditions to thrive and to achieve academic success.

This work was developed within the scope of the project nr. 51 “BLOCKCHAIN.PT - Agenda Descentralizar Portugal com Blockchain”, financed by European Funds, namely “Recovery and Resilience Plan - Component 5: Agendas Mobilizadoras para a Inovação Empresarial”, included in the NextGenerationEU funding program.

Resumo

A popularização da tecnologia blockchain e as suas principais propriedades têm incentivado o desenvolvimento de novas ideias de negócios, e a reestruturação de negócios tradicionais, como o sector imobiliário. O mercado imobiliário inclui processos de mediação complexos e ineficientes. O aluguer de um imóvel envolve múltiplas entidades com diferentes responsabilidades e interesses. Por conseguinte, é imperativo estabelecer uma relação de confiança entre as partes através de intermediários, tais como, notários, bancos, agências imobiliárias ou escritórios de advogados para evitar eventuais litígios. Embora um intermediário garanta confiança, o processo atual apresenta ainda alguns inconvenientes em termos de eficiência, custos, e segurança dos dados. Propomos uma plataforma de arrendamento residencial inteligente assente em blockchain que permite criar e gerir contratos de arrendamento residenciais com um elevado nível de privacidade, segurança e conformidade com o Regulamento Geral Europeu de Proteção de Dados (GDPR). A introdução da tecnologia blockchain pretende reduzir a intermediação de custos elevados, tempo gasto em burocracias e resolver falhas de segurança de dados, proporcionando mecanismos antifraude transparentes e transacções imobiliárias seguras. A plataforma proposta permite que proprietários anunciem as suas propriedades e que possíveis inquilinos as consultem e apresentem propostas de arrendamento. Além disso, permite assinar digitalmente e gerir contratos bem como realizar pagamentos automáticos mensais ou únicos, utilizando stablecoins como método de pagamento.

Palavras-chave: Blockchain · Smart Contracts · Real Estate · Smart Rent · GDPR

Abstract

The popularization of blockchain technology and its key properties have encouraged the development of new business ideas and the restructuring of traditional businesses such as real estate. The real estate market includes complex and inefficient mediation processes. Renting a property envolves multiple entities with different responsibilities and interests. Therefore it is imperative to establish a trustful relationship between parties through intermediaries such as notaries, banks, real estate agencies, or law firms to avoid eventual disputes. Although an intermediary ensures trust, the current process still has some drawbacks concerning efficiency, costs, and data security. We propose a blockchain-based residential smart rental platform, that allows to create and manage residential rental contracts with a high level of privacy, security and compliance with the European General Data Protection Regulation (GDPR). The introduction of the blockchain technology aims to reduce intermediation high costs, time spent on bureaucracy and address data security flaws, providing transparent anti-fraud mechanisms, and secure and reliable real estate transactions. The proposed platform allows landlords to advertise their properties and potential tenants to consult them and submit rental proposals. Moreover, it allows to digitally sign and manage contracts as well as make automatic monthly or one-time payments, using stablecoins as a method of payment.

Keywords: Blockchain · Smart Contracts · Real Estate · Smart Rent · GDPR

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Blockchain technology in recent years has been growing in popularity at a vertiginous pace, mainly due to the 2008 white paper entitled *Bitcoin: A Peer to Peer Electronic Cash System* [Nak08] published pseudonymously by *Satoshi Nakamoto*. This whitepaper proposed an innovative network which became the first ever blockchain application, broadening horizons by unlocking a whole new spectrum of innovative possibilities reshaping some sectors of society, such as the real estate.

The Blockchain technology key features include network decentralization, data persistence, data immutability, participant anonymization, and transaction traceability [YMRS19]. Together they compose a secure system that is increasingly attracting new business ideas. Sectors such as government, private management, healthcare, insurance, electronic voting, and real estate are slowly converting their old business models to the blockchain. In the real estate industry, the incorporation of blockchain technology into the current business process has the potential to revolutionize the way real estate transactions are conducted. The Blockchain key features allow solving traditional problems related to the acquisition, sale, and rental of real estate assets, such as high intermediation costs, time-consuming bureaucracy, lack of standardized processes, and the inability to guarantee information security since information is shared through multiple insecure channels.

The traditional property rental process can be adapted to the blockchain by applying smart rental contracts which are a type of smart contracts. The term smart contract designates executable code residing and executing on the blockchain to automatically enforce the terms of a contract when certain conditions are met. Smart contracts' self-verifying, self-executing, tamper-proof, traceable and irreversible properties [MPJ18] make them a reliable and robust solution for developing blockchain-based real estate projects.

The establishment of a rental requires an agreement between at least two parties that should be sealed through a contract. Renting a property is a complex process not only in terms of regulations but also in terms of lease type and duration. A property can be divided into multiple fractions, and is therefore susceptible to different types of leases. Additionally, a rental may have different durations which further increases its complexity. A property as well as its fractions can be represented through tokens. A Token is seen as a digital representation of a real asset's economic value [VBS20]. Therefore it is possible to tokenize a property into one or multiple tokens for renting all or part of a property.

Despite the technological innovations and advancements, legislation and regulations in the great majority of nations have yet to be discussed or implemented, making the process of renting real estate assets through blockchain highly unpredictable under the law, as real estate is a highly regulated sector. Nevertheless, the development of applications within the European Union must comply with the general data protection regulation (GDPR [Eur16]). As a consequence, privacy issues must be taken very seriously when building GDPR-compliant blockchain-based systems.

This dissertation presents a platform design, conceived for Unlockit, a DLT company operating in the real estate industry. We propose a blockchain-based residential smart rental platform, that allows to create and manage residential rental contracts with a high level of privacy and compliance with the European general data protection regulation (GDPR). The proposed platform permits landlords to advertise properties and tenants to browse them before engaging into a rental agreement. The platform provides a graphical user interface, serving as an interface between users and the blockchain network. The interface allows users not only to advertise and visualize properties, but also to manage, digitally sign rental contracts and automate monthly or one-time payments through the support of stablecoins such as USDC or USDT, as a method of payment.

1.1 Objectives

This document proposes a blockchain-based platform aiming to address the existing problems of renting a residential property, taking into account factors such as GDPR compliance and country legislation. Ultimately, we aim for a platform assuring the following requirements:

1. ***Blockchain*** - The platform should adopt a blockchain to allow transparent, auditable and secure transactions through a decentralized system.

2. ***GDPR compliance*** - The platform should comply with the European general data protection regulation (GDPR).

3. ***Different rental types*** - The platform should support multiple contracts, accounting for diverse property rental types and durations.

4. ***Automated payments*** - The platform should incorporate a reliable and automated method for processing rental payments. It should offer monthly and one-time payment options.

5. ***User Interface*** - The platform should provide a user-friendly interface. It should facilitate contract creation, management, signing, and payment execution.

1.2 Thesis Outline

The remainder of the document is structured as follows. **Chapter 2** introduces necessary prior concepts of real estate, blockchain and GDPR compliance, allowing for a clear comprehension and application of our platform. **Chapter 3** proposes a platform for the problem, beginning by describing the design and infrastructure requirements, followed by discussing the architecture components and closing by demonstrating the architecture's implementation. **Chapter 4** presents the methodology for evaluating and validating the platform and their respective results. Finally, **Chapter 5** describes the plans outlined for the future development of this work and concludes by summarizing its objectives and achievements.

Chapter 2

Related Work

In this chapter we present the related knowledge required to understand our platform. We start by introducing in **Section 2.1** general real estate concepts, going deeper into further concepts related to our target topic such as residential renting and its rental types and durations. In **Section 2.2** we address the concept of blockchain as well as its permission categories. Next, in **Section 2.2.2** and **2.2.3** we introduce the blockchain underlying concepts such as smart contracts, and smart contract platforms. In the following **Section 2.3**, we dive into a detailed analysis of the Hyperledger Fabric permissioned blockchain. Then, in **Section 2.4**, we introduce the concept of wallet, explaining its importance and relationship with the blockchain. Afterwards, in **Section 2.5** we explain the GDPR's relevance and its applicability to the blockchain technology. Next, in **Section 2.6** we discuss some general blockchain use cases followed by introducing the concept of tokenization of real estate assets, and lastly, we discuss and compare specific blockchain use cases in real estate.

2.1 Real Estate

The oldest documented history of real estate business is estimated to have been discovered in the Stone Age approximately 12,000 years ago, in inscriptions and cave drawings. It is believed that even back then, shelter owners rented out their shelters in exchange for some form of currency or essential goods. As time went by, civilizations established land purchase and ownership rights. Property ownership and acquisition were a sign of wealth and social status, as well as a vital source of income. Real estate is undoubtedly a structural element of any civilization and highly regulated by governments. Nowadays, it is common to sell, acquire, and rent properties to single or multiple individuals through detailed and structured contracts which comply with extremely detailed rules, rigorously approved by the world's most diverse governments.

2.1.1 Residential Renting

The real estate business is vast and complex, therefore this document is dedicated exclusively to exploring solutions for renting private residential properties. According to Eurostat [Eur21] in **Figure 2.1**, the latest data reveals that in 2020, 70% of the EU population lived in a home of their own, while the remaining 30% lived in rented houses. Portugal, in particular, reported a population percentage of 77.3% living in its own house and 22.7% living in rented houses.

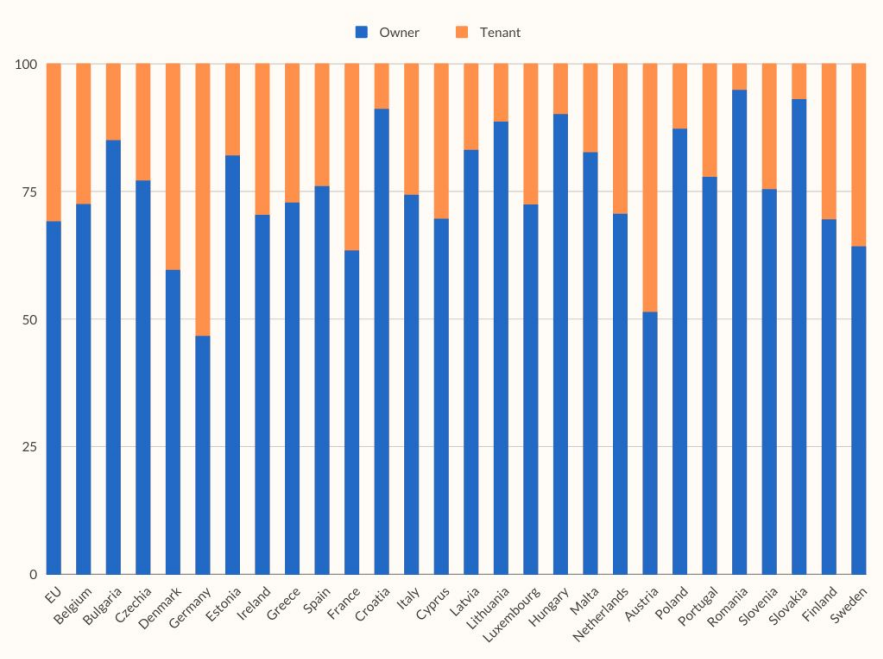


Figure 2.1: Percentage of people living in the EU in owned or rented households in 2020 [Eur21]

Renting a residential property requires a written contract, agreed upon and signed by two parties, a tenant and a landlord. A contract should outline the terms and conditions of a rental, which includes the obligations and rights of both parties, respecting the laws of the country or state in which the property is located. A rental agreement primary goal is to legally assist landlords and tenants in avoiding disputes. If conflicts eventually arise, the contract should address them. Rental contracts can be quite complex and contain a wide range of regulations and specifications concerning the property and the parties involved. However, all contracts should contain a minimum set of mandatory information to identify the parties and set out in clear terms the purpose of the agreement. Although a contract may go into much further detail, it should contain at least the following data depicted in **Table 2.1**

Table 2.1: Essential data for a residential rental contract

Tenant	**Landlord**	**Property**	**Rental**
Identification	Identification	Address	Beginning date
Contact	Contact	Typology	Duration
Guarantor		Area	Type
		Other	Amount
			Deposit
			Payment frequency
			Payment schedule
			Rental rights
			Obligations

When establishing a rental agreement for a residential property, it is important to clearly understand and distinguish between its distinct categories. The duration of the agreement is typically what differentiates them. **Table 2.2** depicts the different types of rentals.

Table 2.2: Types of rental agreements

Rental Type
Fixed-Term
Automatic Renewal
Month-to-Month
Short-Term or Seasonal
Subrental
Room Rental

Fixed-term rentals have an expiration date. This type of agreement is used when a tenant agrees to rent a property for a specific period of time at a predetermined fee. Fixed-term rentals employ calendar dates to determine the rental's start and termination dates. The agreement expires when the set term expires. At that time, the tenant and landlord may sign a new rental agreement with updated dates and other details. Nevertheless, the tenant may opt to leave at that moment [Con22].

Automatic renewal is a form of rental that automatically renews after a set period of time. These rental agreements are in effect until either the tenant or the landlord notifies the other party in advance that they wish to terminate the agreement [Con22].

Month-to-month rentals benefit landlords who are unwilling to compromise on renting their property for extended periods of time, giving them more freedom to manage their property on a month-to-month basis [Con22].

Short term or **seasonal** contracts are used to rent a property for short periods of time, usually during holiday periods. Such contracts are usually held for a few days, but should not exceed 31 days [Con22].

Subrental agreements allow tenants to sublet a rental property to another tenant with the landlord's consent. This type of tenancy is ideal when tenants want to sublet their rented property for brief periods of time while relocating so that they don't compromise their tenancy agreement [Con22].

Room rental agreements are more detailed contracts in which a tenant agrees to share the property with single or multiple tenants. These agreements require more detail regarding property partitions, utility, and cohabitation with other tenants [Con22].

Traditionally, a real estate transaction requires intermediaries and public services such as real estate agents, notaries, and land registries. Rental contracts, in general, do not demand the engagement of these intermediaries, however, they are frequently validated with the support of real estate agents and lawyers. In most countries they are not forced to be registered in land registries, however, for tax and legal purposes, they must be reported to the competent authorities [GT20]. As a result, the engagement of those intermediaries in rental agreements is not a common procedure. As technology currently evolves, the most recent trend lies in prototyping new real estate ideas in the blockchain technology. Blockchain aspires to behave like an intermediary, capable of notarising rental agreements by ensuring their validity and certifying that they were agreed upon on a certain date. It intends to provide a transparent, immutable, traceable, and intermediary-free rental system capable of reducing costs, bureaucracy, tax evasion, preventing fraud and vulnerable tenants from legal evasion. [GT20]. In the following section, we will go over how blockchain technology operates, its permission types, and how it can benefit the residential rental industry.

2.2 Blockchain

Since 2008, blockchain technology has grown in popularity, especially due to the valuation of the Bitcoin [Nak08] network, a result of Nakamoto's white paper publication, where he described an innovative cryptocurrency network becoming not only the first ever blockchain application, but also, the first ever application to solve the double-spending problem, where previously, one unit of currency was spent simultaneously more than once. Despite the fact that its popularity has only increased since 2008, the technology was not born upon the appearance of Bitcoin. Its principles and mode of operation originally emerged in the late 1980s and early 1990s [YMRS19].

A blockchain also commonly referred to as a distributed ledger is essentially an append-only data structure, similar to a distributed database, responsible for maintaining the transaction history, yet its transactions values cannot be rewritten once confirmed. The distributed ledger has as participants, a group of peers, also known as nodes, who do not fully trust each other. They are in charge of controlling, storing, and sharing information, enabling a peer-to-peer system without a central data management authority, also known as an intermediary or trusted third party. Traditionally, such third parties refer to major institutions such as banks, governments, or currency-issuing entities that seize control and secure transactions. The blockchain allows for the absence of such institutions due to three important pillars: Firstly, the distributed architecture, made possible through the use of both distributed ownership as well as a distributed physical architecture. Secondly, the cryptographic mechanisms, namely hash functions and digital signatures which safeguard the integrity of transactions and conformity between blocks. And lastly, the consensus algorithms established between nodes to ensure the system reliability.

Blockchain, as the name implies, is a continuously growing sequence of cryptographically chained blocks storing data. New blocks are inserted by peers who own a copy of the blockchain, and all must come to a consensus on its state before updating it. The blockchain is composed by three core pieces, the blocks, the chain, and the network. A block is mainly a set of transactions recorded in a ledger during a certain period of time. Each block is chained cryptographically with the preceding block, in which every new block contains in its header the hash of the previous block's header, except for the first block designated 'genesis', which has no previous block and therefore no hash, hence the initial hash is normally set to all zeros. From a simple perspective, a block contains two sections, the header and the body.

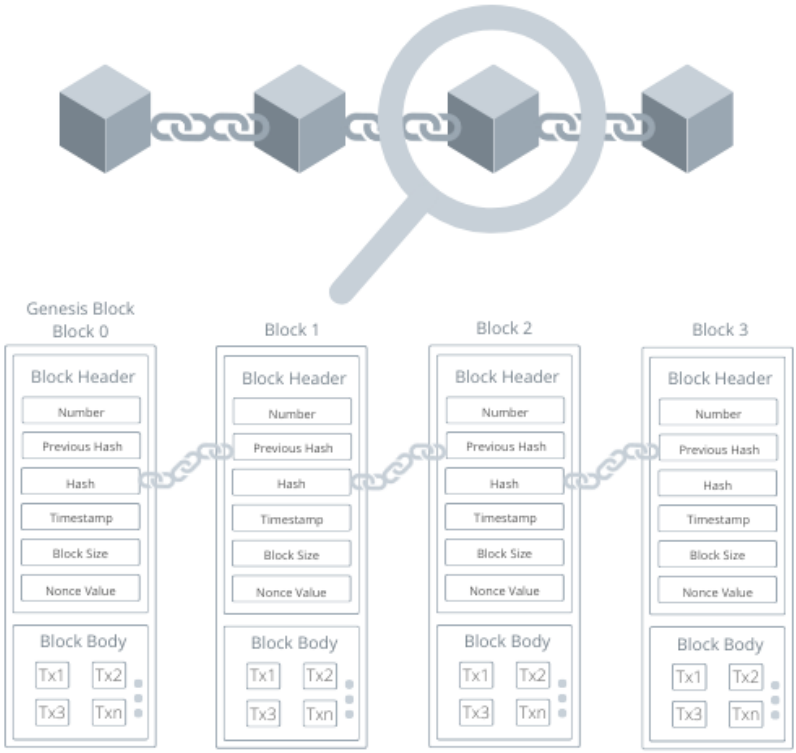


Figure 2.2: The blockchain structure in detail

As shown in the **Figure 2.2**, the block header carries important metadata that uniquely identifies it, whereas the block body holds a list of validated and authentic transactions that have been registered to the network. It may also contain other additional information. **Table 2.3** further clarifies the concepts.

Table 2.3: The structure of a block

Block Header	**Block Body**
Block number	List of transactions
Previous block header hash	Other data
Hash of the block data	
Timestamp	
Block size	
Nonce value	

A transaction is a transfer of tokens, between participants of the blockchain network. Tokens are usually the so-called cryptocurrencies, however, they may represent, on a business-to-business basis, a way of registering assets, or activities that occur on digital or physical assets. Transactions enable the establishment of trust between participants through the use of asymmetric keys, which consist of a mathematically related public and private key pair, where the private is used to sign the transaction while the public verifies its signature. This process is known as encryption/decryption and is vital for signing and verifying signatures as well as ensuring transactions authenticity, integrity and the non-repudiation of the sender.

Submitting a transaction requires the existence of two cryptographic addresses. These addresses are the previously mentioned public keys. Participants in blockchain systems are thus represented anonymously by a pair of keys, a public key used to identify the participant and receive tokens sent by other participants, and a private key to authorize transactions. Their anonymization is therefore a core concept that guarantees the privacy of participants and the security of transactions.

When performing a transaction a complex and functional network is required to validate and confirm it. The network is composed by nodes, subcategorized into full and light nodes, which interact in different ways. Full nodes, also known as miners, store the entire blockchain and secure the network by running a consensus algorithm allowing transactions to be validated and confirmed. They have a massive influence on the network, as they ensure its decentralization, since backups are created all over the world, and every peer updates and synchronizes the ledger. Full nodes select transactions from a set of pending transactions, aggregating them to generate a new block. Light nodes merely deliver their transactions to the full nodes. They are not concerned with storage or validation details.

It is costly and time-consuming to run a full node, so participants, are incentivized through a reward for their services offered by the blockchain network. The reward is usually a digital coin or cryptocurrency, like Bitcoin [YMRS19].

The consensus mechanism is all about defining which node publishes the next block. Depending on the blockchain's purpose, a consensus model may be adopted and implemented from one of the many available. Several new models are emerging, we will highlight the two most common, the Proof of Work and the Proof of Stake.

In the proof-of-work (PoW), all miners compete to be the first to solve a computationally intensive puzzle in order to successfully append the next block to the blockchain. Solving the puzzle is the "proof" of effort and time spent. This process is called mining, and agents who mine are also known as miners. Finding the puzzle solution is difficult, but verifying a valid solution is easy. This allows all the full nodes to easily validate any proposed next block, and disregard any proposed block that would not satisfy the puzzle [YMRS19]. Deepening the concept, Each node in the network calculates a hash value of the block header. The block header contains a nonce and miners frequently change it to obtain different hash values. The consensus requires that the calculated value must be equal to or less than a given value [ZXD+17]. When a node reaches the target value, it broadcasts the block to all other nodes, who then have to verify if the hash value is accurate. If valid, then miners append this new block to their respective blockchains [ZXD+17].

Although highly unlikely, multiple nodes may discover the accurate nonce value simultaneously, hence, legitimate blocks can be created concurrently originating two different branches in the blockchain. The largest branch in extension will be taken as genuine, while the other will become an orphan. This process is known as forking.

Proof-of-work demands high computational power, using exorbitant amounts of energy, thereby becoming a major disadvantage of its use. Bitcoin is one of several blockchains using PoW as its consensus algorithm. The power consumed for Bitcoin mining is currently projected to be greater than Ireland's electricity usage [OM].

Proof-of-stake (PoS) aims to address the energy consumption problem created by proof-of-work. This model is ruled by the amount of stake owned by each participant, where the probability of a new block being appended to the blockchain depends on the participant's total amount of cryptocurrency invested into the blockchain [YMRS19]. There is no need to perform resource-intensive computations since a participant is randomly selected to append a new block to the blockchain, and, once successful, collect the reward [Sal21].

In short, we discussed in this section the emergence of blockchain technology made popular with the appearance of Bitcoin in 2008 [Nak08]. Then, we presented its structure in detail, which, in a nutshell, it's all about securing transactions through its distributed network of nodes, each one storing a copy of the blockchain. Its key features include network decentralization, data persistence, data immutability, participant anonymization, and transaction traceability. We will

continue with the analysis of two underlying blockchain matters. First, we will understand which participants are allowed to participate in the consensus algorithm by categorizing blockchains based on their permission mode, and then, we will discuss what are smart contracts and what are they used for.

2.2.1 Permissionless Versus Permissioned

Permissionless or Public blockchains such as Bitcoin [Nak08], Ethereum [But16], or EOS [Gri17] are decentralized ledgers that enable any node to append blocks without requiring approval from an authority. They are transparent, meaning that every new block is visible to all nodes in the network. Their code is open source, therefore available to any participant to join and typically secure, due to public source-code exposure. Permissionless blockchains are generally used to transact cryptocurrencies since transactions are transparent, immutable, and intermediary-free. It is possible, in permissionless blockchains, for a light node to become a full node and vice versa. A light node can become a full node by downloading and storing a copy of the ledger. This way it is able to validate transactions and contribute to the network. On the other hand, a full node can become a light node by deleting its copy of the ledger. Therefore, any node can eventually access the blockchain to read and write, meaning that anyone can validate and issue transactions. Consensus algorithms such as Proof of Work or Proof of Stake are essential for maintaining trust, efficiency, and network decentralization, however, they tend to be computationally expensive.

Table 2.4: Categorization of blockchain according to permission mode *[Tas19]*

Types	**Read**	**Write**	**Commit**	**Example**
Permissionless	Anyone	Anyone	Anyone	Bitcoin, Ethereum, EOS
Permissioned	Authorized Participants	Authorized Participants	Authorized or Subset of authorized participants	Hyperledger Fabric, R3 Corda, Canton

Permissioned or Private blockchains, in contrast to the prior permission mode, only allow a restricted set of nodes to read and write the ledger. Permission is granted by a centralized or decentralized authority. Permissioned blockchains may present the same properties as Permissionless blockchains, such as network decentralization, transparency, persistence of transactions,

and consensus mechanisms. However, consensus mechanisms tend to be less computationally expensive and faster, as participants have their identities known, so any fraud or malicious act is easily identifiable. Some instances of permissioned blockchains are Hyperledger Fabric [DMH17], R3 Corda [BCGH16] and Canton [Tea20]

2.2.2 Smart Contracts

The smart contract concept was first introduced in 1994 by Nick Szabo. Szabo described its purpose as a way of avoiding the requirement for a third party and preventing malevolent or unintended activities while meeting standard contract criteria such as payment periods, liens, confidentiality, and even enforcement [S+94]. Although the term first appeared in the 1990s, it became popular after the introduction of Bitcoin, when Russian-Canadian programmer Vitalik Buterin published in 2014 a paper entitled A next-generation smart contract and decentralized application platform [B+14], idealizing a platform extensible to other uses beyond the simple exchange of cryptocurrencies. In 2016, Vitalik Buterin launched the Ethereum [But16] platform, being regarded as the second generation of the blockchain. It is a programmable platform whose network work through smart contracts allowing the creation of decentralized applications, transfer of digital assets, and transfer of cryptocurrencies. Ethereum uses Ether as its cryptocurrency.

A smart contract is an executable code that resides and executes on the blockchain to enforce the terms of an agreement [AvM17]. They ensure automated enforcement of the terms whenever certain criteria are satisfied, meaning that, once arbitrary predefined requirements are satisfied, digital assets are released to the concerned parties [B+14].

Smart contracts feature self-verifying, self-executing, tamper-proof, traceable and irreversible properties, meaning that all details present in a smart contract are public, cannot be changed once deployed, cannot be tampered with, as they are verified by all the peers, and its execution occurs automatically. [MPJ18].

A smart contract is composed by an address that uniquely identifies it, an account balance, private storage, and executable code, which secures the contract logic. The contract storage and balance constitute the contract state. When the contract is triggered, the state is stored on the blockchain, and, therefore updated [AvM17]. To invoke a smart contract, a transaction must be sent to the smart contract's unique address. Then, the transaction must be validated and

executed by all nodes to reach a consensus on modifying the contract state. After the consensus agreement, a smart contract can execute itself in an automated manner based on the type of transaction received. It can read and write to its private storage, save tokens into its account balance, send/receive tokens or messages to/from other smart contracts and can even create new smart contracts [AvM17].

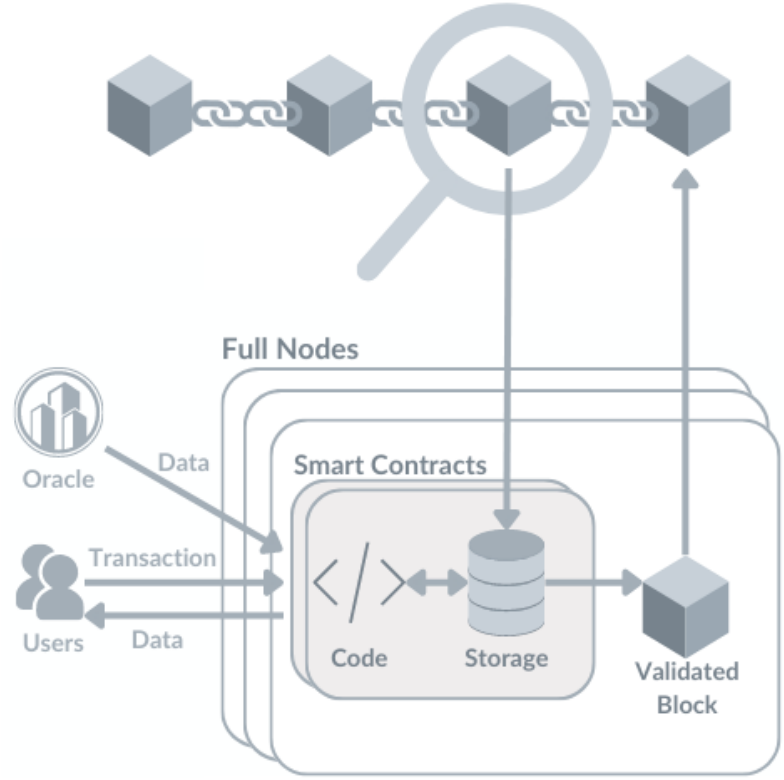


Figure 2.3: A Smart contract structure in detail

Smart contracts, can be categorized into two types, deterministic and non-deterministic. A deterministic smart contract does not require any information from a party external to the blockchain, whereas a non-deterministic smart contract must rely on oracles or data feeds, which are mechanisms external to the blockchain used to gather the necessary data to execute the contract. The term oracle is derived from Greek mythology and refers to a person who can connect directly with God and predict the future. Oracles on the blockchain, unlike Greek mythology, do not foresee the future but rather collect and store data from the real world. An oracle is defined as something that provides external data to the blockchain in a way that the blockchain can interpret it. Oracles can gather data such as the exchange rate of crypto assets, weather conditions, political events, sports events, and so on [Cal20].

The implementation of smart contracts depends on two essential matters: the blockchain and the programming language. Note that, not all blockchains are able to run smart contracts, and their implementability is dependent on the chosen programming language. Different blockchain platforms support smart contracts differently. Some platforms may simply support a simple scripting language, whilst others support far more sophisticated programming languages [ZLK+19]. Ethereum was the first blockchain protocol to enable users to write smart contracts, although other protocols, including, Hyperledger Fabric, and R3 Corda, are also able to do so.

Besides all the aforementioned properties, smart contracts also provide the underlying technology for DApps to operate. The term DApp or decentralized application refers to an application that is executed by multiple users over a decentralized network [Wu19]. Decentralized applications are the latest type of applications. They are identical to traditional applications, except that the valuable data and operations are stored in smart contracts on a blockchain. Furthermore, they interact with smart contracts through transactions, namely contract requests, and provide services depending on them [Wu19].

Smart contracts are intended to innovate, and eventually replace traditional legal contracts, however, their ability to be enforced is dependent on government and political decisions. In the next section we will discuss and compare the main platforms that support their development.

2.2.3 Smart Contract Platforms

A growing number of platforms are being deployed to support the development of smart contracts. In this section we intend to discuss platforms that enable application development in general. We chose them taking into account their community popularity as well as their technical stability. **Table 2.5**, compares five smart contract platforms that we will be discussing along with their advantages and disadvantages.

First off, Ethereum is a platform that allows smart contracts to be written in languages such as Solidity, Serpent, or Mutan. A smart contract's code is compiled and executed in the Ethereum Virtual Machine (EVM) execution environment. Its network features a permissionless mode requiring PoS as its consensus algorithm, which eliminates the high energy consumption of its previous PoW consensus algorithm by requiring participants to hold and stake Ethers, guaranteeing the security of the network. Ethereum adopts an account-based data model, where each node is identifiable by its digital wallet [ZXD+20]. Its permission mode combined with its

Table 2.5: Comparison of smart contract platforms
[Tea20] [ZXD+20]

	Ethereum	**Fabric**	**Corda**	**EOS**	**Canton**
Application	General	General	Digital Currency	General	General
Permission	Permissionless	Permissioned	Permissioned	Permissionless	Permissioned
Execution environment	EVM	Docker	JVM	WebAssembly	Daml
Language	Solidity Serpent, Mutan	Java, Golang Node.js	Java Kotiln	C++	Daml
Consensus	PoS	PBFT	Raft	BFT-DPOS	SMR
Data model	Account-based	Key-value pair	Transaction-based	Account-based	Transaction-based
GDPR compliance	Very weak	Very Weak	Average	Very Weak	Strong
Transaction privacy	Very weak	Weak	Average	Very Weak	Strong

consensus algorithm makes Ethereum a fully transparent platform with scalable performance. Ethereum does not guarantee privacy nor compliance with GDPR regulations. These concerns must be addressed by applications built on top of it.

Unlike Ethereum, Hyperledger Fabric network features a permissioned mode, with no underlying cryptocurrency. Its network consists of nodes whose identities are given by a membership service provider. There are three different types of nodes that undertake different responsibilities, namely client nodes, peers nodes, and ordering service nodes. The peer nodes maintain the ledger state which is used by the client nodes to propose transactions to execute and broadcast them for ordering. The ordering service nodes set the order of all transactions. However, they do not interact in the execution or validation processes [PRLT21]. The platform originally implements Solo, a consensus process based on voting, nevertheless, alternative consensus protocols, such as Practical Byzantine Fault Tolerance (PBFT), could be plugged in.

Fabric allows private and confidential transactions to be performed through a channel, which is essentially a private communication subnet between two or more entities on the network. Each transaction on the network is executed on a channel, where each entity must be authenticated and authorized in order to transact [Doc20]. Fabric supports smart contract and distributed applications (Dapps) development in Java, Golang, or Node.js. Their code runs on Docker containers, which compared to EVM, provides a more efficient execution environment, reducing overhead, yet applications are less isolated. Hyperledger Fabric employs a key-value model, in which data is stored in blockchains as key-value pairs.

R3 Corda is a permissioned blockchain platform mostly adopted by firms in the financial sector and mainly used to develop digital currency applications. Corda is written in Kotlin and supports smart contract development in either Kotlin or Java running on top of the Java Virtual Machine (JVM) execution environment. It has a tansaction-based model, which consists in the absence of digital accounts or wallets. Tokens are stored in a list of unspent transaction outputs, or UTXOs. Each of these transactions has an amount and a spending criteria. Existing unspent transactions are consumed, and new unspent transactions are produced in their place. This model is focused on safeguarding privacy. It is challenging to firmly link multiple currencies to a single owner, since each time a transaction is received, a new address is used. Therefore, participants are encouraged to produce a new address for each received transaction. Corda uses Raft [HSMC15] as its consensus algorithm. Instead of broadcasting in blockchains, the platform employs a point-to-point messaging system where users must define the message recipients as well as the exact information to be transmitted [ZXD$^+$20]. In corda, each node represents an entity that can either interact with each other publicly or privately. Each node is equipped with several services and applications called CorDapps allowing them to communicate and transact with other nodes in the network. From a general perspective, Corda provides transaction privacy, fast operational speeds, cost optimization and efficiency of intercompany cooperation.

EOS platform similarly to Ethereum, features a network with a permissionless mode. It supports the development of smart contracts, whose code is compiled in the WebAssembly (Wasm) execution enviroment, which executes faster than Docker and EVM, although it only supports traditional languages like C++, Rust and C. EOS combines two consensus algorithms, Byzantine Fault Tolerance (BFT) and Delegated Proof of Stake (DPOS). The platform employs an account-based data model, where each node is identifiable by its digital wallet, and every account may have a human-readable name reference. EOS, like Ethereum, due to its permission mode, promotes transparency, so GDPR compliance is naturally reduced.

Canton is a platform whose purpose is the development of distributed applications involving multiple organizations. Canton implements authorization and privacy models through the Daml runtime environment. Daml is a programming language that enables the development of smart contracts whose distinguishing features are built-in authorization and privacy models [Tea20]. Similar to Corda, it has a transaction-based data model. Although Canton can support any programming language with the same transaction model, it is preferable to write smart contracts in Daml, because it is specifically conceived to model complicated multi-party interactions across multiple domains. Daml interpreter translates a Daml expression deterministically into a transaction, and Canton then guarantees the transaction is secure and atomic [Tea20]. Regarding GDPR compliance, Canton complies with GDPR data minimization principles. Canton is auditable and is an appropriate platform for regulated environments, such as Real Estate. It stands out from the previous platforms due to its high scalability, and compliance with the EU GDPR requirements.

2.3 Hyperledger Fabric

In this section, we will discuss the Hyperledger Fabric platform in greater detail. We will analyze its purpose, key concepts, and behavior. Hyperledger Fabric is a blockchain project under the Hyperledger Foundation, established in 2015 by the Linux Foundation, with the aim of pushing forward blockchain technologies across industries. Fabric, incorporates a ledger, operates with smart contracts, and offers participants the ability to manage their transactions.

As discussed in **Section 2.2.3**, Hyperledger Fabric is a permissioned blockchain designed for business applications. It offers high levels of confidentiality, resilience, flexibility and scalability. Its modular architecture adapts to different sectors and business concepts. Supports pluggable consensus protocols with the goal of meeting a variety of needs. It stands out for being the pioneer blockchain platform to allow the development of smart contracts, also referred to as chaincode, in general-purpose programming languages such as Java, Go or Node.js.

Fabric has a ledger subsystem consisting of two merged components, the world state and the transaction log, shown in **Figure 2.4**. The world state component represents the ledger's state at a specific moment, functioning as its database. Meanwhile, the transaction log captures all transactions leading to the current world state, serving as its update history. The ledger, in essence, merges the world state database with the transaction log history. The world state uses CouchDB as its default database, due to its NoSQL nature, which offers a JSON-based document

storage. CouchDB allows data to be stored in a schema-free format, guaranteeing flexibility when structuring stored documents. Fabric provides the capability to establish channels, enabling specific set of participants to form a distinct transaction ledger, as a way to avoid sharing transactions with other participants not included in the set, thus assuring transaction privacy.

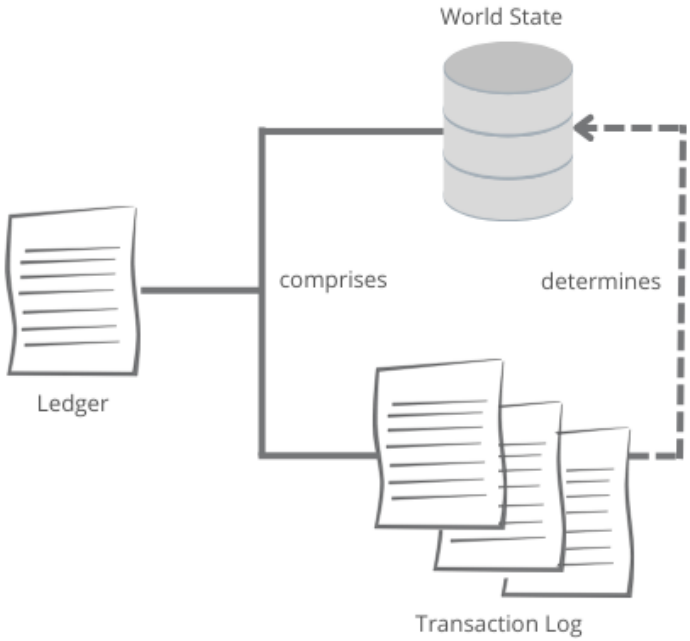


Figure 2.4: Structure of a Hyperledger Fabric ledger

Each participant holds a copy of the ledger for each Hyperledger Fabric network and channel they are part of. To transact on the network, participants must be registered on a channel. The registration is carried out by a trusted MSP (Membership Service Provider), designed to identify and authorize participants, allowing them to engage in network transactions within a designated channel. It is important to note that all network participants have their identity identifiable through digital certificates. These certificates are generated through a public key infrastructure and are linked to organizations, network components and applications [Hyp23].

Organizations are entities that represent different stakeholders in the blockchain network such as institutions, businesses, or any group that participates in the network. Each organization has its own set of peers, endorsing peers, and orderers. Peers are a key player in the network, since they are responsible for managing ledgers and chaincodes. They are network nodes that handle transaction proposals and endorsements by running the Fabric Gateway service. The Gateway acts as an intermediary between peers and applications. It simplifies the interaction process between applications and the Fabric network. Peers within an organization share the same

ledgers and chaincodes. They communicate with other peers within their organization and, if necessary, with peers from other organizations to validate and endorse transactions.

Smart contracts, known as chaincode, are a fundamental part of Hyperledger Fabric, since it is through them, that it is possible to transact on the network. They are executed whenever an external application needs to interact with the ledger. Chaincodes interact with the world state to create and query assets through read and write operations. Assets are digital representations of real world items such as real estate properties, pieces of art, healthcare records and so on. In Fabric, they are represented as a collection of key-value pairs in binary and/or JSON format. Chaincode defines and manipulates assets through its functions business logic. Functions operate on the ledger's current state database and are triggered by a transaction proposal. When a chaincode runs, it produces a series of key-value updates (write set) that can be sent to the network and integrated into the ledger on all participating nodes. State changes occur due to chaincode invocations, known as "transactions", submitted by the participants involved. Every transaction results in a collection of key-value pairs of assets, which are recorded in the ledger by invoking chaincode functions such as create, update or delete.[Hyp23].

Updating the ledger requires three phases. The first phase consists in the submission, execution and endorsement of the transaction proposal. An application submits a signed transaction proposal to the gateway service. The gateway service, responsible for endorsement, selects an appropriate peer to execute the transaction by invoking the specified chaincode. The executing peer generates a proposal response, including the read-write set, signs it, and returns it to the gateway. This process is repeated for each organization required by the chaincode's endorsement policies. The gateway service then collects the signed proposal responses, creating a transaction envelope and is sent back to the client application for final signing via the SDK.

The second phase consists in submitting and ordering the transaction. In this phase, the signed transaction envelope is sent from the client SDK to the gateway service, which forwards it to an ordering node. The ordering node validates the signature and orders the transaction. Subsequently, the ordering service groups the transaction with others into blocks. These blocks are then distributed to all channel peers for validation and commitment to the ledger.

The third and final phase consists in validating and committing the transaction. Each peer validates the transaction by ensuring the client signature matches the original proposal and that, the read-write sets, status responses, and endorsements are aligned. Transactions are marked valid or invalid based on these checks. Valid transactions are then committed by each peer to the

channel ledger, constituting an immutable ledger update. The channel's world state is updated only with results from valid transactions. Subsequently, each committing peer sends the client a commit status event, providing proof of the ledger update.

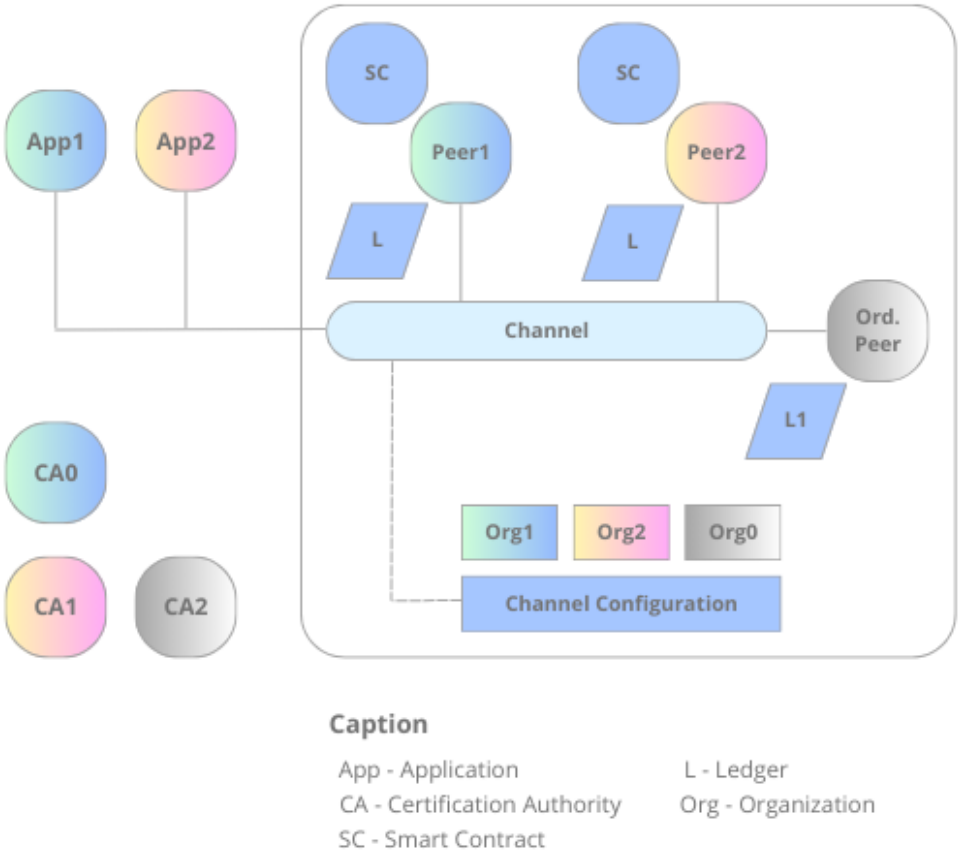


Figure 2.5: Structure of a Hyperledger Fabric network

The **Figure 2.5** shows the structure of a simple Hyperledger Fabric network with a single channel. We start by looking at the right-hand side of the image. It shows a large rectangle surrounding several elements of the figure. This rectangle represents the blockchain network. Within it, there are 3 organizations, organization 0, 1 and 2, (Org1, Org1, Org2) which decide to create a network to transact with each other through the channel. The network requires a channel configuration to define the policies of each organization on the channel. Each organization joins a peer on the channel. Organization 1 (Org1) joins peer 1, organization 2 (Org2) joins peer 2 and organization 0 joins the ordering peer, which is the network's ordering service. All peers have a copy of the ledger (L), and peers 1 and 2 have the smart contract/chaincode (SC) installed on them to propose transactions. Organizations 1 and 2 (Org1, Org2) interact with the channel through the applications shown on the left of the image, located outside the network (App 1,

App2). In the bottom left corner of the figure, we observe the certification authorities (CA0, CA1 and CA2) which generate the certificates for the peers, organizations and applications.

In this section, we provided a detailed examination of the Hyperledger Fabric blockchain. We delved into its sophisticated architecture, complexities and operational processes. In the next section, we will introduce the concept of wallet and explain its relationship with the blockchain technology.

2.4 Wallet

In this section, we will discuss the concept of wallet, its usefulness, relationship with blockchains and its different types. Digital wallets existed long before the blockchain technology appeared. With the emergence of the blockchain, crypto wallets began to appear and started gaining popularity as cryptocurrencies became widely used. Crypto wallets are a new type of digital wallet, and they offer a secure environment for accessing and transacting on blockchains. They are essentially, software applications used mainly to manage cryptocurrencies. Although it may seem contradictory, a wallet keeps track of the balance of cryptocurrencies, but it does not actually contain them. Instead, it keeps track of the user's balance and give him access to the blockchain where the cryptocurrencies are recorded. To allow users to access and manage cryptocurrencies, a wallet stores private keys or seed phrases, which are essential for authorizing and signing transactions on a blockchain [JB22].

2.4.1 Wallet Types

Wallets are categorized based on how they access the network, where they are stored, whether they store the entire chain or just parts of it and whether the keys are stored locally or on a remote server [HSB23]. Regarding network access, wallets can be categorized as either hot or cold, based on their connectivity.

Hot or online wallets, require an Internet connection, and come in the form of applications installed on a computer or smartphone. These applications can either be web wallets, which are browser applications that do not require installation, desktop or mobile wallets, which are applications that are downloaded and installed on the user's devices. Cold or offline wallets do not require Internet access and come in two forms, hardware or paper wallets. Hardware wallets are special devices designed specifically to store private keys securely, while paper wallets are simply pen or pencil notes of private keys.

Let us now define in greater detail the fundamental differences between the different kinds of technology-based wallets, from which we exclude paper wallets.

Figure 2.6 [HSB23] depicts the different wallet types.

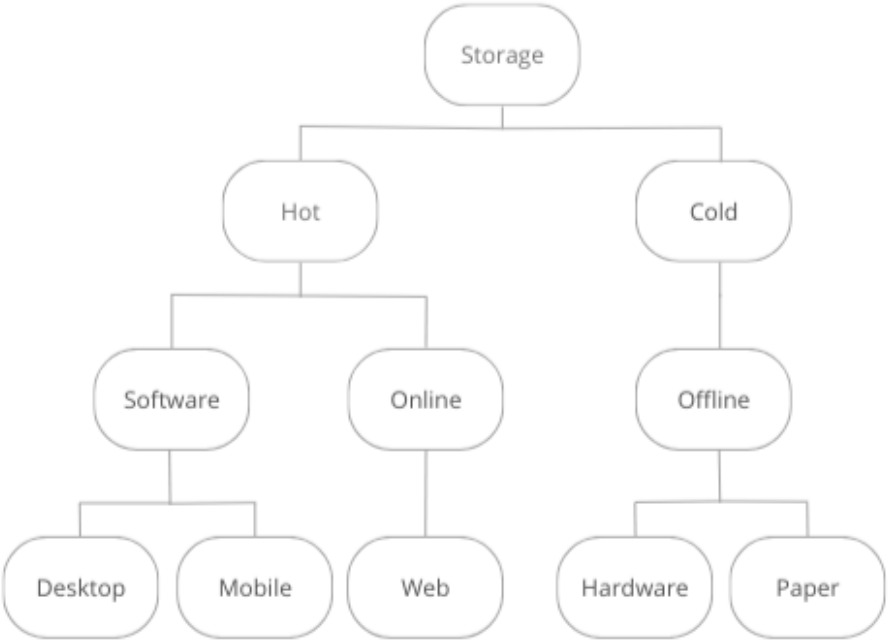


Figure 2.6: Wallet types

- **Desktop wallets** are software applications that users download and install on their computers enabling them with full control over their keys and funds. Private keys are stored in the computer's file system and may eventually be encrypted to provide an extra layer of security. Generally, these applications offer recovery mechanisms, such as generating a seed phrase to allow users to access their private keys in case of loss [HSB23].
- **Mobile wallets** demand the download and installation of an application on a mobile device. They enable users to handle their private keys and generate a seed phrase as a way of recovering the private keys.
- **Web wallet** is an online service that allows users to interact with a blockchain via a web browser interface, or extension. Users can access their funds from any device with an internet connection. They are convenient, but less secure, as they are owned by third-party providers responsible for storing private keys.
- **Hardware wallets** are physical devices designed to store private keys offline. They allow users to have complete responsibility and control over their private keys. They are

considered the most secure type of wallet, however they need to be combined with another type of wallet to access the internet. Great security implies great risk, so in case of damage or loss, there is no chance of recovery unless the seed phrase is backed up.

2.5 GDPR

So far, we have studied significant concepts related to the blockchain technology, and we realized that there is a broad spectrum of possible business ideas to explore. However, we must understand how businesses are regulated in the European Union. To this end, we are going to introduce a vital law for any digital business wishing to thrive on EU soil.

GDPR or General Data Protection Regulation [Eur16] is an EU data privacy and security law, that sets standards for collecting and processing personal data of EU citizens or residents who contract services or goods. A company providing services to citizens or EU residents, even if it does not operate in EU territory, may face high monetary penalties if it fails to handle and process consumer information [Wol18].

The term personal data refers to any information related to an individual who can be directly or indirectly identified, such as names, email addresses, locations, ethnicity, gender, biometric data, religious beliefs, web cookies, or political opinions [Wol18]. Any reversible process, such as generating pseudonyms, blockchain public keys or encrypting data, could reveal the original data which would be violating the GDPR. Conversely, anonymized data is not considered personal data, so the GDPR does not apply.

The GDPR sets out eight rights that apply to all consumers. These include, the right to access and request data, the right to be informed about the processed data, the right to data portability which permits data to be transferred at any time to a new service provider. The right to be forgotten, allowing customers to withdraw or erase their personal data. The right to object and the right to restrict processing allow to stop all or part of the processing when a consumer requests. The right to be notified requires companies to inform customers when there is a security breach, while the right to rectification allows consumers to update and rectify their data.

Companies wishing to create or adapt business ideas to the blockchain must comply with the GDPR law. However, the applicability of the GDPR to blockchain raises some questions, especially when applied to permissionless blockchains. A permissionless blockchain is supported

by transparent and immutable records, meaning that records are publicly available, and once published, can no longer be deleted, thereby violating some GDPR rights including the right to be forgotten. In these types of environments, compliance can be established by forcing consumers to accept certain terms before access is granted [CMS19]. Another alternative is to adopt off-chain storage to store consumers personal data. Alternatively, companies can also employ a permissioned blockchain which due to their restricted access, may allow them to ensure GDPR compliance.

Applications wishing to use the blockchain technology to store personal data must consider the fact that, the blockchain technology permanently records data, and the right to be forgotten demands any data associated with a consumer to be erased on request. Given the circumstances, it is necessary to develop a technique to avoid violating this right. At the time of this writing, storing personal data directly on the blockchain remains impossible. Nevertheless, there are two approaches to circumvent the problem. We can either store personal data on the blockchain using a crypto-shredding or a hashing technique.

In essence crypto-shredding consists of storing encrypted data on the blockchain. Logistically it is a complex and unsecure option due to the issue of where to store the encryption key. Data is encrypted, therefore, losing the key would result in a permanent information loss. The hashing technique is more reliable and secure. A hash is a cryptographic summary of a message, practically impossible to reverse, and really simple to verify. Its application to the blockchain consists in converting personal data into a hash and then storing it on the blockchain. In the future, we believe that blockchain platforms will have built-in mechanisms to handle this problem.

2.6 Blockchain Use Cases

In this section we will discuss the applicability of the blockchain technology across different industries. We will go over some broad blockchain use cases before diving into specific blockchain use cases in real estate.

2.6.1 Blockchain General Use Cases

Operating a blockchain is only reasonable when several untrustworthy participants are not willing to reach an agreement with a trusted third party [WG18]. Undoubtedly, the banking and financial sectors are the most eager to invest in the technology [ZS18], mainly due to the ap-

pearance of cryptocurrencies. Sectors such as, government, private management, healthcare, insurance, electronic voting, and real estate are gradually adapting their traditional business ideas to the blockchain [GR18].

We briefly introduce some applications that, through blockchain, are slowly reinventing and innovating some segments of society. Borderless is a government platform based on smart contracts that enables the provision of a wide range of economic and legal services, such as legal entity registration, notary services, marriage, notary services, and financial transactions [GR18]. E-Residency [SB17] is an electronic identity system used by Estonia. E-residents are able to access a broad range of Estonian public e-services, as well as the EU business environment. Identification cards and related digital keys are used to grant access to services [GR18]. MedRec is a blockchain-based medical system intended to deliver secure, transparent and scalable access to medical data records [GR18]. Gilgamesh is an educational platform aiming to share knowledge that works through Ethereum smart contracts. Readers, critics, and authors may gather to discuss a piece of writing. Users are encouraged through Gil digital coins to keep them engaging with the content through sharing, discussion and writing. The coins can then be used to purchase other academic digital material [D$^+$21].

We just mentioned a handful of examples out of many that we could have mentioned. Industries such as Education, Internet of Things, Commerce, Finance, Intellectual property, Government and Real Estate have already been developing new ideas and businesses ready to be explored with the help of blockchain technology and smart contract platforms.

Before diving into specific blockchain use cases in real estate, we first need to introduce the concept of tokenization and its applicability to real estate.

2.6.2 Tokenization of Real Estate Assets

Before we begin discussing tokenization, we would like to emphasize the concept distinction between token and cryptocurrency. Cryptocurrencies similarly to fiat currency represent a store of value and are generally used as a mean of payment. Although sometimes referred to as digital tokens, cryptocurrencies are generally considered to be assets in themselves, with the token merely representing the coin's stored value [VBS20], whereas tokens are seen as digital representations of real assets economic value.

The applicability of blockchain technology to the real estate industry has triggered a new realm of possibilities that were previously unthinkable, such as the tokenization of a property. Tokenization refers to the process of creating virtual tokens to represent ownership of a real estate interest [VBS20]. It also allows a property or asset to be fractionated into multiple tokens stored on the blockchain. In this process, an asset is divided into several fractions through a smart contract, generating a token for each fraction, meaning that whoever acquires one of these tokens is considered part owner of one of the fractions of the real estate asset. In fact, a token can represent more than just a fraction of a real estate asset. **Table 2.6** shows its different possible representations.

Table 2.6: Representations of a token in real estate
[Den]

Token Representations
Ownership of part of a real property
Ownership of the entire real property
An equity interest in an entity that controls real property
An interest in a debt secured by real property
A right to share in the profits generated by real property

According to **Table 2.6** note that a token can either represent a fraction or a whole property. Tokenization is necessary to convert a property's real value to the digital world. Landlords seeking to rent their property or part of it, may do so by generating one or multiple tokens. In a more clear way, a property owner, for instance a landlord, can either sell or rent his property through tokenization. He must first select a tokenization platform before entering his data and property details. Afterwards he tokenizes the property according to what he wishes. As an example, he might want to divide the property into several bedrooms, or into certain divisions therefore generating a token for each divided fraction. Should he not choose to divide the property he may also generate a token to cover the entire property. The tokens are released on the platform to allow tenants and investors to acquire them in order to become the property tenants or (co-)owners respectively [FCE].

As mentioned at the beginning of this section, a token is a digital representation of a real asset's economic value. There is a particular type of token designated Non-Fungible Token or in short NFT. NFT's represent a unique asset, they cannot be replicated, and cannot be exchanged

for equivalent tokens. The term fungible in essence means that an asset can be exchanged for an equivalent asset. For example, a cryptocurrency can be exchanged for another cryptocurrency with the same value, however a non-fungible asset cannot be exchanged for anything similar since it is unique. These properties define NFTs as the ideal technique for digitally representing assets such as real estate.

According to Smith et al. [SVB+19], the tokenization of real estate is beneficial on several levels. Affordability, which allows a small capital investment in real estate. Increased liquidity, enabling an asset to be sold quickly, minimizing the investor's risk, as well as ease of investing and lower transaction costs. At the time of this writing, the major vulnerability of tokenization concerns the legal component, which is still poorly explored, lacking regulation, as different countries have distinct laws and policies. Having understood the concept of tokenization, we will next discuss a few specific use cases that relate blockchain to real estate.

2.6.3 Blockchain Use Cases in Real Estate

The traditional real estate system faces some drawbacks including high intermediation costs, time-consuming bureaucracy, lack of standardized processes, and non assurance of information security since information is shared through multiple insecure channels. The introduction of smart contracts opens up a whole set of new solutions to tackle the aforementioned issues.

A smart rental contract is regarded as the latest paradigm for renting a property based on the blockchain technology. A smart rental contract should be able to execute the predefined terms of a rental agreement on its own. Furthermore, if both parties have not signed a formal rental agreement, it is imperative that a smart rental contract be able to provide evidence of the rental's existence for related dispute concerns. Since the technology is still in its early stages, there is some uncertainty concerning the use of smart rental agreements. It is yet unclear if the contracts may be used as evidence in court and whether judges would accept them as evidence of eventual disputes [YTK22].

The remainder of this section will address several existing or developing smart rental projects or newly established companies. We will be focusing on projects that resemble a smart rental model that we idealize for Unlockit.

Rentible

Rentible is a community-focused company and a decentralized platform for real estate rental management. It is helpful for tenants looking to rent an accommodation for at least a 3-month period and wishing to pay in digital currency for the security deposit and rent payments. Its ecosystem is supported through blockchain technology that aims to automate processes and services through a decentralized ecosystem. Rentible establishes a real estate lease management platform on the Ethereum blockchain through a DApp that introduces a new model where tenants and landlords can intuitively send or receive lease payments in different digital currencies in a decentralized way [Tea22].

Rentible platform uses the RNB as its utility token. Holding RNBs is required to access features across the ecosystem and benefit from rewards such as executing smart contracts, paying for rentals, accessing rewards and so on. Rentible marketplace is planned to include, alongside residential properties, also commercial properties, coworking spaces, and virtual properties in the metaverse. Their users, either landlords or tenants, are able to interact with the application through mobile or web applications. Rental agreements are translated into smart contracts and executed in DApps allowing users a new level of accessibility. Smart contracts process in the background deposit and lease agreement issues and then are synchronized with a user friendly UI enabling users to interact with it. Users can choose their preferred payment method for rentals. For instance, landlords can request rent payment in a specific digital currency via an intuitive UI that records transactions on a dashboard as proof of payment, while the utility token RNB is used as the internal unit to cover the service fee [Tea22].

SmartRealty

SmartRealty is a company based in Washington, USA. Its goal is to enable users to execute, register and enforce real estate transactions using a smart contract system known as SmartRealty contracts. Those contracts are used to implement and maintain property acquisition and rental agreements. SmartRealty contracts assist in the establishment of standards for paying rent or purchasing a home that, if not satisfied, instantly terminate a contract. The platform is powered by three components: the contractual platform, which allows both parties to establish contracts in accordance with state laws, the property listing platform, which allows landlords to advertise their homes on the web, and the RLTY tokens, used for payments. Payments can actually be made with any cryptocurrency or fiat currency, however, they will be automatically converted to RLTY tokens to properly record and track payments [SMA18].

At the time of this writing, SmartRealty is applying smart contract technology to residential leases and offering a platform for landlords to advertise their rental properties and homes to potential tenants. They intend to integrate residential and commercial real estate transactions in the future, including rentals, sales, and even non-traditional transactions like owner financing and option leases [SMA18]. For now, according to their official project documentation [SMA18], the blockchain platform has yet to be decided. However, there are currently two preferred options, Ethereum and Polkadot.

Midasium

Midasium is a start-up company based in London, UK, that is building a global ecosystem for residential and commercial property where transactions can be tracked in real time, such as title registrations and rental agreements. Its proof of concept at the time of writing this document is supported by a permissioned blockchain, entitled Midasium blockchain, responsible for hosting smart contracts. The platform is designed for independent landlords or property managers to manage the cash flow of their property portfolios [Mid17]. The platform is distinguished by three prominent features. The first is the security of bond deposit, which holds funds securely for the rental term and only releases them upon dual signature of tenant and landlord, with any deductions paid to the landlord as agreed by both parties. The second is the reconciliation of tenancy ledger which automatically reconciles revenue from rent deposits with the tenancy ledger, and the third is the expense management meaning that all maintenance costs are deducted using income from the tenant's bond deposit or rent [Mid17].

RentPeacefully

RentPeacefully is a blockchain-based platform that allows landlords to advertise properties for rent and create rental agreements employing smart contracts on the Ethereum blockchain. The procedure of renting a property is rather straightforward. A landlord lists his property, waits for a tenant to express interest, and then creates a rental agreement that both the tenant and landlord must sign in order for it to be legitimate. The smart contract is then signed and deployed on the Ethereum network. Any party can engage with it to start a dispute, pay the rent, or withdraw the funds. The system is further intended to process maintenance requests, payment deposits, and mediation tasks [Ren18].

The Bee Token

The Bee Token is the token that powers the Beenest network [Bee18] facilitating transactions on the platform. Beenest is an open-source, decentralized short-term house-sharing network built on top of a set of Bee protocols with the aim of connecting hosts and guests without charging commissions. The platform allows hosts to advertise their properties and guests to find accommodation. Its protocols are open Ethereum protocols that can supply the Beenest network with three core systems: A payment system that allows two authenticated peers to send and receive money that is held in Bee tokens, even after a successful exchange of services between the two parties [Bee18]. A decentralized arbitration system for resolving user disputes, with positive incentives to build a network of legitimate arbitrators and negative incentives to discourage fraudsters [Bee18], and a reputation system that combines a genuine identification derived through a trusted fingerprinting protocol on the Ethereum blockchain with a rating determined by a transparent, immutable review and scoring interchange between P2P entities such as guests and hosts [Bee18].

Projects Comparison

As we know, blockchain technology is still barely explored in the real estate industry. Consequently, there are still few companies developing projects that relate both blockchain and real estate concepts. The projects we have presented throughout this section, are the first to explore the unknown and therefore their development is still at an embryonic stage. In **Table 2.7**, we compare the properties of the aforementioned projects.

It is noticeable, by looking at **Table 2.7**, the different development stages. The Rentible project is the most stable on the market, and, together with the RentPeacefully project, are the only two, that already provide services to the public. All the others are still Proof-of-Concept intending to integrate the market. Nearly all the projects are supported by the Ethereum blockchain, except Midasium which has developed its own blockchain.

We believe that using the Ethereum blockchain was a decision based on its popularity and its large developer community. The Ethereum blockchain features a permissionless mode, meaning that transactions are transparent and traceable, therefore it is difficult to provide privacy mechanisms. Neither platform mentions in its official documentation how to deal with privacy issues or how to comply with the European GDPR regulation. We believe that is due to the fact that the projects target audience lies mostly outside of the European Union.

Table 2.7: Comparison between blockchain real estate projects

	Rentible	**SmartRealty**	**Midasium**	**RentPeacefully**	**The Bee Token**
Blockchain	Ethereum	Ethereum, Polkadot	Midasium	Ethereum	Ethereum
Permission	Permissionless	Permissionless	Permissioned	Permissionless	Permissionless
Proof of Concept	No	Yes	Yes	No	Yes
Token	RNB	RLTY	No Token	No Token	Bee Token
Payment Method	Digital Currency	Digital / Fiat Currency	Not Specified	Ether	Not Specified

The Midasium project, the only one featuring a permissioned mode, is still in its infancy, so it does not address privacy issues or GDPR compliance. The SmartRealty project, although still considered to be a proof-of-concept, has clearly defined its operational process. The only uncertainty lies in the decision on which blockchain platform should be adopted, Ethereum or Polkadot.

Regarding the payment methods, each platform implements its own approach. The Midasium and The Bee Token projects do not currently specify how payments are handled. The Rentible project allows payments with digital currency, using RNB tokens to cover the service fee. The SmartRealty project allows payments in both digital and fiat currency, which are then converted into RLTY tokens and recorded on the blockchain. Finally, the RentPeacefully project allows payments through Ethers taken from an Ethereum wallet.

The referred projects, although innovative, lack core properties such as transaction privacy and compliance with EU GDPR regulation. Our solution aims to overcome these flaws and provide a service robust, reliable and compliant with the EU GDPR.

Chapter 3

Platform Design

In this chapter, we will introduce a blockchain-based residential smart rental platform. We will decompose its complexity into four phases to ensure the reader does not feel lost. In a first phase, we will begin by briefly describing the current problems in the real estate industry, we address. In a second phase we will generally explain how the platform operates, mentioning its different components. In a third phase, we will discuss its architecture pre-decisions, architecture, blockchain-application interaction, rental payment and rental process. Finally, in a fourth phase, we will discuss the platform's implementation, focusing on the technology stack before concluding with an explanatory presentation on the application's design and user-application interaction.

3.1 Current Problems

As discussed in the **Section 2.1**, the current real estate rental process has several issues that our proposed platform addresses, namely:

- High intermediation costs
- Time-consuming and over-bureaucratic processes
- Information shared over multiple unsecured channels
- Lack of transparency and traceability
- Lack of automatic rental payment mechanisms

3.2 Overview

Our proposed platform aims to solve the problems mentioned in **Section 3.1** by providing a blockchain-based residential smart rental platform, that allows landlords and tenants to manage real estate transactions, through an application that interacts with a Hyperledger Fabric blockchain. Parties can create, sign, manage and delete rental agreements over rental units through a smart contract. Rental units refer to entire properties or single rooms. The platform implements a form of real estate tokenization functionality for entire properties or single rooms. Other real estate token representations mentioned in **Section 2.6.2**, such as ownership, equity interest or interest in a debt secured by a property, have not been included for complexity reasons and irrelevancy with the residential rental matter. The application includes a front-end enabling landlords to advertise their rental units to interested tenants, and integrates functionalities to ensure compliance with the GDPR and to execute rental payments securely. In simple terms, the application is composed by two main components, a front-end that provides different functionalities which together enable interaction between participants, and a back-end responsible for handling the business logic and data processing. The back-end interacts with multiple components, such as, a Hyperledger Fabric blockchain, a dedicated database that stores participants' personal information, and some external entities such as an authorization provider and wallets for transacting crypto payments.

3.3 Architecture Pre-Decisions

In this section we will explain the pre-architectural decisions for our platform, focusing on two key selections, the blockchain platform and the programming language required to write the smart contracts.

3.3.1 Smart Contract Platform

As previously stated in **Section 2.2.3**, we discussed and analyzed several blockchain platforms able to support the development of smart contracts. We will briefly explain why Hyperledger Fabric is the most suitable blockchain for our platform, by comparing its pros and cons with other platforms. We considered decision factors such as core functionality, privacy, GDPR compliance mechanisms, pricing and community support.

Rental contracts in most countries are not obliged to be registered in land registries, however, for tax and legal purposes, they must be reported to the competent authorities. Consequently,

our platform requires to know the identity of the parties involved. Furthermore, we believe that disclosing the identity of the involved parties will fortify user's trust in our platform. As you know beforehand, permissioned blockchains require participants to identify themselves in order to join the network. As a result, when comparing all the studied platforms, we excluded Ethereum and EOS, as they feature a permissionless permission mode, leaving us to discuss among three permissioned platforms, Hyperledger Fabric, Corda, and Canton.

Looking at **Table 2.5** in **Section 2.2.2**, at first glance, the choice of Canton seems obvious, as it stands out for its strong features regarding transaction privacy and GDPR compliance. However, their enterprise version maintained by Digital Asset is extremely expensive for a newly established startup like Unlockit. In addition, it is still a relatively new platform with little community support and a shortage of published documents. Considering these two crucial factors, Canton, for now, does not fit what we desire.

Between Fabric and Corda, we decided based on the amount of community support, which somewhat allows for faster development of a final product. According to our research, Hyperledger Fabric is adopted by more projects having a larger developer community. Another tie-breaking factor is that Corda's paid enterprise mode is recommended for high-volume transaction needs rather than its standard mode. The fact of paying for software at an early stage of a startup should be avoided. Given these reasons, we considered it suitable to adopt Hyperledger Fabric.

3.3.2 Smart Contract Programming Language

The programming language choice for the development of smart contracts is always a platform dependent matter. As stated in **Section 3.3.1**, the proposed platform is based on the Hyperledger Fabric blockchain. Thus, there is a restricted set of programming languages that we could have chosen. **Table 2.5** in **Section 2.2.2** mentions three: Java, Golang, and JavaScript. In the context of our platform, opting for JavaScript was driven by the need for consistency with our blockchain interface server, also coded in JavaScript and designed to interact with the Hyperledger Fabric blockchain. We will explain more about the server in the next section.

3.4 Architecture

In this section we will introduce the architecture of our platform. We will begin by presenting its high-level architecture and progressively detail its components. Afterwards, we will explain the rental and payment procedures, and conclude with a demonstration of the platform's implementation and application design.

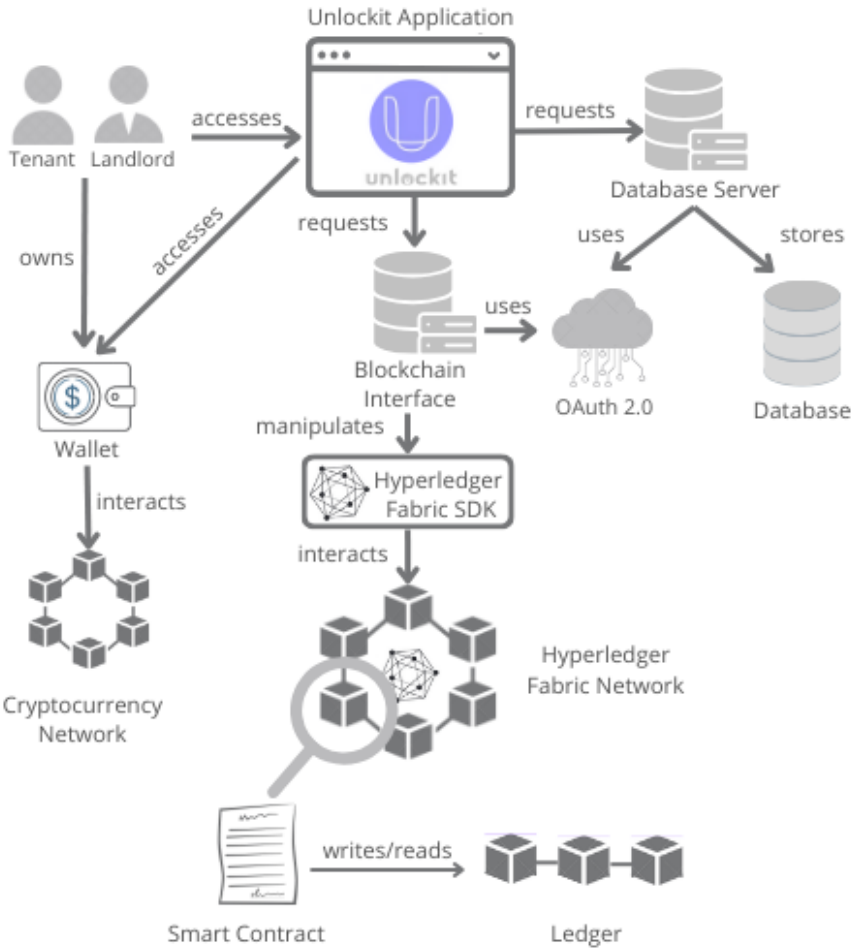


Figure 3.1: Platform architecture

The platform architecture represented in **Figure 3.1** illustrates the interaction between participants. Upper in the figure we observe that both landlords and tenants can access the Unlockit application to manage their real estate transactions. The application interacts with three different components that are interconnected to fulfill each other's needs. The application sends requests to two back-end servers and these dispatch them to different components. Initially, the idea was to develop a single server, however, it became necessary to create a second one to deal with different responsibilities and security.

3.4.1 Database Server

The rightmost server in the picture represents a server interacting with a protocol and a component, namely, OAuth 2.0 [Har12] and a database. OAuth 2.0 is an authorization protocol, provided by a trusted third-party entity, that authorizes users to access the application. This protocol allows users to grant Unlockit's application limited access to their third-party provider account without sharing their credentials. It provides a secure way for users to log into Unlockit's application using their third-party provider account information. Logging in to the application is simple. A user logging in is redirected to the third-party provider's authorization endpoint. Users authenticate on their third-party provider's website and are then redirected back to Unlockit's application with an authorization code and an access token that can access the third-party provider's resources on their behalf. This token is added to each request to the server to guarantee the user's continuous authorization and authenticity.

Regarding the database, it was created to store and maintain data within three interconnected tables. These tables, namely, "User", "Advertise" and "PropertyPhoto" are linked to each other and to the records stored on the blockchain. Briefly and orderly, the first table stores and manipulates information related to user registration, personal information, and attributes required for the application's business logic. The second table stores and manages information related to an advertisement's publication. And finally, the last table stores only images associated with a property, published in an advertisement. We will revisit these tables later in **Section 3.4.5**.

The database server provides the application with an API containing a set of endpoints. The API endpoints are displayed in the **Figures 3.1**, **3.2** and **3.3**.

Table 3.1: Database server - User endpoints

Endpoint	**Description**
api/user/loggedin	Checks if the user is logged in
api/user/time	Gets authorization token expiration time
api/user/login/get/userId	Gets user Id
api/user/login	Logs the user in and creates user initial record
api/user/get/id	Gets user by Id
api/user/update/id	Updates user by Id
api/user/delete/id	Deletes user by Id

Table 3.2: Database server - Advertise endpoints

Endpoint	**Description**
api/advertise/register	Creates a new advertise
api/advertise/user/get/all/id	Gets all advertises of a user Id
api/advertise/user/get/advertises	Gets all advertises from user's list of advertises ids
api/advertise/location/get/all/location	Gets all advertise by location
api/advertise/get/id	Gets advertise by Id
api/advertise/update/id	Updates advertise by Id
api/advertise/delete/id	Deletes advertise by Id

Table 3.3: Database server - Property Photo endpoints

Endpoint	**Description**
api/property-photo/register	Creates a new property-photo
api/property-photo/get/id	Gets property-photo by Id
api/property-photo/property/id	Gets the property-photo associated with a property Id
api/property-photo/delete/id	Deletes property-photo by Id

3.4.2 Blockchain Interface

Going back to the **Figure 3.1**, we can observe the other server right below the application, referred to as the blockchain interface. It is responsible for interacting with the Hyperledger Fabric blockchain through the SDK.

The SDK is the software development kit that provides APIs for our platform to submit transactions, query the ledger, and listen for events emitted by smart contracts on the network. The SDK includes a set of client libraries that make it easy for the application to use the APIs. It also includes command line tools that allow to manipulate the network, to manage user identities, and to deploy or update smart contracts.

The server receives requests from the application, processes them and then submits the requests through the SDK API to the Hyperledger Fabric network. Similarly to the database server, this server, interacts with the OAuth 2.0 authorization protocol to authorize users. All requests from the application require this authorization and authentication, which is the first security barrier between the application and the blockchain.

The server provides the application with a simple API containing four endpoints: "Signup", "Login", "Evaluate" and "Submit". **Figure 3.4** illustrates the server's API with a short description of each one.

Table 3.4: Blockchain interface server endpoints

Endpoint	**Description**
signup	Authorizes user, registers on blockchain, creates credentials and identities, stores in wallets and creates encrypted user id
login	Authorizes user and verifies encrypted user id
evaluate	Authorizes user and executes chaincode read-only functions
submit	Authorizes user, verifies encrypted user id and executes chaincode write-only functions

The first endpoint is only used when a new user is registered in the application for the first time. It is therefore the least used endpoint, but the most complex, as it requires the execution of more operations. It starts by authorizing and authenticating the user using the OAuth 2.0 protocol. It then registers the user on the blockchain and assigns him his credentials, which are represented by a pair of public and private keys, where the public key is represented by a digital certificate. These credentials are used to generate two identities, a public and a private one. The public identity only holds the digital certificate that allows to retrieve the public key. The private identity stores both the digital certificate and the private key. Both identities are extremely important for the security of users and the blockchain. They are stored in two separate file system wallets, A public accessible to the blockchain, and a private accessible to the user only.

After this, the user's public key is used to encrypt the user id and the result is stored on the blockchain. This step provides the second barrier of protection, ensuring that every transaction made by the user to change the state of the blockchain is validated. This validation is necessary

whenever a request is made to the blockchain to change its state, and consists of verifying whether the user encrypted id on the blockchain, when decrypted using the user private key, matches the original user id. In other words, the user needs to decrypt the encrypted id stored in the blockchain, using his private key, and verify, if the result of the decryption process matches the original user id before encryption. If and only if this is verified, the submission is sent.

Finally, a request is sent to the smart contract/chaincode to execute the function "CreateEncryptedId" and persist the data in the blockchain world state. This function registers the encrypted Id in the format shown in **Figure 3.2**. If everything goes as expected, a response is sent to the application and the user may start using it. Further on, we will explain more about the smart contract/chaincode and its available functions.

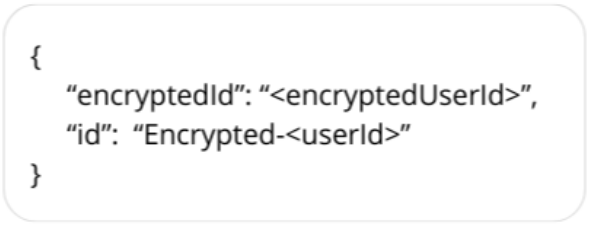


Figure 3.2: Record of an encrypted id representation on the blockchain

The second endpoint, "login", is used whenever it is necessary to log in to the application. The blockchain network must always be active in order to allow users to log in to the application. The endpoint begins by authorizing and authenticating the user using the OAuth 2.0 protocol. It then submits a read request to the smart contract/chaincode function "ReadAsset" to read the encryptedId record of the user's id previously created during the "signup" endpoint, shown in **Figure 3.2**. It then checks whether the user's private wallet exists and extracts the private key. The private key is then applied to the encrypted user id, resulting in the decrypted user id. It then checks whether this decrypted user id equals the original user id. If so, the user's identity is verified and the user is logged into the application.

The third endpoint, "evaluate", is the simplest, as it does not require changing the network's world state. It only executes read functions. Like the others, it starts by authorizing the user using the OAuth 2.0 protocol. If successful, runs one of the read functions provided by the smart contract/chaincode. The function to be executed depends on the application's needs when requesting the blockchain interface server.

The fourth and final "submit" endpoint is used whenever a user wants to make a change to the world state of the blockchain network. It authorizes the user using the OAuth 2.0 protocol, and then, validates him by verifying its encrypted id record, similarly to the process performed during the login endpoint. If validated, the user is able to submit a transaction. In other words, the server is able to call a smart contract/chaincode write-function. And once again, the function to be executed depends on the application's needs when requesting the blockchain interface server

3.4.3 Hyperledger Fabric Blockchain

Taking another look at the **Figure 3.1**, we find the Hyperledger Fabric network representation below its SDK. Despite the figure's simple representation, our network is actually significantly more complex.

We implemented and deployed a Hyperledger Fabric blockchain on a Kubernetes cluster. A Kubernetes cluster is a set of interconnected physical or virtual machines running Kubernetes, an open source container orchestration platform. The cluster enables automated deployment, scaling and management of containerized applications. It consists of a master node that controls the cluster and the worker nodes where the containers are deployed.

Having a Kubernetes cluster ensures efficient use of resources, high availability and simplifies the management of complex applications in a distributed computing environment. We chose to implement the Hyperledger Fabric blockchain on a Kubernetes cluster to simplify the management and scalability of its components, such as peers, orderers and CA's, guaranteeing the network's adaptability to changing workloads.

For simulating a Kubernetes cluster without the overhead of running it on separate virtual machines or cloud instances, we used the KinD tool, short for Kubernetes in Docker. It's a lightweight tool designed for local development, testing, and debugging. It allows to create, run and manage a kubernetes cluster inside a Docker container. In our setup, we employed KinD to establish a Kubernetes cluster. The cluster enables the deployment of Hyperledger Fabric network elements, effectively simulating a real network environment locally.

To deploy the Fabric blockchain in our KinD cluster, we employed two tools, the kubectl plugin and the hlf-operator. Kubectl is the Kubernetes command line tool, while hlf-operator is a Kubernetes operator designed specifically to manage Hyperledger Fabric networks. Hlf-operator

simplified the network set-up process by automating the management of Fabric components using Kubernetes resources. It was further useful as it allowed us to continuously deploy, scale and manage the Fabric network through kubectl commands.

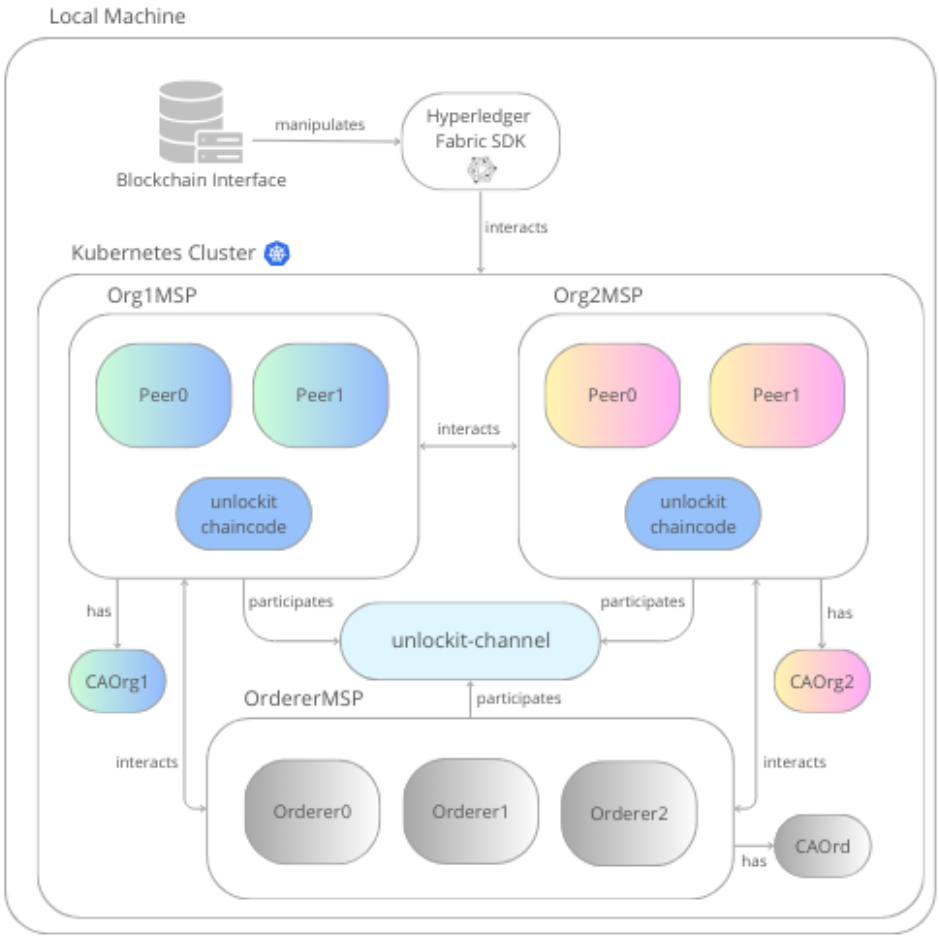


Figure 3.3: Architecture of the Hyperledger Fabric network on kubernetes

Figure 3.3 illustrates a local environment of a machine running our platform. Inside the machine we can observe the blockchain interface server manipulating the Fabric network SDK to interact with the network hosted on a KinD cluster. In this diagram we omitted the remaining components of the architecture depicted in figure **3.1**, so that we could focus our attention on the Hyperledger Fabric network.

Our Hyperledger Fabric network runs inside a KinD cluster and contains three organizations, Org1, Org2, Org3. Each one has its own Membership Service Provider and Certification Authority, represented in the figure **3.3** by Org1MSP, Org2MSP, OrdererMSP, CAOrg1, CAOrg2 and CAOrd respectively. The CAs manage user identities, issue certificates and handle authentication processes within the organization. All organizations participate in the same channel, which consequently implies that all peers within those organizations participate in the same channel

as well. Both organizations, Org1 and Org2, have the same number of peers, Peer0 and Peer1, and the same chaincode deployed on them. Each peer maintain a copy of the ledger and a copy of the chaincode associated with the channel. The organization marked in grey, represents the organization in charge of the network's consensus. It is composed by three ordering peers who establish the consensus of the transactions, validating and maintaining the network.

In our platform, when the application submits a transaction to the Fabric network, the request is sent to the blockchain interface server, which uses the SDK to submit the request to either organization 1 or 2. The selected one, sends the transaction proposal to its two peers for approval. They simulate it and provide an approval signature. Once approved, the SDK sends the approved transaction to the ordering organization's to be ordered by its ordering peers. At this stage, the ordering peers, group the transactions into blocks, reach a consensus on the order of the transactions and then create blocks containing the endorsed transactions. Afterwards, the blocks are disseminated to the peers of all the participating organizations (Org1, Org2) for validation and commitment to their ledgers.

Our system is specifically designed to allow state changes in the blockchain solely through transaction submissions to organization 1. Consequently, organization 1 holds the read and write permissions, while organization 2 is limited to read-only access. Essentially, organization 2 can only execute query transactions within the network. This deliberate design ensures that the organization 1, (Unlockit organization), retains exclusive authority over ledger modifications, while the remaining organizations would exclusively act as network auditors.

3.4.4 Smart Contract

Now that we understand the structure of our network, lets examine its smart contract, and the assets it manipulates. As mentioned in the previous section, our Hyperledger Fabric network contains only one smart contract in charge of interacting with the assets stored on the blockchain network.

As mentioned in **Chapter 2** in **Section 2.3**, assets are digital representations of real-world objects such as a real estate property. They are represented as a collection of key-value pairs in binary and/or JSON format. Our platform is designed to contain six assets, namely, "PropertyAsset","ContractAsset", "Proposal", "RentalInfo", "Payment" and "EncryptedId", all registered on the blockchain through different chaincode functions.

Let us briefly explore the purpose of each one. The "PropertyAsset" record represents a physical property. The "ContractAsset" record encapsulates digital rental contract details, including contractual information and digital signatures from both parties. The "Proposal" record represents a tenant's proposal for a particular advertisement. The "RentalInfo" record provides an analytical analysis of all proposals made for an advertisement. It is only created when a rental contract is confirmed and records two metrics, the number of proposals made to an advertisement by all potential tenants, and the proposal with the highest value. The "Payment" record, registers the payments linked to a rental contract. Finally, the "EncryptedId" record, as explained in the **Section 3.4.2**, serves to verify users' identities before letting them submit transactions to change the state of the blockchain.

Now that we have understood the different assets and their respective utility, let us examine how the chaincode manipulates them. Hyperledger Fabric chaincode works similarly to a common API, offering CRUD (Create, Read, Update, Delete) operations. However, it distinguishes itself due to the fact that, data management is specifically targeted towards the blockchain.

Our chaincode offers an API containing several functions targeting different records. **Tables 3.5**, **3.6**, **3.7**, **3.8**, **3.9**, **3.10**, **3.11**, and **3.12**, describe the chaincode's API functions for each each asset/record presented plus three more records necessary to run the platform. Each table row contains the function name, visibility and description.

Table 3.5: General chaincode functions

Function	**Visibility**	**Description**
ReadAsset	public	Reads any asset
GetAllAssetsByAssetType	public	Gets all assets of a given asset type
AssetExists	public	Checks if asset exists
readAssetObject	private	Reads any asset as an object
getAllAssetsByAssetTypeAsObject	private	Gets all assets of a given asset type as an object

Looking at the tables, we notice that there are public and private functions. The private ones are only accessible from within the chaincode, meaning that they can be called by other chaincode functions. The public ones can be called by the Fabric SDK through an application request.

Table 3.6: Property Asset chaincode functions

Function	**Visibility**	**Description**
CreatePropertyAsset	public	Registers a new property asset
DeletePropertyAsset	public	Deletes a property asset
ReadAllPropertiesByLandlordId	public	Reads all properties linked to a landlord
getAllPropertiesByLandlordId	private	Gets all properties linked to a landlord

Table 3.7: Contract Asset chaincode functions

Function	**Visibility**	**Description**
CreateContractAsset	public	Registers a new contract asset
UpdateContractAsset	public	Updates an existing contract asset
DeleteContractAsset	public	Deletes an existing contract asset
ReadPropertyActiveContract	public	Reads an active contract asset linked to a property asset
ReadAllContractsByPropertyId	public	Reads all contracts linked to a property asset
getActiveContract	private	Gets an active contract asset if exists
getAllContractsByPropertyId	private	Gets all contract assets linked to a property asset

Every public function altering the blockchain state validates the user id. When the blockchain interface server invokes any of these functions, sends a request, with the user id as the first argument, previously verified by the server. During execution, this id is compared with another id, also provided as an argument. The function proceeds only if the ids match. To make it clearer, we will demonstrate with two functions.

The "CreatePropertyAsset" function receives the user id as the first parameter and the other parameters relate to the property being created. One of them is the "landlordId", which must match the user id of the user submitting the request. The function only proceeds its execution if the user id is equal to the "landlordId". Let us now take as an example the "DeletePropertyAsset" function which receives two parameters, the user id and the property id.

Table 3.8: Proposal chaincode functions

Function	**Visibility**	**Description**
CreateProposal	public	Registers a new proposal record
UpdateProposal	public	Updates an existing proposal record
DeleteProposal	public	Deletes a proposal record
ReadAllProposalsByContractId	public	Reads all proposals linked to a contract asset
getAllProposalsByContractId	private	Gets all proposal records linked to a contract asset
getHighestProposal	private	Get the highest proposal from all the proposal records

Table 3.9: Rental Info chaincode functions

Function	**Visibility**	**Description**
CreateRentalInfo	private	Registers a new rental info record

Table 3.10: Payment chaincode functions

Function	**Visibility**	**Description**
CreatePayment	public	Creates a payment record linked to a contract asset
UpdatePayment	public	Updates a payment record linked to a contract asset
DeletePayment	public	Deletes a payment record linked to a contract asset
VerifyPayment	public	Verifies if a payment record is confirmed
getContractByPaymentId	private	Gets contract asset linked to a payment record
processMonthlyPayments	private	Processes all monthly payments for all contract assets.

Table 3.11: Encrypted Id chaincode functions

Function	**Visibility**	**Description**
CreateEncryptedId	public	Creates a encrypted id linked to a user
DeleteEncryptedId	public	Deletes an encrypted id linked to a user

Table 3.12: Utils chaincode functions

Function	**Visibility**	**Description**
getCurrentRealWorldDate	private	Gets the current real world time based on the user location
extractDateFromString	private	Manipulates date
extractDate	private	Extracts date from specific date format
isOrgValid	private	Checks whether the organization to which the transaction is submited is valid

Note that, only the owner of the property must be able to execute this function. To verify this, we use a private chaincode reading function that retrieves the JSON of the property's asset. From here, we obtain the "landlordId", since the property asset contains this information. Finally, we compare the two ids and check whether they are the same.

Both of these examples are a sample of what is done similarly in other public chaincode functions. By doing so, we guarantee that all landlords are able to create properties, but only the landlord who owns a given property can delete it. We guarantee that all landlords are able to create rental contracts, but only the landlord of a given contract can delete and update it. We guarantee that all tenants are able to submit proposals, but only the tenant who submitted a given proposal can delete it. Finally, we guarantee that all landlords are able to create payments, but only the landlord and tenant of a given payment can update and delete it.

There are also other constraints that apply to specific public functions within the chaincode. For instance, a property, a contract, or a payment record, cannot be deleted if there is an active rental contract. Moreover, a user cannot delete his data from the platform if he is involved in an ongoing rental contract, either as a landlord or tenant, or if there are pending payments to receive or send.

3.4.5 Data Storage Architecture

Having explored the different components within our platform's architecture, we will now illustrate their interconnection, to provide a comprehensive understanding of how data storage components collaborate.

Figure 3.4: The data storage architecture

Figure 3.4 shows a compact diagram of how data is managed and stored on the different components of our platform. It offers a detailed perspective on the interconnections between database tables and their relationships with the blockchain records. Each square is labeled with the table or record name, the storage type, and multiple attributes. The storage type is an acronym represented on the square's top right corner indicating "DB" for Database and "BC" for Blockchain. The relationships between squares are represented by arrows, whose tips indicate ownership or reference.

3.4.6 Rental Payments

Traditional payment methods have long been a fundamental aspect of financial transactions. These conventional methods involve various channels such as cash, checks, credit cards, and bank transfers. However, in recent times, there has been a notable shift towards cryptocurrency payments, offering a decentralized and secure alternative.

Cryptocurrency payments, particularly in stablecoins like USDC and USDT, have gained substantial traction due to their inherent stability tied to real-world assets. There are several types of stablecoins. Those that are backed by physical assets, those that are fiat-collateralized which are held by a real-world currency such as the US Dollar, and those that are managed algorithmically by an algorithm that maintains a stable value by controlling its supply. Stablecoins, provide a level of predictability in value, addressing the price volatility often associated with cryptocurrencies.

In the context of our platform, we can observe by looking at the left-hand side of **Figure 3.1**, that we focus on stablecoin payments on a cryptocurrency network. In the figure, each user of the application has access to its own wallet, serving as a storage space for their stablecoins. The wallet's primary function is to engage with the cryptocurrency network by submiting and receiving transactions. Our platform enables users to access their wallets securely through the application. This capability is made possible as the application seamlessly integrates with various wallet types. Depending on the user's preference, the wallet can be accessed through a browser extension, a mobile or a desktop application.

To perform a transaction, its necessary to know two pieces of information about the receiver. The amount to be sent and the address to which we want to send the amount. In our platform, tenants make rental payments by sending funds from their wallets to the landlord's cryptocurrency network address. We chose to execute payments on a cryptocurrency network due to its open nature, promoting a democratic financial ecosystem.

Our platform maintains a payment record on the Fabric blockchain associated to a rental contract. **Figure 3.5** displays an expanded version of the payment record extracted from **Figure 3.4**. The payment record is composed by multiple attributes essential for validating rental payments.

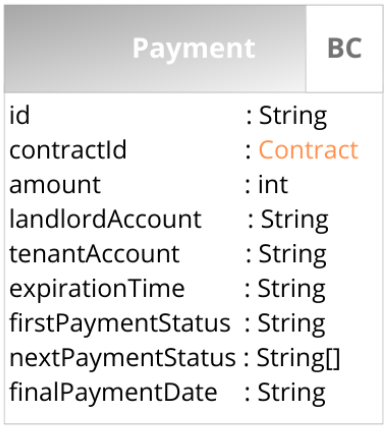


Figure 3.5: Payment record

Each payment record contains: a reference to the contract id linked to it, the amount to be paid, the sender and recipient accounts, represented by their cryptocurrency network address, the first payment's expiration time, the first and subsequent payment's transaction status, if applicable, and the payment's final date.

There are two types of rental contracts, long and short term. Depending on the rental type, the payment will be executed monthly or in one single sum. If it is a short-term contract, it will last less than a month, and therefore the payment will only be made once. On the other hand, if it is a long-term contract, it will last longer than a month and the payment will be made on several occasions. In this case, after the initial payment, the smart contract will process all subsequent payments on the first day of each month.

The payment record is adaptable to different types of contracts. It maintains a status for each rental payment. The status represents whether or not a given transaction has already been confirmed on the cryptocurrency network. All payment records hold a status for the first payment. However, only long-term contracts require their payment record to hold the status of the following rental payments.

Figures 3.6, and **3.7** illustrate two interaction diagrams demonstrating the payment process from the landlord's and tenant's perspectives, as explained in this section.

As an example, a short term contract, lasting 25 days, has a payment record holding a single reference to the status of the first payment transaction, whereas a long term contract, lasting say, 3 months, has a payment record, holding reference to the status of the first and the two

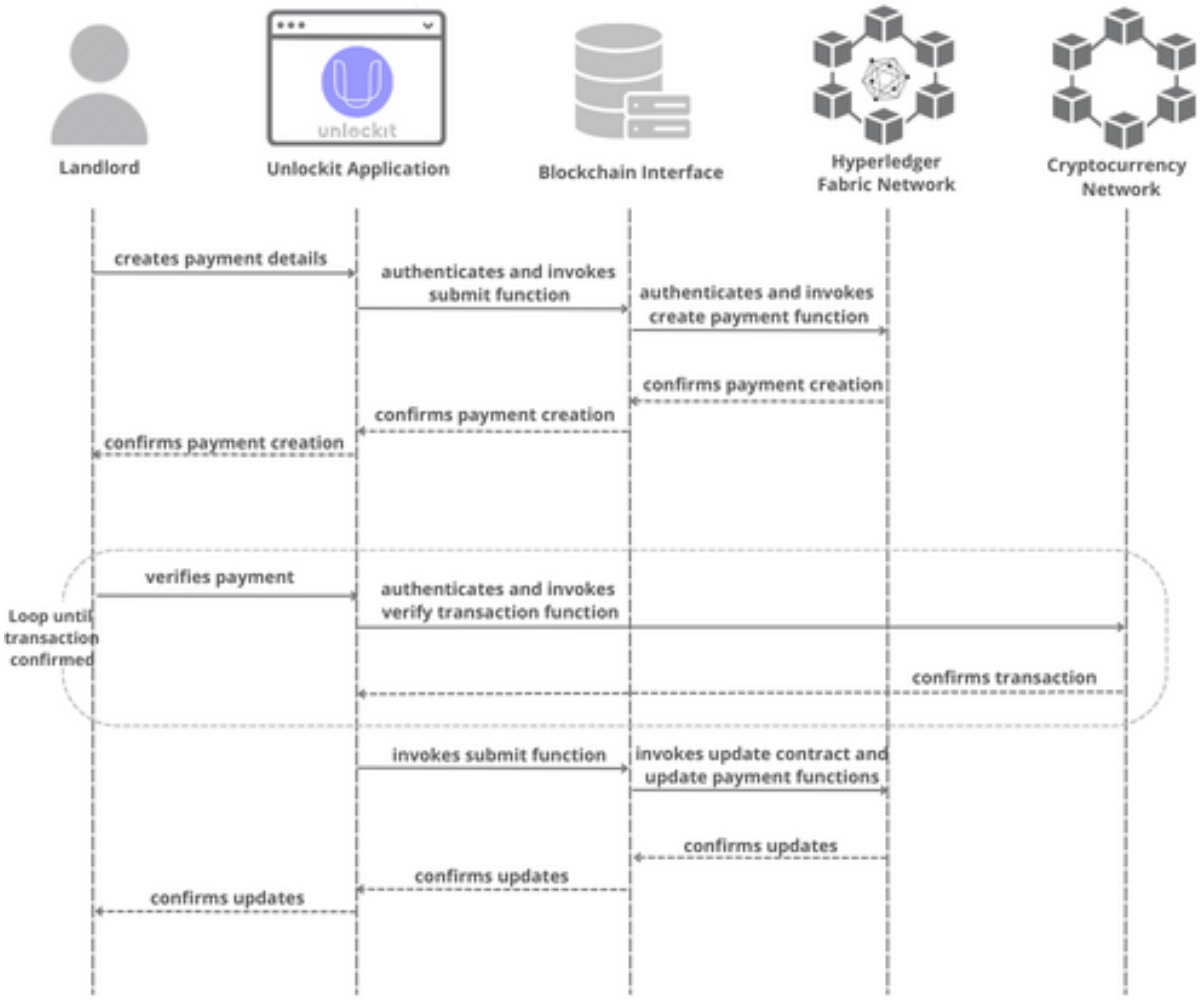


Figure 3.6: Interaction diagram depicting a landlord creating and verifying a payment

subsequent payment transactions. The first payment is coupled with an expiration time defined by the landlord on the platform, which represents the time limit within which, the tenant must make the payment. Finally, the last attribute is used to identify the end date of a contract's last payment.

The first payment is always generated by the landlord through the application. The application sends the request to the blockchain interface server, which then invokes the chaincode function to create a payment. However, in order to generate the payment for the following monthly rents, it is necessary to automate the process, avoiding manual user interaction.

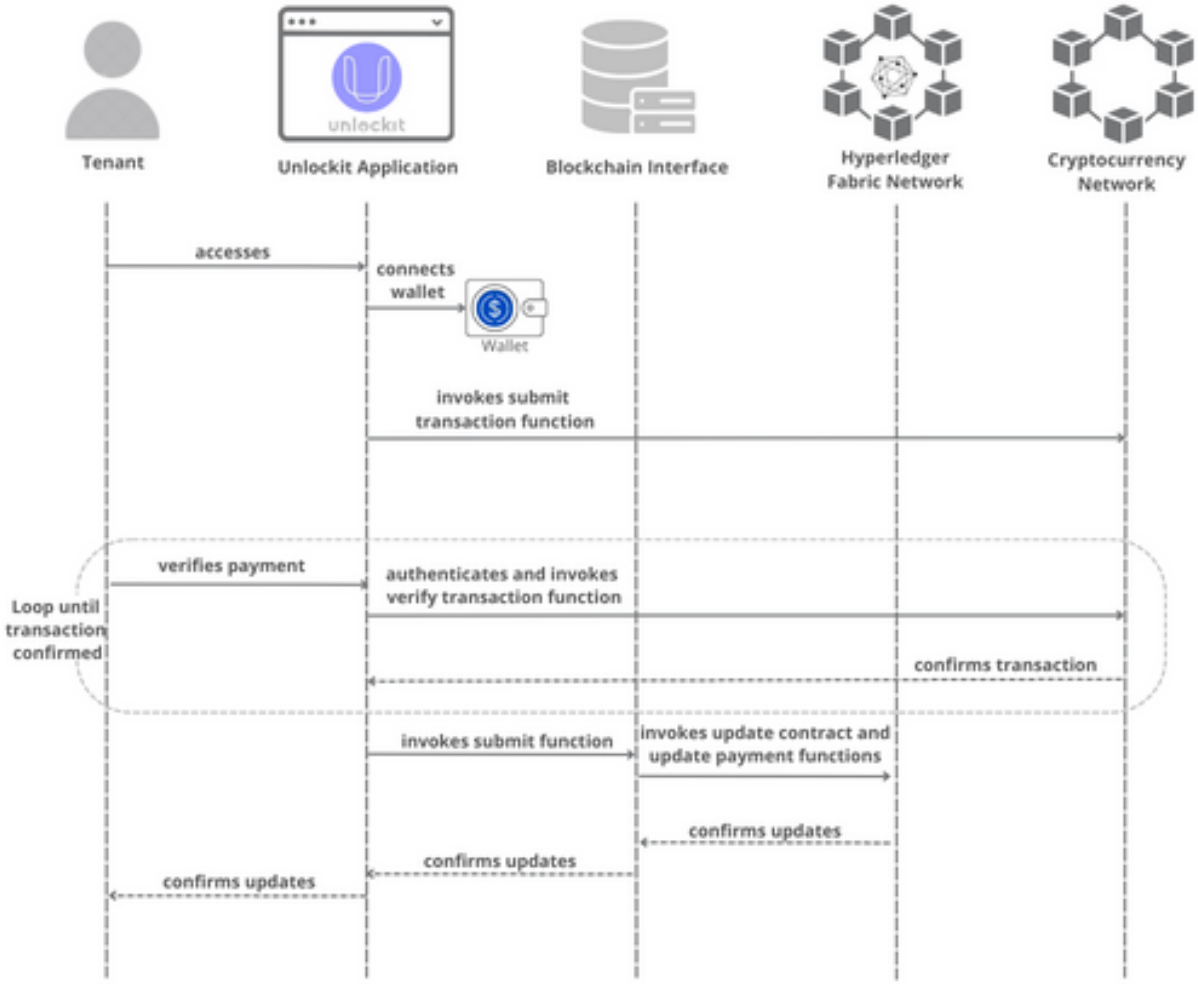


Figure 3.7: Interaction diagram depicting a tenant sending and verifying a payment

Our platform uses an external scheduler service, a cron job in kubernetes, to invoke the "processMonthlyPayments" chaincode function at midnight in the first day of each month, processing all pending payments for all rental contracts.

3.4.7 Rental Process

Until this point, we have introduced all the platform components. However, we still have not explained how the process of renting a property actually works. Lets understand it. **Figure 3.8** illustrates the steps required to establish a rental contract.

Users, whether tenants or landlords, log into the application through a third-party provider's authorization mechanism, which assigns them with their third-party provider's user unique id. When a new user accesses the application for the first time, he must authenticate with the third-

party provider. Subsequently, his identity is registered on the Fabric blockchain, paired with a generated key pair, later stored in a public and private wallet. The verification and authorization mechanism provides trust between application users, since all users are authenticated by a trusted third-party provider, and identified on the blockchain by their user unique id. This is intended to prevent fraud in a subsequent rental process.

Figure 3.8: The rental process

Once authenticated, users can access the application to consult the advertisements list. However, they are only able to create advertisements and submit rental proposals after registering their personal information in the application.

After registering their personal data, landlords may start creating advertisements. Creating an advertisement requires defining the property and rental contract details. Ideally, the property details should be unique, to ensure a unique representation of the property on the blockchain.

Although not implemented in our platform, a property should be verified before being published by checking, for instance, its title deed or tax registration.

The platform permits advertising different types of properties, ranging from entire houses to different apartment typologies and even single rooms. The contract duration influences two aspects, the type of rental and the payment procedure it supports. According to the rental types discussed in **Section 2.1.1**, we support fixed-term, month-to-month, short term and room rentals. The fixed-term modality is part of the long-term contracts, as it requires continuous payments over a period of time longer than one month. The month-to-month and short-term modalities behave differently. They are part of the short-term contracts lasting less than or equal to one month. Room rentals do not depend on the contract duration, but instead on the type of space within a property. It is therefore possible to define a long-term or short-term contract for this kind of rental.

Once defined all the advertisement details, the landlord publishes it on the platform. Three records are created during the publication, namely, advertisement, property and rental contract. The advertisement record, is stored in the database and holds two references, one for the property and the other for the rental contract. The property and the rental contract records are stored on the Fabric blockchain network, and are represented as NFTs.

The rental contract NFT is assigned with the landlord's digital signature. Briefly, a digital signature is a cryptographic technique used to verify the authenticity and integrity of a digital message. It guarantees that the sender is genuine and the content has not been tampered with during transmission. To create a digital signature, the sender uses his private key to encrypt a hash of the message. The recipient can then use the sender's public key to decrypt the hash and verify its authenticity. If the decrypted hash matches the recalculated hash of the message received, the digital signature is valid, confirming the identity of the sender and the integrity of the content. As for the digital signature generated by the landlord, the message is a concatenated string containing a few contract details such as the, contract id, property id, term, initial, final dates and conditions. The digital signature is therefore generated by applying the SHA256 hash, chosen for its security, to the concatenated string, followed by the user's private key. The landlord's digital signature applied to a rental contract guarantees the non-repudiation of the landlord who signed it. A tenant wishing to verify the landlord's digital signature is allowed to do it, by applying the landlord's public key to the landlord's digital signature.

Once an advertisement is active, a potential tenant consults the advertisements list and submits a digitally signed proposal to the advertisement. Since the proposal record contains a reference to the contract record, the tenant produces a digital signature for the same concatenated string as the landlord did when creating the contract. The proposal represents the tenant's acceptance of the contract terms set by the landlord, together with the adjustable rental amount that the tenant wishes to submit.

The landlord is notified in the application about a potential tenant's interest and checks the proposal submitted. Note that, the landlord may have several proposals for the same advertisement. At this point, the landlord can opt to either reject or accept the proposal. If rejected, the potential tenant will be notified in the application and may resubmit another proposal for the same advertisement. If accepted, the advertisement will be temporarily unavailable to other users until payment is confirmed or rejected. Once the proposal has been accepted, the landlord inserts his payment details including a configurable payment time limit and sends it to the tenant. If the payment transaction is not registered on the cryptocurrency network within the configured time limit, the proposal is rejected and the advertisement is made available to all users again. As for the payment details, the amount will be automatically displayed on the platform according to the tenant's proposal amount. The landlord simply has to select the stablecoin (USDC, USDT) he prefers to be paid in, and enter his cryptocurrency network account public key, which is the alphanumeric address associated with his cryptocurrency network wallet.

The payment details together with the payment deadline are sent to the tenant's email address. After receiving the details, the tenant connects his wallet through the application, enters the payment details and submits the transaction. The wallet can be connected by opening its browser extension or by scanning a QR code to open it on the tenant's mobile device. Once the transaction is sent to the cryptocurrency network, the proposal can no longer be rejected, unless the transaction expires, which does not usually happen.

The time for confirming a transaction on a cryptocurrency network is variable, therefore, we can not tell when a transaction will be effectively confirmed. Depending on the cryptocurrency network implemented, a transaction may take seconds, minutes or hours to be confirmed. Therefore, both parties have to be constantly verifying it until one party actually confirms it. As soon as one of the parties verifies the transaction confirmation, the platform updates the rental contract's NFT, by adding the new tenant id along with the tenant digital signature obtained from the proposal previously submitted by the tenant.

Depending on the rental type, long or short term, the payment will be executed monthly or in one single sum, as explained in the **Section 3.4.6**. If it is a short-term contract the payment will only be made once, while, if it is a long-term contract, it will be made on several occasions. In this case, after the initial payment, the smart contract will process all subsequent payments on the first day of each month. All subsequent payment details are generated using the same information as the first one, including the landlord's cryptocurrency network address and the exact rental amount. However, unlike the first, the subsequent payments do not require a payment time limit.

3.5 Implementation

Now that we have learned about the platform's architecture, lets discuss the actual implementation. This section explores the technologies used to build the described architecture, providing practical insights about our application. **Figure 3.9** depicts the platform high-level architecture with the implemented technologies.

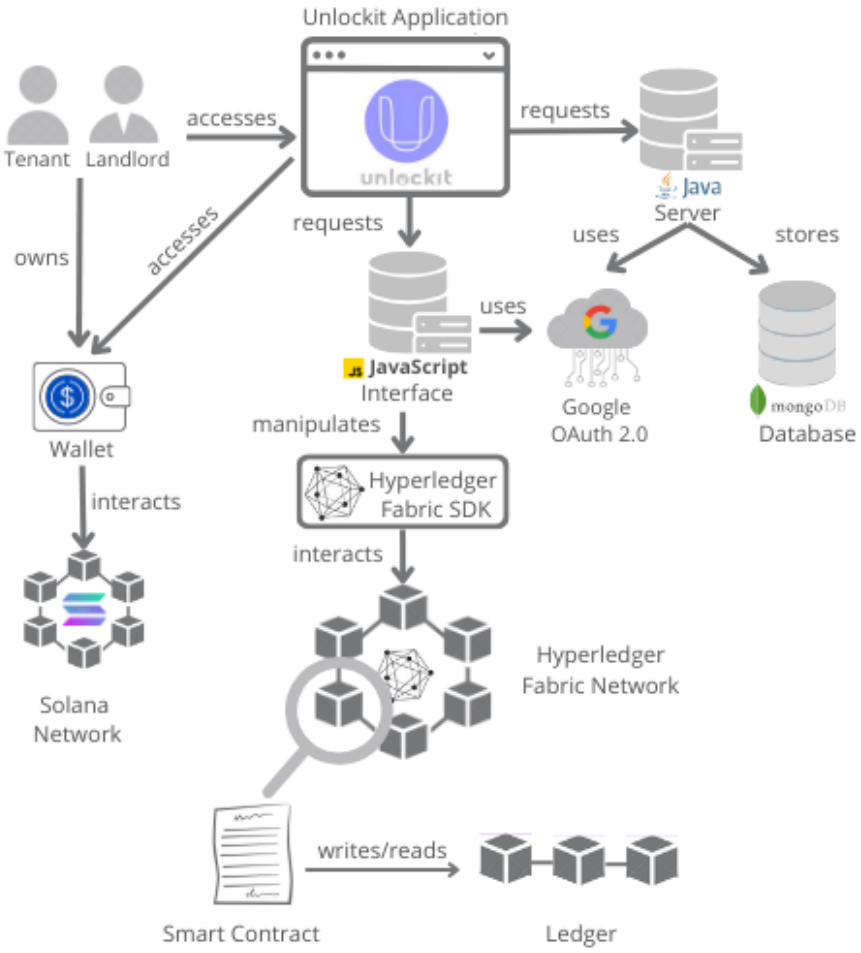


Figure 3.9: Platform architecture implementation

As mentioned briefly earlier in **Section 3.2**, our platform provides an application with a user-friendly front-end, implemented in the React framework. The application is equipped with a complex back-end involving different components. A user authorization protocol, OAuth 2.0 provided by Google, to authorize and uniquely identify users. A server written in java that interacts with an external Mongodb database to store users' personal data, and a dedicated server written in JavaScript/Typescript running on Node runtime environment that interacts with the Hyperledger Fabric network through an SDK.

MongoDB was selected as the database solution because of its flexible, schema-less document model, which accommodates the dynamic nature of user data without the need for predefined structures. Utilizing a JSON-like format, MongoDB's BSON storage facilitates seamless integration with the application code, aligning with the platform's objectives of simplicity and ease of data management. This approach aligns with the principles of NoSQL databases, offering scalability and adaptability to evolving user data requirements. Furthermore, MongoDB provides built-in features to enhance data security, including encryption at rest and in transit. The database supports encryption of data files, ensuring that sensitive user information is safeguarded when stored on disk. Additionally, It incorporates TLS/SSL encryption to secure data during transit between the application and the database. This encryption protocol helps protect against potential eavesdropping and unauthorized access to the data during transmission.

Regarding payments, the application integrates a Web3 wallet through a browser extension or a mobile application for processing payments using Stablecoins in Solana, a high-performance blockchain platform. Numerous cryptocurrency network platforms supporting stablecoins were available for consideration, however, Solana was chosen due to its fast transaction confirmation times, processing thousands of transactions per second, making it an efficient choice for real-time transactions. Additionally, Solana's low gas fees enhance the cost-effectiveness of cryptocurrency payments, making it an attractive option for our application and for testing purposes.

3.5.1 Application Design

To conclude this chapter, we will present our application design aligned with the implementation technologies covered in the last section. We will illustrate the different pages of the application and understand the interaction between application-user and between landlord-tenant. The application is organised into eight different web pages, each, with a unique interface design responding to specific user needs. **Table 3.13** identifies all these web pages along with with a description of their respective functionalities.

Table 3.13: List of application web pages

Web Page	**Description**
Inital	The application's initial page
Login / Register	Logs in or registers users in the application and on the blockchain
Search	Allows users to search for property advertisements by location
Listings	Lists and filters the set of property advertisements selected by location. Allows tenants to select a property from the listings and contact or submit a rental proposal to the landlord
Advertises	Allows landlords to create advertisements for their rental properties
Contracts	Allows landlords to monitor the status of their rental contracts, receive multiple proposals for a contract, accept or reject proposals, connect their wallets, generate and verify rental payments. Allows tenants to consult the status of their rental proposal submissions, connect their wallets and verify payments.
Properties	Allows landlords to inspect their rented or vacant properties, and tenants to inspect their rented properties
Profile	Allows users to insert their personal information

Lets now demonstrate the interface of each page and understand how users interact with them.

Initial

The initial page shown in **Figure 3.10** is the entry point to our platform. The interface allows users to proceed to the login / register page

Login / Register

The login / register page shown in **Figure 3.11** allows users to access the platform through the OAuth 2.0 protocol provided by Google. Once authorized, users are redirected to the search page.

Figure 3.10: Application initial page

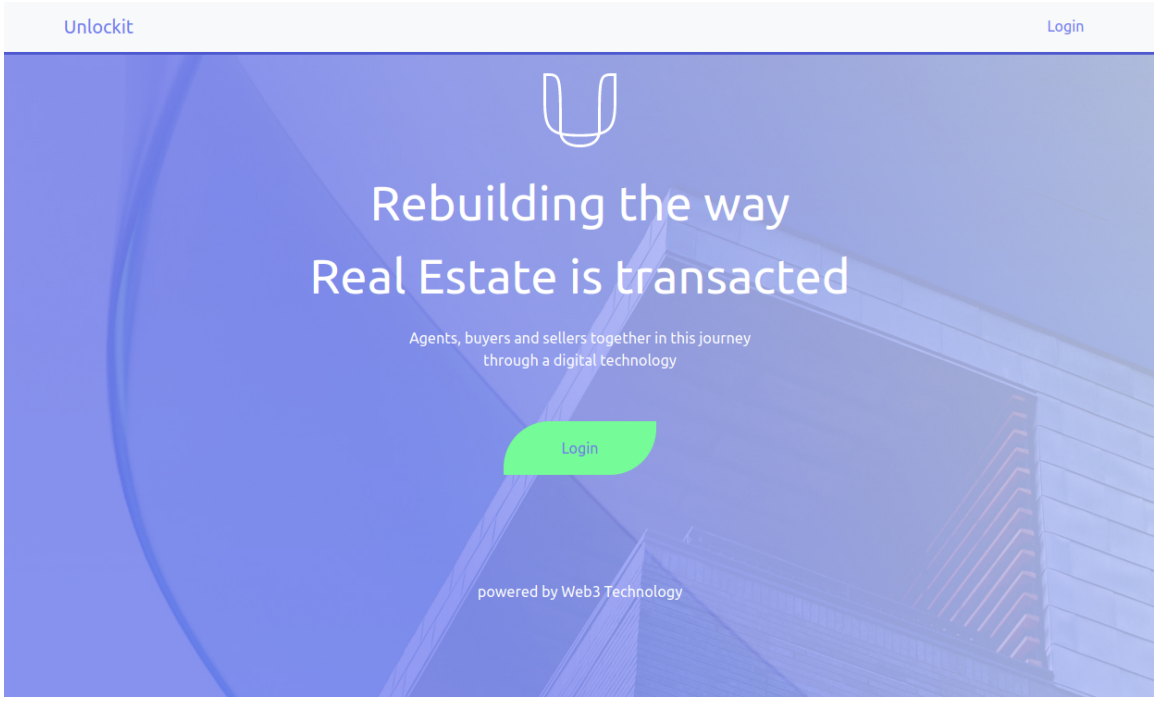
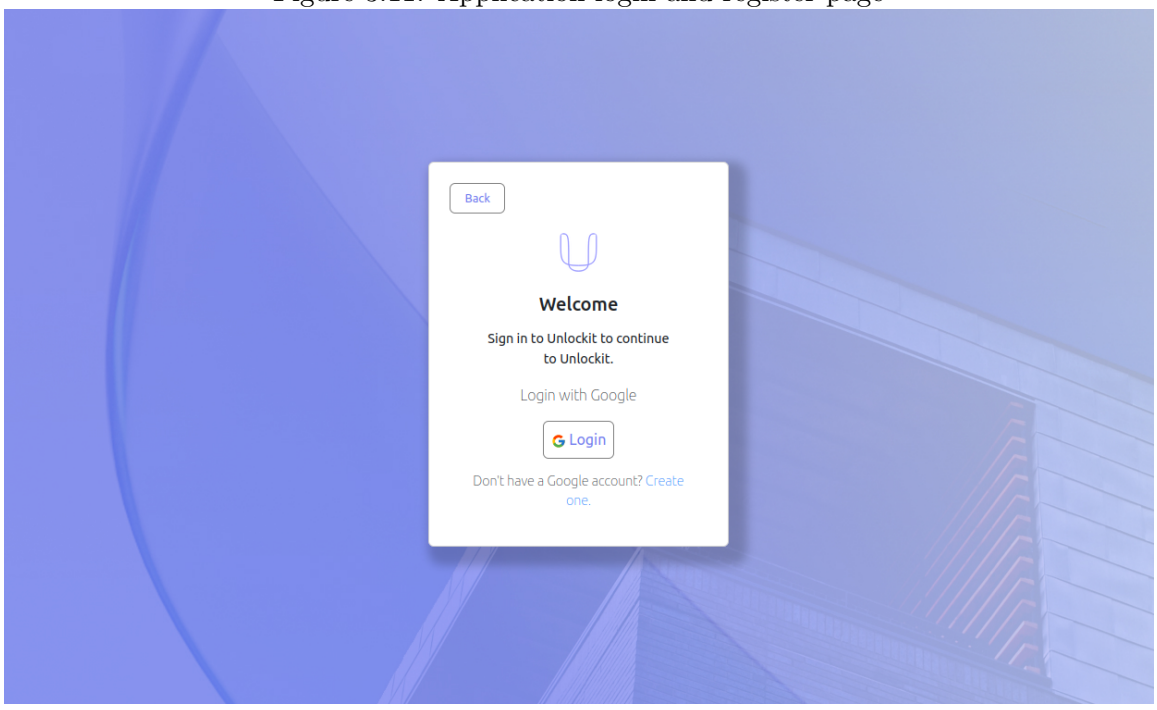


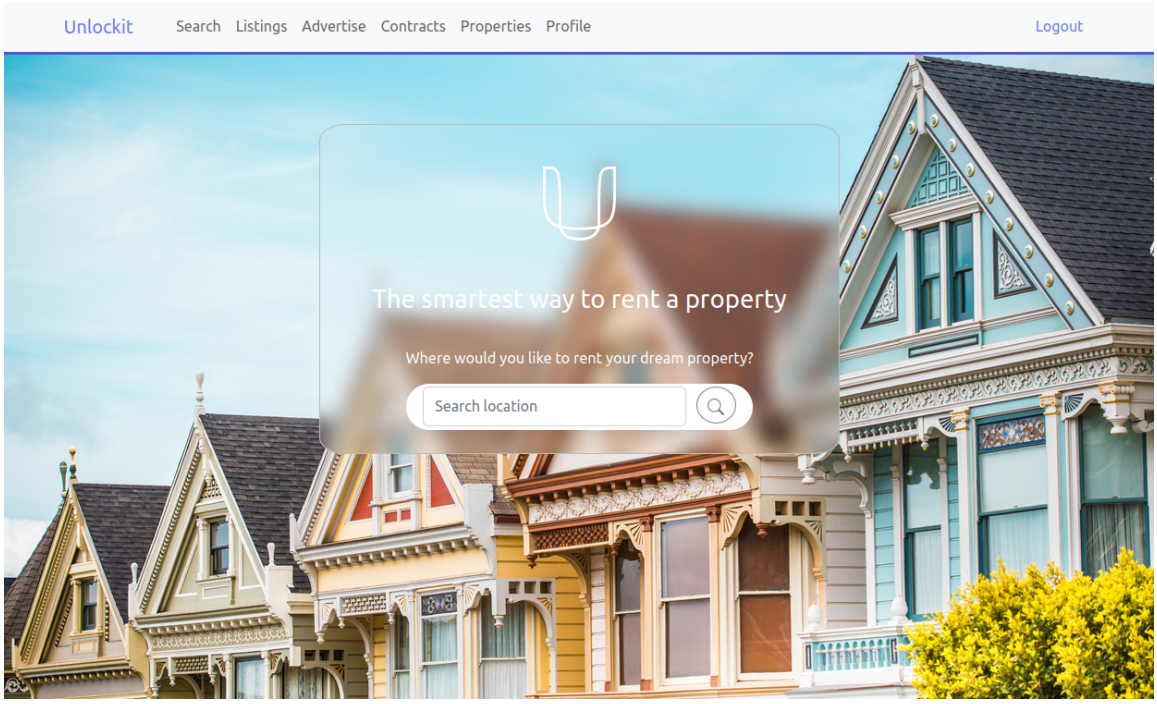
Figure 3.11: Application login and register page



Search

The search page shown in **Figure 3.12** allows users, regardless of whether they are acting as tenants or landlords, to search for property advertisements by location. The search results redirect users to the listings page.

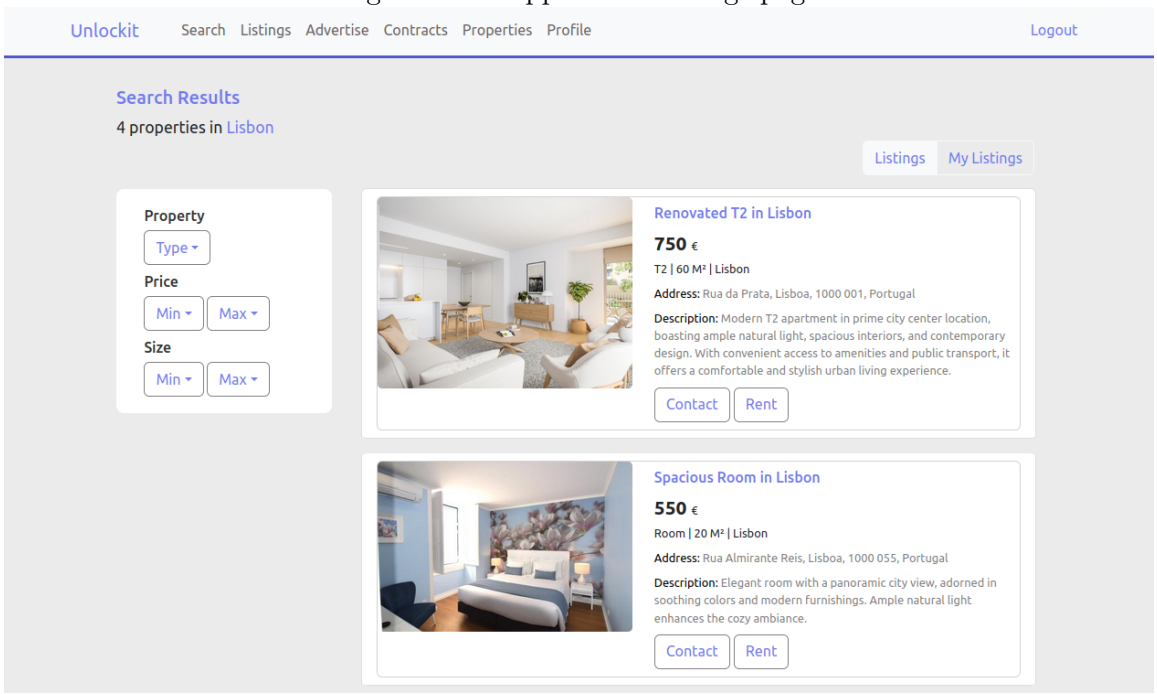
Figure 3.12: Application search page



Listings

The listings page shown in **Figure 3.13** is complex as it enables several actions to be taken.

Figure 3.13: Application listings page



Users are able to view their own or other property advertisements. However, only tenants may view the buttons within an advertisement. They may contact the property's landlord by clicking on the "Contact" button and submit a rental proposal by clicking on the "Rent" button.

Figure **Figure 3.14** illustrates the interface after clicking on the "Rent" button, and **Figure 3.15** illustrates the interface after clicking on the "Contact" button.

Figure 3.14 shows the area, where a tenant can review the rental contract details, propose a new rental amount, and submit a rental proposal by clicking on the "Submit Proposal" button. After submitting a proposal, the tenant is redirected to the contracts page.

Figure 3.15 shows the area where a potential tenant can access a landlord's contact information for future contact.

Figure 3.14: Application listings page - rent

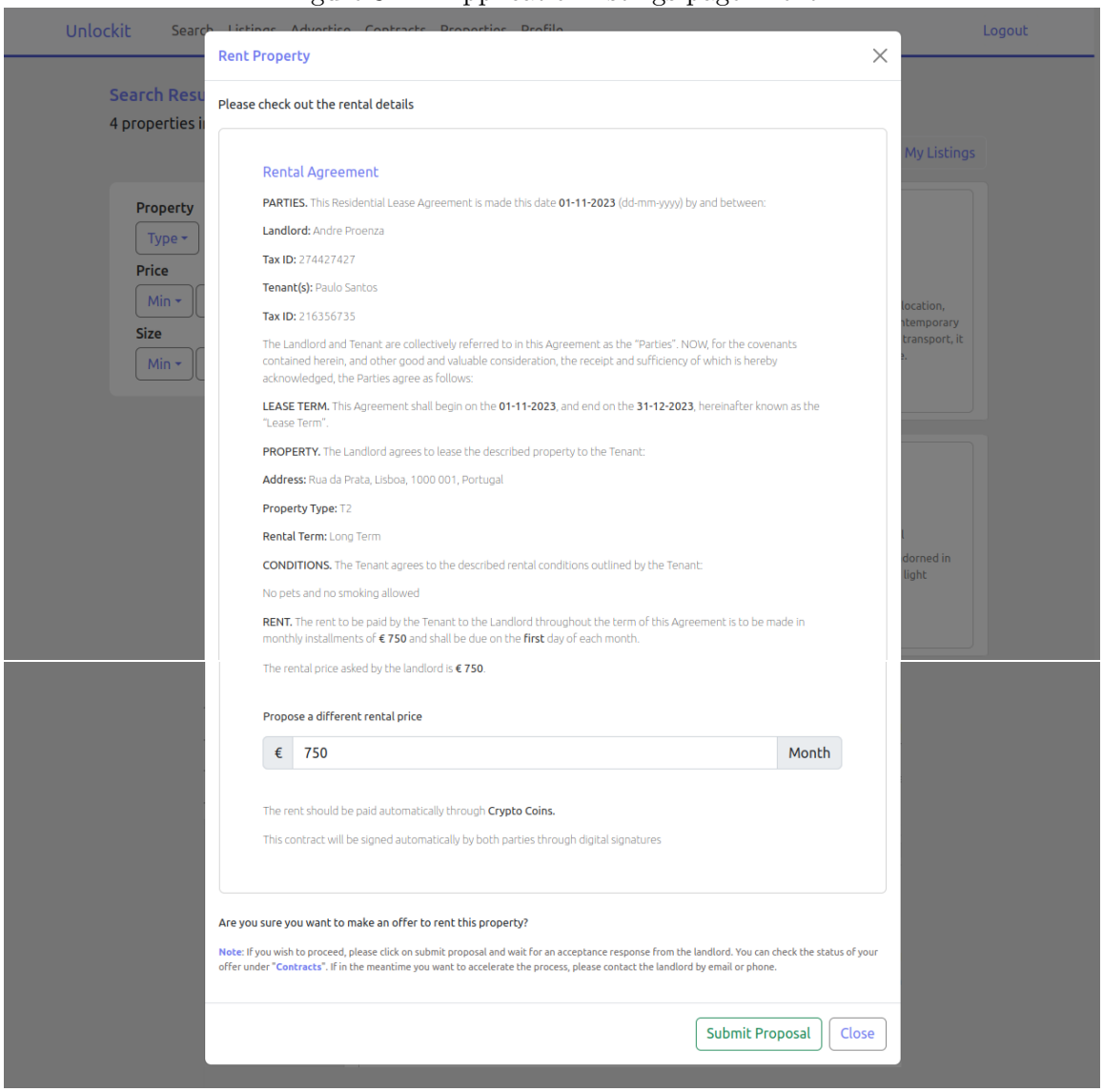
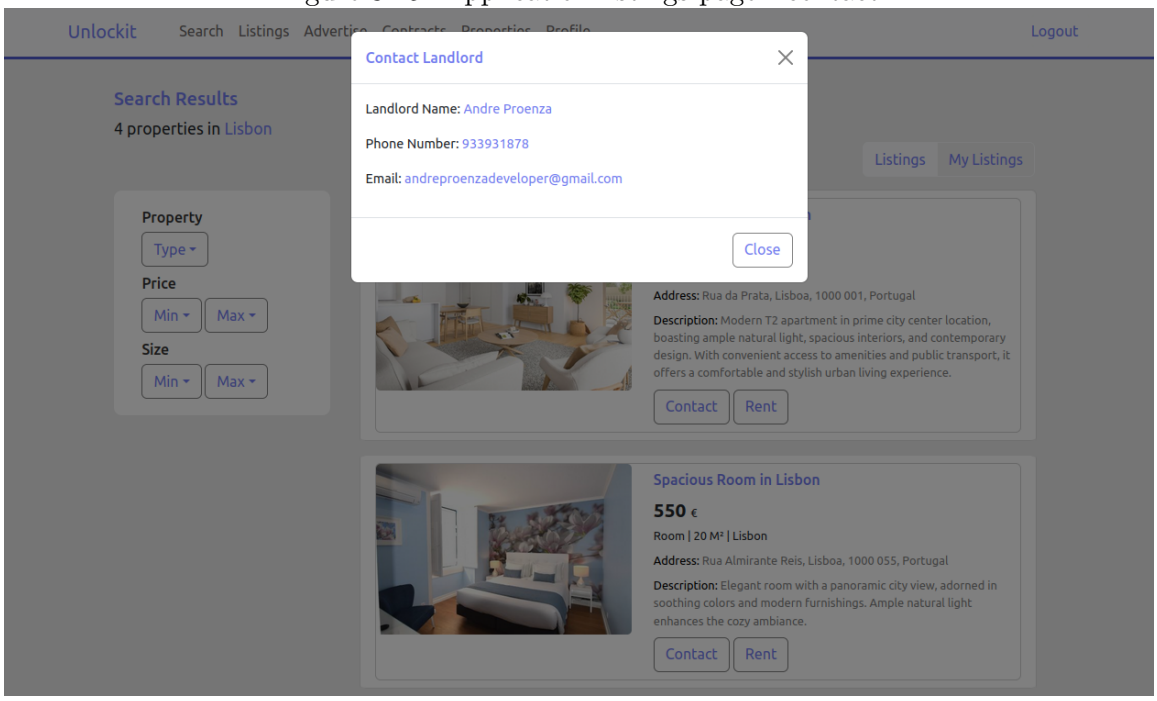


Figure 3.15: Application listings page - contact



Advertises

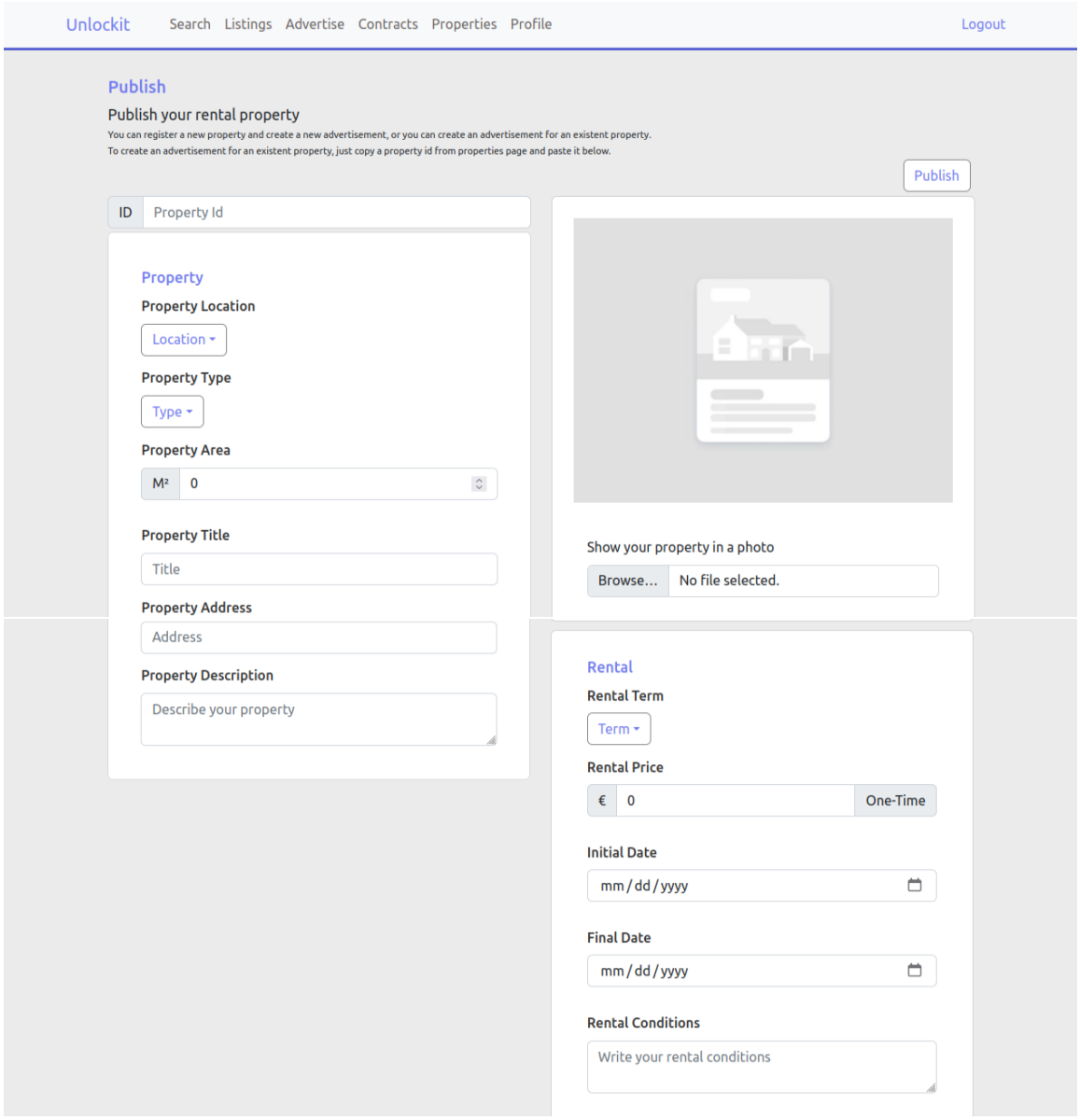
The advertise page shown in **Figure 3.16** allows landlords to publish property advertisements. Within the interface, landlords insert two distinct sets of information, the property description and the rental contract details. Once a contract ends and the property becomes vacant, landlords can easily republish the same property without redefining it. They only need to input their property id, and the property fields are automatically populated. However, landlords must still specify the new rental contract details. Additionally, once the advertisement is published, users can check the property's rental status on the contracts page.

Contracts

The contracts page is undoubtedly the most complex and important page on our platform. This is where landlords and tenants view their rental contracts. Once a landlord publishes an advertisement, the respective contract is listed on the left-hand side of the page. In this area, a landlord can manage multiple contracts, each representing a different property linked to a different advertisement. **Figure 3.17** exemplifies a scenario where a landlord has three contracts, linked to three advertisements, each one published for a different property.

By clicking on the "+" button, a landlord loads a set of submitted proposals from different tenants. Once loaded, a landlord may click on the "magnifying glass" button to open the full rental contract on the right-hand side of the page, as shown in **Figure 3.18**.

Figure 3.16: Application advertise page



There, the landlord has the option to either reject or accept the tenant's proposal. In case of rejection, the tenant receives a notification on the same page. If the landlord chooses to accept, he must initiate a payment request. To that, he must first connect his wallet by clicking on the "Connect Wallet" button as shown in **Figure 3.19**. Then he must select a desired wallet type and provider as shown in **Figure 3.20**. And finally, he must click on the "Request Payment" button, to request the first rental payment to the tenant, as shown in **Figure 3.21**.

Figure 3.17: Application contracts page - landlord contracts

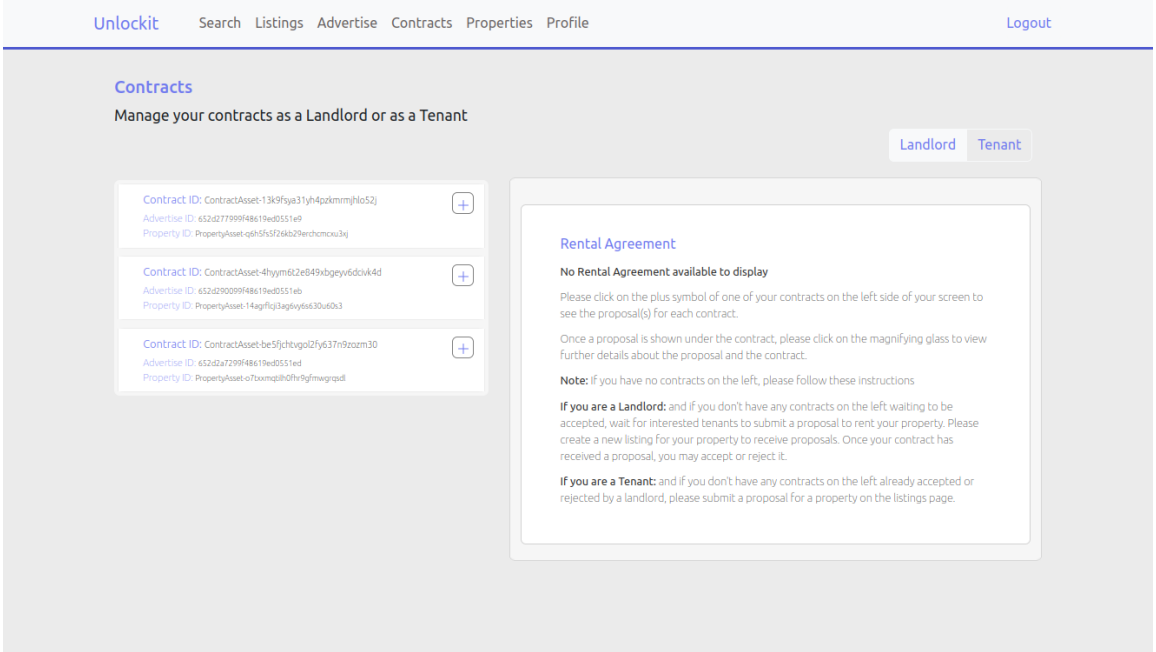


Figure 3.18: Application contracts page - landlord contract proposals

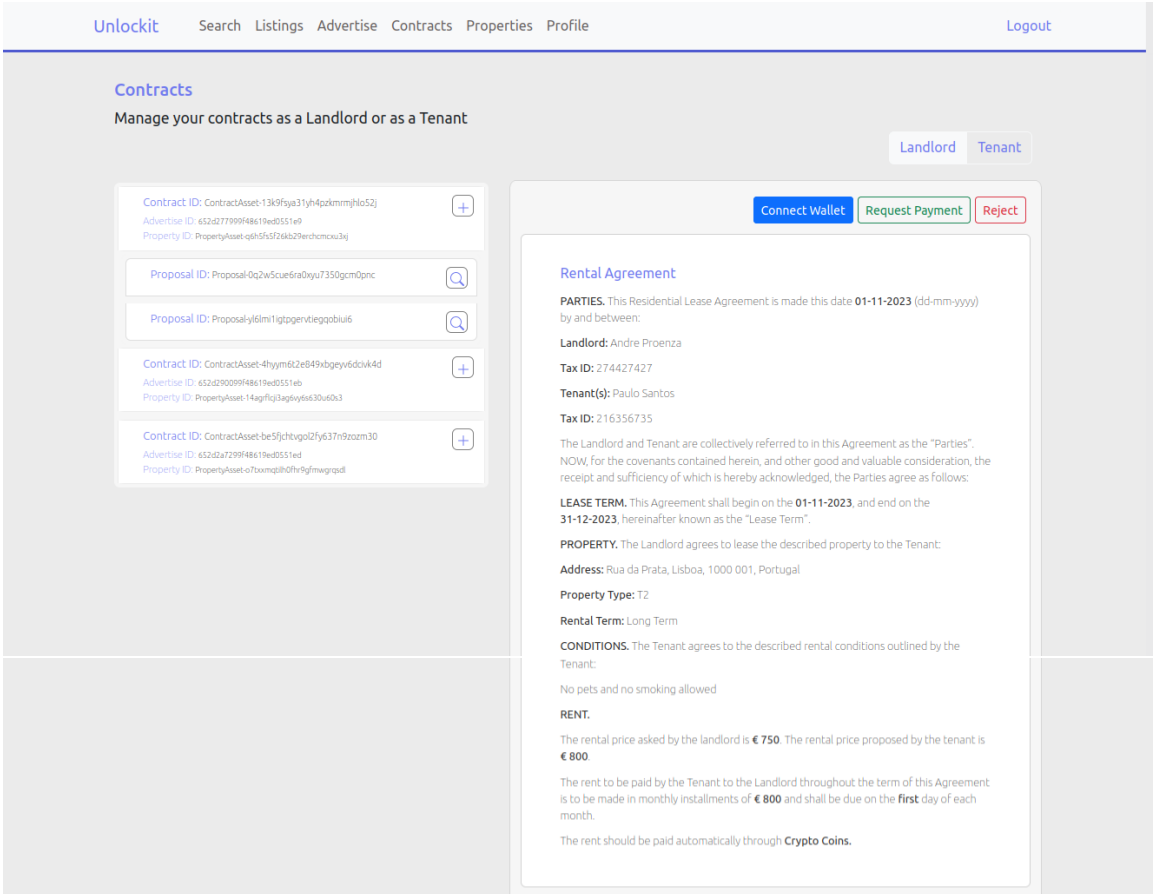


Figure 3.19: Application contracts page - connect wallet

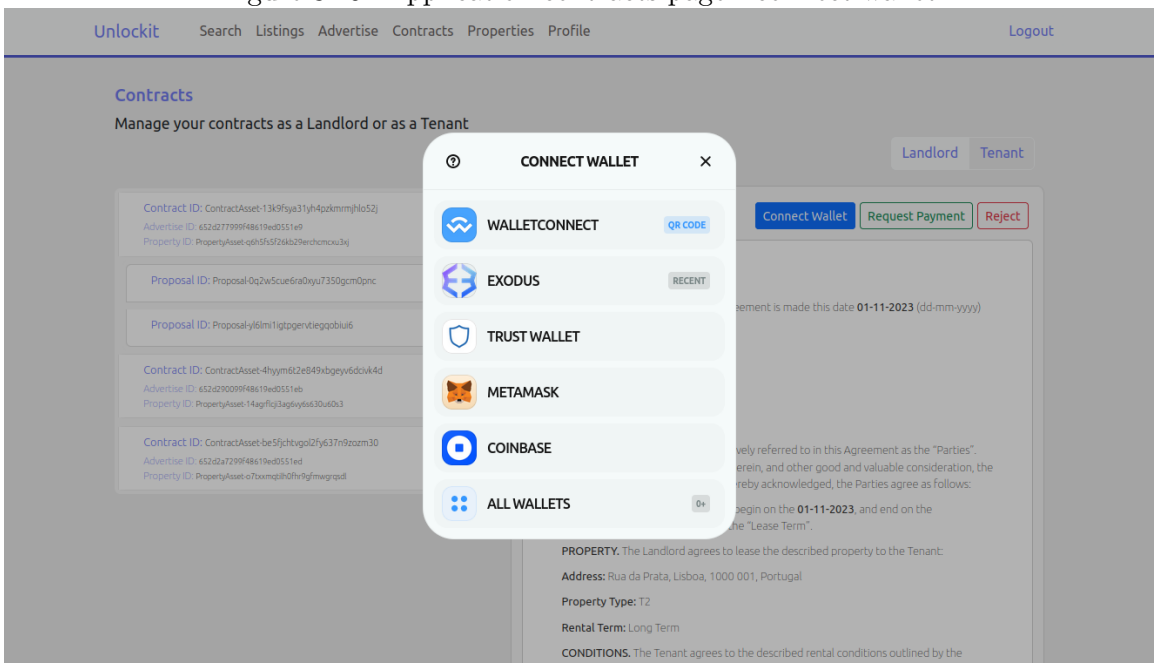
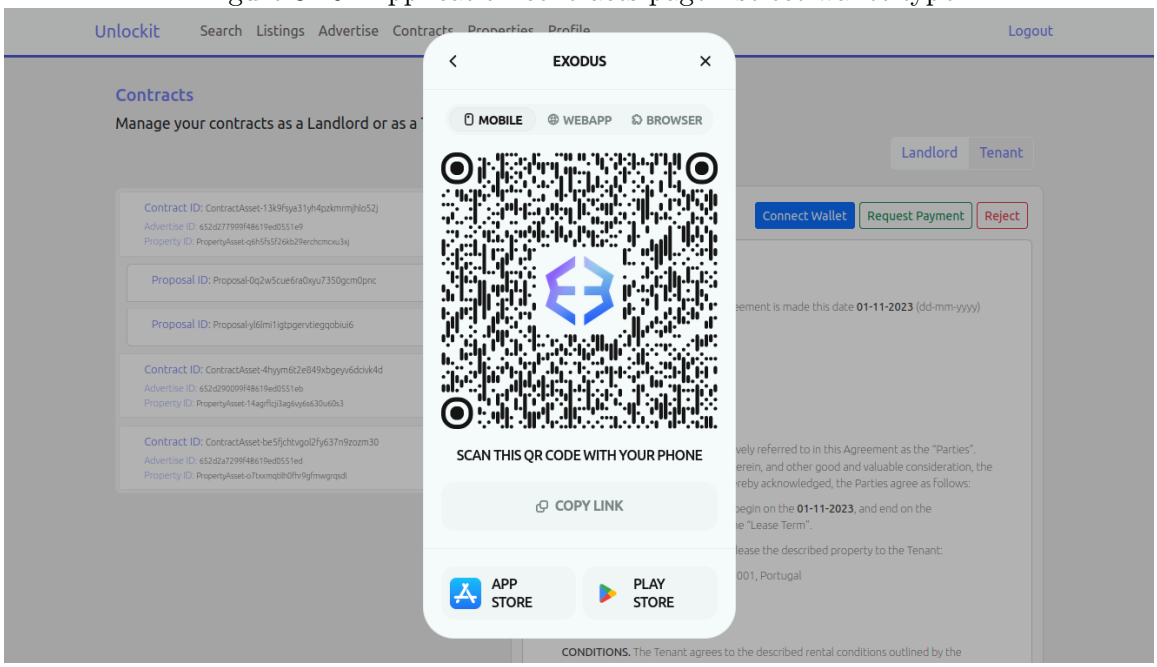


Figure 3.20: Application contracts page - select wallet type



A new modal window will open to handle the payment request. This window displays the cryptocurrency network address and rental amount fields automatically filled in. The landlord only needs to fill in the payment time limit. Once all the fields are populated, the payment request is sent to the tenant via email. The tenant must make the payment within the time limit to avoid losing the contract. Until the payment is submitted, the application displays to the tenant the interface depicted in **Figure 3.22**.

Figure 3.21: Application contracts - payment request

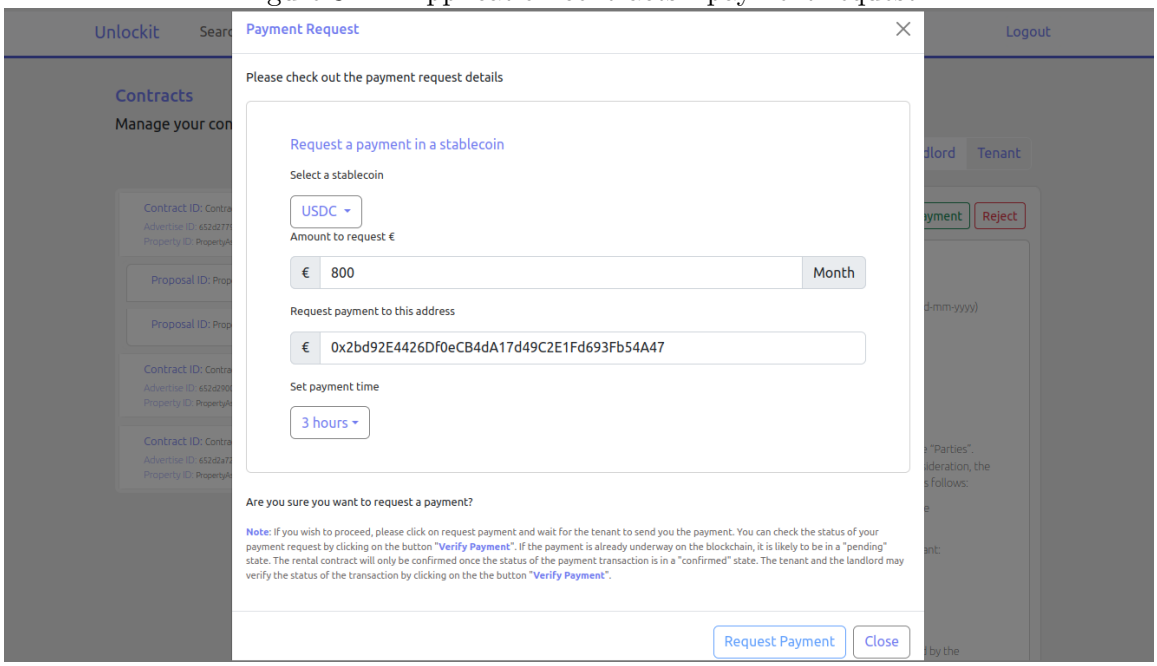
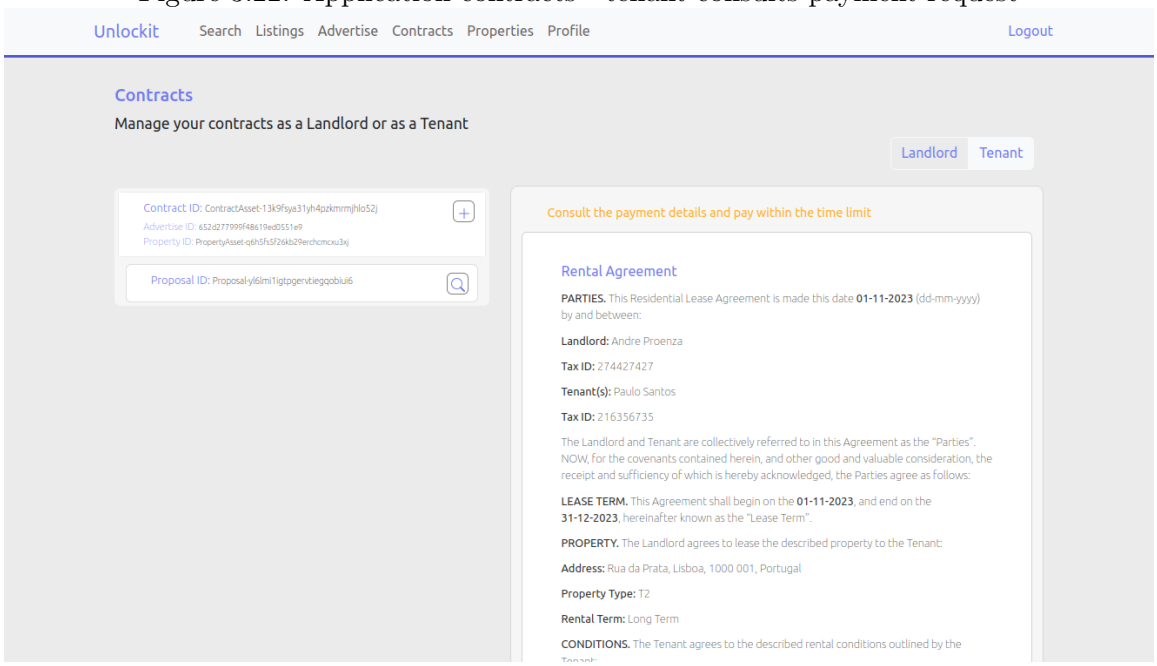


Figure 3.22: Application contracts - tenant consults payment request



After sending the payment, it takes a while for the transaction to be confirmed. This depends on the speed of the network. Both the landlord and tenant must confirm the transaction by clicking on the "Verify Payment" button, as shown in **Figure 3.23**. As soon as one of them confirms the transaction, the rental is confirmed, and the interface shows to both parties the payment and contract confirmation, as illustrates **Figure 3.24**.

Figure 3.23: Application contracts - payment verification

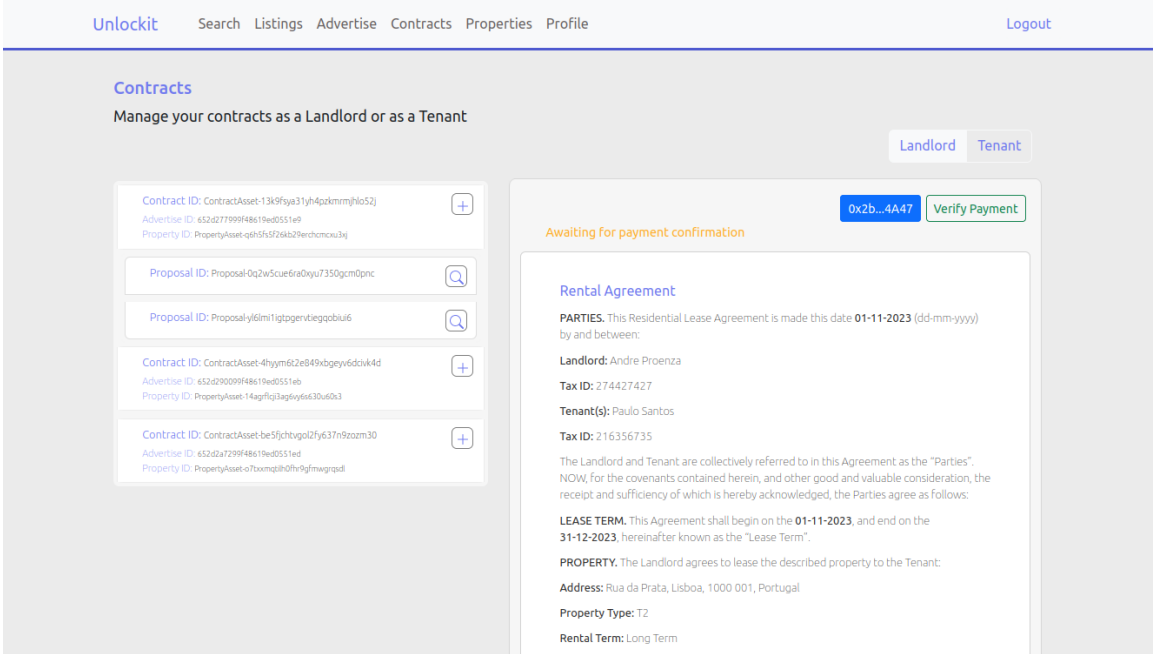
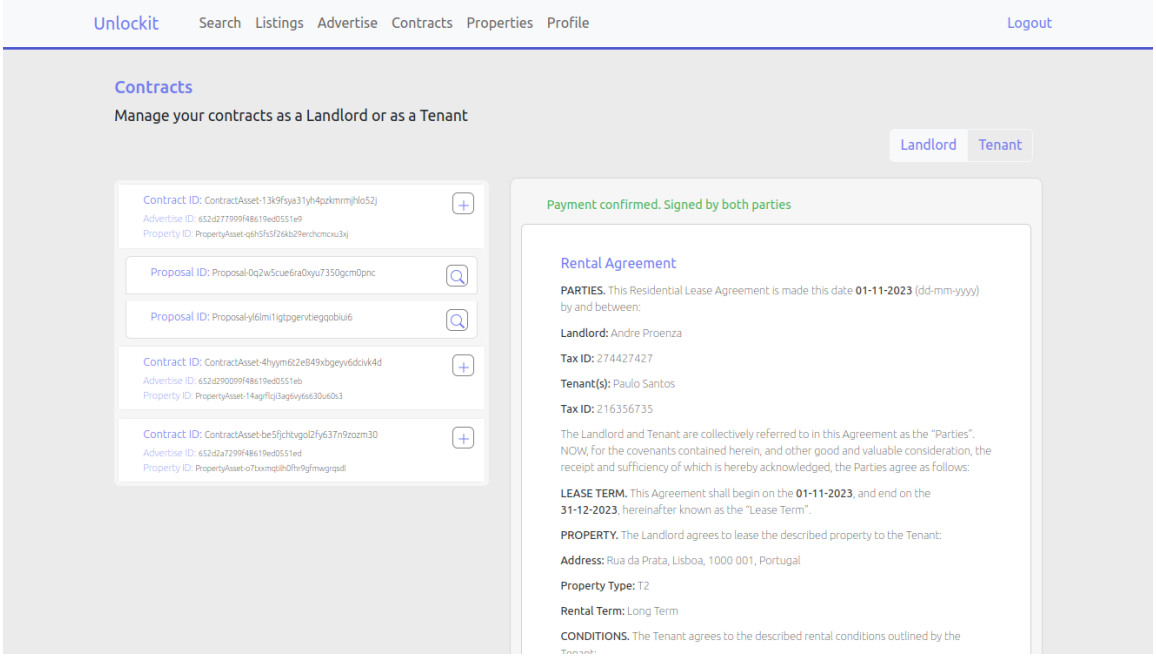


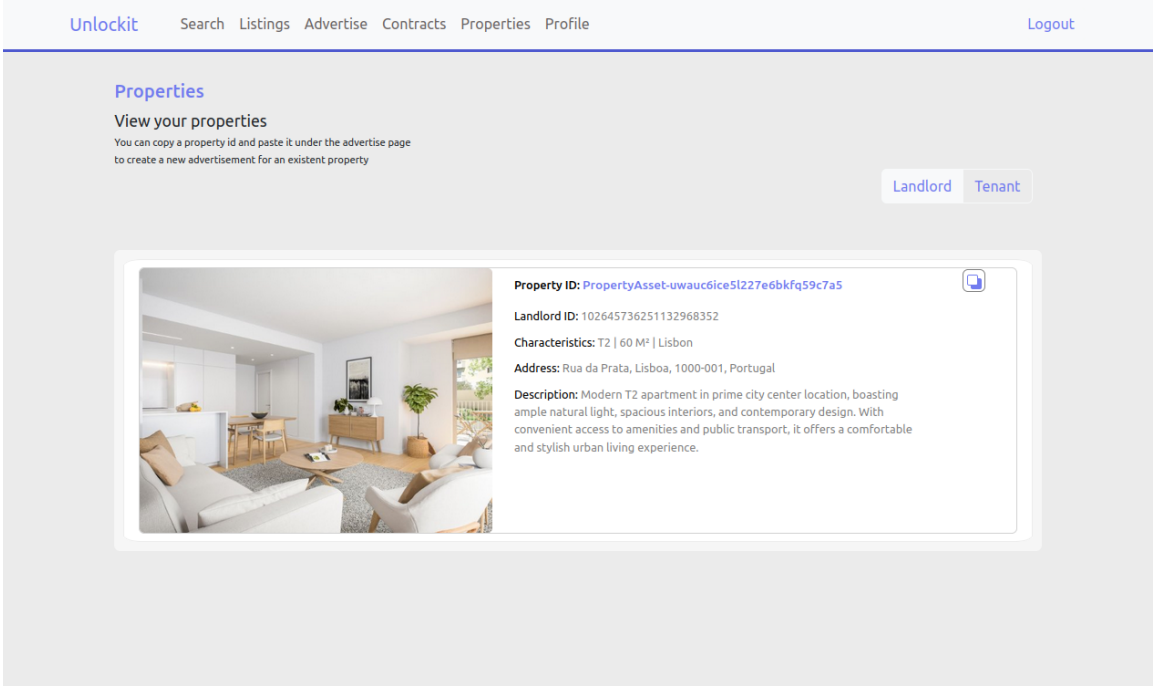
Figure 3.24: Application contracts - payment confirmation



Properties

The properties page shown in **Figure 3.25** allows landlords to search for their rented or vacant properties and enables tenants to search the properties they currently rent. Landlords also have the option to copy a property id and paste it on the advertise page to create a new advertisement for a vacant property.

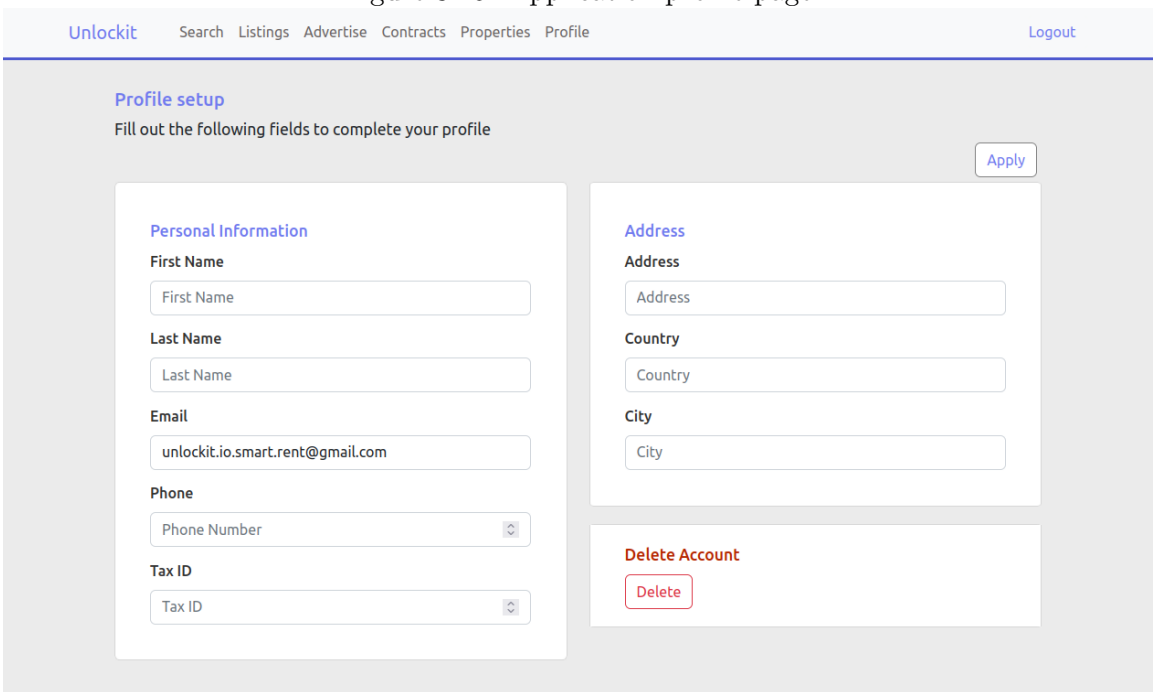
Figure 3.25: Application properties page



Profile

The profile page shown in **Figure 3.26** enables users to insert their required personal information for renting properties or creating advertisements. Furthermore, the page offers an account deletion mechanism, allowing users to permanently delete their data from the platform.

Figure 3.26: Application profile page



3.6 Summary

During this chapter, we have described and analyzed the proposed platform on several levels. We will now summarize how the platform successfully addresses the current problems mentioned in the beginning of this chapter.

The platform significantly reduces the high costs associated with the intermediation of traditional entities by implementing a blockchain-based platform, allowing real estate transactions to be decentralized in a secure manner. The architecture previously presented, includes the support of a blockchain network, an easy-to-use application that provides digitally signed rental contracts, allowing the platform to replace traditional bureaucratic procedures for more efficient, transparent and secure mechanisms, eliminating the need for multiple non-secure communication channels. Furthermore, the use of a database combined with a blockchain network, enhances information security, increases transparency, traceability and minimizes the risks of fraud and tax evasion. And finally, the adoption of automatic payment mechanisms to collect monthly rents, revolutionizes traditional inconsistent payment procedures, offering an automated and hassle-free alternative.

In essence, the platform establishes a new paradigm in the real estate industry, providing a viable ecosystem that, with further refinement, has the potential to become widely applied in the European real estate rental industry.

Chapter 4

Platform Evaluation

In this chapter, we will evaluate the platform in terms of to its adherence to European standards and performance benchmarks. Our goal, is to provide information on the applicability and robustness of the platform in the real world. The evaluation begins by analysing the platform's compliance with the European Union's General Data Protection Regulation (EU GDPR), and ends with a detailed examination of its performance, focusing on the responsiveness of the application, smart contract and database.

4.1 GDPR Compliance

As mentioned in **Chapter 2**, **Section 2.5**, the GDPR is an EU privacy and data security law that sets standards for collecting and processing personal data of EU citizens or residents that contract services or goods.

Our platform is designed to provide real estate services to European citizens. Consequently, it manipulates customer sensitive data. Therefore, the platform must comply with the GDPR, fulfilling the seven principles of protection and accountability and the eight user privacy rights. We will go through each of them and understand how the platform complies with them. However, before we begin, we need to recap how a user is identified on the platform.

We know for a fact, that, the blockchain technology, regardless of its implementation, keeps immutable, irreversible and transparent records of data. Therefore, any personal user information introduced into the blockchain cannot be removed, and can be seen by any entity interacting with it. Both of these aspects severely violate the European data protection and privacy rights. To overcome this situation, our platform integrates a database managed by Unlockit, which internally handles users' personal data. Any, and all administration of users' personal data is

Unlockit's responsibility. The application uses an external authorization service, to authenticate and authorize users to access the platform. From this process results an access token from where we extract a unique id which identifies the user. This identifier is extremely important as it bridges the connection between the user's personal data on the database, and his records on the blockchain, such as properties, contracts, proposals, payments and so on. A user is only and exclusively identified on the blockchain by a unique id. Having this clear, lets now understand how the platform adheres to the seven data protection principles.

1. *Legality, Fairness, and Transparency*

   Data processing must be legal, fair, and transparent to the individuals concerned.

   Most of the processed data, such as contracts, properties, proposals and payments are stored on the blockchain. The blockchain ensures the principle of transparency and fairness, as data processing can be viewed by an authorized individual. With regard to legality, personal data is collected only if the user wishes to continue on the platform to begin renting or listing properties.

2. *Purpose Limitation*

   Data should only be processed for legitimate, explicit purposes communicated to the data subjects during collection. The personal data collected by the platform is only acquired to identify a user and its collection is explicitly stated on the platform.

3. *Data Minimization*

   Collect and process only the necessary data essential for the specified purposes.

   The personal data is collected for user identification purposes only. Data relating to property, rental contracts and payments details are collected and processed solely for the purpose of establishing rental contracts between parties.

4. *Accuracy*

   Personal data must be kept accurate and kept up-to-date. The platform keeps up-to-date personal data in the database, and the latest blockchain records in the world state.

5. *Storage Restriction*

   Store personally identifiable data only for the necessary duration required for the stated purpose. The user's personal data is kept until the end date of the last active rental contract or the last pending payment. If the user has no active rental contracts, or pending payments, his personal data is kept until he quits the platform.

6. *Integrity and Confidentiality*

   Processing must guarantee appropriate integrity, and confidentiality, such as through encryption methods. The platform keeps personal data encrypted and secure in the database. Property, rental contracts, proposals and payment records are secured in the blockchain. The integrity of blockchain transactions is ensured by constantly validating users' encrypted IDs. The database tables are encrypted by an encryption key stored within the database, which is protected by a master key stored in a keyfile on the MongoDB server.

7. *Accountability*

   The data controller is accountable for demonstrating compliance with these principles as per GDPR regulations. Blockchain records are transparent and therefore available for auditing at any time. Users' personal data, on the other hand, can be requested for auditing from Unlockit whenever necessary.

So far, we have assessed and confirmed the platform's compliance with the previous data protection principles. The next step involves evaluating compliance with the eight privacy rights. The GDPR acknowledges eight privacy rights for data subjects, empowering individuals with increased control over the data they share with Unlockit.

1. *The right to be informed*

   Individuals have the right to know how their data will be used. The platform collects personal data with users' consent. Users will only be able to access its services, such as establishing rental contracts or publishing advertisements, once they authorize the collection of their personal data.

2. *The right of access*

   Individuals can request access to their personal data that an organization holds. A user may access, verify and update his personal data at any time on his profile page.

3. *The right to rectification*

   Individuals can request corrections to inaccurate or incomplete data. A user can rectify and update his personal data on his profile page at any time.

4. *The right to erasure (The right to be forgotten)*

   Individuals can request the deletion of their personal data. As stated at the beginning of this section, users are identified by an id. This id serves as an unique identifier linking the user's personal data in the database with the associated blockchain records. This right allows a user to delete all his data from the platform whenever he wishes. However, there

are some restrictions imposed by the platform. A user can only delete his information if no rental contracts are active and no payments are pending. Should this not be the case, the user must wait for the last active contract to end or complete any missing payments. If these conditions are met and the user wishes to delete his data, he may do so on his profile page using the delete account mechanism. By deleting his account, his personal data in the database is erased, and the blockchain records associated with the user id, such as property, rental contracts, payments, and proposals, are also deleted from the world state. However, they are always available for auditing in the transaction log.

5. *The right to restrict processing*

   Individuals can limit the way an organization uses their data. By entering their data, users grant the platform permission to process it for identification purposes only. Should the platform use the data for another purpose, users would eventually, be able to configure their usage. At the moment, this right is not applicable.

6. *The right to data portability*

   Individuals can obtain and reuse their data for their own purposes. Users may access their personal data from the profile page anytime, allowing them to use profile data for purposes beyond the platform.

7. *The right to object*

   Individuals can object to the processing of their data in certain circumstances. This right does not apply to our current platform, since users authorize the handling of their data solely for the purpose of user identification. Therefore, there are no other circumstances to object to.

8. *Rights in relation to automated decision making and profiling*

   Individuals have safeguards against the risk of a potentially damaging decision being made without human intervention. If our platform ever uses automated algorithms to evaluate, for instance, rental contracts based on specific criteria, users have the right to contest and seek human intervention if they believe the decision was unjust or discriminatory, providing a safeguard against biased or unfair automated decisions. Since our application does not have automated decision-making and profiling mechanisms, this right does not apply.

In summary, the preceding assessment ensures that our platform correctly processes users' personal data, guaranteeing full compliance with the EU GDPR's data protection principles and user privacy rights.

4.2 Performance

In this section we will evaluate the platform's API through performance tests using key metrics such as response time, throughput and error rate. To conduct these measurements, we used Postman performance testing tool, to simulate real-world traffic.

The tests were performed on a local execution environment running the platform, consisting of a virtual machine running Linux Mint Mate 21.2, 64-bit operating system with 3 virtual processors, 6.4 GB of RAM memory and 150 GB of storage. The virtual machine is hosted on a Linux Mint Cinnamon 20, 64-bit operating system, equipped with a dual-core Intel processor operating at a base clock speed of 1.60 gigahertz, with 7.5 GB of RAM memory and 480 GB of storage.

The platform has a collection of twenty-five public functions, available for testing, distributed across three APIs: the smart contract, the database server and the blockchain interface server. However, although indispensable, presenting independent tests for all of them, would overload this document. Therefore, we decided to test the platform's two most significant processes, which involve accessing various public and private functions. They are, the process of publishing a property and the process of submitting a proposal.

As explained in **Section 3.1**, the process of creating an advertising involves authorizing the user in the application, validating his identity on the blockchain, creating the property and rental contract records on the blockchain and updating all the database tables related to the landlord that publishes the advertisement. As for submitting a proposal, it also involves authorizing the user in the application, validating his identity on the blockchain, creating a proposal record on the blockchain and updating the advertise and user database tables related to the tenant that submits the proposal.

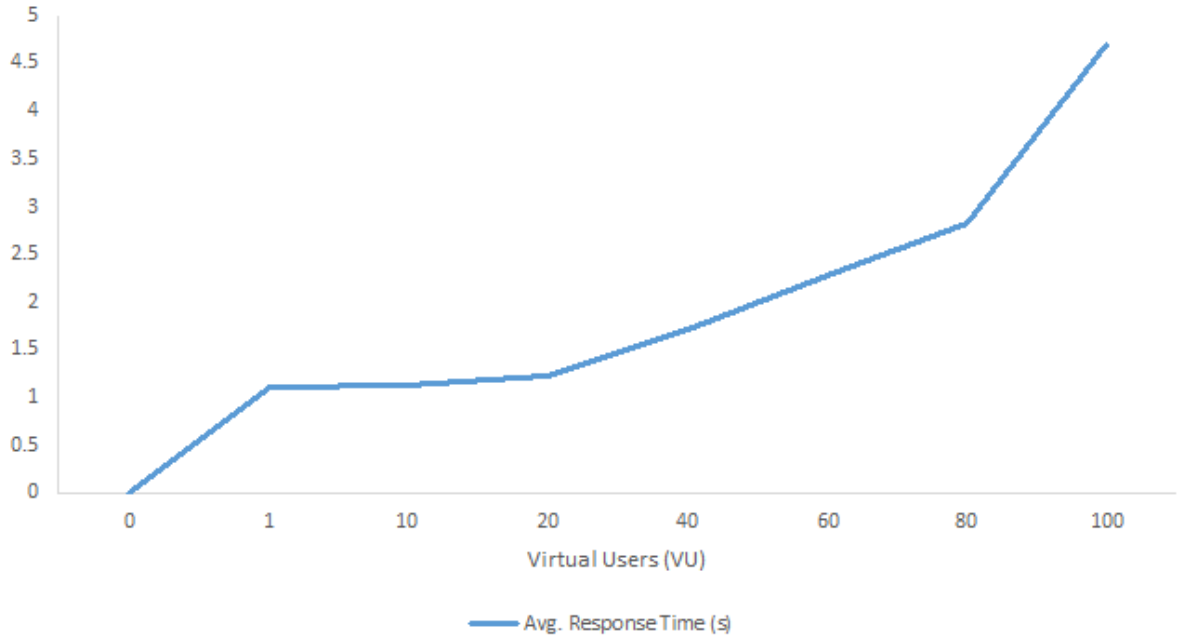
We will measure each one of these processes against the time it takes for a request to travel from the application to the back-end components, be processed, and come back again (response time), and the number of requests processed by the platform per unit of time (throughput). The tests are executed with a continuously increasing number of virtual users, ranging from 1, 10, 20, 40, 60, 80 up to a maximum of 100 virtual users. For each of these sets of virtual users, which send parallel requests to the platform, we measured 10 times, each lasting a 1 minute, and calculated the average response time, throughput and error rate metrics. Based on these tests, we created six graphs, two for each analyzed metric.

Each graph plots one variable against the number of virtual users, where the x-axis indicates the number of virtual users (VU) and the y-axis shows performance metrics, such as, response time (blue line), throughput (golden line) and the error rate (red line). The first three graphs, **Figure 4.1**, **4.2** and **4.3**, measure the average response time, throughput and error rate, that it takes to publish an advertisement. This involves evaluating some of the platform functions in the following order: "CreateContractAsset" (smart contract), "CreatePropertyAsset" (smart contract), "registerPropertyPhoto" (database), "getUserById" (database), and "registerAdvertise" (database). **Table 4.1** summarizes, for each metric, the average results obtained from the performance tests conducted for the process of publishing an advertisement.

Table 4.1: Publish advertisement process - performance summary

Avg. Requests Sent	**Avg. Throughput**	**Avg. Response Time**	**Avg. Error Rate**
664	7.13 req/s	2.14 ms	0.24 %

Figure 4.1: Publish advertisement process - performance response time



The blue line graph, **Figure 4.1**, describes the average response time, that peaks, when VU count is at its peak, measuring approximately 4.6 s. The gold line graph **Figure 4.2**, describes the average throughput, that increases as the number of VUs rises, reaching the peak throughput of 10 req/s at around 80 VUs. After this point, the average throughput starts to decrease as the platform becomes overloaded and is no longer able to handle the load effectively. The red line

Figure 4.2: Publish advertisement process - performance throughput

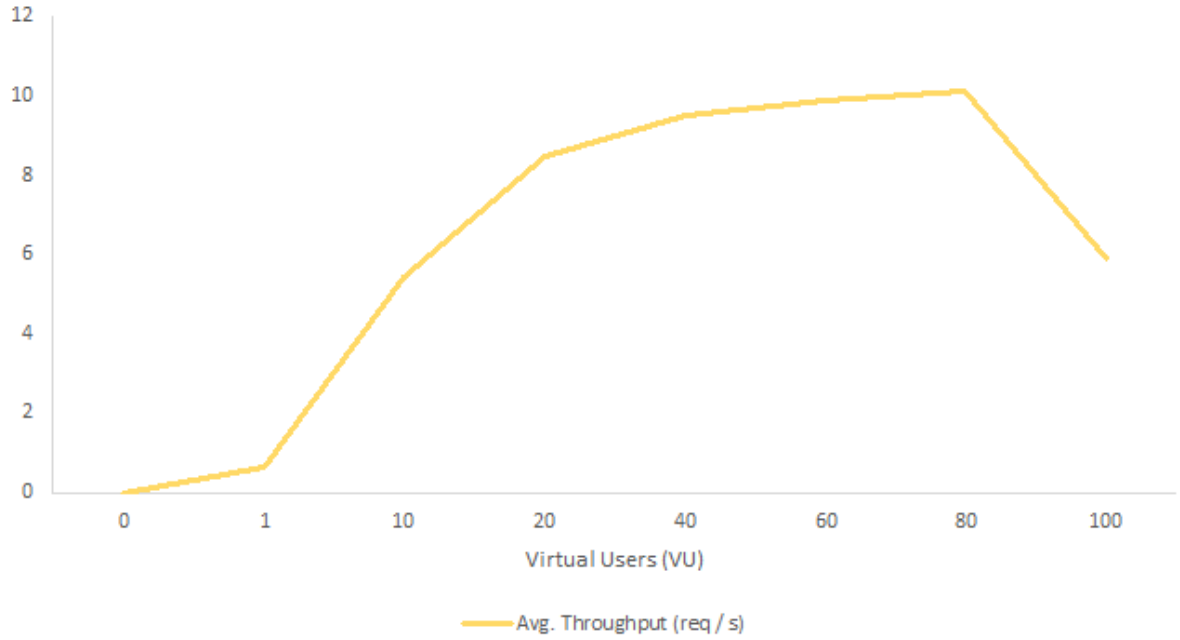
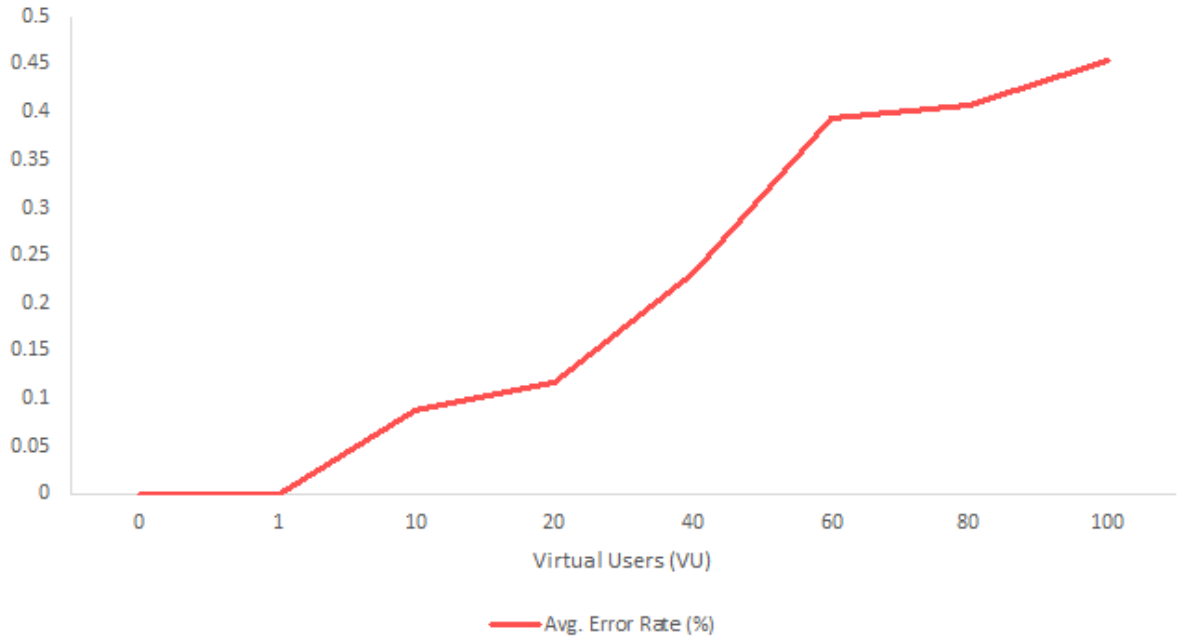


Figure 4.3: Publish advertisement process - performance error rate



graph, **Figure 4.3**, describes the average error rate, that remains relatively low throughout the tests, but it does start to increase slightly as the number of virtual users increases, reaching a maximum error rate of approximately 0.45% when VUs peak at 100.

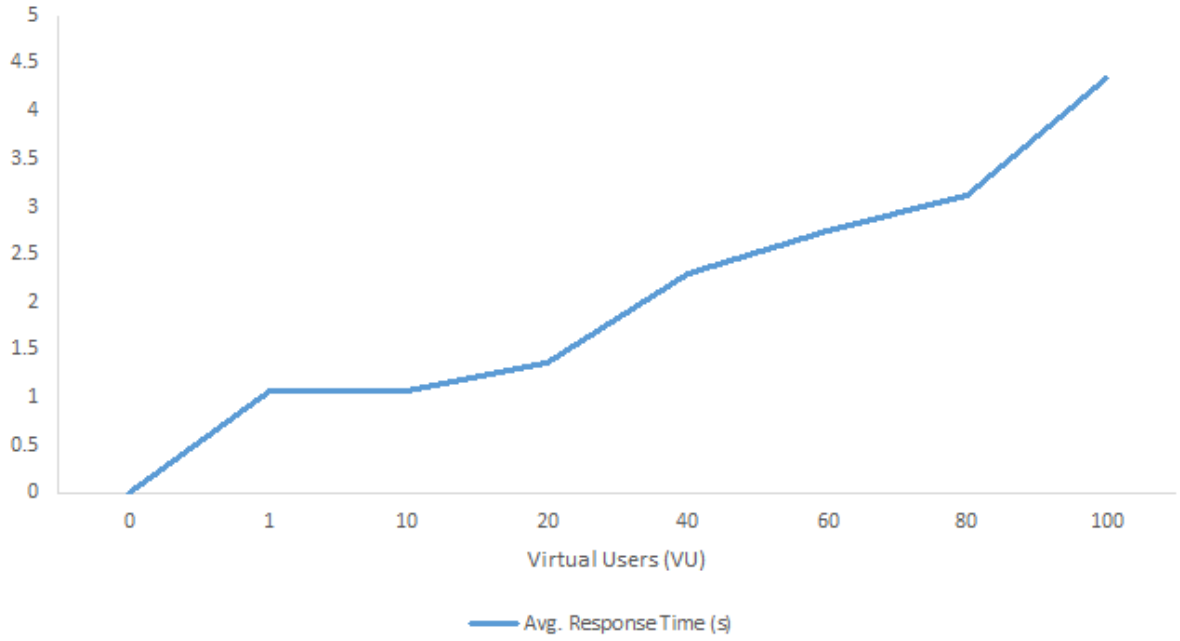
The last three graphs, **Figure 4.4**, **4.5** and **4.6** measure the average response time, throughput and error rate that it takes to submit a proposal. This involves evaluating some of the

platform functions in the following order: "CreateProposal" (smart contract), "updateUserById" (database), and "updateAdvertiseById" (database). **Table 4.2** summarizes, for each metric, the average results obtained from the performance tests conducted for the process of submitting a proposal.

Table 4.2: Submit proposal process - performance summary

Avg. Requests Sent	**Avg. Throughput**	**Avg. Response Time**	**Avg. Error Rate**
569	5.90 req/s	2.29 ms	0.33 %

Figure 4.4: Submit proposal process - performance response time



The blue line graph, **Figure 4.4**, shows the average response time increasing as the number of VUs increases. The highest average response time observed is approximately 4.3 s. This suggests that the platform is becoming saturated as more and more users are trying to access it, indicating a potential correlation between load and response time which can be a concern for scalability. The golden line graph, **Figure 4.5**, describes the average throughput increasing linearly as the number of VUs increases until it reaches a peak at around 60 VUs, registering approximately 7.7 req/s. This indicates that the platform is capable of handling up to 60 VUs efficiently. However, as the number of VUs increases beyond 60, the average throughput starts to decrease, suggesting that the platform reached its maximum capacity. The red line graph,

Figure 4.5: Submit proposal process - performance throughput

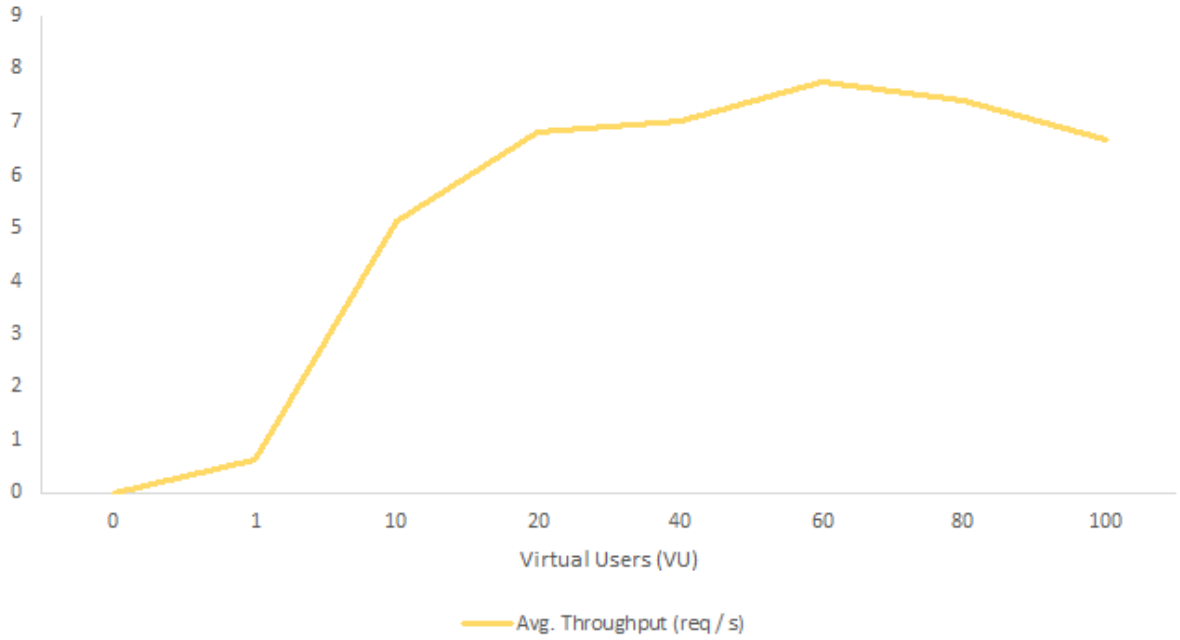


Figure 4.6: Submit proposal process - performance error rate

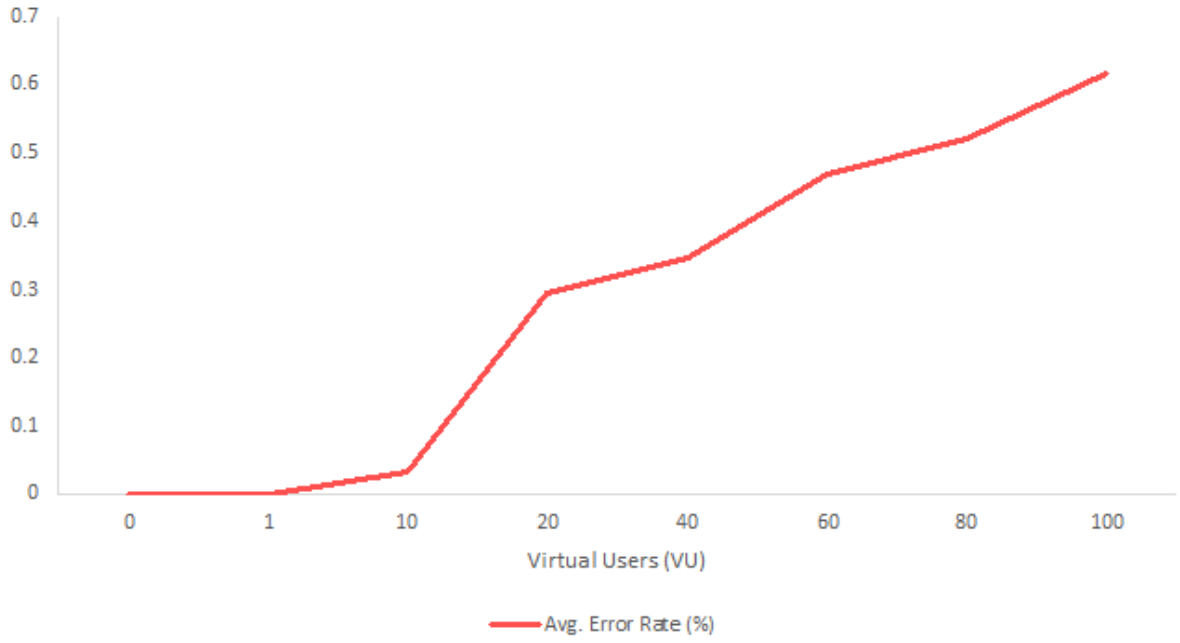


Figure 4.6, describes the average error rate, that remains relatively low but increases gradually as the number of virtual users increases, reaching a maximum error rate of approximately 0.61% when VUs peak at 100.

The tests show a significant error rate, which indicates that the platform might need optimization or resources enhancement to reduce this error rate. According to our measurements, only the functions that submit transactions to the smart contract experience failures.

The most common and unique error is the, "ECONNRESET", which stands for "Connection Reset by Peer". It occurs when a TCP connection between two machines is unexpectedly terminated by the peer, which can happen when there is network congestion due to high resource utilization, causing the connection to reset. Considering this evidence, in a future iteration we would recommend increasing the number of peers in organization 1 and the number of ordering peers in organization 3, to handle a higher volume of requests.

In this evaluation we demonstrated the platform's compliance with the European data protection regulation (GDPR) and tested the platform's performance when exposed to a high number of parallel requests. We concluded that refining the Hyperledger Fabric blockchain network could enhance the performance of the analyzed processes, making the platform more suitable for real-world deployment.

Chapter 5

Conclusions

Real estate industry currently faces complex and inefficient rental property processes, therefore, formulating innovative solutions is imperative. This dissertation presented a platform design conceived for Unlockit, a DLT company operating in the real estate industry. We proposed a blockchain-based residential smart rental platform, designed to solve real estate rental existing problems through mechanisms that guarantee the integrity, security and transparency of real estate transactions. Our platform innovates and streamlines traditional property rental processes by excluding intermediaries and centralized entities through the integration of the latest smart contract technology supported by the blockchain technology. The platform leverages Hyperledger Fabric's permissioned blockchain to provide secure and transparent residential rental services. It offers seamless listing and consultation of residential properties, as well as the ability to create, manage and sign digital rental contracts between landlords and tenants, with digital signatures. Furthermore, it integrates convenient and automated services for paying monthly or one-time rents using stablecoins, ensuring a safe and hassle-free payment experience. This dissertation is organized to begin by introducing the required key concepts to easily understand the platform. Its is followed by a presentation describing the platform's pre-architecture requirements and a detailed explanation covering the architecture components. Then it defines the platform's behaviour describing its complex data storage architecture and its rental and payment procedures. It concludes with a platform evaluation from two perspectives, the platform compliance with the GDPR, assessing its adherence to data protection laws and privacy rights, and, the platform performance, assessing aspects such as the platform throughput and response time of transactions going from the application to the Hyperledger Fabric blockchain. We hope that our research efforts, as well as our platform design, will encourage further research on the topic as a way to continue innovating and pushing the real estate industry towards a more digital, secure, and transparent future.

5.1 Achievements

The purpose of this dissertation was to conduct a research project to interconnect the concepts of real estate and blockchain with the intention of providing a platform capable of innovating the real estate industry by reformulating its traditional residential property rental processes. This work enabled the development of a platform design to provide effective solutions to the currently identified problems. The platform, due to decentralization, significantly reduces intermediation costs, simplifying processes that were previously time-consuming, bureaucratic and dependent on centralized real estate entities. Information is now shared securely via a unified platform, eliminating the vulnerabilities associated with multiple unsecured channels. The platform increases transparency and traceability, mitigating risks related to fraud and tax evasion. It meticulously follows the European data protection directives defined in the GDPR, and introduces automatic monthly rent payment mechanisms, solving the previous obsolete traditional form of payment. Ultimately, we believe that, we have managed to provide a new real estate ecosystem that, if refined, could be applied to the European real estate rental industry.

5.2 Future Work

To conclude the dissertation, we will discuss potential enhancements that could be implemented in future iterations on top of the developed platform.

5.2.1 Integrate Know Your Costumer Policy

In the architecture implementation illustrated in **Figure 3.9**, the OAuth 2.0 protocol provided by Google authenticates and authorizes users to access the platform. However, Google provider does not fully verify its customers, meaning that, any user with a Google account may access the platform. In a future iteration, it would be useful to update this mechanism to a more restrictive one that adopts the know your customer policy, allowing users to be fully identified through their identification document, photos or videos to prevent rental frauds. From an implementation point of view, this mechanism could be implemented using the identification system of each EU country. In Portugal's case, it could be implemented using the digital mobile key software provided by its government. Ideally, the platform should allow any citizen residing in an EU country to authenticate using the identification software provided by his country.

5.2.2 Verify Property

The platform enables the creation of multiple advertisements for different properties. Nevertheless, it does not implement any mechanism to verify their existence. It is therefore of high priority to iterate on this aspect. An ideal platform would involve verifying both the ownership and spatial existence of a property. Ownership could be verified through a title deed or a tax record presented by a landlord, whereas spatial existence could be confirmed by address verification using a specialized software.

5.2.3 Property Tokenization

The platform could offer a mechanism to allow users to tokenize their properties into multiple divisions generating multiple NFTs. Users would introduce their property details into the platform, and the platform would partition the property into multiple rentable fractions.

5.2.4 Refine Hyperledger Fabric Network

The platform supports a Hyperledger Fabric blockchain network with a single channel and two organisations. In a future iteration, as requests and different records increase, more channels could be implemented, to process only specific records, and improve the platform's scalability. Existing organizations could join more peers to further increase scalability, and new organisations could be incorporated to improve the network's security.

5.2.5 Integrate Real Estate Agencies

The platform is designed to facilitate residential rental transactions directly between two participants, a tenant and a landlord. However, it could be possible to integrate real estate agencies. The agencies would act as landlords on the platform, holding several advertisements representing different properties. Despite the fact that these advertisements would be managed by the agency, they would actually be owned by the agency's clients, who would hire the agency's intermediary services to interact with the platform on their behalf just like a traditional agency.

5.2.6 Support Other Payment Methods

The current implementation provides a single payment method using two stablecoins, USDT and USDC, which are recorded on the Solana blockchain for its speed and low transaction fees. Future enhancements might include integrating traditional payment methods alongside other stable or volatile cryptocurrencies on different permissionless blockchain networks.

Bibliography

[AvM17] Maher Alharby and Aad van Moorsel. Blockchain-based smart contracts: A systematic mapping study. *arXiv e-prints*, pages arXiv–1710, 2017.

[B+14] Vitalik Buterin et al. A next-generation smart contract and decentralized application platform. *white paper*, 2014.

[BCGH16] Richard Gendal Brown, James Carlyle, Ian Grigg, and Mike Hearn. Corda: an introduction. *R3 CEV, August*, 1(15):14, 2016.

[Bee18] Beetoken Team. Beetoken Whitepaper, 2018. https://www.allcryptowhitepapers.com/beetoken-whitepaper/.

[But16] Vitalik Buterin. Ethereum: platform review. *Opportunities and Challenges for Private and Consortium Blockchains*, 2016.

[Cal20] Giulio Caldarelli. Understanding the blockchain oracle problem: A call for action. *Information*, 11(11):509, 2020.

[CMS19] CMS. The tension between GDPR and the rise of blockchain technologies, 2019. https://cms.law/en/media/international/files/publications/publications/the-tension-between-gdpr-and-the-rise-of-blockchain-technologies.

[Con22] Inc ContractsCounsel. Rental Lease Agreement - Types of Rental Lease Agreements, 2022. https://www.contractscounsel.com/t/us/rental-lease-agreement.

[D+21] Navin Duwadi et al. A systematic review on blockchain in education: Opportunities and challenges. *Nepalese Journal of Management Science and Research*, 4, 2021.

[Den] Dentons Team. The tokenization of real estate: An introduction to fractional real estate investment. https://www.dentons.com/en/insights/articles/2022/september/6/the-tokenization-of-real-estate.

[DMH17] Vikram Dhillon, David Metcalf, and Max Hooper. The HyperledgerProject. In *Blockchain enabled applications*, pages 139–149. Springer, 2017.

[Doc20] Hyperledger Fabric Docs. Hyperledger Fabric Channels, 2020. https://hyperledger-fabric.readthedocs.io/en/latest/channels.html.

[Eur16] European Parliament and European Council. Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 april 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation), April 2016.

[Eur21] Eurostat. House or Flat – Owning or Renting, 2021. https://ec.europa.eu/eurostat/cache/digpub/housing/bloc-1a.html.

[FCE] FCE Media. Real estate tokenization: why and how it works. https://fcegroup.ch/en/news/text/id394-2022-06-05-real-estate-tokenization-why-and-how-it-works.

[GR18] Julija Golosova and Andrejs Romanovs. Overview of the blockchain technology cases. In *2018 59th International Scientific Conference on Information Technology and Management Science of Riga Technical University (ITMS)*, pages 1–6. IEEE, 2018.

[Gri17] Ian Grigg. EOS-an introduction. *White paper. https://whitepaperdatabase. com/eos-whitepaper*, 2017.

[GT20] Rosa M Garcia-Teruel. Legal challenges and opportunities of blockchain technology in the real estate sector. *Journal of Property, Planning and Environmental Law*, 12(2):129–145, 2020.

[Har12] Dick Hardt. The OAuth 2.0 Authorization Framework. RFC 6749, October 2012.

[HSB23] Sabine Houy, Philipp Schmid, and Alexandre Bartel. Security aspects of cryptocurrency wallets—a systematic literature review. *ACM Computing Surveys*, 56(1):1–31, 2023.

[HSMC15] Heidi Howard, Malte Schwarzkopf, Anil Madhavapeddy, and Jon Crowcroft. Raft refloated: Do we have consensus? *ACM SIGOPS Operating Systems Review*, 49(1):12–21, 2015.

[Hyp23] Hyperledger Fabric Foundation. Hyperledger Fabric Documentation - Release 2.5, 2023. https://hyperledger-fabric.readthedocs.io/en/release-2.5/.

[JB22] Kim Peiter Jørgensen and Roman Beck. Universal wallets. *Business & Information Systems Engineering*, pages 1–11, 2022.

[Mid17] Midasium Team. Midasium Whitepaper, 2017. https://midasium.herokuapp.com/smart-tenancy.

[MPJ18] Bhabendu Kumar Mohanta, Soumyashree S Panda, and Debasish Jena. An overview of smart contract and use cases in blockchain technology. In *2018 9th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*, pages 1–4. IEEE, 2018.

[Nak08] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. 2008.

[OM] Karl J O'Dwyer and David Malone. Bitcoin mining and its energy footprint. In *25th IET Irish Signals & Systems Conference 2014 and 2014 China-Ireland International Conference on Information and Communications Technologies (ISSC 2014/CIICT 2014)*.

[PRLT21] Julien Polge, Jérémy Robert, and Yves Le Traon. Permissioned blockchain frameworks in the industry: A comparison. *ICT Express*, 7(2):229–233, 2021.

[Ren18] RentPeacefully Team. What is RentPeacefully?, 2018. https://rentpeacefully.com/.

[S+94] Nick Szabo et al. Smart contracts. *white paper*, 1994.

[Sal21] Fahad Saleh. Blockchain without waste: Proof-of-stake. *The Review of Financial Studies*, 34(3):1156–1190, 2021.

[SB17] Clare Sullivan and Eric Burger. E-residency and blockchain. *Computer Law & Security Review*, 33(4):470–481, 2017.

[SMA18] SMARTRealty Team. SMARTRealty whitepaper, 2018. https://smartrealty.io/wp-content/uploads/2018/01/SMARTRealty-Whitepaper-v1.pdf.

[SVB+19] Julie Smith, Manasi Vora, Hugo Benedetti, Kenta Yoshida, and Zev Vogel. Tokenized securities and commercial real estate. *Available at SSRN 3438286*, 2019.

[Tas19] John Taskinsoy. Blockchain: a misunderstood digital revolution. things you need to know about blockchain. *in Things You Need to Know about Blockchain (October 8, 2019)*, 2019.

[Tea20] Digital Asset Canton Team. A Daml based ledger interoperability protocol. 2020. http://canton.io.

[Tea22] Rentible.io Team. Rentible whitepaper, 2022. https://whitepaper.rentible.io/.

[VBS20] Joseph Vincent, Steven Bender, and Peter Smirniotopoulos. Session 5: Real estate tokenization. 2020.

[WG18] Karl Wüst and Arthur Gervais. Do you need a blockchain? In *2018 Crypto Valley Conference on Blockchain Technology (CVCBT)*, pages 45–54. IEEE, 2018.

[Wol18] Ben Wolford. What is GDPR, the EU's new data protection law?, 2018. https://gdpr.eu/what-is-gdpr/.

[Wu19] Kaidong Wu. An empirical study of blockchain-based decentralized applications. *arXiv e-prints*, pages arXiv–1902, 2019.

[YMRS19] Dylan Yaga, Peter Mell, Nik Roby, and Karen Scarfone. Blockchain technology overview. *arXiv e-prints*, pages arXiv–1906, 2019.

[YTK22] Kai-Jie Yong, Eng Siang Tay, and Dennis WK Khong. Application of blockchain smart contracts in smart tenancies: A malaysian perspective. *Cogent Social Sciences*, 8(1):2111850, 2022.

[ZLK+19] Weiqin Zou, David Lo, Pavneet Singh Kochhar, Xuan-Bach Dinh Le, Xin Xia, Yang Feng, Zhenyu Chen, and Baowen Xu. Smart contract development: Challenges and opportunities. *IEEE Transactions on Software Engineering*, 47(10):2084–2106, 2019.

[ZS18] Kaspars Zīle and Renāte Strazdiņa. Blockchain use cases and their feasibility. *Applied Computer Systems*, 23(1):12–20, 2018.

[ZXD+17] Zibin Zheng, Shaoan Xie, Hongning Dai, Xiangping Chen, and Huaimin Wang. An overview of blockchain technology: Architecture, consensus, and future trends. In *2017 IEEE International Congress on Big Data (BigData Congress)*, pages 557–564. IEEE, 2017.

[ZXD+20] Zibin Zheng, Shaoan Xie, Hong-Ning Dai, Weili Chen, Xiangping Chen, Jian Weng, and Muhammad Imran. An overview on smart contracts: Challenges, advances and platforms. *Future Generation Computer Systems*, 105:475–491, 2020.