Blockchain Based Residential Smart Rent

André da Silva Proença
andre.proenca@tecnico.ulisboa.pt
andreproenza.github.io

Instituto Superior Técnico
Universidade de Lisboa, Portugal

Advisors
Prof. Miguel Pupo Correia
Tiago Dias - Unlockit

Cooperation

Abstract. The popularization of blockchain technology and its key properties have encouraged the development of new business concepts as well as the restructuring of traditional businesses such as real estate. The real estate market includes complex and inefficient mediation processes. Acquiring, selling or 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 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 introduction of blockchain technology intends to reduce intermediation high costs, property legalization, time spent on paperwork and address data security flaws, providing transparent anti-fraud mechanisms, and secure and reliable real estate transactions. The proposed platform will allow landlords to advertise properties and tenants to browse them as well as digitally sign and manage rental contracts.

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

Table of Contents

1 Introduction

Blockchain technology in recent years has been growing in popularity at a vertiginous pace, mainly due to the 2008 white paper release entitled *Bitcoin: A Peer to Peer Electronic Cash System* [1] published pseudonymously by *Satoshi Nakamoto*, proposing an innovative network which became the first ever blockchain application, enabling the broadening of horizons by unlocking a whole new spectrum of innovative possibilities reshaping some sectors of society.

The Blockchain technology key features include network decentralization, data persistence, data immutability, participant anonymization, and transaction traceability [2]. Together they compose a compact and secure system that is increasingly attracting new business ideas. Sectors such as, government, private management, healthcare, insurance, electronic voting, and real estate are all 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 and sale of real estate assets, such as high intermediation costs, time-consuming bureaucracy and administration, lack of standardized processes, high legalization property costs, and the inability to guarantee information security since information is shared through multiple insecure channels.

The traditional property leasing 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. Their self-verifying, self-executing, tamper-proof, traceable and irreversible properties [3] make them a reliable and robust solution for developing blockchain-based real estate projects.

The establishment of a lease 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 state regulation 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 lease 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 [4]. Therefore it is possible to tokenize a property into one or multiple tokens for leasing 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 leasing 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 [5]). As a consequence privacy issues must be taken very seriously when building a system.

We propose a blockchain-based 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 will permit landlords to advertise

properties and tenants to browse them before engaging into a rental agreement. The platform will provide a graphical user interface, UI, which will serve as an interface between users and the blockchain system. The UI will not only allow users to advertise and visualize properties, but also allow them to manage, write and digitally sign rental contracts.

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. In the end we aim for a solution assuring the following requirements:

1. ***Blockchain*** - The solution should adopt a permissioned blockchain to allow transparent, auditable and secure transactions through a decentralized system.
2. ***GDPR compliance*** - The solution must comply with the European general data protection regulation (GDPR).
3. ***Different rental types*** - The solution should support multiple rental contracts, taking into consideration the different types of rentals within a property and their durations.

1.2 Document Overview

The remainder of the document is structured as follows. **Section 2** introduces necessary prior concepts of real estate, blockchain and GDPR compliance, allowing for a clear comprehension and application of our solution. **Section 3** exposes the problem and proposes a solution as well as its architecture, design and infrastructure requirements. **Section 4** describes the solution evaluation methodology, which focuses on validating and verifying the solution. **Section 5** describes the expected plans for the future development of this work. Finally, **Section 6** summarizes the achievements of this work, concluding it.

2 Related Work

In this section we present the related knowledge required to comprehend and apply our solution. 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 lease types and durations. In **Section 2.2** we address the concept of blockchain as well as its permission categories. In **Section 2.3** we introduce the blockchain underlying concepts such as smart contracts, smart contract platforms and Daml programming language. Afterwards, in **Section 2.4** we explain the GDPR's relevance and its applicability to blockchain technology. Finally, in **Section 2.5** we discuss some general blockchain use cases followed by describing the concept of tokenization of real estate assets, and lastly, we introduce and compare some specific blockchain use cases in real estate.

2.1 Real Estate

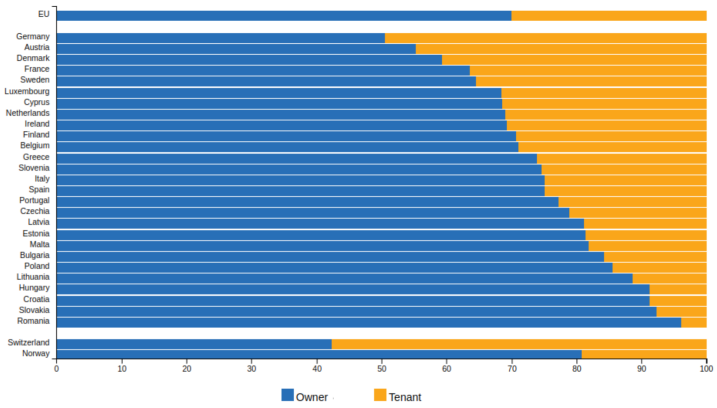
The oldest documented history of real estate business is estimated to have been discovered in the Stone Age approximately 12,000 years ago, in inscriptions and cave drawings. It is believed that even back then, shelter owners rented out their shelters in exchange for

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.

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 [6] in **Figure 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.

Fig. 1: Percentage(%) of people living in households owning or renting their home in 2020 Source: Eurostat [6]



Renting or leasing 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 lease, which includes the **obligations** and **rights** of both parties, respecting the laws of the country or state in which the property is located. The lease agreement's primary goal is to legally assist landlords and tenants in avoiding disputes. If conflicts eventually arise, the contract should address them. A lease is able to secure different types of residential properties as well as different types of typologies within the property. A rental contract can be quite complex and contain a wide range of regulations, 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 1**

Table 1: Essential data for a residential leasing contract

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

When celebrating a lease 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 depicts the different types of leases.

Table 2: Types of leasing agreements

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

Fixed-term leases have an expiration date. This sort of rental lease agreement is used when a tenant agrees to rent a property for a specific period of time at a predetermined fee. Fixed-term leases employ calendar dates to determine the lease's start and termination dates. The agreement expires when the set term lease expires. At that time, the tenant and landlord may sign a new lease with updated dates and other details. Nevertheless, the tenant may opt to leave at that moment [7].

Automatic renewal is a form of lease 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 [7].

Month-to-month leases 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 [7].

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 [7].

Sublease 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 sublease their rental property for brief periods of time while relocating so that they don't compromise their tenancy agreement [7].

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 [7].

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 when establishing the contract. 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 [8]. 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 as such an intermediary, capable to notarise 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, paperwork, combating the black market, reducing tax evasion, preventing fraud, corruption, finance instability, and vulnerable tenants from legal evasion [8]. In the following section, we will go over how blockchain technology operates, its permission types, and how it can benefit the residential rental market.

2.2 Blockchain

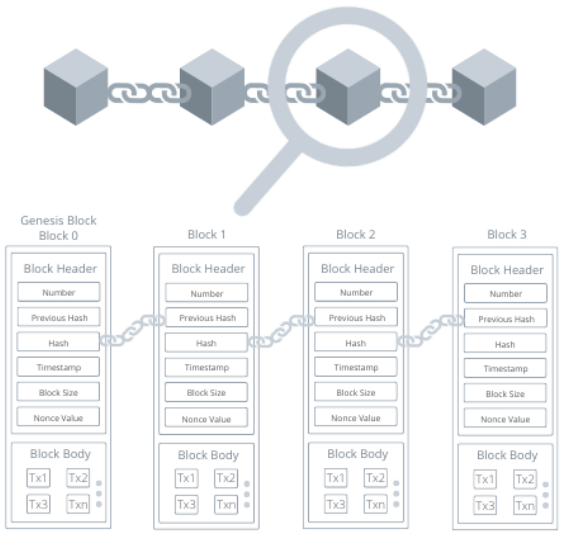
Since 2008, blockchain technology has grown in popularity, especially due to the valuation of the Bitcoin network, a result of Nakamoto's white paper publication, where he described an innovative cryptocurrency network becoming not only the first ever blockchain application, but also, the first ever 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 [2].

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 transactions history, yet its transactions values cannot be rewritten once confirmed. The dis-

tributed 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: The **distributed architecture**, made possible through the use of both distributed ownership as well as a distributed physical architecture, the **cryptographic mechanisms**, namely hash functions and digital signatures which safeguard the integrity of transactions and conformity between blocks, and 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 of 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**.

Fig. 2: The blockchain structure in detail



As shown in the **Figure 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 3 further clarifies the concepts.

Table 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 who do not trust each other 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** and **integrity**.

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 [2].

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 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 [2]. 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 [9]. 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 [9].

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 [10].

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 [2]. There is no need to perform resource-intensive computations since a participant is randomly selected to append to the block, and, once successful, collect the reward [11].

In short, we discussed in this section the emergence of blockchain technology made popular with the appearance of Bitcoin in 2008 [1]. 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**. Next, we will analyze which participants are allowed to participate in the consensus algorithm by categorizing blockchains based on their **permission mode**. Afterwards, we will discuss what are **smart contracts** and what are they used for.

Permissionless Versus Permissioned

Permissionless or **Public** blockchains such as **Bitcoin**, **Ethereum** [12], or **EOS** [13] 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.

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 and 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** [14], **R3 Corda** [15] and **Canton** [16]

Table 4: Categorization of blockchain according to permission mode [17].

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

2.3 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 [18]. 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** [19], idealizing a platform extensible to other uses beyond the simple exchange of cryptocurrencies. In 2016 **Vitalik Buterin** launched the **Ethereum** [12] platform, being regarded as the second generation of the blockchain. It is a programmable platform whose network works through smart contracts allowing the creation of decentralized applications, transfer of digital assets, and transfer of cryptocurrencies. **Ethereum** uses **Ether** as its cryptocurrency.

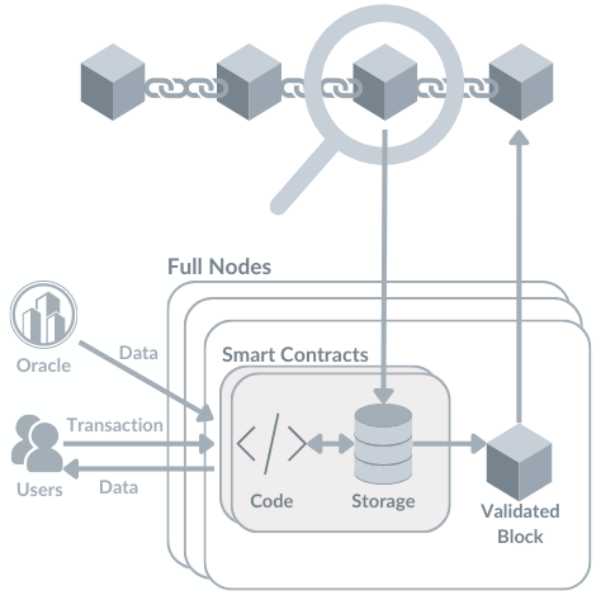
A smart contract is an executable code that resides and executes on the blockchain to enforce the terms of an agreement [20]. They ensure automated enforcement of the terms whenever certain criteria are satisfied, meaning that, once arbitrary predefined requirements

are satisfied, digital assets are released to the concerned parties [19].

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. [3].

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 [20]. 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 [20].

Fig. 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 real or crypto assets, weather conditions, political events, sports events, and so on [21].

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 [22]. **Ethereum** was the first blockchain protocol to enable users to write smart contracts, although other recent protocols, including, **Hyperledger**, and **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 [23]. 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 [23].

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. That being said, in the next section we will analyze and compare the main platforms that enable the development of smart contracts.

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 5**, compares the **5** 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 and requires **PoW** as its consensus algorithm, which is computationally intensive and not particularly viable when considering energy expenses. Ethereum adopts an **account-based** data model, where each node is identifiable by its digital wallet [24]. Its permission mode makes Ethereum a fully transparent platform which reduces scalability performance, **privacy** and **GDPR compliance** [25] of 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 [26]. 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.

Table 5: Comparison of Smart Contract Platforms [16] [24]

	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	PoW	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

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 [27].**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 the 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 [28] 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 [24]. 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, every account may have a human-readable name reference. **EOS**, like Ethereum, due to its permission mode, promotes transparency, so aspects such as **privacy** and **GDPR compliance** are 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 in-built authorization and privacy models [16]. 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** [16]. 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.

Next we will take a closer look into the **Daml** programming language, its features and behavior, as well as taking a brief glimpse of a smart contract example written in **Daml**.

Daml

Daml or **Digital Asset Modeling Language** is a Haskel-inspired functional programming language designed to develop smart contracts for distributed enterprise workflows. It is an open-source language conceived for a wide variety of business processes. It does not depend on any particular blockchain system, making it a universal solution for operating with

multiple blockchain platforms, or even traditional database architectures. **Daml** enables programmers to reduce the amount of time dealing with blockchain complexity by allowing them to focus on their business logic solutions. It is an exceptionally human-friendly language allowing both humans and machines to read the information contained in smart contracts. Professionals interested in contract revision, such as lawyers, may quickly comprehend its code meaning by reading and analyzing it. **Daml** provides developers an easy development pipeline, allowing smart contracts to be written in less time, with shorter code and fewer mistakes

One of its key features is the provision of **privacy**. Daml privacy model provides privacy at the level of subtransactions, where, a party learns only those parts of the ledger changes that affect the contracts in which the party has a stake, and the consequences of those changes [29]. **Daml** keeps track of which parties have access to the smart contract details and then transmit that information to the blockchain. Being aware of this, the blockchain transmits the data to the party's physical node on a network, only if the party is allowed to see it.

Daml was specially designed for blockchains with a **permissioned** mode. This is certainly beneficial for strengthening privacy, however, a permissioned network might increase the risk of corruption and manipulation, since network owners could manipulate consensus rules, mining, or immutability. Such factors lead to a limited network decentralization.

Despite **Daml** rapid growth, it is still a relatively new technology, and therefore does not yet have a strong developer community, As a consequence, there is a shortage of specialists to work with it. **Table 6** depicts **Daml** advantages and disadvantages.

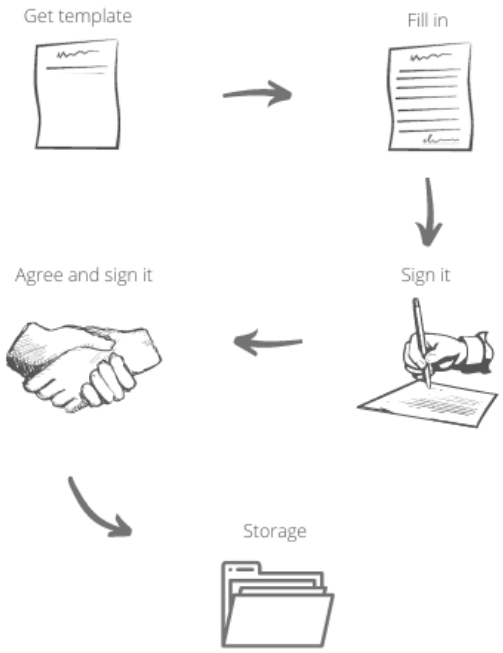
Table 6: A comparison between Daml advantages and disadvantages

Advantages	**Disadvantages**
Strong privacy	Limited decentralization
Easy development flow	Small community
Human-Machine readability	Shortage of specialists
Portability across multiple blockchains	-

Daml Smart Contracts

Before we dive into a practical smart contract case, we should first recall the life of a **traditional contract**. **Figure 4** illustrates a traditional celebration of a contract agreement. In real life, to celebrate an agreement between multiple parties, first it is necessary to obtain a legal paper contract template that complies with specific local government regulations. Afterwards, the contract is filled out and signed. Once signed, it is handed over to the other party(ies) to be reviewed and signed upon agreeing to the terms.

Fig. 4: Life of a Traditional Contract



After all parties have signed, it is manually stored in an archive and then enforced. A contract frames two types of participants, the **signatories** and the **observers**. The **signatories** are the participants whose authority is required to create the contract or archive it. Each contract must have at least one signatory. A contract is visible to all **signatories**, however it may also be visible to **observers**, such as regulators, or specific law enforcement entities. The terms of the contract may or may not be changed. Usually they are changed, to correct or add new details. If an amendment is made, the current contract becomes null and void, and therefore archived. Then a new contract with a similar model is filled out and signed, replacing the previous one.

Daml aims to portray in code the life of a traditional contract. A contract written in Daml is actually a template and not a contract, however, a contract is an instance of a template. **Figure 5** illustrates a code example of a contract agreement between two parties regarding a property lease.

Beginning from the top, every **Daml** file defines a module. The template name, in this case "Lease" is defined right away. A **template**, represents a self-issued, non-transferable token [29]. From line 5-10 the variables are declared, just like in a traditional approach when declaring the interested parties. We defined a **tenant** and a **landlord**, as well as the **property** to be leased, the **amount** to be paid, and the **duration** of the lease. These are only a handful of several variables that may be declared to enhance the contract complexity.

Fig. 5: An example of a lease agreement written in Daml

Lease.daml

```
1  module Lease where
2
3  template Lease
4    with
5      tenant   : Party
6      landlord : Party
7      property : Text
8      amount   : Int
9      duration : Date
10     -- other details
11   where
12     signatory tenant
13     observer landlord
14
15     agreement
16       "I hereby agree that by signing this contract
17       I become this property tenant for the agreed
18       duration and I will comply with all contract terms"
19
20     key tenant : Party
21     maintainer key
22
23     choice AgreeWithTerms : ContractId Lease
24       with
25         newTenant   : Party
26         newDuration : Date
27       controller tenant
28       do
29         -- business logic
30         -- ...
31         create this with
32           tenant = newTenant
33           duration = newDuration
34
```

In lines 12-13 we set the permissions for some variables. Party "tenant" is a contract **signatory** whereas party "landlord" is an **observer**. Lines 15-18 exhibit text outlining the agreement represented by this contract. As a result, its terms are stated in a human-readable fashion facilitating an easy understanding.

Lines 20-21 indicate the contract **key**. A contract **key** is optional and is used to uniquely identify a contract. Keys can assume any value other than the **contract ID**. A key is always accompanied by single or multiple **maintainers**. Just as the signatories own a contract, the maintainers own a key. They prevent double allocation or incorrect lookups. **Maintainers** must be contract signatories since the key is part of the contract [29].

Lines 23-33 provide a contractual choice. Choices in practice operate just like normal func-

tions. We created one choice called **AgreeWithTerms** which aims to handle the business logic behind the property leasing. It returns a reference to a contract of type **Lease** taking as parameters a Party "newTenant" and a date "newDuration". Additionally, line 27 states that the choice is controlled by the **tenant**, meaning that the **tenant** is the only party allowed to exercise it. Right below, the "**do**" block defines the logic that the choice must perform when exercised. It is not implemented in this example, although we could easily do a few variable validations for instance. Finally, lines 31-33 are in charge of **creating** and **archiving** the contract. In fact, there is no explicit instruction to archive it. **Consuming** choices by default archive the contract when exercised as opposed to **nonconsuming** choices that only archive a contract when explicitly requested.

In this section we learned about **Daml's** behavior, structure, and utility. We evaluated its advantages and disadvantages and examined step by step a specific but simple example of a smart contract. In the next section, we will analyze an important European data protection regulation, which is currently vital for the development of any business, that offers services to European citizens, including real estate

2.4 GDPR

GDPR or General Data Protection Regulation [5] 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 within the EU, may face high monetary penalties if it fails to handle and process consumer information [30].

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 [30]. Any reversible process, such as generating pseudonyms for instance generating blockchain public keys or encrypting data, could reveal the original data which would be violating GDPR. Conversely, anonymised 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 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 [31]. 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 technique 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.

2.5 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.

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 [32]. Undoubtedly, the banking and financial sectors are the most eager to invest in the technology [33], mainly due to the appearance 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 [34].

We briefly introduce some applications that, through blockchain, are reinventing and innovating some segments of society. **Borderless** is a government platform based on smart contracts enabling the provision of a wide range of economic and legal services, such as legal entity registration, notary services, marriage, notary services, and financial transactions [34]. **E-Residency** [35] 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 [34]. **MedRec** is a blockchain-based medical system intended to deliver secure, transparent and scalable access to medical data records [34]. **Black** is a digital insurance company on the blockchain, which opens the centralized insurance market for crowdfunding. **Gilgamesh**

is an educational platform aiming to share knowledge that works through Ethereum smart contracts. Readers, critics, and authors interact 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 [36].

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.

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 [4], 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 [4]. 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 7** shows its different possible representations.

Table 7: Representations of a Token in Real Estate [37]

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 7** 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 lease 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 [38].

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. [39], 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.

Blockchain Use Cases in Real Estate

The traditional real estate system faces some drawbacks including high intermediation costs, time-consuming bureaucracy and administration, lack of standardized processes, high legalization property costs, 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 property leases based on the blockchain technology. A smart rental contract should be able to execute the predefined terms of the smart rent contract on its own. Furthermore, if both parties have not signed a formal rental agreement, it is imperative that a smart rent contract be able to provide evidence of the lease'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 [40].

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 [41].

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 [41].

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 [42].

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 [42]. For now, according to their official project documentation [42], 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 lease 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 [43]. The platform is distinguished by three prominent features. The first is the security of bond deposit, which holds funds securely for the lease 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 [43].

RentPeacefully

RentPeacefully is a blockchain-based platform that allows landlords to advertise properties for rent and create lease 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 [44].

The Bee Token

The Bee Token is the token that powers the **Beenest** network [45] 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 [45]. A decentralized arbitration system for resolving user disputes, with positive incentives to build a network of legitimate arbitrators and negative incentives to discourage fraudsters [45], 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 [45].

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 8**, we compare the properties of the aforementioned projects.

It is noticeable, by looking at **Table 8**, 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.

Table 8: 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

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.

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 Mi-

dasium 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.

3 Solution Proposal

In this chapter we will introduce our proposed solution as well as its implications and benefits. We will decompose its complexity into several phases so that the reader does not feel lost. In a first phase, we will discuss the existing problems we wish to overcome. In a second phase we will generally explain how the solution operates and how participants interact with each other. At last, in a third phase, we will discuss the chosen blockchain platform, its architecture, design, features, blockchain-application interaction, and compliance with EU GDPR legislation.

3.1 Existing Problems

As discussed in the **Section 2.1**, the current real estate renting process has several issues that our proposed solution aims to address. Some of them are:

- High intermediation costs
- Time-consuming and over-bureaucratic processes
- Information shared over multiple unsecured channels
- Lack of transparency and traceability
- Susceptibility to fraud and tax evasion
- Lack of automatic rental payment mechanisms
- Lack of tools to manage rented properties

3.2 Overview

Our proposed solution aims to solve the existing problems mentioned in **Section 3.1** by providing a reliable blockchain-based platform that allows landlords and tenants, through an application that interacts with a blockchain, to manage real estate transactions. Parties can create, sign modify, delete, and manage rental agreements over rental units through smart contracts. The application will feature a front-end that will allow landlords to advertise their rental units to interested tenants. The application is expected to provide a variety of rental contract templates which will comply with local regulations and GDPR. The application will also feature a back-end that will be responsible for interacting with the blockchain platform as well as handling data.

3.3 Solution Decisions

In this section, we will discuss which platforms were selected for implementing our solution, and what reasons led to their choice over others.

Smart Contract Platform

As previously stated in **Section 2.3**, we discussed and analyzed several blockchain platforms able to support the development of smart contracts. We will now evaluate their pros and cons to determine which one is best suited for our solution, taking into consideration decision factors such as core functionality, privacy, GDPR compliance mechanisms, pricing, community support, and Daml compatibility

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 solution. 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 5** in **Section 2.3**, at first glance, the choice of Canton seems obvious, as it stands out for its strong features regarding transaction privacy, GDPR compliance, and Daml compatibility. 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, we believe that Canton, for now, does not fit what we desire.

Both Fabric and Corda support Daml, thus, 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 believe it is reasonable to move forward towards the adoption of Hyperledger Fabric platform.

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**, the proposed solution will be based on the Hyperledger Fabric platform. Thus, there is a restricted set of programming languages from which we may choose. **Table 5** in **Section 2.3** mentions three: Java, Golang, and Node.js. However, although not listed in the table, according to recent community contributions, Fabric already supports writing Smart contracts in Daml. Daml presents unique features. It is human-friendly, allowing even non-IT professionals to understand a contract. It offers strong privacy, easy development flow and portability among other blockchain platforms. Given these criteria, we believe it is the ideal candidate to implement our solution.

3.4 Architecture and Design

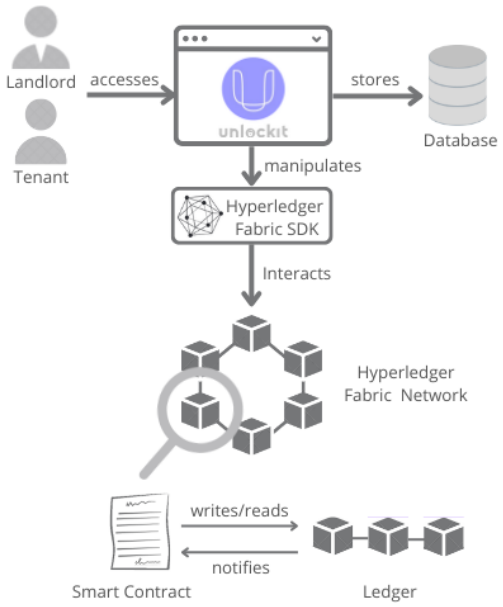
The proposed solution will be supported by the Hyperledger Fabric blockchain. Fabric will be in charge of safeguarding and keeping rental contracts private by implementing different channels to protect landlord and tenant data. To interact with a Hyperledger Fabric network, our solution requires an application that interacts with the network using the Hyperledger Fabric SDK.

The SDK is the software development kit that provides APIs for our solution 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 stop or start the network, to manage user identities, and to deploy or update smart contracts.

Our solution will provide an application, featuring an easy-to-use user-friendly front-end, to be implemented in a framework like React.js, and a back-end to be developed in Node.js or similar technology. The application will allow landlords to advertise their residential units, and tenants to search for them. A tenant, having decided which property wishes to rent, sends a note to the landlord. After agreeing on the terms, the landlord fetches a smart contract template available on the platform and signs it together with the tenant, establishing a valid rental contract. The application will then interact with Fabric's SDK to publish the contract on the network.

The system high-level architecture is depicted in **Figure 6**.

Fig. 6: Proposed Solution Architecture



Regarding the system's GDPR compliance, it is known that blockchain records data permanently and immutably. European GDPR legislation, on the other hand, stipulates the right to be forgotten, which states that all personal data must be erased when requested by a consumer. As mentioned in **Section 2.4**, there are currently two techniques to overcome this problem, namely crypto-shredding and hashing.

Our solution aims to adopt the hashing technique combined with an external database which will store the users personal data in plaintext along with the hash of that same data. The Hyperledger Fabric blockchain will record the hashes associated with the users personal data, inside smart rental contracts. Therefore the contracts will not expose users confidential information, as they only contain a hash, and as previously stated it is practically impossible to reverse a hash data.

So, should a user wish to delete his personal data from unlockit's service, his personal data would be erased from the external database, yet, the hash stored in the blockchain would stay permanently. So, even though the hash is kept on the blockchain, the right to be forgotten is not violated since the hash would not identify any record in the external database.

The solution architecture represented in **Figure 6** 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 the Hyperledger Fabric blockchain network by manipulating the Hyperledger Fabric SDK. The SDK creates and monitors smart contracts with the intent of committing and verifying transactions on the network.

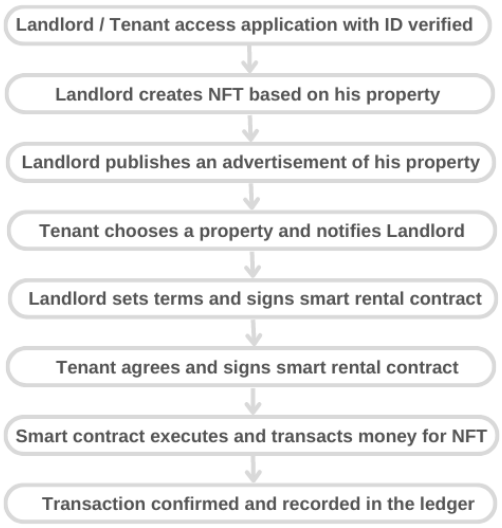
Compliance with European GDPR legislation is achieved by combining an external database and a hashing technique. Landlords and tenants will have their personal data recorded in an external database and will have a hash associated with that personal data on the Hyperledger Fabric blockchain.

3.5 Rental Process

Until this point, we have introduced the solution's architecture and design. We will now understand the rental process in more detail. **Figure 7** illustrates the steps required to formalize a rental agreement. We will explain the entire process in further detail. A tenant or landlord accesses the platform and registers through a trusted third-party verification mechanism. This verification mechanism will provide trust between application users, since all users are verified by a legitimate institution, preventing fraud in a later rental process.

After an initial identity verification, unlockit's platform will allow a landlord to create an NFT of the entire property or part of it, and then store it in its wallet. The landlord will then publish an advertisement linked to the created NFT. A tenant, after being initially verified, is able to search for, and select a property.

Fig. 7: The process of renting a property



After the selection, the tenant notifies the landlord about his interest. The landlord selects a rental template on the platform, customizes it, and describes the terms of the agreement. Finally signs it and sends it to the interested tenant. The tenant agrees and signs the contract.

Immediately, the smart contract is executed, the money is sent to the landlord's wallet and the NFT to the tenant's wallet. The contract is then confirmed by the network and consequently recorded in the ledger. Eventual intentions to modify or terminate a smart rental contract are accomplished by creating a new one.

3.6 Rental Payments

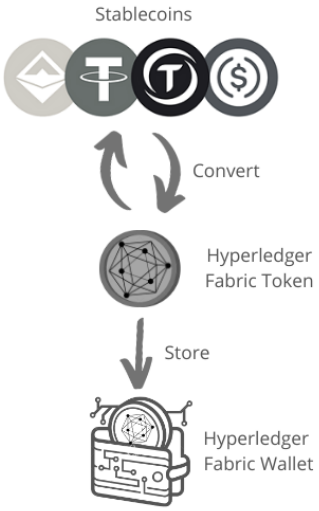
So far we have discussed the solution architecture and the rental process in detail. However, we still need to address the issue of transactions payment and money. We propose a payment strategy to handle transactions on the Hyperledger Fabric network. For that, we first need to briefly introduce two related concepts, wallets and stablecoins.

A wallet is a digital storage location used to store digital assets such as tokens used for payments like Hyperledger Fabric tokens, known as FT tokens, non-fungible tokens or NFT's, smart contracts and other digital data. A wallet contains the identity of an owner. An owner is able to manage and manipulate his digital assets by accessing his wallet.

Stablecoins are a type of digital currency designed to avoid the volatility of cryptocurrencies such as Bitcoin or Ether and maintain a constant, stable value. 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.

Our solution relates these two concepts. Recent community developments introduced in Hyperledger Fabric version 2.0, the FabTokens, or in short FT's. FT's are stored in a participant's digital wallet and allow them to pay for transactions and services on the Fabric network. Due to volatility, our solution intends to provide a payment method based on stablecoins. The Hyperledger Fabric platform allows the creation of its own stablecoin, nevertheless, we are looking for a more stable and recognized stablecoin in order to build trust with the users. However, since the Hyperledger Fabric network only supports specific tokens, it is necessary to implement a conversion mechanism to convert stablecoins into FabTokens to allow their registration in the ledger. **Figure 8** illustrates the described payment process.

Fig. 8: The Payment Process



4 Evaluation Methodology

The methodology for evaluating our solution will go through four stages. The first stage will involve the development and implementation of the solution architecture. Initially, we will prioritize the construction of the solution's infrastructure, focusing on the front-end and back-end application that will allow the common user to access the platform easily. Next, we are going to deploy and configure the Hyperledger Fabric platform. We will configure which peers and organizations have initial access, as well as setting up the communication channels between peers and establishing the connection between application and Fabric through the Hyperledger Fabric SDK. Furthermore, during the first phase, we intend to design the smart contract templates in accordance with different regulations, taking into account GDPR compliance.

The second stage will involve validating our prototype with potential users of the solution, such as landlords and tenants, who want to try out the system without an actual commitment of their properties or money. We intend to obtain validation from established

organizations in the real estate industry, such as real estate agencies and real estate agents who could provide us with specialized feedback. The validation aims to gain feedback on the functionality, appearance, and engagement of the application with the end user.

The third stage focuses on exhaustively verifying whether the solution complies with GDPR legislation, by ensuring that smart rental contract templates offer privacy and comply with data protection criteria. We will test how two parties establish a secure channel between each other and how they guarantee privacy.

Finally, the fourth and final stage will consist of testing the solution regarding performance. We intend to evaluate the system's latency from the moment a lease is agreed, modified, or canceled to the point where changes are committed to the network.

5 Schedule of Future Work

Future work is scheduled as follows:

- **January** - Validate the solution architecture and set up the development environment
- **February** - Design and implementation of the application front-end and back-end which includes designing and developing the smart rental templates.
- **March** - Implement and configure Hyperledger Fabric blockchain and set up the connection between application and Fabric.
- **May** - Implement smart contracts in Hyperledger Fabric, define contract logic and ensure compliance with GDPR legislation
- **June** - Writing the dissertation document and a scientific paper describing the project.
- **July** - Final revision and finish dissertation.

6 Conclusion

Real estate industry currently faces complex and inefficient rental property processes, therefore, formulating innovative solutions is imperative. This document proposes a solution to overcome the existing problems through an application that interacts with Hyperledger Fabric blockchain providing services for listing properties, signing, modifying, and terminating leases through the latest smart contract technology.

This document explores the key concepts required to easily understand and apply our solution. We began by introducing the term real estate, its existing problems, the types and durations of a contract, and the type of information that should definitely be included in a rental contract. Then, we introduced the blockchain concept and its features. We categorized it into different permission categories and evaluated their benefits and drawbacks. Afterwards we explained the definition of smart contracts, followed by a discussion about which platforms and programming languages support their development, with particular focus on Daml. Later we explained the EU GDPR legislation and emphasized its importance.

Next, we explored some general blockchain use cases, then we introduced the process of tokenizing a real estate asset, and lastly, we dived into more specific blockchain use cases in

real estate. We reviewed and compared some companies and business ideas, most of which, were merely proof-of-concept, yet, we found some already operating on the market.
We hope that our research efforts, as well as our proposed solution, 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.

References

1. S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2008. [Online]. Available: https://bitcoin.org/bitcoin.pdf
2. D. Yaga, P. Mell, N. Roby, and K. Scarfone, "Blockchain technology overview," *arXiv e-prints*, pp. arXiv–1906, 2019.
3. B. K. Mohanta, S. S. Panda, and D. Jena, "An overview of smart contract and use cases in blockchain technology," in *2018 9th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*. IEEE, 2018, pp. 1–4.
4. J. Vincent, S. Bender, and P. Smirniotopoulos, "Session 5: Real estate tokenization," 2020.
5. 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.
6. Eurostat. (2021) House or Flat – Owning or Renting. [Online]. Available: https://ec.europa.eu/eurostat/cache/digpub/housing/bloc-1a.html
7. I. ContractsCounsel. (2020) Rental Lease Agreement - Types of Rental Lease Agreements.
8. R. M. Garcia-Teruel, "Legal challenges and opportunities of blockchain technology in the real estate sector," *Journal of Property, Planning and Environmental Law*, 2020.
9. Z. Zheng, S. Xie, H. Dai, X. Chen, and H. Wang, "An overview of blockchain technology: Architecture, consensus, and future trends," in *2017 IEEE International Congress on Big Data (BigData Congress)*. IEEE, 2017, pp. 557–564.
10. K. J. O'Dwyer and D. 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)*.
11. F. Saleh, "Blockchain without waste: Proof-of-stake," *The Review of Financial Studies*, vol. 34, no. 3, pp. 1156–1190, 2021.
12. V. Buterin, "Ethereum: platform review," *Opportunities and Challenges for Private and Consortium Blockchains*, 2016.
13. I. Grigg, "EOS-an introduction," *White paper. https://whitepaperdatabase. com/eos-whitepaper*, 2017.
14. V. Dhillon, D. Metcalf, and M. Hooper, "The HyperledgerProject," in *Blockchain enabled applications*. Springer, 2017, pp. 139–149.
15. R. G. Brown, J. Carlyle, I. Grigg, and M. Hearn, "Corda: an introduction," *R3 CEV, August*, vol. 1, no. 15, p. 14, 2016.
16. D. A. C. Team, "A Daml based ledger interoperability protocol," 2020. [Online]. Available: http://canton.io
17. J. 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.
18. N. Szabo *et al.*, "Smart contracts," *white paper*, 1994.
19. V. Buterin *et al.*, "A next-generation smart contract and decentralized application platform," *white paper*, 2014.
20. M. Alharby and A. van Moorsel, "Blockchain-based smart contracts: A systematic mapping study," *arXiv e-prints*, pp. arXiv–1710, 2017.
21. G. Caldarelli, "Understanding the blockchain oracle problem: A call for action," *Information*, vol. 11, no. 11, p. 509, 2020.

22. W. Zou, D. Lo, P. S. Kochhar, X.-B. D. Le, X. Xia, Y. Feng, Z. Chen, and B. Xu, "Smart contract development: Challenges and opportunities," *IEEE Transactions on Software Engineering*, vol. 47, no. 10, pp. 2084–2106, 2019.
23. K. Wu, "An empirical study of blockchain-based decentralized applications," *arXiv e-prints*, pp. arXiv–1902, 2019.
24. Z. Zheng, S. Xie, H.-N. Dai, W. Chen, X. Chen, J. Weng, and M. Imran, "An overview on smart contracts: Challenges, advances and platforms," *Future Generation Computer Systems*, vol. 105, pp. 475–491, 2020.
25. M. Valenta and P. Sandner, "Comparison of Ethereum, Hyperledger Fabric and Corda," *Frankfurt School Blockchain Center*, vol. 8, pp. 1–8, 2017.
26. J. Polge, J. Robert, and Y. Le Traon, "Permissioned blockchain frameworks in the industry: A comparison," *ICT Express*, vol. 7, no. 2, pp. 229–233, 2021.
27. H. F. Docs. (2020) Hyperledger Fabric Channels. [Online]. Available: https://hyperledger-fabric.readthedocs.io/en/latest/channels.html
28. H. Howard, M. Schwarzkopf, A. Madhavapeddy, and J. Crowcroft, "Raft refloated: Do we have consensus?" *ACM SIGOPS Operating Systems Review*, vol. 49, no. 1, pp. 12–21, 2015.
29. D. A. S. GmbH, "Daml SDK Documentation Version : 2.4.0," p. 1251, 2022. [Online]. Available: https://docs.daml.com/
30. B. Wolford. (2018) What is GDPR, the EU's new data protection law? [Online]. Available: https://gdpr.eu/what-is-gdpr/
31. CMS. (2019) The tension between gdpr and the rise of blockchain technologies. [Online]. Available: https://cms.law/en/media/international/files/publications/publications/the-tension-between-gdpr-and-the-rise-of-blockchain-technologies
32. K. Wüst and A. Gervais, "Do you need a blockchain?" in *2018 Crypto Valley Conference on Blockchain Technology (CVCBT)*. IEEE, 2018, pp. 45–54.
33. K. Zīle and R. Strazdiņa, "Blockchain use cases and their feasibility," *Applied Computer Systems*, vol. 23, no. 1, pp. 12–20, 2018.
34. J. Golosova and A. Romanovs, "Overview of the blockchain technology cases," in *2018 59th International Scientific Conference on Information Technology and Management Science of Riga Technical University (ITMS)*. IEEE, 2018, pp. 1–6.
35. C. Sullivan and E. Burger, "E-residency and blockchain," *Computer Law & Security Review*, vol. 33, no. 4, pp. 470–481, 2017.
36. N. Duwadi *et al.*, "A systematic review on blockchain in education: Opportunities and challenges," 2021.
37. Dentons Team. The tokenization of real estate: An introduction to fractional real estate investment. [Online]. Available: https://www.dentons.com/en/insights/articles/2022/september/6/the-tokenization-of-real-estate
38. FCE Media. Real estate tokenization: why and how it works. [Online]. Available: https://fcegroup.ch/en/news/text/id394-2022-06-05-real-estate-tokenization-why-and-how-it-works
39. J. Smith, M. Vora, H. Benedetti, K. Yoshida, and Z. Vogel, "Tokenized securities and commercial real estate," *Available at SSRN 3438286*, 2019.
40. K.-J. Yong, E. S. Tay, and D. W. Khong, "Application of blockchain smart contracts in smart tenancies: A malaysian perspective," *Cogent Social Sciences*, vol. 8, no. 1, p. 2111850, 2022.
41. R. Team. (2022) Rentible whitepaper. [Online]. Available: https://whitepaper.rentible.io/
42. SMARTRealty Team. (2018) SMARTRealty whitepaper. [Online]. Available: https://smartrealty.io/wp-content/uploads/2018/01/SMARTRealty-Whitepaper-v1.pdf
43. Midasium Team. (2017) Midasium Whitepaper. [Online]. Available: https://midasium.herokuapp.com/smart-tenancy
44. RentPeacefully Team. (2018) What is RentPeacefully? [Online]. Available: https://rentpeacefully.com/
45. Beetoken Team. (2018) Beetoken Whitepaper. [Online]. Available: https://www.allcryptowhitepapers.com/beetoken-whitepaper/