UNIVERSITY OF THESSALY

SCHOOL OF ENGINEERING

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

Collective Space Management with the Use of Blockchain

Diploma Thesis

Apostolos Filippidis

Supervisor: Manolis Vavalis

July 2025

UNIVERSITY OF THESSALY

SCHOOL OF ENGINEERING

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

Collective Space Management with the Use of Blockchain

Diploma Thesis

Apostolos Filippidis

Supervisor: Manolis Vavalis

July 2025

ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Συλλογική Διαχείριση Χώρων με τη Χρήση Blockchain

Διπλωματική Εργασία

Απόστολος Φιλιππίδης

Επιβλέπων/πουσα: Μανόλης Βάβαλης

Ιούλιος 2025

Approved by the Examination Committee:

Supervisor **Manolis Vavalis**
Professor, Department of Electrical and Computer Engineering, University of Thessaly

Member **Konstantinos Lalenis**
Emeritus Professor, Department of Planning and Regional Development, University of Thessaly

Member **Dimitrios Polychronopoulos**
Professor, School of Architecture, Democrition University of Thrace

DISCLAIMER ON ACADEMIC ETHICS AND INTELLECTUAL PROPERTY RIGHTS

«Being fully aware of the implications of copyright laws, I expressly state that this diploma thesis, as well as the electronic files and source codes developed or modified in the course of this thesis, are solely the product of my personal work and do not infringe any rights of intellectual property, personality and personal data of third parties, do not contain work / contributions of third parties for which the permission of the authors / beneficiaries is required and are not a product of partial or complete plagiarism, while the sources used are limited to the bibliographic references only and meet the rules of scientific citing. The points where I have used ideas, text, files and / or sources of other authors are clearly mentioned in the text with the appropriate citation and the relevant complete reference is included in the bibliographic references section. I also declare that the results of the work have not been used to obtain another degree. I fully, individually and personally undertake all legal and administrative consequences that may arise in the event that it is proven, in the course of time, that this thesis or part of it does not belong to me because it is a product of plagiarism».

The declarant

Apostolos Filippidis

Diploma Thesis

Collective Space Management with the Use of Blockchain

Apostolos Filippidis

Abstract

Cooperative management means making a collective effort through communication and cooperation towards a goal. The decentralized nature of blockchain technologies fits like a glove in this context. Motivated by a project built in 2019, called Wareblock, which proposes renovating and leasing abandoned warehouses, through tokenized crowdfunding, we present the following work.

This thesis consists of four parts. The first part is to create a business plan. This is to show that a property investment using blockchain technology is viable and in what way it could be achieved. In the second part, this thesis explores technologies, ways, and models that combine blockchain with cooperative management through a comprehensive and thorough review. The third part is a small effort to continue the Wareblock platform by implementing a management and lease contract infrastructure, using two projects discovered in our review. Finally, we present a real-world case study that ties everything together.

The result is a group of smart contracts that can be used effectively as a back-end should the platform be implemented fully. Secondly, a guide for building a business plan for the majority of properties. And third, we lay a foundation for a continuance in this work, through the literature review and implementation, which offers a great deal of guidance and information to a potential governance or management system for property investment.

Keywords:

blockchain, cooperative management, collaborative property management, decision-making, governance, smart contracts, Wareblock, business plan, Solidity.

Table of contents

List of figures

List of tables

Chapter 1

Introduction and Motivation

1.1 Presentation of the subject

Blockchain technologies have developed rapidly in recent years, offering innovative solutions to a wide range of industries. This includes the real estate sector and property management. In particular, blockchain enables collective ownership and management systems, while maintaining a truly decentralized nature. Smart contracts give tokens utility beyond their financial value. A token might represent a title of ownership. By buying this specific token, one buys a percentage of the property. A token might also represent voting power in a decision-making model. Buying the token enables the holder to participate in decision-making within the group of token holders.

This thesis explores the application of blockchain technologies in the context of cooperative space management, where individuals collectively own and manage a property. The study is motivated by a project developed in 2019 titled Wareblock. [2] presents the implementation of the project and provides insight on the matter. The project focuses on the reusing of abandoned buildings, mainly targeting abandoned tobacco warehouses in the northern Greek area. Wareblock can create a unique token that represents the ownership of the warehouse. The token is then distributed to investors by participating in a crowd-fund. The collected funds are then used for the uptake of the building. The building is eventually leased, and investors can collect their monthly share.

Reviewing both the paper and the implementation (in Chapters 3 and 4 respectively), we notice that co-management is a topic left for future investigation and implementation. To approach this topic, we follow a general approach to the matter. In this thesis, we investigate

and analyze different concepts that are closely or remotely related to management with the use of blockchain. Collaboration or perhaps a collective dynamic is a key feature we expect to find. The investigation covers topics such as blockchain management, decision making, governance, governance tokens, and Decentralized Autonomous Organizations management schemes.

The review contains literature spanning from published papers to gray literature, to implementations and projects. We identify smart contracts that align closely with the goals of collaborative management. In this thesis, we present an effort to combine three different implementations to make the first step towards the original vision, presented in [2]. The implementation addresses the post-crowdfund part. Using the warehouse token, supplied by the crowdfund, token holders can create proposals and vote on property matters. Individuals can also rent the property, and proceeds from rent are distributed to token holders.

Participants in crowd-funding purchase warehouse tokens in proportion to their deposited amount. For this reason, as far as the property is concerned, the participants are considered investors. An investor needs to be presented with a business plan of the property's operation before buying tokens. In this thesis, we present a guide on how a business plan can be created. The presented structure follows a break-even plan. This is a rational strategy which in the end leaves investors with their original deposit and a property share at hand.

To conclude the thesis, we present a case study of a real building. We present calculations and projections following the business plan for the property. We discuss the use after renovation, budget and operation costs, and calculate the token price and lease price to break even. We also calculate the realistic cost of the blockchain network using our smart contract implementation.

1.2 Thesis Organization

The rest of this thesis is organized as follows.

- The remainder of this chapter 1.3 presents the theoretical background necessary to read this thesis.

- Chapter 2 presents a detailed guide on how to build a business plan for a cooperative property management project, including budget, risks, and break-even analysis.

- Chapter 3 reviews the literature on blockchain technologies that have to do with cooperative management, governance mechanisms, decision making, and related projects.
- Chapter 4 analyzes the software presented in Chapter 3, describes the implementation of the system (which integrates the three contracts), and presents experimental results.
- Chapter 5 provides a case study that applies the business plan and the system developed to a real building.
- Chapter 6 concludes the thesis by summarizing the findings, drawing conclusions, and making closing remarks.

1.3 Theoretical Background

1.3.1 Blockchain and Smart Contracts

Blockchain is a storage technology in which data are not controlled by a centralized entity [6, 7]. The database is in the form of a ledger that is replicated and stored on multiple computers worldwide. The computers known as nodes comprise the blockchain network. Each node in a blockchain network communicates directly with all other nodes, forming a P2P (Peer-to-Peer) network. Typically, the ledger is not stored continuously but is divided into blocks. Each block also contains the cryptographic hash of the previous block, forming a blockchain.

An important part of a blockchain network is its consensus mechanism. Consensus means agreement between nodes with the same version of the database. Different types of consensus mechanisms vary depending on the implementation. A consensus mechanism is the protocols, incentives and ideas that lead the blockchain network to agreement [8]. In chapter 3, we mention decision making by consensus. This means that the entire community is on board with a decision instead of voting-based decision-making. In that case, only the majority of the community is on board.

[9] is a blockchain that does not just store a distributed ledger. It is a distributed database that holds accounts and transactions, but also stores an entire machine state. The system is known as the Ethereum Virtual Machine (EVM), which is useful because of smart contracts. Smart contracts are specific programs that can be executed by the EVM. Ethereum was the first blockchain to accommodate smart contracts, but it is currently a common practice for

newer blockchains. A high-level language like Solidity can be used to write a smart contract. Compiling the solidity file produces bytecode that can be executed by the EVM.

In the blockchain, assets exist in the form of tokens. A token can be programmed using a smart contract, which means that it can represent a variety of things. The smart contract can produce a single token or an entire token supply. As we discuss in chapter 3, there are certain token standards, such as ERC-20. This means that they are programmed to have a specific functionality. For example, ERC-20 is a divisible token that can have its fractions distributed to the blockchain community of users. The tokens, like cryptocurrencies, can be transferred or exchanged. These actions are performed using transactions. Nevertheless, initiating a transaction is costly because a fee must be paid, which is called Gas.

Since a token can be programmed using smart contracts, a token can be anything. In the case of this thesis, a token might be the title of ownership of a real estate property. To create such a token, the property must undergo a tokenization process called minting.

1.3.2 Business Plan Related Terms

Chapter 2 provides the design of a financial plan for the renovated property. The following provides insight regarding some of the terms used. A budget is a plan of financial operation that includes expenditures and the way to finance them. As explained in [10], banks and guides describe a business budget as a spending plan based on the expected income and expenses of the company. A business plan typically includes a budget as part of the financial projection of a company. The budget helps plan cash flow, allocate resources, and even set safety reserves. It guides many decisions, even ones that come down to hiring personnel or cutting costs.

An expense is any cost made in the process of generating revenue, according to [11]. Examples of expenses are payments to suppliers, wages, rent, and marketing costs. Typically, in a business plan, expenses are broken down by category. There are operating expenses such as rent and utilities, fixed costs such as wages, and variable costs such as rent and supplies. Managing expenses is crucial to a business plan in order to achieve profitability and break-even goals.

The break-even point is the sales level at which a business's total revenue is equal to it's total costs. [12] describes it as the point at which the business realizes no net income or loss. Planning for a break-even goal means determining exactly what the company needs to sell

and at what cost to cover all the costs of doing business.

According to Investopedia [13], Return on Investment (ROI) is a profitability metric used to assess the efficiency of an investment. It is calculated by dividing the net profit by the initial cost of the investment, typically expressed as a percentage.

[14] explains that a surplus is the amount by which resources exceed needs. In budgeting, a surplus occurs when income or revenue exceeds planned expenditures. In other words, it is the money left after all costs have been paid. In business planning, it is good practice to maintain a surplus. It is a contingency that ensures that the venture can withstand cost overruns.

Gross income for a business is the revenue from sales minus the cost of goods sold (in sales the cost of goods sold is referred to as COGS). Investopedia [15] explains that it represents the money earned from core operations, before accounting for other expenses. [16] states that net income is the bottom-line profit after all expenses and costs have been deducted from the total revenue.

Chapter 2

Business Plan for Cooperative Property Investment

While we discuss the concepts of cooperative management, we keep in mind that ultimately token holders participate in an investment. Typically, real estate investments are slow in returns, but provide stability through long-term contracts. This chapter presents a business plan to break even in the end. It is common for property investments to break even in a span of 5-10 years.

Ultimately, the purpose of buying a warehouse token is to see a return after the property is leased. Due to the nature of the exchange, it is not always obvious, but this is an investment. The token holder is considered an investor. For this reason, a business plan should be available to the investor before buying tokens.

We summarize the property business plan in the following list:

1. Find and mint an abandoned building.
2. Sell fractions of the ownership of the building to raise capital.
3. Use the money raised to renovate the building and to cover expenses.
4. Lease the building.

Table 2.1: The property business plan simplified.

In this chapter, we will expand on the above to create a more detailed step-by-step financial plan. We will discuss project expenses, token pricing, rental income, return-on-investment,

and risks. To conclude the business plan, we analyze an application of this financial model using a case study.

2.1 Step-by-step Plan

[2] explains that since the target property is typically an abandoned warehouse (or building in general), we expect ownership to be unclear. The first step Wareblock makes in a case like that is to negotiate a deal with the rightful owners of the building (which can be the state, a bank, or private entities). In exchange for complete access and lease rights, Wareblock offers them share tokens.

Once ownership is resolved, the building can be minted and becomes a blockchain-based asset. This means that a token supply of our choosing is created and available for distribution through crowdfunding. The crowdfund rewards participants with a number of tokens corresponding to the amount they deposited.

There is a distinction to be made here between traditional crowdfunding and the one used by Wareblock. A crowdfunding process is a means to raise capital. Typically, people can show their support for a project by making donations. If that project becomes a product and the project developers start a company, the donors are rewarded with discounts or products. Wareblock instantly rewards donors with a token that represents a share of the property. The Wareblock platform is designed to use an all-or-nothing funding model for crowdfunding. This means that the investor deposits are swapped into DAI and stored until the project reaches the target goal. If the goal is not reached, all token purchases are refunded. The use of DAI ensures that investor deposits are not affected by volatility.

2.2 Expenses

The uptake of the building is the largest and most important expense. The next step, after securing the right of access, but prior to minting the property, is to contact multiple construction agencies and receive offers. The decision between different offers is left to the judgment of the management team.

This leads us to the next expenditure that has to do with project management while in construction. In the chapters 3 and 4 we refer to an outside entity, used to help realize the

decisions made by the investors. Here we refer to a different management entity that is placed in that position by Wareblock and not by investor decision. The expense is the salary of the project manager or the project management cost in general. Depending on the platform's scope, this may be managed internally or outsourced.

Other expenses involve platform costs and blockchain fees. Platform costs have to do with the operating costs of the Wareblock platform. Depending on the deal between Wareblock and property investors, compensation for the platform service can be one of the following: commission-based, subscription-based, or equity-based. Commission-based means that a flat amount is to be transferred to Wareblock in exchange for its service. Subscription-based means that for as long as the Wareblock platform is used for property management, a monthly subscription must be paid. Equity-based means that a share of the generated tokens is transferred to Wareblock upon creation.

Blockchain fees refer to the gas costs of minting the property, token creation, and crowdfunding. Any other blockchain-related expenses are to be included in this number. In chapter 4 we show how these numbers can be calculated in detail.

Expense	Description
Uptake	Full cost of renovation and repurposing the building.
Manager Salary	Project management and coordination.
Blockchain costs	Gas fees and costs for minting, token creation, crowdfunding.
Platform fees	Operational or service cost of using the Wareblock platform.

Table 2.2: Expense Report

2.3 Calculation of Budget & Token price

The sum of all expenses should be the budget needed for this project. [2] poses a question as an open issue. The question is if token holders would be required to invest more capital in the future. In other words, what happens if the budget is not enough? To address this, we can add a small surplus to the budget for a case of emergency. The surplus can be added at a rate depending on the volatility of the project. This is a fail safe that ensures liquidity in a

potential crisis. Once construction is complete, the surplus can be returned to investors. The following is a formula for the calculation of the budget.

$$Expenses = UptakeCost + ManagementCost + BlockchainFee + PlatformFee$$

$$Budget = Expenses + SurplusRate * Expenses$$

Once the budget is calculated, we can determine the token price by dividing to the total token supply. We have to figure out what percentage of the token supply is available to the crowdfund before calculating the token price. There is a small percentage reserved for the platform and a small percentage reserved for the original owner. If either of those deals does not involve tokens, we can set the corresponding percentage equal to 0 in the following formulas:

$$OwnerTokens = OwnerPercentage * TotalSupply$$

$$WareblockTokens = WareblockPercentage * TotalSupply$$

$$AvailableTokens = TotalSupply - OwnerTokens - WareblockTokens$$

$$TokenPrice = \frac{Budget}{AvailableTokens}$$

Once crowdfunding is complete, the tokens will be in the possession of investors. This token serves multiple functions. It serves as a share of the property. The token supply is equivalent to the full ownership of the building. This means that the token has a value that can increase over time. This creates an opportunity for a market that involves property tokens. The token can also be used to collect profits from property leasing. In addition to financial benefits, the token also allows the holder to participate in management and decision making.

2.4 Lease Strategy & Timeline

After construction is complete, the property is leased for a fixed period. Our business plan should include a brief construction period at the beginning and a lease period for the rest of the timeline. The timeline is created, depending on the projection of the construction company regarding the duration of the renovation and conversion of the building. The lease period is set depending on how soon the investors wish to see returns. Since this is a project

with low investor input, we assume that they expect a short-term investment. On the other hand, this is a lease plan that involves construction, so we expect it to have a substantial duration.

	Research, Negotiation with owners, Communication with construction agencies, Minting
3-6 Months	Crowdfunding: **Investors enter here**
1-2 years	Construction and preparation of the building.
3-more years	The building is leased or available for lease.
	Break-Even Point
Post-Plan	The investors can decide their property's future.

Table 2.3: Project Timeline

To calculate returns, we consider multiple scenarios for building occupancy. We consider three scenarios: Pessimistic, Realistic, Optimistic. Before approaching property owners, Wareblock researches the property and the market to propose a plan based on market interest. Taking this into account, it is fair to assume a yearly occupancy of 60% or more.

Scenario	**Average Yearly Occupancy**
Pessimistic	60%
Realistic	80%
Optimistic	95%

Table 2.4: Occupancy Scenarios

2.5 Operating Costs

In addition to the expenses of the project in general, there are some monthly costs associated with the operation of the building. Maintenance costs and emergency repairs are unlikely to appear in the first few years of renovation. Although it is good practice to plan by setting

up a maintenance fund for an emergency. Depending on the nature of the deal between investors and tenants, we must consider utilities. Many businesses require more power, thus a more expensive electricity bill. Insurance is important to ensure that in case of an emergency, investors' assets are secure. Property management is thoroughly discussed in chapter 3. It is a person or entity that helps realize the decisions made by token holders. The management cost has been included in the budget as a different expense.

Expense	**Description**
Maintenance	Emergency fund for repairs.
Utilities and Services	Monthly utility and service bills
Insurance	Coverage for property damage
Property Management	Investor-made decision handling internally or externally
Legal	Contracts, filings, compliance and documentation

Table 2.5: Operating Costs

2.6 Goal: Break Even

A proper business plan will allow investors to break even at the end of the lease period. At a selected occupancy rate for the entire lease period, the gross income after subtracting operating costs should match our budget.

$$GrossIncome = Lease * 12 * Years * Occupancy$$

$$NetIncome = GrossIncome - OperatingCosts$$

To break even, the net income must be equivalent to the budget.

$$NetIncome = Budget$$

So:

$$Lease * 12 * Years * Occupancy - OperatingCosts = Budget$$

By rearranging, we get:

$$Lease = \frac{Budget + OperatingCosts}{12 * Years * Occupancy}$$

In a cooperative property investment, the goal typically includes two parts. The first part is to break even through the lease income. By calculating the operational costs, we can ensure a viable monthly lease. That way we can break even at the selected year. Post-plan, investors can decide how they will profit from their token. It is up to them to decide between an exit strategy or keeping the token and continuing to collect the monthly dividends. Another possibility is for the building to be appraised and sold. In this case, the investors will be compensated in exchange for their shares.

2.7 Risks & Mitigation

An investment is not without risk. Many risks might involve a project of this kind. The first is its nature. Renovating a building that is old and abandoned is a difficult task. The first risk has to do with crowdfunding. [2] addresses this issue. Each investment is stored in DAI on-chain. If crowdfunding does not meet its goal, the original investment is returned to the investor. In this way, the platform ensures that the investor is not at a loss.

To account for low occupancy, there are two solutions. The first, which we cover in 2.4, is to create occupancy scenarios. Using a realistic scenario, we can make safe calculations to avoid being overrun. The second solution is research. The property should accommodate a business that has both space and potential to grow.

Calculating a small surplus is a great way to deal with emergencies. This means calculating a budget that covers a surplus and setting a small fund to cover emergency expenses. Finally, the plan provides an exit strategy. This means that besides rent collection and decision-making capabilities, there is more to this token. After the property is appraised, each investor's share can be sold. This creates an exit strategy for investors and an entry for interested parties.

Risk	Mitigation
Underfunded Crowdfund	Refund in DAI if goal is not met.
Low Occupancy	Plan using a pessimistic or realistic scenario.
Cost Overrun	Use a surplus and set up a fund in case of emergency.
Token Liquidity	Post-plan building appraisal, the token can sold.

Table 2.6: Risks and how to mitigate them

2.8 Limitations of the plan

The final concern of this business plan should be its limitations. Unlike scalable businesses, in the case of property leasing, there is only one source of income, rent. This is a common concern when dealing with real estate investments, so it is important to let investors know why this does not affect their profits. The way to deal with this is to offer stable long-term leases. This ensures a long-lasting period of guaranteed returns. It is also important to note that this is a plan that aims to break even at the end. This means that in the end the token holder has collected their initial initial investment through monthly dividends. At that point, they can choose to continue earning monthly or sell their tokens.

Chapter 3

Literature Review

Since the objective of this study is to continue the work found in [2], it is in our best interest to examine the view of the authors on the matter. We keep the first part (which is associated with crowd-funding) as is and continue to the latter, which is the management part. For investors to manage their property, the author proposes a decentralized REIT or dREIT.

The decentralized Real Estate Investment Trust (REIT) would be the means used to participate in decisions and receive profit, should the property be leased. The author proposes the DAO scheme for governance, which is an investor proposal and decision-making through voting. Each voter would hold voting power equivalent to their percentage of shares.

In the proposed scenario, voters are encouraged to make decisions that benefit the property by improving its infrastructure or increasing its value, thus increasing their returns. This might not be enough to incentivize all parties, so the author proposes another way to do so, which is with token vesting. This means that, with time, the voting power of long-time investors would increase. To gain perspective, an investor would be far more eager to make decisions that last in the long term but also actively participate in the voting process.

3.1 Blockchain & Management

[17] addresses a problem, also mentioned in [2], that due to its high financial barrier, an individual might find it difficult to invest in real estate. The authors present Bitland, which is a real estate platform that allows fractional ownership. We notice that while property management is mentioned [17, p. 1], there is no implemented management part. This tells us that it

is left without but in need of implementation. Naturally, some questions appear regarding the nature of the management. Would an investor manage the property or an outside participant, and in what way?

We will not expand on crowdfunding, although it is worth mentioning the categorization made in [18, p. 1], which classifies crowdfunding into 3 categories. These are crowd-based, prize-based, and equity-based. The first is the traditional crowdfunding process, in which individuals make donations to support a cause or fund a product they wish to see on the market. The second differs only in the fact that that donors are promised something in return, for example, a product the company makes. The third is equity-based crowdfunding, in which donors are rewarded with equity in the hope of profit in the long term.

In [18, p. 4-5], the authors present an equity-based crowdfunding platform where donors have control of the funds through voting. They propose to include an extra role that manages the project and makes decisions, called the program manager. In case the program manager wishes to spend any of the funds, they must have the investors' permission, which is granted to them by vote. This has been discussed in [2, p. 12], where the authors suggest the use of a beneficiary managing the property. We present literature that expands on this idea in the following sections.

3.2 Governance

3.2.1 Governance In General

Regarding decentralized governance systems, [19, p. 1-4, 12-14] presents a systematic review of the literature until 2022 that focuses on the governance of smart cities. The review focuses mainly on how different technologies can be combined with blockchain and smart contracts to create a truly decentralized and self-governed smart city. To conclude their work, the authors endorse blockchain as a means for truly decentralized governance, explaining that it enables transparency and efficiency in decision-making. We can confidently say that the authors' view on the matter has been heard throughout the community since the paper was cited a total of 97 times.

Governance has many different approaches that vary according to the governed entity. Using blockchain for governance is an idea inspired by the natural decentralized and communicative nature of blockchains. [3, p. 1-2] analyzes six principles for the governance of

blockchain networks. We study these principles with a level of abstraction, meaning that while the governed entity is different, the objective remains the same. We can focus on the common elements, such as the way investors and other benefactors make decisions that affect the blockchain's future.

In the proposed framework, principle one refers to the level of centralization, in other words, the degree to which a single entity controls the decision-making process between the investors. The authors consider three degrees of decentralization, low, medium, and high, which respectively translate to single source of authority, multiple authorities, and no clear authority. Our case of managing the property could contain such levels of third-party intervention. Considering that a single person making decisions is definitely not the case, the low level of decentralization is immediately discarded. The highest level is to have investors make decisions without supervision or third-party involvement. Lastly, the medium level it to have an individual or an authority making proposals or even filtering out the ideas that lead to loss.

During this thought process, it is important to keep in mind that in our case, the medium level of centralization can also be achieved by voting power. This means that a single investor can really affect the voting process by holding a high percentage of shares. Although we deem that a single individual holding the majority of shares is a scenario that goes against the nature of what we are trying to achieve. The Wareblock project focuses on uptaking and utilizing abandoned buildings that no investor showed interest in buying. Hence, it is highly unlikely that a single individual will have immense voting power from the beginning. However, an investor can buy more shares at a later time, which is a cause for further investigation.

[20] explains the governance differences between a high degree of decentralization and a low degree. A completely centralized governance structure means concentrated authority. This can be harmful to platform participants because owner interests might collide with their own. On the other hand, in a completely decentralized environment, all platform participants have governance rights. This can affect the speed of decision making since conflicts between community members may appear. The findings suggest that a moderate level of centralization can provide stability and efficiency in the governance process.

Instead of analyzing the degree of centralization, [21] explains that the primary governance methods are three. These are founder-based, council-based, and expressive representation. In founder-based governance, the the authority is concentrated on a single individual. That individual has the deciding vote on every decision. Council-based governance refers

to a group that typically consists of founders, early adopters, and contributors. Representative expression is to have community members vote directly or by delegation on governance matters.

Of the five remaining principles [3, p. 5-7, 12], the second, third, and fourth are the most substantial. Principle number two states that stakeholders need to receive aligning incentives, which is also mentioned in [2, p. 5] and [4, p. 3337]. Principle three states that in order to build trust, the voters' reasons and arguments should be shared, making the decision process transparent. Number four states that accountability should be enforced, meaning that stakeholders would answer for their decisions. The stakeholders' decisions immediately affect property value, so accountability applies directly.

3.2.2 Blockchain Governance

In [22, p. 1-5], the authors explore the way Ethereum is governed, targeting 3 main focus points. How open and transparent is governance itself, how the decision-making process works, and to what effect does governance affect token prices. The latter, we deem irrelevant to our work and focus on the first two objectives. The study was published in 2024, so we can get a recent view of how the Ethereum blockchain is governed.

The authors' findings showed that Ethereum governance is both open and transparent. It works by making proposals and discussing them before deciding. In the beginning, individuals share their proposals, which are called Ethereum Improvement Proposals (EIP). Then, the proposals are openly and thoroughly discussed in public forums. This means that formally, Ethereum operates without a central authority, and ideas and proposals emerge naturally. However, the study showed that although more than 600 people have made EIPs, most of the actual changes have been proposed by a small group of people. To be more accurate, 20% of the total proposers are responsible for more than 60% of all EIPs. The next stage is community discussion, which takes place online. Typically more than a 1000 members join the discussion. Research showed that EIPs proposed by popular individuals receive more attention. For a proposal to be implemented, it must receive support by rough consensus. Proposals that make changes to the Ethereum core move to a second discussion stage before implementation. The second discussion is conducted by developers, and the average participant number is around 21. Lastly, findings show that only one third of EIPs proceed to development and proposals made by popular authors are more likely to be approved.

From the above, we arrive at the conclusion that a small group of individuals is responsible for the majority of changes in Ethereum, despite its community being open and permissionless. That is why it falls into the category of medium degree of decentralization, as described in [3, p. 5]. There is a part of the Ethereum community that is responsible for the majority of the actions taken. This has also been discussed in [23]. Finally, Ethereum governance is built on trust. Almost 89% of EIP authors and 79% of developers are transparent with their identity. The authors of [2, p. 12] are concerned with investor KYC, but as we can see, members of the Ethereum community provide identification willingly.

[24] proposes a fair governance model, explaining that the permissionless nature of blockchain does not promote fairness. The proposed model is based on seven principles. The first principle is active participation between all parties in an honest and limitless manner. Secondly, the model accommodates laws. The laws are presented in the form of platform rules that treat everyone equally. The third principle has to do with efficiency. The model suggests that resources are used efficiently and without waste. Transparency is the fourth principle. All governance processes should take place in an open and transparent manner. In addition, information about governance matters must be available to everyone. Principle number five states that a fair governance structure must be responsive. In this way, members feel valued and are more likely to participate in the governance process. The sixth principle dictates that the model should be consensus-oriented, to ensure that all individual opinions are heard. The last principle states that the model should hold decision-makers responsible for their actions.

3.2.3 Governance Tokens

In [2, p. 4], it is suggested that when a property is minted, a number of tokens are generated. The total supply of tokens represents complete ownership of the property. Another suggestion is that one divisible token can be generated per minted property. Such tokens are called fractional tokens, and their fractions are what an investor would receive, should they take part in the crowdfund. The author suggests ERC-20 tokens, which by specification can be divided into fractions and their fractions can be distributed.

As stated in [25], every system has different rules. This means that the platform's rules dictate what privileges holding a token, or a number of tokens, grants. Governance tokens are fractionable tokens that allow token holders to vote, to change specific parameters of the

system. Such tokens are ERC20Votes, [26], which is an extension of ERC-20. An important note is that governance tokens are restricted in the control they provide. This is natural considering that our objective is to use governance tokens to cooperatively manage a property and not the platform itself.

3.3 Decision-Making

It becomes clear that we should shift our focus towards decision making. Management should be achieved through group decision making, because fully automated management is not able to handle difficult and complex tasks [4, p. 3338], [2, p. 5].

[4] introduces an organizational model named Crypto Management, which aims to improve decision making by addressing three main problems: lack of reliable data sharing (data asymmetry), difficulty in establishing trust (trust asymmetry), and time-related obstacles that cause delays (time asymmetry). Tackling these problems is of great interest to our work since challenges appear when individuals collaborate to govern or manage an entity.

The authors present a solution which involves combining five technologies and techniques to tackle the above-mentioned problems. The first is to use Blockchain technology. Doing so ensures that all the information each individual needs is easily accessible, transparent, and trustworthy. The hierarchical structure of a DAO fits in this instance and embodies the second method, in which members participate equally in decisions. In the scenario presented by the authors of [2] each individual has a say equivalent to their share of the property. However, the idea remains the same that decisions are made through community participation and vote.

Smart contracts embody the means of execution and are another technology listed in the model. As mentioned above, the use of smart contracts helps automate any task decided by voters, thus minimizing time costs. To ensure that all voting parties have access to the same information, the authors suggest data federation as the next technique. Data federation means data, openly circulated in the network. This ensures that all voting parties trust each other and the data provided to them.

The last technique is to use decision incentives. More specifically, NFTs and standard cryptocurrencies are the main incentives that are used in crypto management. It is worth mentioning that in our case, along with financial compensation, a very strong incentive is

the objective itself. This means that the well-being of the property is an issue that concerns all parties and can only be achieved by efficient management. This leads to better payout, as well as a longer life expectancy for the property. [2, p. 5] explains that investors aim to make decisions that increase the value of the property. A higher property value translates into higher income for the investor.

Name	**Model**	**Voter Incentive**	**Data Federation**
Crypto Governance	DAO	Cryptocurrency, NFTs	Data circulates the network; all parties hold the same trustworthy data
Wareblock dREIT	Undecided	Cryptocurrency, long term increase in property value, increase in voting power and profit	Undecided
Governance Principles	6 Governance principles; different voter roles	Vary based on role, mostly monetary or means for improvement	-

Table 3.1: The table shows how the proposals from [2] and [3] would fit the one proposed in [4]. The **Technology** and **Means** are the same for the three frameworks, which are blockchain and smart contracts respectively.

At this point, we return to the governance of blockchain networks, since decision making is crucial to their management. [23] analyzes the decision-making of blockchains in two axes. The first axis is on-chain/off-chain. This means actions taken within or outside the blockchain system, as explained in [23, p. 2]. A Gemini team, in [27], note that blockchain governance used to take place completely off-chain at first. Stakeholder coordination took place through means such as conferences, forums, and email. [28] provides a description of the two governance mechanisms. Off-Chain governance methods include governance processes that do not take place within the blockchain network. This includes conferences, forums, social networks etc. On-Chain governance is the exact opposite. The governance-related functions are

provided by the platform and take place on the blockchain. Typically, to participate in the process, one must hold the blockchain's native token.

An example provided is that of MakerDAO, the organization that has published and currently manages DAI. DAI is a stable-coin that is bind to the price of the US dollar. To participate in the governance process, an individual must own an MKR token, which is a MakerDAO native token. This allows them to access forums to participate in discussions associated with governance, and to vote on proposals. At this point, we remind the reader that in the case of property management, the On-Chain/Off-Chain matter is important. The objective is to have a group of property owners, make decisions regarding their collectively owned property. Since we are studying how to utilize blockchain technology to achieve that, it is natural that the management will take place On-Chain.

The second axis is community-driven/institution-driven. Community-driven means that a broad community of voters is responsible for the actions taken, while institution-driven means that a central authority, such as an institution, makes decisions. The authors present a way to analyze decision making in blockchain networks, by creating a framework that breaks down blockchains in five dimensions. The dimensions are decision makers, incentives, access, coordination, and approval conditions. By doing so, they are able to create an axis system of the two previously mentioned axes, which makes clear that no blockchain decision-making system is purely on-chain or off-chain and institutional-driven or community-driven. Decision making can be described as a spectrum between them. This means that a blockchain can have most of its actions happen within its system but allow some actions to happen outside. The same applies to institution-driven/community-driven.

[29] proposes a DAO framework for governing non-profit organizations. The authors express their concern that token-based voting is harmful for a collaborative governance system, so they propose that a reputation-based system be used instead. The reason is that token-based voting can lead to concentration of power. [29, p. 5] states that in a significant number of decentralized finance platforms, more than half of the total governance tokens are held by a few individuals. The authors deem that to be a problem and their solution is to have only individuals who have gained a certain level of reputation, are allowed to participate in decision-making. This does not fall too far from the case for some platforms. An example is that of Ethereum, which we reviewed in the previous section, in which proposals made from distinguished individuals are far more likely to reach the implementation stage.

[23, p. 12-13] reviews blockchain platforms to analyze who the real decision-maker is in each one. The authors explain that blockchain's nature is contrary to having a single individual making decisions. That is the case for the transaction level, so they examine whether this applies to the bigger picture as well. Regarding the Tezos blockchain, the findings show that while the decisions are made by the community, there is a single entity that exerts influence and determines the final decisions. In Ripple and Stellar, two other blockchains that they review, the decision-making process is in the hands of a central entity, an institution. On the other hand, the Decred network has all of its processes happening on-chain and it relies on having an active community with strong participation. The platform strives to preserve this state because a governing minority would lead the platform to its doom. The last of the platforms, the authors review, is Bitcoin, which is a community-driven blockchain, with most of its decisions made On-Chain.

The authors conclude that the decision-making method finally has to do with the purpose of the blockchain. This means that since Bitcoin and Decred are decentralized currencies, they both rely on community participation and do not focus on whether their processes take place on-chain or off-chain. Meanwhile, the target audience for blockchains, such as Stellar and Ripple, are banks, hence the need for a strong and centralized decision-making approach. Collaborative ownership and management of a property is a decentralized governance matter. This prohibits having a single centralized entity make decisions. It is our goal that investors have control over their funds and share of the property. This raises the question: to what extend should a centralized factor exist? For example, there could be an individual whose purpose would be to research and present only the few proposals that have ensured profit. This relieves a great degree of difficulty in decision making while preserving investor control over the property.

3.4 Decentralized Autonomous Organizations & Governance

[2, p. 5] suggests the Decentralized Autonomous Organization (DAO) scheme for governance, because it is a widely used practice. DAOs are collectively-owned, internet-based organizations governed by token holders. Their defining feature is that there is no central authority. Decisions are made by the community and are realized with the use of smart contracts. [30, p. 6] explains that the use of DAOs has benefits, such as improved decision

making, freedom of speech and fairness.

[30, p. 7] examines DAOs and analyzes how their governance works. It also studies the way improvement proposals are made and how the community decides in the end. The study shows that 70% of the DAOs use relative quorum voting, while 19% uses weighted voting. The study also explains that some voting practices discourage participation. For example, various platforms require a minimum balance of tokens to submit proposals. For proposals that suggest changes to the platform, it is a common practice to request implementation code upon submission of the proposal. The above facts lead to the conclusion that wealthy individuals or platform developers are a step ahead in the voting process. This means that more often than not, a DAO falls in the medium decentralization category from [3, p. 5].

Another finding of [30] is that DAO voting processes can be executed on-chain or off-chain, as we discuss in the previous chapter. On-chain voting means votes are executed as transactions and are recorded directly onto the blockchain. A drawback of on-chain voting is that voters are required to pay gas fees, but the outcome of the vote gets automatically implemented using smart contracts. To solve this problem, communities use vote delegation. This means that a person without sufficient funds to submit a vote can choose a delegate to vote on their behalf. For example, two individuals who support the same proposal can have one of them vote with twice the voting power but half the cost. [2, p. 7] proposes gas station networks to reduce costs. In off-chain voting, votes are not submitted as transactions, but stakeholders are prompted to sign their vote with their wallet. A different means is used to vote, which is usually web-based. Afterwards, the implementation process begins.

[30, p. 7] also performs a case study of a platform called Compound. Compound is a platform built on Ethereum that lets users borrow money from a pool of digital assets. In 2020 the platform administrators introduced a token named COMP and replaced themselves with a community governance system. The reason for this action was to replace central authority with a decentralized governance system. Holding a COMP token translates into voting power, used to govern the platform. By design, COMP has no expectation of profit to avoid exploiting it for financial gain. To date, Compound's governance is performed on-chain using Governor, an open source system, developed by Compound. Governor offers great functionality for proposals. Examples of its features are proposal initiation, cancellation, queuing, and execution and for votes, vote casting and delegation.

A drawback is that vote casting costs gas, so users either use delegates or use other means

to vote. The way this works is by discussing proposals in forums. We notice many similarities between Ethereum governance and the DAO model [22]. In fact, the creator of Ethereum, Vitalik Buterin, endorsed the idea in 2014 [30, p. 6]. Compound is an open-source and available system for use. This means that in our project we can adapt it to our needs and use it as a means of governance.

An approach that should be included is that of Tally, [31], a protocol designed for governance. Tally gives stakeholders the ability to maintain voting rights to a DAO but still utilize their governance tokens' value. The way it works is that a shareholder can stake their governance token, without losing their voting power, and in return they receive a Liquid Staked Token (LST). The LST is equivalent to the governance token and has financial value instead of governance value. The platform also allows for vote delegation. As a means to motivate participation, the platform revokes voting rights from abstinent community members. Their voting rights are redistributed to the community. In the case of property management, the governance token has value as a fraction of the property ownership title. We will not expand on Tally though it is important to note voting rights redistribution, as a form of voter incentive.

[5] presents a way to govern DAOs using the eight principles for governing commons by Elinor Ostrom. This can be achieved by considering DAOs as digital commons. According to Ostrom ([5, p. 2]), common pool resource systems are characterized by low excludeability and high subtractability. This means that such systems require participation, but misuse by a single individual can degrade the system. In the case of property management, Ostrom's principles can be applied by analyzing the way they are applied for DAOs.

The first principle ([5, p. 4]) states that it is essential to establish clearly defined boundaries. There must be clear roles and responsibilities for members of the community. In property management, the investor is responsible for making decisions for the property. We have mentioned the possibility of a third party making proposals or moderating the property owners in some way. Should that individual hold any voting power? And if so, what would that be? Would they be able to veto against a proposal, should they deem it harmful for the property and its value? The second principle explains how the rules of the DAO should align with the environment of the system. It states that to manage is a property, the system should have built-in rules that encompass property management and allow it to run smoothly.

The third principle is to collectively make decisions, a practice that is applied in the ma-

jority of DAOs. The fourth and fifth principles have to do with the way community members act and how the rules are enforced. The fourth principle suggests monitoring, which means transparency between community members. The fifth proposes graduated sanctions, which means using the public ledger to quickly locate dishonest behaviors and penalize individuals who try to cause harm to the community. The sixth principle refers to conflict resolution, which is important to quickly resolve arguments. Finally, the seventh principle has to do with enabling the government to provide laws that support the DAO infrastructure. Property management requires different laws, but that is something we will leave to the legislators. We will not go into details regarding the eighth principle, since it has to do with large-scale applications.

Ostrom's Principles	**Property Management**
1. Clearly Defined Boundaries	Clear rules regarding the investors and the third party moderator, should there be one
2. Aligning Rules and Environment	Proper infrastructure of rules and roles that matches the management of a property
3. Collective decision-making	dREIT - System that allows all investors to vote upon the property's future
4. Monitoring	Having investors make decisions transparently; rule enforcement
5. Graduated Sanctions	Penalizing individuals acting maliciously; Using ledger to quickly locate rule violations
6 Conflict Resolution Mechanisms	Decisions taken by vote; individuals must conform to the majority's decision.
7. Government Regulation	Reforming of current laws to accommodate fractional property management

Table 3.2: The table shows how Ostrom's principles for governance [5] could be applied in the case of property management [2].

Both [32] and [33] endorse Ostrom's principles, each in a different way. The first expands on the original principles by presenting a framework to govern digital commons with the help of blockchain on a global scale. Stern's critique of Ostrom's principles is used to identify six affordances provided by the blockchain that directly address the commons governance issue.

The first is tokenization, which plays an important role in governance. Tokens represent ownership and voting power and are the foundation of any governance model. The second is formalization and enforcement of the platform rules. These rules exist in the form of smart contracts and are enforced automatically as part of the system. The third and fourth affordances refer to enhancing decentralization by creating strong organizations, such as DAOs, and by building infrastructure for collaboration platforms, to govern commons. The fifth affordance has to do with transparency, and the sixth with trust. As we have mentioned in previous chapters, blockchain systems enable transparency through open, tamper-proof data and secure interactions between parties.

[33] combines Ostrom's principles with blockchain governance literature to propose a conceptual DAO framework for commons governance. The proposed DAO scheme consists of 3 elements. The governance structure, the enabling technology, and the community. [33, p. 8] explains it as so; Commons governance is a result of the combination of the three elements. The governance structure, a constitution, as it is characterized by the authors, is the organizational scheme, that is backed by rules and processes and is the base to support a community. These processes would include monitoring to ensure nothing would get out of hand, and a court to penalize responsible parties if it does.

The enabling technology is the set of technological means that allow the forming of a community under the above-mentioned rules. It consists of a platform easily accessible by individuals and completely decentralized to eliminate the factor of a central authority. Smart contracts are the means of enforcing constitutional rules. Lastly, there is the community, which by design allows forming smaller communities. As part of the constitution, participation is required so that everyone can collaborate and decide upon matters regarding the commons.

3.5 The case of Governor

In this section, we are going to analyze the Governor contract. As discussed in a previous chapter, Governor was implemented as a means of governance for the Compound DAO, in an effort to remove central authority from it. This allows COMP token holders to propose, vote, and implement changes regarding the Compound protocol, [34, p. 10]. Governor Alpha is the original governance contract implemented, while Governor Bravo is the second iteration.

Governor Bravo expands Alpha and addresses some of its limitations. We will focus on the workings of Governor Alpha and Bravo, as well as the OpenZeppelin Governor, which is an effort for improvement of the original.

3.5.1 Compound Governor

[34, p. 12] explains that Compound's Governor consists of 3 main functionalities. These are the Proposal Mechanism, the Voting Process, and the Timelock. The Proposal Mechanism allows holders of the COMP token to create proposals. To be more specific, a proposer should be an individual with a minimum of 1% of the entire COMP supply delegated to them. Afterwards, follows the voting process, in which voting power of each individual is proportional to the number of tokens held, or delegated. The voting parties can vote for or against until a certain number of votes for either side is reached. The last function is the timelock, a mechanism that queues the voted-for proposal's actions for execution. Timelock stalls for a specific time period, thus leaving time for reviewing and preparation.

[35] explains that until 2022, the minimum amount of COMP required to submit a proposal was 100,000. The tokens can be held by ownership or delegation. Although this was an outrageous amount of COMP, it never stood in the way of effective governance and collaboration, as explained by the Compound team. Later, after discussion, Compound lowered that threshold. The change was proposed, voted, and implemented. The proposals were divided into two categories. Any user with 100 COMP in their account can create autonomous proposals. These become governance proposals if they reach the above delegate threshold. Since this article was written in 2020, we notice that these numbers have shifted, probably adjusted by the platform upon vote. Currently, any individual holding 100 COMP can submit an autonomous proposal, and it becomes a Governance proposal upon reaching 25,000 COMP delegated. When a governance proposal is made, a two-day review period occurs; then, the voting period begins.

The voting period is 3 days. In order to proceed to the next phase, a proposal must agree with the majority and at least 400,000 votes need to be cast. Individuals who opt for vote delegation select the account of their preference, and their voting power is transferred to that account. In the first implementations of Governor, a voting party could choose between for or against, while in the latest, the option to abstain from voting has been granted. Should the circumstances not be met, the proposal gets canceled and the 100 COMP locked by the

proposal creator are returned to them. If the proposal is voted, the Timelock queues it for execution. Timelock allows a two-day period for the community to review the proposal and prepare for the change.

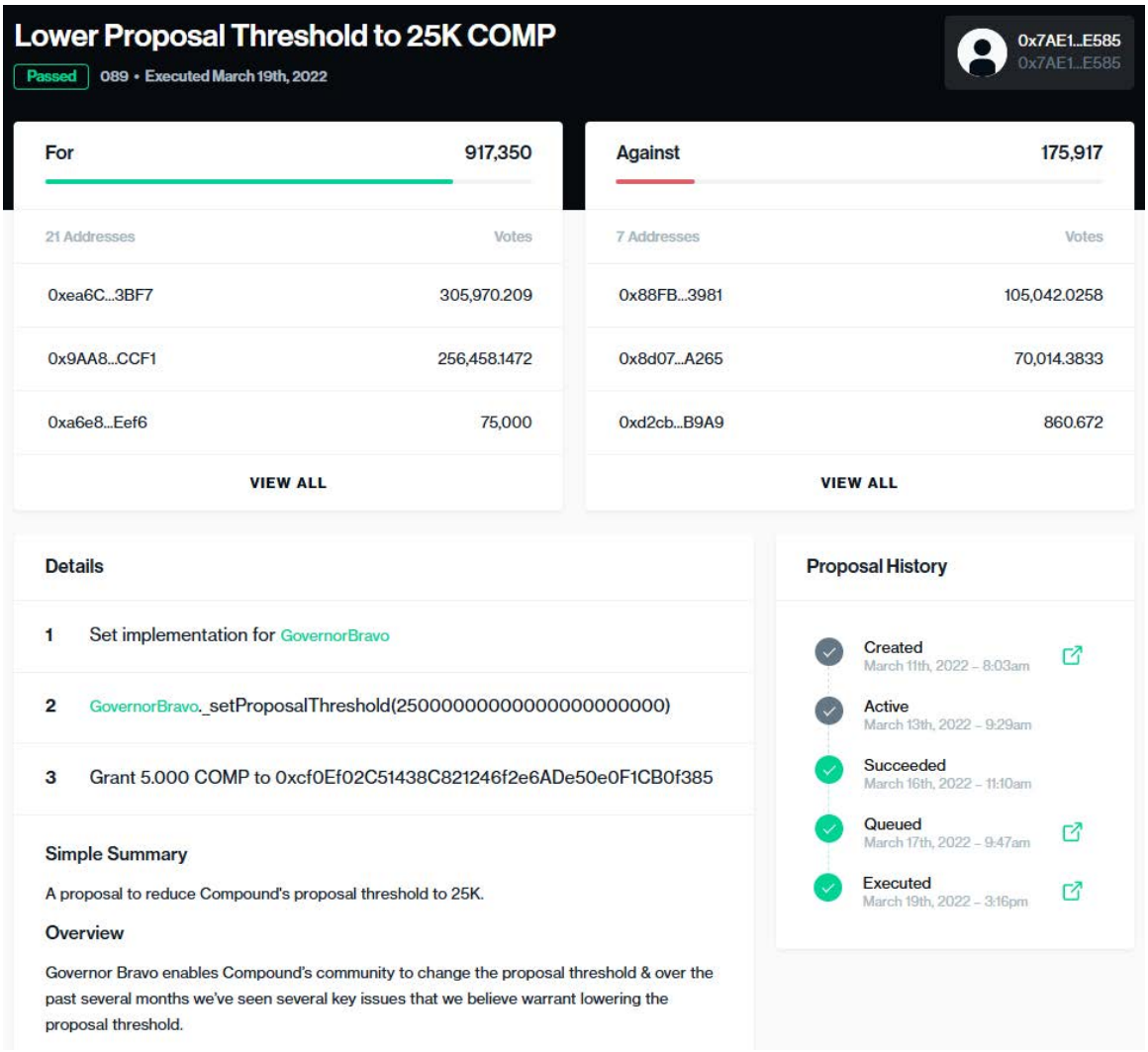

Figure 3.1: Compound Governor Proposal: The image shows the official Compound governance proposal to lower the proposal threshold to 25k COMP, openly available at [1].

Figure 3.1 shows the initial proposal that was made in 2022, to lower the COMP threshold from 100,000 to 25,000. The proposal passed with 917,350 votes for and 175,917 votes against. Listed, under each vote are the voting parties. As we discussed in a previous chapter, a commonly used practice is for voters and delegates to have their identity public. This enhances trust, as it provides a more complete and thorough understanding of opinions and views. The figure also shows the details and description of the proposal. On the right, there is a clear history following the exact time periods of the 3 phases. Not present at the image, but available should anyone visit the proposal website, is a full proposal description and forum,

and also a simulation of the running code for testing.

When a proposal is made, it has already been implemented by the proposer. [36] explains that a Compound governance proposal is a piece of executable code that will be executed automatically as soon as the two-day period passes. The proposal is submitted using the **propose** function, [37], which by design includes parameters such as the new function signatures, their arguments, and the target account addresses. This means that the proposer is responsible for implementing their proposal. Not a trusted third party, but a member of the voting parties. For the case of property management, we could implement a collaborative management contract such as the one presented here. The investors would create and vote on proposals. The proposer, a member of the community would be the one to realize each decision. Vote delegation can be implemented as well. Should an investor not have the time to actively participate in the voting they can choose an account to make decisions for them.

3.5.2 OpenZeppelin Governor

[38] presents OpenZeppelin's implementation of Governor. While it seems to be in favor of the Compound Governor protocol, and while it has gained some popularity, it is difficult to adjust it to other platform's needs. As a result, this implementation of Governor is compatible with systems that work with Governor Alpha or Bravo. The contract offers the same primary features. These are proposal creation, voting, and execution. This implementation is somewhat a generalization of Compound Governor that can be applied to a wider variety of instances. To eliminate confusion, from now on, we will address OpenZeppelin Governor as Governor, and should we want to address Compound Governor, we will use Governor Alpha or Governor Bravo.

The differences in the two implementations lie in the details. Governor's users can make proposals, vote, and execute them. The nature of these proposals is for the platform designer to decide. The threshold to submit a proposal is adjustable as well as the time period in which the proposal will remain active, before the voting commences. The voting period is configurable, and so is the quorum for closing a vote. There is no longer a compulsory Timelock mechanism. The platform can be designed to fit any needs, and even have the proposals be executed immediately. Lastly, Governor allows for the configuration of voting method. While Governor Alpha and Bravo support only ERC-20 based voting, OpenZeppelin Governor is designed to support NFT-based voting and quadratic voting.

Feature	OpenZeppelin Governor	Compound Governor
Proposal Threshold	Customizable	25,000 COMP required
Voting Delay	Customizable	2 days
Voting Period	Customizable	3 days
Quorum	Customizable	400,000 COMP
Execution	Custom Timelock or Immediately	Timelock
Voting Mechanism	ERC-20 or NFTs or Quadratic	ERC-20 only

Table 3.3: Key differences of OpenZeppelin Governor and Compound Governor as explained in section 3.5

3.6 Micro REIT Contract

In this section, we review a piece of software uploaded to GitHub, as part of a university project, [39]. It was developed as a part of the ”Smart Contracts and Decentralized Blockchain Applications” course at the University of Basel. The software was designed to accommodate micro-REIT transactions along with property management. To elaborate, the contract enables fractional property ownership, tax-payments and government regulation, and tenant payments in the case of rental. The contract also accommodates automatic distribution of earnings to each shareholder, depending on their percentage of ownership. In the GitHub directory, the developers include a demonstration of the contract logic and have made a presentation video, showing a demonstration of its core features, [40].

The entirety of the code is located in a single contract, named realEstate.sol. The contract is deployed by the government or at least a governing agent. This means that there is a benefactor that oversees the shareholder and tenant transactions. The developers refer to this party as the government, and for the rest of the analysis, so will we. After the government deploys the contract, it is responsible for setting the tax rates, enforcing laws against bad actors and distributing funds. The hands-on management part is the responsibility of the main property owner who willingly mints their property. After the property is minted, the ownership of the property is represented by a sum of shares, which are all in the hands of the main property owner. Other accounts can purchase shares from the main property owner and become shareholders of the property. The rent received from tenants is distributed by the smart contract. The government receives its share, and the rest is divided among the shareholders according to their percentage of shares.

By design, each realEstate.sol contract is used to manage a single property. This means that the government must redeploy the contract for every property. Upon contract deployment, a fixed number of shares is generated (which in this instance is 100 shares; 1 share is equal to 1% of the property). In table 3.4, we can see the difference between the token used in the project and the ERC-20 token standard. In our benefit, the code can be easily modified to accommodate ERC-20 tokens. That way we can introduce a voting system like Governor, which is presented in the previous sections.

Feature	**Micro-REIT Share Tokens**	**ERC-20**
Fungible	Yes; 100 identical and equivalent shares	Yes
Divisible	No	Yes; 18 Decimals
Ownership Model	Percentage based	Balance Based
Governance	Owner makes decisions	No standard method

Table 3.4: The table shows the differences between the tokens used in the Micro-REIT Contract and the ERC-20 token standard.

At this point, we take a closer look at each role. The acting roles in this contract are 4; the government, the main property owner, the shareholder, and the tenant. As explained previously, the government is responsible for the deployment of the contract. They can add or remove stakeholders and seize assets if necessary. The government also collects taxes and using the **distribute** function, sends the corresponding dividends to shareholders. The main property owner controls the property. They are the one who approves who gets to be a tenant, they set the rent prices and control exactly how many months of rent should be prepaid. The main property owner is also a shareholder. The shareholder can offer their shares or buy shares from another shareholder. Should the shareholder hold at least 51% of all shares, they can claim ownership of the property. Upon receiving their profits, they can withdraw them on demand. Lastly, we have the tenant who needs to be approved by the main property owner and pays the rent.

The Real Estate contract is interesting for a number of reasons. The first reason is that it is based on Ethereum and is implemented in Solidity. This means that we can modify it and use it for a different project, in this case property management. By modifying property share tokens to implement the ERC-20 token standards, we can provide a larger number of shares. Also, we can use ERC-20 tokens to implement a voting system, or use one like governor. In

this way, the property can be managed collaboratively and not by a single individual.

3.7 Parking Associated Applications

[41] presents ParkCoin, a parking platform for the city of Ghent in Belgium, in cooperation with Ghent University. The platform allows users to register parking their car in a certain zone, by paying with PARK tokens. Park is the platform's native token and can be bought using Ether. The platform's interface allows users to select a parking zone and pay for their parking. A single contract deployed on the Ethereum network is responsible for registering vehicles and managing parking.

There are two main roles in the application. The admin is the owner of the contract. They have the right to change the tariff of a zone. They also control the price of PARK tokens. The second role is the user. The user can buy and send tokens to a selected address. They can also park by sending the corresponding token amount along with their license plate.

ParkCryption, [42] is a UK-based platform that enables worldwide parking using the blockchain. It offers an application that has two uses, finding parking spots and offering parking spots. Users can list their own, private parking space and receive dividends when it is used. On the other side, users who wish to park can use the listed parking spots, after paying a small fee in cryptocurrency. The platform incorporates two blockchain networks, the Binance smart chain and the Ethereum blockchain. Users can pay for parking using PARK tokens, which are ParkCryption native tokens.

[43] addresses the issue of urban parking in a similar way. It does so by presenting a platform based in India, called ParkSpace. The platform also features two roles, the individual looking to lease their parking space, and the individual who seeks to park their car. In contrast to ParkCryption, ParkSpace does not feature a native token. Users can rent a parking spot and pay using a MetaMask wallet.

[44] addresses the same problem in a similar way, but on a larger scale. They propose ParkChain, a platform where individuals can lease their unused space for a limited amount of time. In contrast to the platforms mentioned above, each parking space is represented by a unique token, [44, p. 7]. These tokens follow the ERC-721 protocol, which represents Non Fungible Tokens (NFTs). The tokens can be rented to customers for profit. Another key difference is that parking spot owners do not lease directly to car owners. Contractors

rent available spaces from their owners. Using that space, they create parking pools that car owners can use to park. Contractors enter a bidding war to claim and rent a parking spot to add to their pool.

[45] suggests a platform called CityPass. The platform is a city-wide application for a South Korean city, Bucheon. The city's residents and businesses can share their unused parking spaces. In exchange, they receive a portion of the parking fee directly.

Platform Name	Description	Location
ParkCoin	Municipality offers parking zones, users park using native tokens.	Ghent, Belgium
ParkCryption	Individuals offer parking spaces, users park using native tokens.	United Kingdom
ParkSpace	Individuals offer parking spaces, users park using standard cryptocurrency.	India
ParkChain	Individuals offer parking spaces, contractors lease them to create parking zones, users park using native tokens.	India
CityPass	Individuals and businesses offer parking spaces, users park using standard cryptocurrency.	Bucheon, South Korea

Table 3.5: Blockchain related parking solutions as presented in section 3.7

3.8 Literature Summary

Title	Year	Topic	Description
A blockchain based platform for the uptake of abandoned historical building	2019	Wareblock	Software
ERC20Votes.sol	-	Governance Tokens	

Introducing the Tally Protocol	2024	Tally	
Compound v2 Docs	-	Compound Governor	
How to setup on-chain governance	-	OpenZeppelin Governor	
REIT Smart Contract (Solidity)	2020	REIT Smart Contract	
REIT Smart Contract (Solidity)	2020	REIT Smart Contract	Video Presentation
The Modern Guide to Digital Governance	2024	Governance	Guide
Bitland - A Decentralized Commercial Real Estate Platform	2022	Real Estate	Platform Description
Decentralized Crowdfunding Using Blockchain	2023	Crowdfund	
Blockchain Technology and Smart Contracts in Decentralized Governance Systems	2022	Governance	Literature Review
Defining Blockchain Governance Principles: A comprehensive framework	2022	Governance	Framework
Reputation-based Decentralized Autonomous Organization for the non-profit sector: Leveraging blockchain to enhance good governance	2022	Governance	
Blockchain-Based Crypto Management for Reliable Real-Time Decision-Making	2023	Decision Making	
Analysis of the Potentials of Blockchain for the Governance of Global Digital Commons	2021	Governance	
Decentralized autonomous organization design for the commons and the common good	2023	DAO Governance	

Decentralized Crypto Governance? Transparency and Concentration in Ethereum Decision-Making	2024	Blockchain Governance	Review
Analyzing Decision Making in Blockchain Governance	2023	Decision Making	Analysis
Governing decentralized autonomous organizations as digital commons	2024	DAO Governance	
Decentralized Governance and Digital Asset Prices	2023	Governance	
Around the Block #13: On the value and risks of governance tokens	2021	Governance Tokens	Article
DeFi Solutions: Decentralized Governance Protocols	2022	Governance	
Blockchain Governance On Decentralized Networks	2023	Governance	
Compound Autonomous Proposals	2020	Compound Governance	
Compound Governance. Steps towards complete decentralization	2020	Compound Governance	
Decentralized Governance Mechanism in Blockchain	2023	Governance	
Decentralized Governance of Digital Platforms	2021	Governance	Study

Table 3.6: The table presents the entirety of the literature, categorized by genre.

Chapter 4

Experimental Results

4.1 Introduction

In this chapter, we review and analyze three projects. All projects are implemented in solidity. The first is the Wareblock implementation. The other two are closely related to the topic of cooperative management. Since we are looking for ways to expand the Wareblock platform, we analyze the following code to determine how we can combine the three. For the reader's ease, we have included in our code review only the important variables, functions, events, etc. To read a detailed description of the structures in the reviewed contracts, the reader is directed to the Appendix A. The review of the contracts spans the sections 4.2, 4.3, 4.4.

4.2 Wareblock

4.2.1 WarehouseToken

The Wareblock back-end consists of four different contracts. `WarehouseToken.sol` describes an ERC-20 token, `WarehouseToken`.

The token variables are:

`string public tokenURI`	is the token link in the Wareblock website.
`address private _owner`	is the account that deploys the `Wareblock` contract, therefore creates the token.
`address crowdsale`	is the account that initiates the crowdfund.

Table 4.1: WarehouseToken variables

The `constructor` produces an ERC-20 token. It uses the `_mint` function to send the entire token circulation to the account that deploys the contract. Because the token inherits the `ERC20` contract, it already includes standard ERC20 functions such as `transfer()`, `approve()`, `transferFrom()`, `balanceOf()`. In the same way, it already includes functions such as `burn()` and `burnFrom()`, because it inherits `ERC20Burnable`.

4.2.2 WarehouseCrowdsale

`WarehouseCrowdsale.sol` is the contract that initiates the crowdfund. The developer does not use the OpenZeppelin Crowdfund contract. This is to ensure that the refund is handled in DAI. That way, the original amount spent keeps its value in case the Ethereum price drops. The contract variables are:

`WarehouseToken private _token`	The warehouse that is sold at the crowdsale.
`address payable private _wallet`	is the account that collects the funds.
`address private _owner`	the account that deploys the contract.
`uint256 private _goal`	the crowdfund goal.
`mapping(address => uint256) private _deposits`	a map structure that maps account addresses to DAI amount deposited. This helps to keep a local record of received DAI and return DAI in the event of a crowdfund being canceled.
`Dai dai`	is an instance of the DAI token. This is used to handle the swap from ether to DAI.
`UniswapV2Router02 uniswapRouter`	is a router used to exchange tokens. Specifically, Wareblock utilizes the Uniswap router to swap Ether to DAI.
`uint256 private _daiRaised`	is the total amount of DAI that has been raised from the crowdfund.

Table 4.2: WarehouseCrowdsale Variables

There is also an event named `Tokens Purchased()`, that is declared here. It is used to keep transaction logs when purchasing `WarehouseToken`. The details stored in the log are the buyer's address, the address that receives the tokens, the number of DAI spent, and the number of tokens bought.

The `constructor` checks that the constructor arguments have been set correctly, using the `require()` function. Then it initializes the contract variables.

In the following table we can see the functions of the WarehouseCrowdsale contract which implement the main buy/refund functionality.

`buyTokens()`	first performs checks on the arguments. The arguments that must be passed are the address of the token receiver, the DAI amount that is needed for the transaction to proceed, and the deadline for swapping ether to DAI. The function handles all the necessary processes to convert Ether to DAI, using a Uniswap router. `_processPurchase()` is called to send the tokens to the selected account. The leftover Ether is sent back to the account that called the function. Then, a `TokensPurchased` event is emitted.
`claimRefund() public`	is used to refund the deposited funds to the account that made the deposit. The `WarehouseToken` that are in their possession are burned, and the DAI they had deposited originally is returned.

`beneficiaryWithdraw() public`	Using the `Dai` external `transfer()` function, it transfers the accumulated DAI to the crowdfund wallet. This function can be used if the crowdfund reaches its goal.

Table 4.3: WarehouseCrowdsale Methods

4.2.3 Wareblock

The main contract that manages Wareblock is `Wareblock.sol`. The contract offers functionality such as the addition of a new warehouse. A new `WarehouseToken` is created for each warehouse. Using `WarehouseCrowdsale` one or more crowdfund processes can be initiated.

In the contract, there are the following structures and variables:

`address public owner`	is account address of contract deployment
`address public daiAddress`	is the address of DAI. This is used in the swapping process since the platform handles transactions with DAI
`Warehouse[] private warehouses`	is used to store the entire warehouse database.

Table 4.4: Wareblock structures and variables

The following structure contains two arrays, one of `warehouseToken` and one of `WarehouseCrowdsale`. That way, it can store the different crowdfunds in the same structure. A different token supply is needed for different crowdfunds, so an array is used as well.

```
struct Warehouse {
    address[] warehouseTokens;
    address payable[] warehouseCrowdsales;
}
```

A structure named `WarehouseLandUseInfo` is used to store a `WarehouseToken` type, a `WarehouseCrowdsale` type, and additional information. The information includes the total token supply, the goal of the crowdfund, the number of DAI raised so far, and the exchange rate. Another variable stored here is `supplyforSale`, that stores the number of the remaining supply of `WarehouseToken`. The structure is as follows:

```
struct WarehouseLandUseInfo {
    address warehouseToken;
    address payable warehouseCrowdsale;
    uint totalSupply;
    uint goal;
    uint daiRaised;
    uint supplyforSale;
    uint rate;
}
```

The following structure contains all the different uses for a structure using the above. It also stores the warehouse name, the `warehouseToken` symbol, the warehouse link, and the crowdfund closing time.

```
struct WarehouseInfo {
string name;
string symbol;
string tokenURI;
uint closingTime;
WarehouseLandUseInfo[] uses;
}
```

There is also an event that is emitted when a new warehouse is created:

`event WarehouseAdded()`: is used to log the new warehouse. The logs contain an array of `WarehouseToken` and an array of `WarehouseCrowdsale`.

The function used to insert a warehouse into the blockchain and initiate crowdfunding processes is `addWarehouse()`. The arguments for the function are `string memory _name` and `string memory _symbol` which are the name and symbol of the warehouse, `uint[] memory _`

which is an array that contains the total supply for each crowdfund,`string memory _tokenURI` which is the link to the warehouse in the Wareblock page, `uint[] memory _goals` which is the crowdfund goals. The arguments also contain `address payable _wallet` which is the account that the accumulated funds will be sent to, and `uint _duration` which is the duration of the crowdfund. The function creates and initializes the necessary `WarehouseToken` and `WarehouseCrowdsale` objects. It adds the created objects to the `warehouses` array and then emits a `WarehouseAdded` event.

4.3 Real Estate Smart Contract

The entirety of the project is located in a single contract named `realEstate.sol`. The contract is deployed by a government or a governing agent, as we discussed in Chapter 3. The main actors in the contract are the government, the property owner, the shareholders, and the renter.

The state variables are initialized first:

`string public name`	is the name of the property.
`string public symbol`	is the symbol of the property token.
`uint8 public tax`	is the tax rate set by the government.
`uint256 public totalSupply`	is the number of tokens generated
`uint256 public rentPer30Day`	is the rent for a 30 day stay.
`uint256 public accumulated`	is the rent collected after tax and before being distributed to the shareholders.
`uint256 public rentalBegin`	is the block number that marks the beginning of rental.
`uint256 public occupiedUntil`	is the block number that marks the end of rental.
`uint256 private _taxdeduct`	is the amount of ether transferred to the government per rent payment.

Table 4.5: realEstate state variables

The account addresses of the acting agents in the contract are also initialized:

`address public gov`	is the governing agent and is the account in which the contract is deployed.
`address public mainPropertyOwner`	is the owner's account. Initially, all the property shares are sent to that account.
`address public tenant`	is the account that pays rent. The main owner of the property must approve the tenant.
`address[] public stakeholders`	is an array that contains a list of stakeholder addresses. The stakeholder can buy shares from the main property owner. The first two entries of the array are the government and the owner

Table 4.6: realEstate address variables

A series of map structures is initialized:

`public revenues`	maps account addresses (`address`) to revenue balances(`uint256`). This is used to keep a record of shareholder balances
`public shares`	maps account addresses (`address`) to token balances(`uint256`). This is used to keep a record of the shares of each shareholder
`public rentpaidUntil`	maps addresses (`address`) to block numbers (`uint256`). This is used to determine the period for which the tenant has rented the property.
`public sharesOffered`	maps addresses (`address`) to number of shares (`uint256`). Is the number of shares each shareholder might want to sell.

`public sharesSellPrice`	maps addresses (`address`) to share price per unit (`uint256`).

The government functions are the following:

`addStakeholder() public onlyGov`	is used to add a shareholder to the `shareholders` array. Also, the government's allowance for seizing tokens over the new shareholder is set to infinity, as we discussed. A `NewStakeHolder()` event is emitted.
`banStakeholder() public onlyGov`	is used to remove a shareholder from the `shareholders` array. The shareholder's tokens are seized using the `seizureFrom()` function. Emits a `StakeHolderBanned()` event.
`setTax() public onlyGov`	is used by the government to set the tax rate. Emits a `ChangedTax()` event.
`distribute() public onlyGov`	is used by the government to distribute funds to shareholders, depending on their number of shares. Emits a `RevenuesDistributed` event for each of the shareholders.
`seizureFrom() public returns (bool success)`	can be used in two scenarios. The first scenario is when the government bans a shareholder and seizes their tokens. The second scenario is when tokens are bought. Emits a `Seizure()` event. The function is not limited to government use only.

Table 4.8: realEstate Government functions

The main property-owner functions are:

`canPayRent() public onlyPropOwner`	is used by the main property owner to approve the tenant. Emits a `CurrentlyEligibleToPayRent()` event.
`setRentper30Day() public onlyPropOwner`	is used by the main property-owner to set the rent for a 30-day period. Emits a `RentPer30DaySetTo()` event.

Table 4.9: realEstate Main property-owner functions

The stakeholder functions are the following:

`offerShares() public`	can be used by the shareholder to offer shares for sale. The shareholder can choose the price of the offered shares. Emits a `SharesOffered()` event.
`buyShares() public payable`	is used to buy shares from a shareholder that has offered. An account must be added to the shareholders list before buying from other shareholders. Only the government can add a shareholder. `SeizureFrom()` is used to transfer tokens and `address.transfer()` is used to transfer funds to the seller. Emits a `SharesSold()` event.
`withdraw() payable public`	is used by shareholders to withdraw the revenue entitled to them. Emits a `Withdrawal()` event.

Table 4.10: realEstate Stakeholder functions

The renter function is the following:

`payRent() public public payable isMultipleOf eligibleToPayRent`	is used by the tenant to pay rent and occupy the property for a block-time period. The function calculates how many tokens are required for the selected time period. Emits a `Rental()` event.

Table 4.11: realEstate Renter function

4.4 OpenZeppelin Governor

As we discussed in the Chapter 3 the Governor contract offers features such as proposal creation, voting, and execution. The key functions and data structures are located in `Governor.sol` and `IGovernor.sol`. The contracts are located at ”@openzeppelin-contracts/contracts/governance”.

The primary data structures in the contracts are:

A struct named `ProposalCore`. The struct contains information, such as the address of the proposer’s account, a timestamp for the beginning of the vote, the vote’s duration. It also contains two boolean variables to determine if the proposal has been executed or canceled.

```
struct ProposalCore {
    address proposer;
    uint48 voteStart;
    uint32 voteDuration;
    bool executed;
    uint48 etaSeconds;
}
```

A map `_proposal` is used to store the proposals. It maps proposal IDs `uint256` to proposal structs `ProposalCore`:

```
mapping(uint256 proposalId => ProposalCore) private _proposals
```

An enumeration named `ProposalState` is used to describe the state of the proposal:

```
enum ProposalState{
    Pending,
    Active,
    Canceled,
    Defeated,
    Succeeded,
    Queued,
    Expired,
    Executed
}
```

A stack is used to ensure that the governance calls are executed in the correct order.

`DoubleEndedQueue.Bytes32Deque private _governanceCall`

A common practice is to have an internal and public function with the same name (`function _<name` and `function <name>()`). This is a common practice where the public function calls the internal. Typically, the public function checks if the inputs are valid, and the internal function assumes that they are.

The key functions of the contract are:

`propose() public virtual returns (uint256)`	The function checks if the proposer is valid. Then checks if the proposer has the votes needed to make a proposal. Finally, it calls the internal function `_propose()`.

Table 4.12: Governor proposal function

Many functions for vote casting are used in the contract and each has a different use. The `IGovernor` interface provides the following:

`castVote()`	is used to cast a vote. It calls the internal function `_castVote()`

Table 4.13: Governor vote casting function

Queueing is used after a proposal has passed to queue it for execution. It is used to ensure that a proposal can not be executed until it is safe. After the selected time period has passed execution commences. While the proposal is in `Pending` state, the proposer can cancel it.

`queue()`	checks if the proposal state is valid and calls the internal function `_queueOperations()`. If the return value is 0 the function reverts, and the contract assumes that queueing was unsuccessful
`execute()`	checks if the proposal has passed. If queuing has been implemented, it also checks if the proposal is in queue. The proposal is divided in multiple governance calls. Each governance call is pushed into the `_governanceCall` stack to ensure that all the governance calls are executed in the correct order. Then, the internal function, `_executeOperations()` is called. `execute()` also handles cleanup, which is to clear the stack and emit a `ProposalExecuted()` event.

Table 4.14: Governor Queue and Execute Functions

`cancel()`	checks if the caller address is the same as the proposer address and then calls the internal function `_cancel()`. There is no need for cleanup, because the proposal never makes it to execution phase.

Table 4.15: Cancellation function

4.5 Experiments

4.5.1 Implementation

In this section, we present an effort to combine the projects reviewed in sections 4.2, 4.3, 4.4. The goal of the implementation is to create a complete back-end of smart contracts that handles token distribution, decision making, and property leasing. The project is available at ”https://github.com/MissingFilename/collaborative-space-management.git”. Important parts of our implementation can be found in the Appendix A

The three programs presented above appear to approach our topic from different angles. The Wareblock platform presented in 4.2, is for the minting of properties and the distribution of tokens. The Governor contracts, described in 4.4, can be used by token holders to propose and vote on property matters. The RealEstate contract, presented in 4.3, goes one step further and implements the property leasing feature. All three of the above programs are implemented in Solidity. This means that they can be modified and combined into a complete project.

The changes made to the Wareblock platform are few. The developer of the platform points us towards this direction, by using specific standards. An example is the use of the ERC20 token. The WarehouseToken was modified to implement the ERC20Votes that has been discussed in the chapter 3. This is a necessary change required by the Governor contract to work with the WarehouseToken. Another change is the implementation of a token factory and a warehouse factory. Contract deployment is a costly process in terms of gas and bytecode size. So, the implementation of small and lightweight modules that handle contract deployment was necessary. Finally, an important change is the use of **yarn** instead of **npm** as a package manager.

Governor had to be implemented from the beginning since the contract is offered in the form of a library. A problem that we faced during implementation is the very heavy bytecode size of the contract. Since 2016 Ethereum has enforced a limit in contract bytecode size for security reasons. [46] explains this change more thoroughly. To address this issue, we have made a lightweight implementation of the governor with basic features only. Governor is an extremely modular contract that allows for a variety of functions.

As we explain in the previous section, Governor executes proposals by calling functions. These functions are used to call other functions or deploy contracts. For processes that take place off-chain, the proposal owner can pass the proposal without a target function, only with

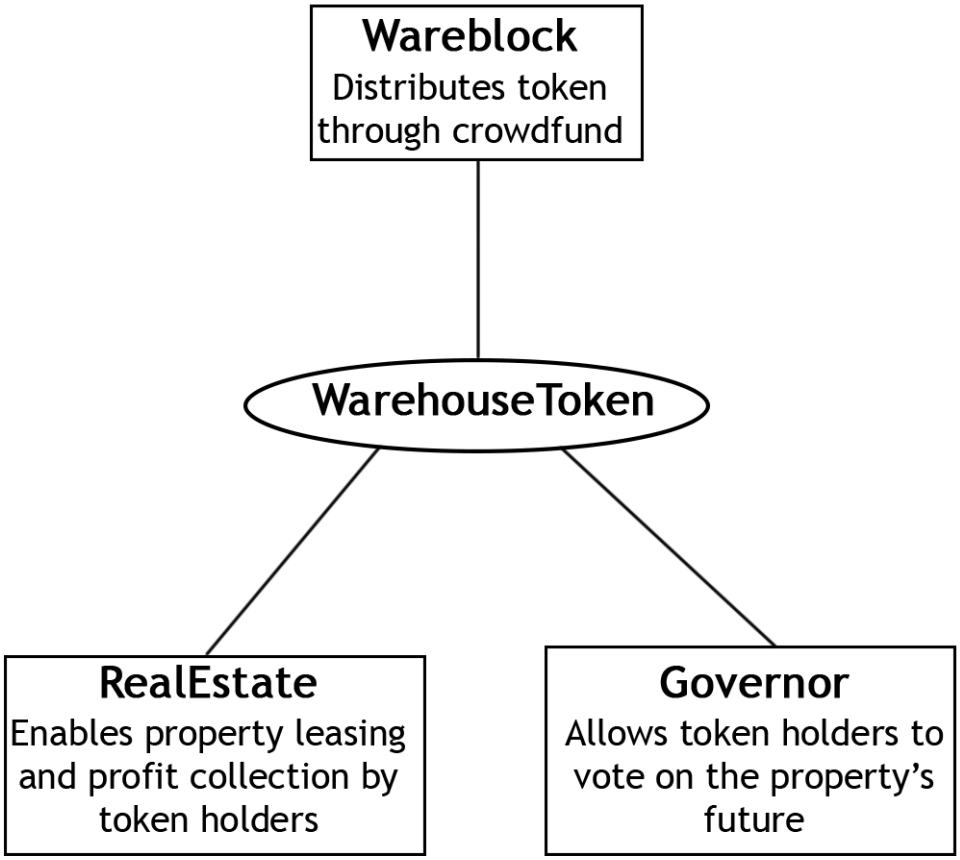

Figure 4.1: Contracts used in implementation: The image shows how the three contracts presented above, would work together.

a description. Since not all property management processes are expected to have migrated to the blockchain, it is essential to have an outside partner (who is probably a property manager) that handles description-only proposals. In this way, the platform can accommodate decisions that are made on the blockchain but contain processes that happen off-chain.

Finally, the lease part of the platform allows WarehouseToken holders to receive their share of the profit when the property is leased. The largest change made to this contract is the removal of shares and the introduction of ERC20 tokens. The contract developer has implemented the shares to share common logic with the ERC20 token, so the transition is relatively easy to implement. Another change is the removal of the main role of the property owner, since there is no clear property owner on the Wareblock platform. All property owner functions are now handled by the government.

4.5.2 Testing

Our aim is to implement a compilation of contracts that handle three parts: token distribution through the crowdfund, proposal passing and vote, and property leasing. After reviewing the implementation of Wareblock, we notice that testing takes place in three different domains. Locally, using the Hardhat testing environment, using Ganache, and using the Ropsten testnet. Similarly, since it is customary to follow this practice, we use three levels of testing.

We use the local Hardhat testing environment as the first level of testing. We implement unit tests that focus on the features we implement. We do not provide tests for the features that have already been tested by the original contract authors. Since Ganache is deprecated, we use the alternative, which is the Hardhat node. A hardhat node is the Ganache equivalent but is built in Hardhat and can also be used for the unit tests. As the third level of testing, the contracts have been deployed on the Sepolia testnet. This is a costly and slow process that requires extreme care.

For gas measurements in local testing, we use the hardhat-gas-reporter package. The gas reporter monitors the testing process and records the gas spent for each of the function calls. Typically, contract deployments are the most expensive calls in terms of gas. Gas measurements for the Sepolia network can be acquired through Etherscan. Etherscan is a block explorer that keeps track of accounts, transactions, and contract addresses. It is also available for the testnet. It records the transactions along with the gas spent on each one. Since a testnet is supposed to mimic the behavior of a real network, by viewing the address of the deployer, we can get all the numbers we need.

4.6 Discussion of the results

The gas measurements produced by the gas reporter are shown in the following table.

Wareblock	
`addWarehouse`	3277367
WarehouseCrowdsale	
`buyTokens`	171248
WarehouseToken	

approve	46381
delegate	95685
transfer	56309
WarehouseGovernor	
castVote	69503
execute	732092
propose	72047
RealEstate	
buyShares	68599
canPayRent	47148
distribute	105428
offerShares	82427
payRent	171745
setRentper30Day	46903
withdraw	29828

Table 4.16: Gas measurements using the Hardhat Gas Reporter. The values in the second column are in Gas units.

Contract	Gas Reporter (Gas Units)	Etherscan (ETH)
WarehouseToken	2087968	
WarehouseCrowdsale		
Wareblock	1622878	0.03430629
TokenFactory	2739257	0.05615607
CrowdsaleFactory	1253155	0.02586608
WarehouseGovernor	2885519	
RealEstate	2021998	

Table 4.17: The table presents measurements from the contract deployments. The second column of the table contains gas measurements from the gas reporter. The third contains gas measurements from Etherscan.

At the time of measurement for the above tables, the gas price on the Sepolia network is 21.139170925 Gwei (0.000000021139170925 ETH). This means that we can check for gas reporter accuracy and reliability by multiplying the gas reporter output to the gas price. The following table contains the computed gas spending based on the gas reporter compared to the actual values.

Contract	Computed Gas (ETH)	Actual Gas (ETH)
Wareblock	0.03430629543	0.03430629
TokenFactory	0.05790562193	0.05615607
CrowdsaleFactory	0.02649065774	0.02586608

Table 4.18: Gas reporter computed gas value in ETH compared to actual testnet value.

At a different time, when the gas price is 34.519183885 Gwei (0.000000034519183885 ETH), we call the `addWarehouse` function. The gas reporter estimation for this function is 3277367 Gas. This means that at a rate of 0.000000034519183885 ETH per Gas unit the function call should cost 0.1131320341 ETH. The actual gas fee provided by Etherscan is 0.11313327 ETH.

From the numbers gathered, we can draw two conclusions. The first is that the gas reporter has an accuracy of three to four digits. This is essential in the calculation of the budget. This means that we can safely use the gas reporter's numbers to make a cost estimation regarding gas fees and other blockchain-related expenses.

Chapter 5

Case Study

5.1 Introduction

In this chapter, we are going to apply the business plan and the implementation (presented in Chapters 2 and 4 respectively) to a real building to build a case study. [2] explains that the Wareblock platform was initially built with the goal of renovating abandoned buildings, specifically abandoned tobacco warehouses in northern Greece. The selected warehouse is a building of great historical importance, located in the city of Kavala. The warehouse is called ”Kapnapothiki EOK” because it belonged to the National Tobacco Organization, before being bought by the Municipality of Kavala.

The building is located in the city center, next to the city hall. Figure 5.1 shows exactly where the warehouse is located. It is a six-story building that was previously used as an old tobacco warehouse. The first floors are currently being used. The ground floor, first, second, and part of the third floor are actively used by the Municipality of Kavala. At the ground level, there is also a tobacco museum run by the Municipality. The remaining upper floors and the remainder of the third floor are currently abandoned, but structurally sound. This makes them an ideal candidate for adaptation and reuse, using the Wareblock platform.

5.2 Proposal

This case study applies the Wareblock model to the currently unused areas of the warehouse. As described above, the eastern part of the third floor, the fourth and fifth floors are abandoned. A structural analysis has green-flagged operations to take place to renovate and

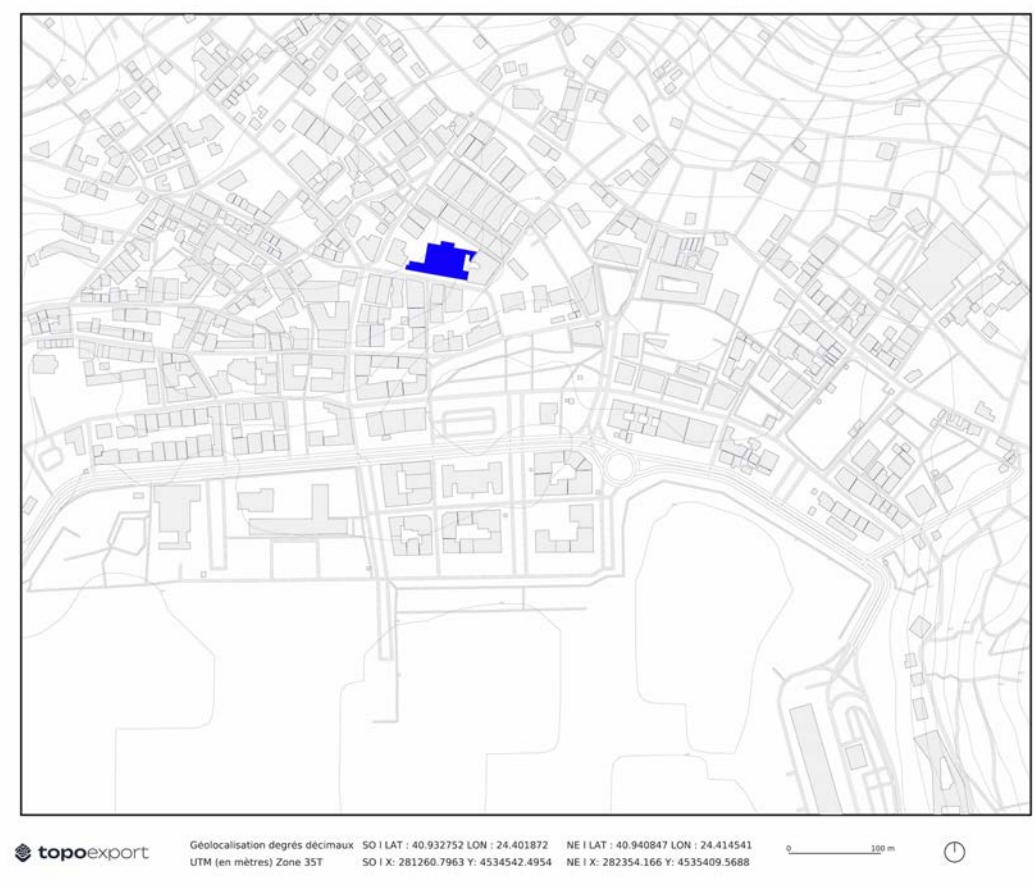

Figure 5.1: Kavala City Center - Topographic Map - Highlighted in blue is the building ”Kapnapothiki EOK” - Source: https://topoexport.com

(a) Schematic of the roof

(b) Aerial Photograph

Figure 5.2: ”Kapnapothiki EOK” - Kavala City Center - Topographic Map - Large Scale - Source: https://topoexport.com

use the building for our project.

The proposal involves transforming the eastern part of the third floor into a gallery space while repurposing the fourth and fifth floors into a cafe. The gallery may operate in association with the tobacco museum located on the ground floor or independently as an exhibition space. The high ceilings of the warehouse provide ample natural lighting to the interior, making it suitable for cultural and commercial use.

In addition to operating as a cafe, the adaptable rooms on the upper floors can host workshops, study areas, screenings, and exhibitions. Given the location of the building in the heart of Kavala and the panoramic view from the upper floors, the proposed concept is likely to attract the public. The combination of culture through the museum and exhibitions, with a study area and a cozy, well-equipped cafe, is expected to attract wide public participation.

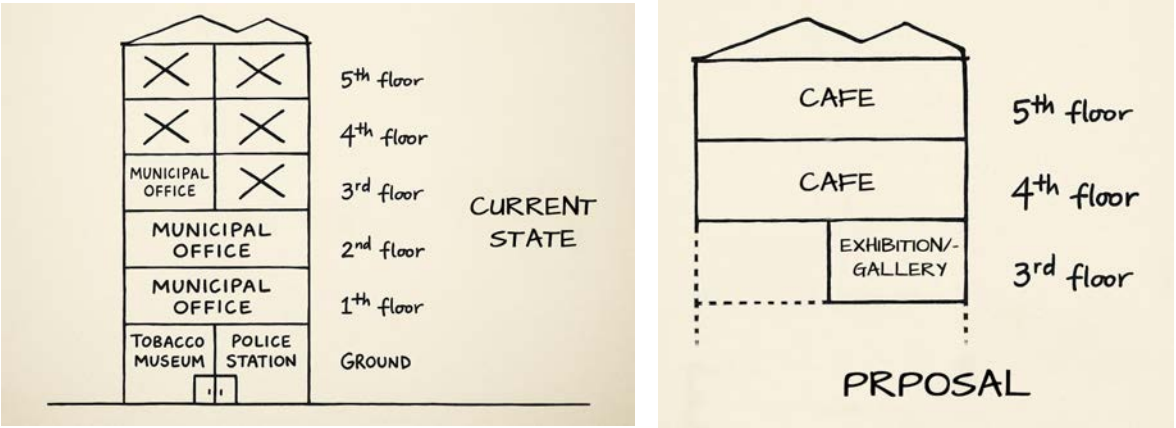

Figure 5.3: Current state of each floor (left) and the proposed use (right).

5.3 Application of our implementation

Assuming that the Wareblock contract has already been deployed, we have to calculate minting, crowdfunding, and eventually decision-making fees. It is essential to calculate the gas fees precisely, to correctly calculate the budget. Also important is the calculation of gas for participating in crowdfunding because investors have to cover it as well. As we explain in Chapter 4 the results from the gas reporter provide an accuracy of 4-5 digits. This is deemed enough to make calculations that will not affect the investment or the investors.

In the minting process, we gather the necessary information and upload it to the Wareblock platform. The WarehouseToken has the following properties. A name, a symbol, and a

URI are used to identify the token. Typically, a name is an identifying characteristic, which means that it is unique. In this instance, we can assume that the name is ”Kapnapothiki EOK”, the symbol is ”KAPEOK”, and the URI is produced by the platform.

For this case study, we use the following numbers. According to https://etherscan.io the average price of Gas at the time of writing is 2.388 GWEI or 0.000000002388 ETH. Also, at the time of writing, the price of Ethereum is at €2.285,88.

The function that handles minting is `addWarehouse()`. As we have discussed in the previous chapter, this function is responsible for minting the warehouse into the blockchain, generating `WarehouseToken` and creating the supply, and initiating crowd-funding or multiple ones according to the plan. The average cost of calling `addWarehouse()`, according to the Hardhat Gas Reporter, is 3,277,367 Gas Units. To calculate the gas fee for calling this function we do the following:

$$GasFee = GasUnits * AvgGasPrice * EthereumPrice$$

$$GasFee = 3277367 * 0.000000002388 * 2285.88$$

$$GasFee = 17.89$$

This gives us a gas fee of €17.89. We can calculate all the gas fees in a similar way. Table 5.1 shows us the calculation of all fees.

Function	**Gas Used (Gas Units)**	**ETH Cost (ETH)**	**Cost (€)**
addWarehouse	3277367	0.00782819	17.89
buyTokens	171248	0.00040891	0.94
buyShares	68599	0.00016384	0.37
WarehouseGovernor	2087968	0.00498571	11.39
propose	72047	0.00017188	0.39
delegate	95685	0.00022843	0.52
castVote	69503	0.00016591	0.38
execute	73292	0.00017502	0.40

Table 5.1: Functions and average gas cost - Implementation, Chapter 4

From table 5.1 we draw the following conclusions. Contract deployments are more expensive than normal function calls, but are executed only once per property. The cost of minting a warehouse and generating tokens is around 18€. We can see that there are two different options to buy. `buyTokens` refers to the crowdfund and is more expensive, while `buyShares` refers to post-crowdfund, after the property has been leased. The option `buyShares` is for people looking for an entry in a later phase. Buying tokens has a gas fee of around €1 per purchase.

5.4 Application of our business plan

Using the map in the figure 5.4 we can calculate an estimate of the area of the building. Using the scale, we estimate that each floor is about $1000m^2$. This means that the building is a total estimate of $2500m^2$. To estimate the construction cost, we can use a safe number of $€400/m^2$, which according to the Technical Chamber of Greece is a reasonable amount. To back this estimation, we take the following into consideration. The building has passed structural assessment so only surface work is needed, besides electrical and plumbing. The building has large, well-lit areas which require little lighting work. The building already has infrastructure for HVAC, elevators, and general accessibility. The transformation does not require industrial standards, since it will be turned into a cafeteria.

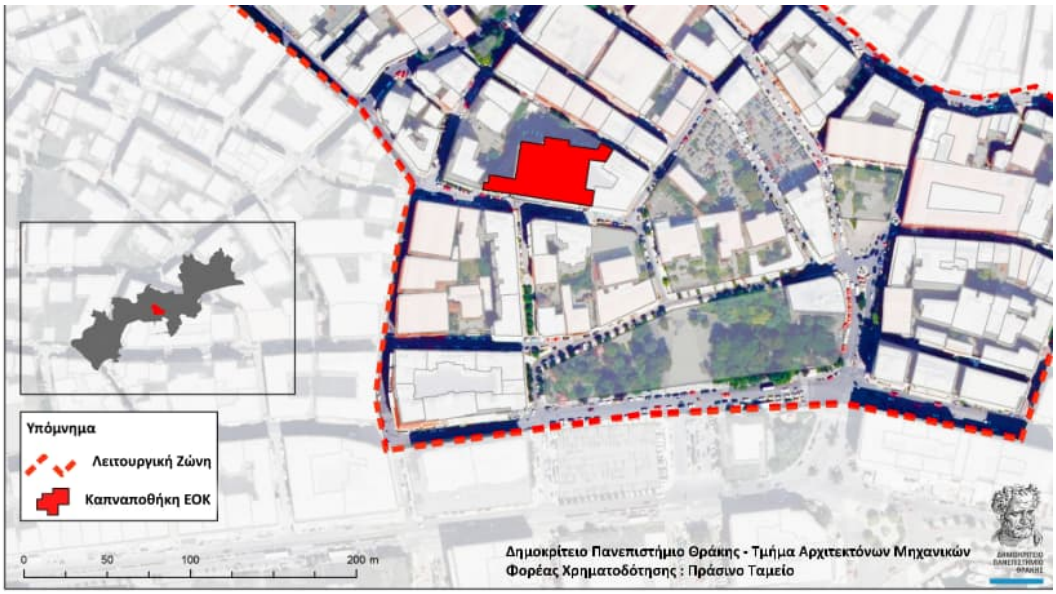

Figure 5.4: ”Kapnapothiki EOK - Position in map and scale - Source: [, εργο...]”

The next step is to calculate the total construction cost. From the following formula, we

get that the total construction cost is €1250000.

$$ConstructionCost = 2500 \times 400 = 1000000$$

The first in the list of priorities is to make a timeline. As we can see from figure 5.2 the duration of the plan is 5 years. The next cost estimation to make is the manager's salary. Considering the length this individual or firm would have to go to, we make a fair estimation of 1200€/month. Since the complete project is going to last for 7-9 years, the salary of the manager is €129600, as we can see from the following.

$$ManagerSalary = 12 * 9 * 1200 = 129600$$

Duration	**Action**
	Crowdfund opens
6 Months	Crowdfunding, token distribution, capital raise.
1.5 years	Renovation, construction, preparation.
5-7 years	The building is leased or available to be leased.
	Break-Even Point, End of Timeline.

Table 5.2: "Kapnapothiki EOK" Uptake and lease - Project Timeline

In this case study, we make the assumption that there is no platform fee, rather the platform gets a token share. Finally, the blockchain costs are very low which means they can be covered by the overhead. To calculate the budget required to start the project, we add all the before-mentioned costs.

$$TotalCost = 1000000 + 129600 + BlockchainCosts = 1129600$$

Adding the costs, we get a total of €1129600. A clever practice is to consider a 10-15% overhead for a contingency. Using the following, we get a budget of €1242560.

$$Budget = 1129600 + \frac{1129600 * 10}{100}$$

$$Budget = 1242560$$

For the token price, we divide the budget to the number of tokens available. We consider 20% of the token supply to be allocated for the Municipality and the Wareblock platform. Considering a token supply of 10000 tokens, the remaining 80% accounts for 8000 tokens. Using the following calculation, we get a token price of €155.32 per token.

$$TokenPrice = \frac{1242560}{8000} = 155.32$$

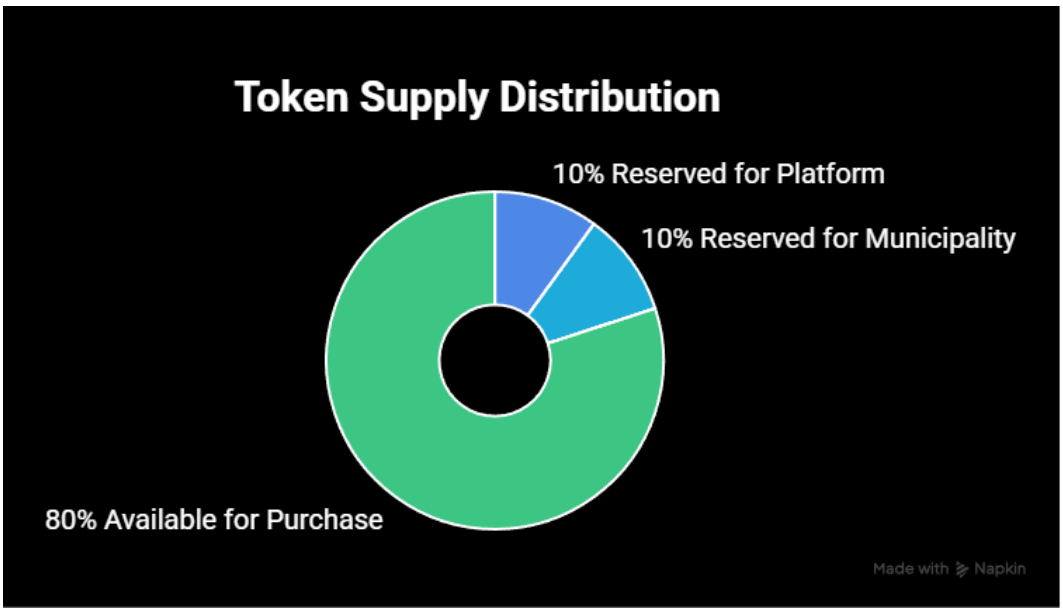

Figure 5.5: Token Supply Distribution - Pie Chart Generated at: https://napkin.ai

The third-floor gallery will be managed by the Municipality of Kavala and will operate as an exhibition related to the tobacco museum, or independent. The gallery will be open to the public with a standard entry fee of 5€. Upon agreement, we expect 20% of the gallery's revenue as compensation for providing the place and the infrastructure. We consider a realistic amount of monthly visitors is a 1000, according to the population, tourism, and overall cultural activity of the city of Kavala. At an entry fee of €5 the gallery is estimated to generate about €12000 annually and €60000 in a 5 year duration. This means that at a rate of 20%, we expect a total return of €12000, from the gallery. We remind the reader that this is a pessimistic-realistic scenario so that we can make safe estimates.

To break even in a span of 7 years, the fourth and fifth floors are to be leased at a price of €18490.47. Using a realistic scenario of 80% occupancy for the building, we make the following calculation.

$$MonthlyLease = \frac{Budget - GalleryRevenue}{(7years * 12months) * 80\%} = \frac{1242560 - 12000}{67.2} = 18490.47$$

To conclude the case study, we make the following remark. The business plan is created using pessimistic-realistic numbers for safety. The outcome is an investment that follows property investment standards, such as the duration or the slow but guaranteed break even. Table 5.3 shows a summary of the numbers calculated in the business plan.

Item	**Value**
Total Budget	€1242560
Token Supply	10000
Total Tokens Available	8000
Token Price	€155.32
Revenue from Gallery	€12000
Monthly Lease	€18490.24

Table 5.3: Summary of the number estimations and calculations in the business plan

In the following images, we present a Wareblock platform concept that we created according to the implementation provided by [2]. The mocks were generated using Figma, a concept design framework at: ”https://figma.com”. Figures 5.6, 5.7, and 5.8 present the introduction to the platform, as well as the means to connect and buy Warehouse tokens. Figures 5.9 and 5.10 show the user inventory and a token example. Finally, figures 3.1 and 5.12 show the proposal creation and fund collection respectively.

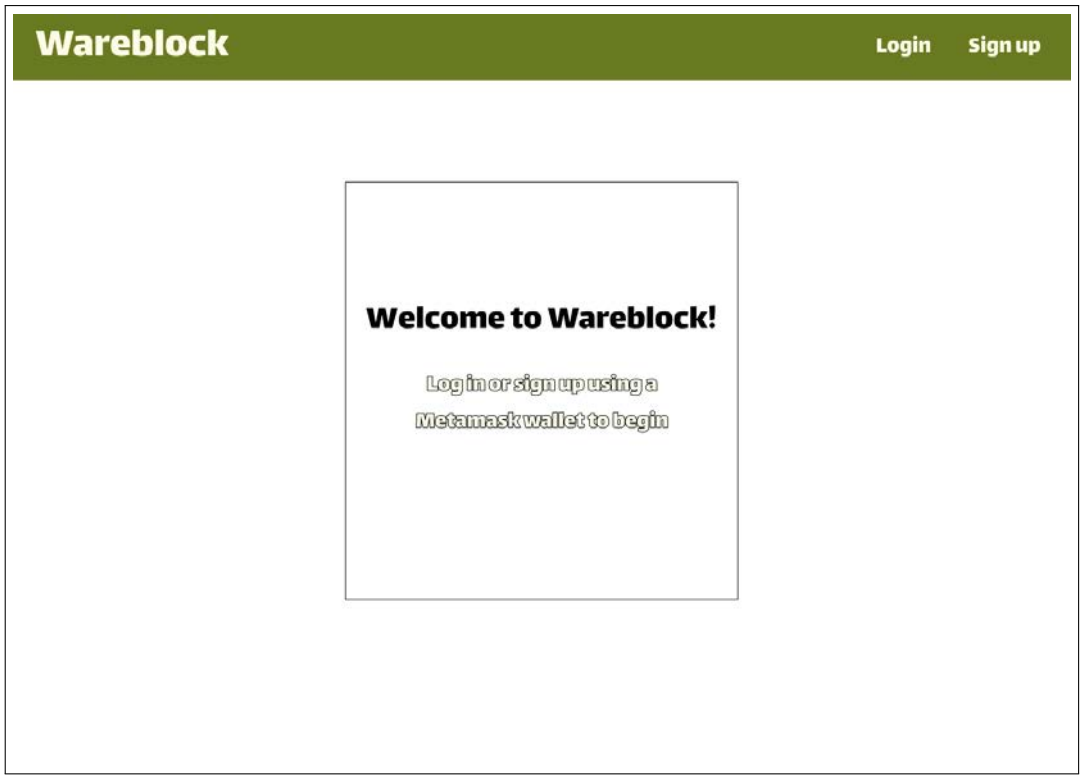

Figure 5.6: Wareblock Mock - Starting Page

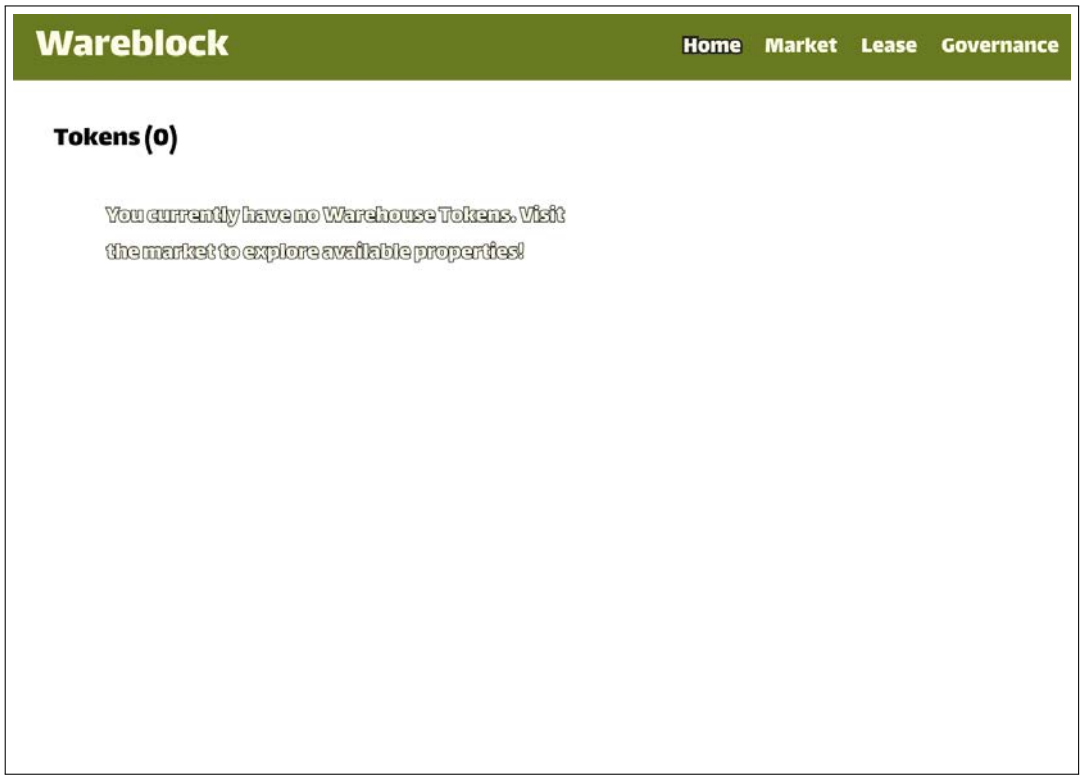

Figure 5.7: Wareblock Mock - Empty Home Page

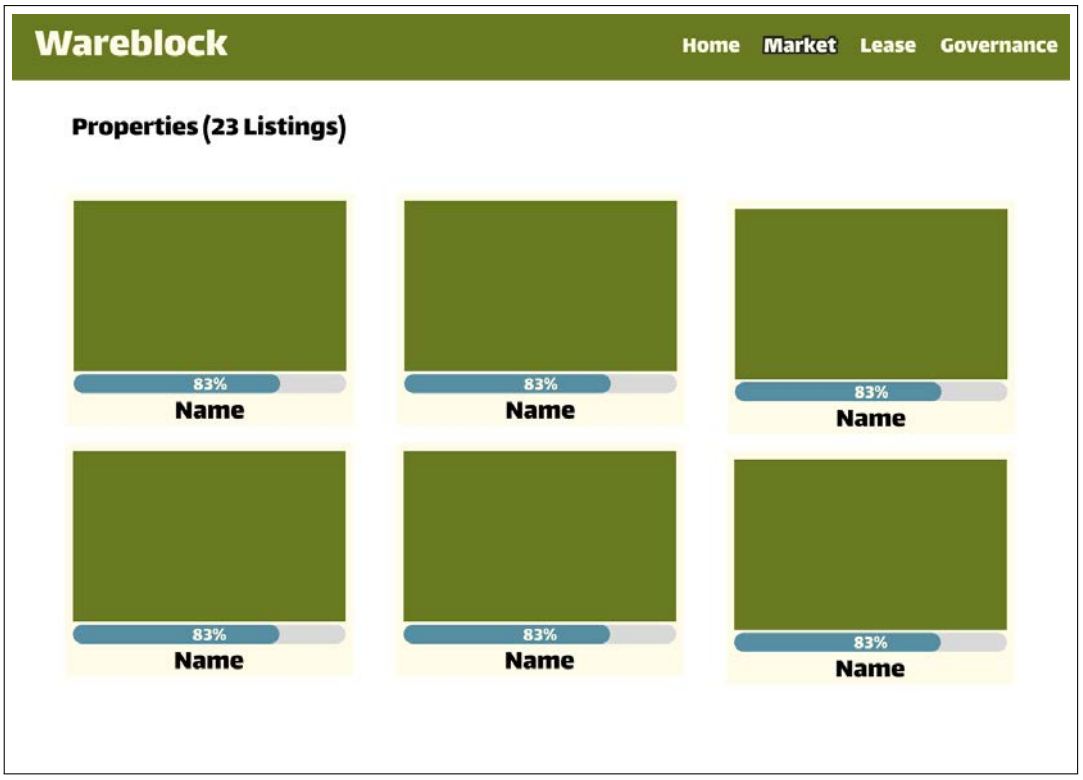

Figure 5.8: Wareblock Mock - Market

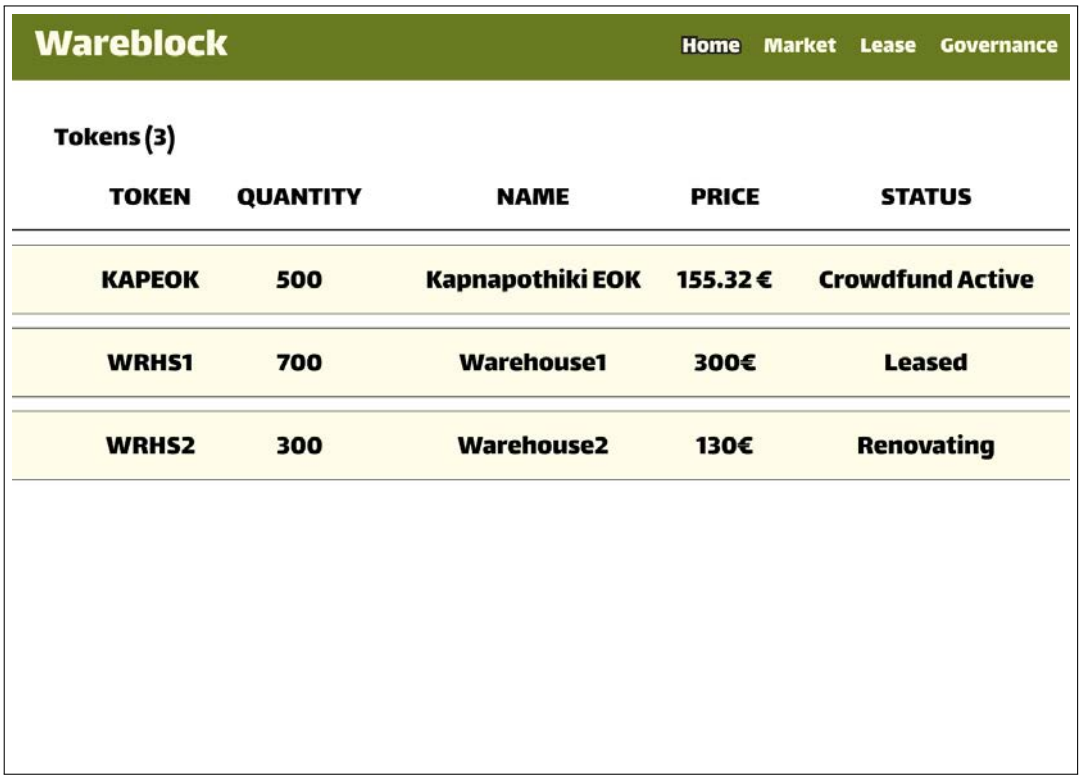

Figure 5.9: Wareblock Mock - Home

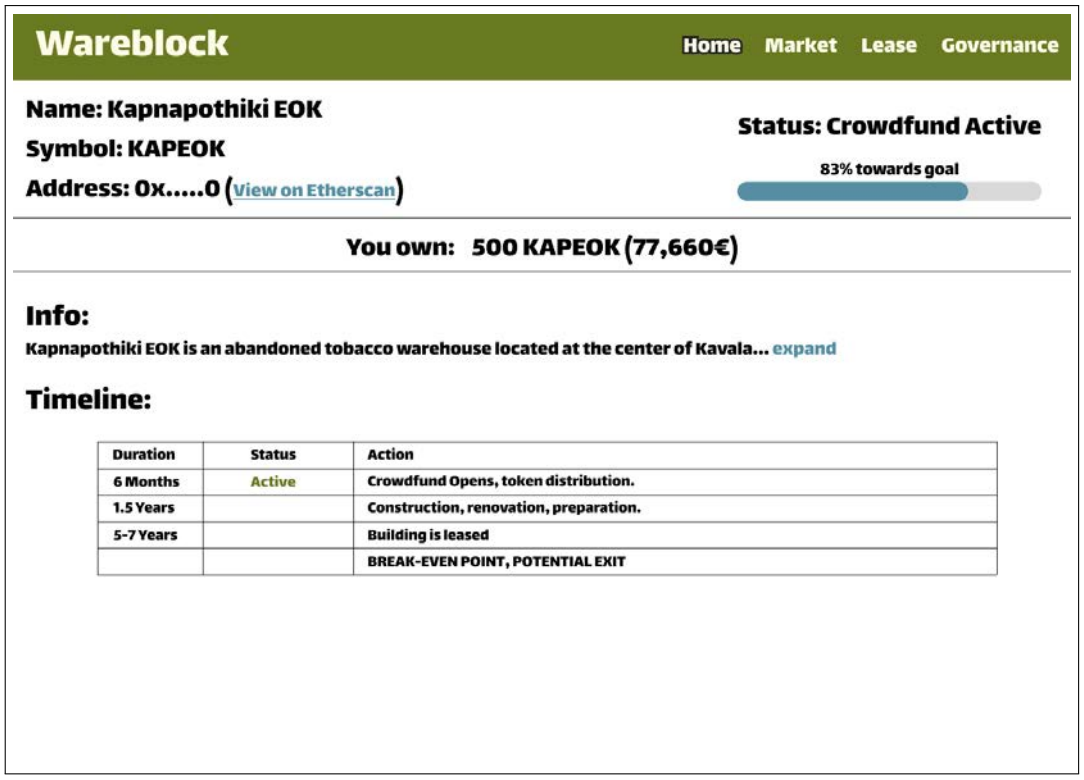

Figure 5.10: Wareblock Mock - Token Info - Kapnapothiki EOK

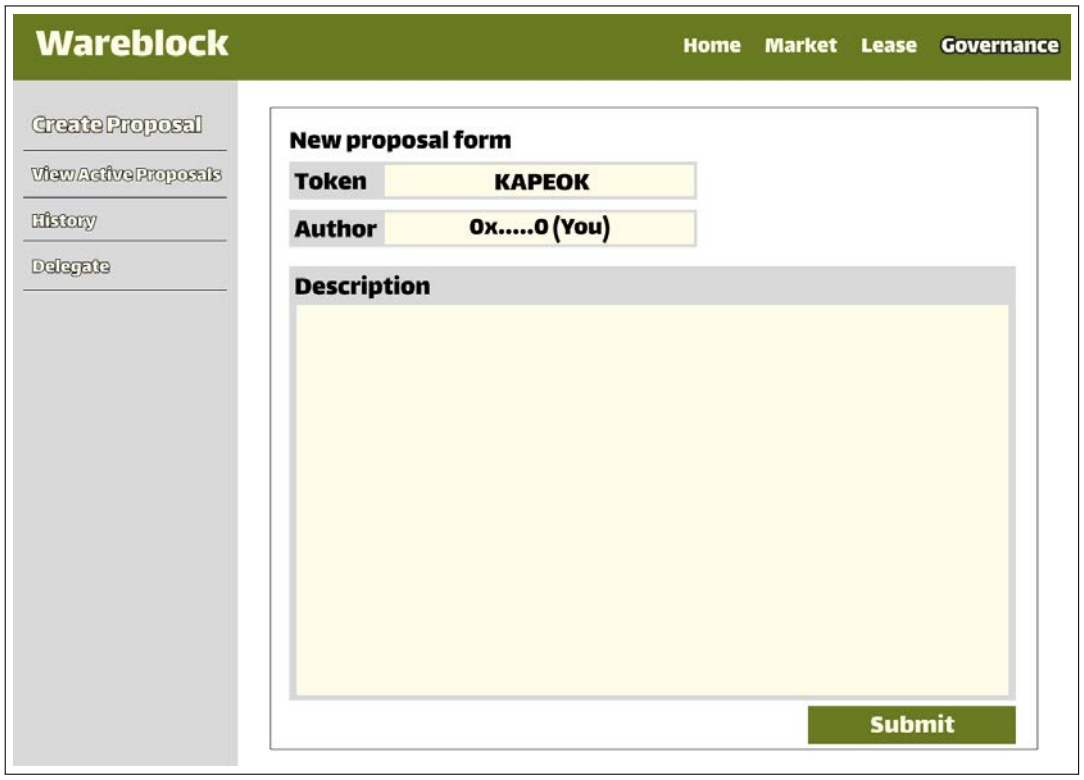

Figure 5.11: Wareblock Mock - Proposal Creation

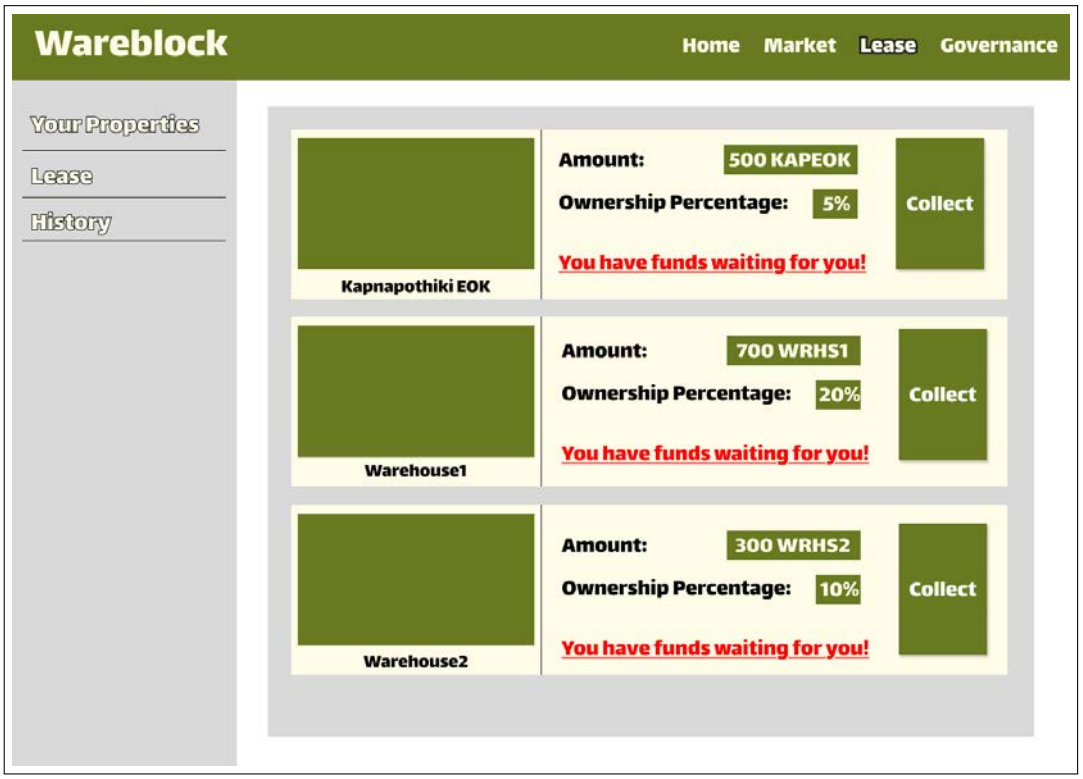

Figure 5.12: Wareblock Mock - Collection of funds

Chapter 6

Synopsis and Prospects

6.1 Synopsis

The objective of this thesis was to investigate and expand on the use of blockchain in the context of cooperative space management. This work was carried out with the ultimate goal of expanding the Wareblock project, introduced in 2019.

As we explain in our literature review, in Chapter 3, the topic involves governance and decision making, collaboration, and collective ownership. In simple terms, a group of people collectively owns an entity and their goal is to manage it. Here, it should be noted that the term management can be used abstractly to translate into different meanings. We deem that the most closely related definition is to decide upon their co-owned property's future. The literature review presented in Chapter 3 investigates blockchain management and primarily revolves around governance and decision-making issues.

In addition to papers and articles, the literature review contained code. The code was reviewed to detect potential tools and smart contracts that can be used to achieve our goal. In this paper, we successfully analyze the use and implementation of these tools. In the first part of Chapter 4, we present a thorough review of the workings of the code. The reviewed contracts include Wareblock, RealEstate, and Governor. The second part of Chapter 4 presents our effort to implement a working decision-making governance system and a lease system, using these tools. The contract is implemented and tested successfully.

Considering that the Wareblock platform proposes an investment in the form of buying tokens, it is imperative that the investor is presented with a detailed business plan. Chapter 2 presents an elaborate scheme for investing in the uptake of an abandoned warehouse, using

the platform. The plan proposes a break-even strategy. In the end, the plan leaves the investor with 100% return on their original investment. The token remains in the possession of the investor, and they can either sell it or keep collecting their monthly earnings with it.

Finally, we applied the implementation and business plan to a real-world building to build a case study. The case study consisted of the blockchain analysis in which we calculate the blockchain cost of minting a building. The findings were then used to build the business plan according to the plan we present in Chapter 2. The result suggests a feasible long-term investment that offers a simple system to rent, receive profits, and cooperate to make decisions.

6.2 Prospects

This thesis focuses on cooperative management, while the ultimate goal is to expand the Wareblock platform. The presented work manages to achieve that, in an organized and well-grounded manner. Chapter 2 explores the financial aspect of a property investment using cryptocurrency, while Chapter 3 explores similar technologies and applications. Chapter 4 presents the next step, by combining the Wareblock implementation with two related ones. The result is a group of smart contracts that act as a token distributor, real estate rental, and governance system.

Noticeably missing from the implementation is the user interface. While we provide the reader with mock designs of the platform, the user interface is left for future implementation. Assuming that the Wareblock project proceeds to receive funding, the implementation will be finished and the platform will begin operation. There is a need for different tasks that translate into job positions. Finally, the platform, acting as a company or DAO, will have to initiate different minting procedures and market them to attract investors.

Upon receiving a fund, the first step would be to assemble a team. Taking into account the needs of the platform, an ideal team would consist of full-stack developers,blockchain engineers, legal consultants, and a project coordinator. In addition, the team would require individuals to handle the property management until the project reaches a threshold where investors can manage it, as well as a person to realize investor proposals. Early expenses should also include an area of operation equipped with appliances.

Another important step that requires attention is the legal aspect of the platform. Currently, there is very little regulation regarding token-based ownership, so it is important to

build a legal background to ensure a safe investment. This will help protect investors, property owners, and tenants. Legal consultants are also useful for negotiating deals with building owners and creating the terms of use of the platform.

The last step is to start working with buildings. For example, a pilot building to initiate and help market the platform as a modern means of investment. This will help place the platform on the market and make a test case to help eliminate potential problems that have to do with design and planning. Finally, it is important to decide how the platform is going to attract investors and how to share the Wareblock vision with them.

Bibliography

[1] Compound | Proposal Detail #89. URL: https://compound.finance/governance/proposals/89.

[2] Eleni Panagou and Manolis Vavalis. A blockchain based platform for the uptake of abandoned historical building *. Technical report, 2019. URL: https://labsgroup.io.

[3] Yue Liu, Qinghua Lu, Guangsheng Yu, Hye Young Paik, and Liming Zhu. Defining blockchain governance principles: A comprehensive framework. *Information Systems*, 109, 2022. doi:10.1016/j.is.2022.102090.

[4] Ge Wang, Juanjuan Li, Xiao Wang, Junqing Li, Yong Yuan, and Fei Yue Wang. Blockchain-Based Crypto Management for Reliable Real-Time Decision-Making. *IEEE Transactions on Computational Social Systems*, 10(6), 2023. doi:10.1109/TCSS.2022.3211331.

[5] Sen Li and Yan Chen. Governing decentralized autonomous organizations as digital commons. *Journal of Business Venturing Insights*, 21, 2024. doi:10.1016/j.jbvi.2024.e00450.

[6] Enis Karaarslan and Enis Konacakli. CHAPTER 3 DATA STORAGE IN THE DECENTRALIZED WORLD: BLOCKCHAIN AND DERIVATIVES. 2020. doi:10.26650/B/ET06.2020.011.03.

[7] Akm Bahalul Haque and Mahbubur Rahman. Blockchain Technology: Methodology, Application and Security Issues. *IJCSNS International Journal of Computer Science and Network Security*, 20(2), 2020.

[8] Consensus mechanisms. URL: https://ethereum.org/en/developers/docs/consensus-mechanisms/.

[9] Ethereum Virtual Machine (EVM). URL: https://ethereum.org/en/developers/docs/evm/.

[10] How to Create a Business Budget. URL: https://business.bankofamerica.com/en/resources/don-t-fear-the-b-word-how-budgets-can-liberate-your-business.

[11] Expense: Definition, Types, and How It Is Recorded. URL: https://www.investopedia.com/terms/e/expense.asp.

[12] 16.4: The Break-Even Point - Business LibreTexts. URL: https://biz.libretexts.org/Bookshelves/Business/Introductory_Business/Book%3A_Introduction_to_Business_(Lumen)/16%3A_Accounting_and_Finance/16.04%3A_The_Break-Even_Point.

[13] What Is Return on Investment (ROI) and How to Calculate It. URL: https://www.investopedia.com/terms/r/returnoninvestment.asp.

[14] What Is a Surplus? URL: https://www.investopedia.com/terms/s/surplus.asp.

[15] What Is Gross Income? Definition, Formula, Calculation, and Example. URL: https://www.investopedia.com/terms/g/grossincome.asp.

[16] Net Income vs. Profit: What's the Difference? URL: https://www.investopedia.com/ask/answers/122414/net-income-same-profit.asp.

[17] Sakshi Sanjay Pande, Shrushti Mandolikar, and Sanjay Shitole. Bitland-A Decentralized Commercial Real Estate Platform. In *IBSSC 2022 - IEEE Bombay Section Signature Conference*, 2022. doi:10.1109/IBSSC56953.2022.10037494.

[18] Arjun Menon, Kaustubh Kadam, Pranav Kumar, and Subash Kumar Shah. Decentralized Crowdfunding Using Blockchain. *SSRN Electronic Journal*, 2023. doi:10.2139/ssrn.4324640.

[19] Adam P. Balcerzak, Elvira Nica, Elżbieta Rogalska, Miloš Poliak, Tomáš Klieštik, and Oana Matilda Sabie. Blockchain Technology and Smart Contracts in Decentralized Governance Systems, 2022. doi:10.3390/admsci12030096.

[20] Yan Chen, Jack I Richter, and Pankaj C Patel. Decentralized Governance of Digital Platforms. *Journal of Management*, 47(5):1305–1337, 2021. doi:10.1177/0149206320916755.

[21] @LCX Team. Decentralized Governance Mechanism in Blockchain - LCX, 8 2023. URL: https://www.lcx.com/decentralized-governance-mechanism-in-blockchain/.

[22] Cesare Fracassi, Moazzam Khoja, and Fabian Schär. Decentralized Crypto Governance? Transparency and Concentration in Ethereum Decision-Making. *SSRN Electronic Journal*, 2024. doi:10.2139/ssrn.4691000.

[23] Lukas Schädler, Michael Lustenberger, and Florian Spychiger. Analyzing decision-making in blockchain governance. *Frontiers in Blockchain*, 6, 2023. doi:10.3389/fbloc.2023.1256651.

[24] Shiva Jairam, Jaap Gordijn, Isaac Da, Silva Torres, Fadime Kaya, and Marc Makkes. A Decentralized Fair Governance Model for Permissionless Blockchain Systems. 2021.

[25] Justin Mart and Ryan Yi. Around the Block #13: On the value and risks of governance tokens | Coinbase, 2021. URL: https://www.coinbase.com/learn/market-updates/around-the-block-issue-13.

[26] openzeppelin-contracts/contracts/token/ERC20/extensions/ERC20Votes.sol at master · OpenZeppelin/openzeppelin-contracts. URL: https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/token/ERC20/extensions/ERC20Votes.sol.

[27] @Cryptopedia Staff. DeFi Solutions: Decentralized Governance Protocols | Gemini, 2022. URL: https://www.gemini.com/cryptopedia/defi-solutions-decentralized-governance-meaning.

[28] @Cryptopedia Staff. Blockchain Governance on Decentralized Networks | Gemini, 2023. URL: https://www.gemini.com/cryptopedia/blockchain-governance-mechanisms.

[29] Yoshiro Saito and John A. Rose. Reputation-based Decentralized Autonomous Organization for the non-profit sector: Leveraging blockchain to enhance good governance. *Frontiers in Blockchain*, 5:1083647, 1 2022. doi:10.3389/FBLOC.2022.1083647/BIBTEX.

[30] Ian Appel and Jillian Grennan. Decentralized Governance and Digital Asset Prices. *SSRN Electronic Journal*, 2023. doi:10.2139/ssrn.4367209.

[31] Introducing the Tally Protocol — tally.xyz. URL: https://tally.mirror.xyz/Drw-uvqhUnJLRxg32sV-sqKZ785-AO85FBaCYeXqxhA.

[32] David Rozas, Antonio Tenorio-Fornés, and Samer Hassan. Analysis of the Potentials of Blockchain for the Governance of Global Digital Commons. *Frontiers in Blockchain*, 4:577680, 4 2021. URL: www.frontiersin.org, doi:10.3389/FBLOC.2021.577680/BIBTEX.

[33] Paul Van Vulpen and Slinger Jansen. Decentralized autonomous organization design for the commons and the common good. *Frontiers in Blockchain*, 6:1287249, 12 2023. doi:10.3389/FBLOC.2023.1287249/BIBTEX.

[34] Ellie Dotson and Tommy Lower. The Modern Guide to Digital Governance, 2024. URL: https://modernguidetodigitalgovernance.com/.

[35] Calvin Liu. Compound Autonomous Proposals | Compound Labs | Medium, 2020. URL: https://medium.com/compound-finance/compound-autonomous-proposals-354e7a2ad6b7.

[36] Robert Leshner. Compound Governance. Steps towards complete decentralization | Compound Labs | Medium, 2020. URL: https://medium.com/compound-finance/compound-governance-5531f524cf68.

[37] Compound v2 Docs | Governance. URL: https://docs.compound.finance/v2/governance/.

[38] How to set up on-chain governance - OpenZeppelin Docs. URL: https://docs.openzeppelin.com/contracts/4.x/governance.

[39] REIT Smart Contract (Solidity) - ethereum_smart_contracts_solidity/contracts at Pirmins_Branch · pzd3mir/ethereum_smart_contracts_solidity, 2020. URL: https://github.com/pzd3mir/ethereum_smart_contracts_solidity/tree/Pirmins_Branch/contracts.

[40] REIT Smart Contract (Solidity) - Tokenized Real Estate & Property Management on Ethereum Blockchain - YouTube, 2020. URL: https://www.youtube.com/watch?v=S_5GT04xLAU.

[41] GitHub - lab9k/ParkCoin: A new parking system using the distributed database ethereum. URL: https://github.com/lab9k/ParkCoin.

[42] PARKCRYPTION – Rent parking spaces using Binance Coin or Ethereum. URL: https://www.parkcryption.co.uk/.

[43] Sanket Shigaonkar, Abhishek Vetal, Pranav Yeolekar, and Suchita Walke. A Decentralized Web Application for Parking Spot Booking and Listing: Improving Convenience and Efficiency in Urban Areas. 2023. URL: http://computers.stmjournals.com/index.php?journal=JoWET&page=index, doi:10.37591/JoWET.

[44] H. S. Jennath, S. Adarsh, Nikhil V. Chandran, R. Ananthan, A. Sabir, and S. Asharaf. Parkchain: A Blockchain Powered Parking Solution for Smart Cities. *Frontiers in Blockchain*, 2(6):460168, 8 2019. URL: https://www.parkaidemobile.com/, doi:10.3389/FBLOC.2019.00006/BIBTEX.

[45] South Korea implements blockchain shared parking, part of smart city initiative - Ledger Insights - blockchain for enterprise. URL: https://www.ledgerinsights.com/blockchain-parking-shared-south-korea-smart-city/.

[46] Hard Fork No. 4: Spurious Dragon | Ethereum Foundation Blog. URL: https://blog.ethereum.org/2016/11/18/hard-fork-no-4-spurious-dragon.

Appendix A

Reviewed Code

This appendix chapter contains the code that we reviewed in Chapter 4.

`setCrowdsaleAddress( address addr ) external OnlyOwner`	is used to set the `crowdsale` variable to the account address that handles the crowdfund. `OnlyOwner` is a modifier that allows only the contract owner to run this method.
`destroyFrom( address account, uint256 amount ) external OnlyCrowdsale`	is used to destroy tokens. The burning process occurs when a crowdfund is canceled. In this case, the tokens that have been distributed are destroyed. Only the owner of the crowdfund can cancel it, so it is natural that a modifier is used to prevent anyone else from doing so.

Table A.1: WarehouseToken Methods

`address[] private _siblings`	An array containing the crowdsale processes that refer to the same property but have a different goal. Multiple crowdsales might take place at the same time.
`uint256 private _rate`	the exchange rate from DAI to `WarehouseToken`

`uint private _openingTime` and `_closingTime`	are timestamps that indicate the initiation and completion of the crowdfund.

`setSiblings( address payable[], memory crowdsales ) external onlyOwner`	used to create a list of currently active crowdfunds.
`siblingsGoalReached( ) public view returns (bool)`	is used to determine whether any of the other crowdfunding processes have been completed
`buyTokensWithDai()`	offers the same functionality as `buyTokens()`, but in this case the purchasing account uses DAI for the transaction.
`_preValidatePurchase() internal view()`	is used to validate the transaction before it takes place.
`_deliverTokens() internal`	uses the `WarehouseToken` function `safeTransfer()` to deliver tokens to the selected account.
`_processPurchase() internal`	calls the `deliverTokens()` function, as well as log the transaction on the `__deposits` map.
`refundAllowed(address) public view returns (bool)`	is used to determine whether an account can have their deposit refunded.

Table A.3: WarehouseCrowdsale Methods

`uint8 public avgBlockTime`	is the average time for a block to be added to the blockchain. The contract measures uses block time for calculations. It is considered to be around 13 seconds.

`uint8 public rentalLimitMonths`	is the number of months the renter can pay in advance. By default, that number is 12
`uint256 public rentalLimitBlocks`	is the number of months the renter can pay in advance, expressed in block time.
`uint8 private decimals`	is the number of decimals of the tokens. In this implementation the number is 0.
`uint256 public totalSupply2`	is used to check that the amount sent for rent will not produce a float when the accumulated profit is divided among shareholders.
`uint256 public blocksPer30Day`	is the number of blocks that are added to the blockchain in a month

Table A.4: realEstate State Variables

`limitadvancedrent() public onlyPropOwner`	the main property owner uses this function to determine how many months of rent can be payed in advance. Emits a `PrePayRentLimit()` event.

Table A.5: realEstate Main property-owner functions

`transfer() public returns (bool)`	is used to transfer shares from one account to another.
`claimOwnership() public`	can be used by a shareholder if they accumulate 50% or more of the total token supply. In this way, they can claim ownership of the property and become the main owner of the property. Emits a `MainPropertyOwner()` event.

Table A.6: realEstate Stakeholder functions

`_propose()`	assigns a proposal ID to the new proposal. Checks the inputs, and the proposal state. Creates a new proposal and places it inside the `_proposals` map. In the end, it emits a `ProposalCreated()` event.

Table A.7: Governor proposal function

`castVoteWithReason()`	is used to cast a vote and provide a reason
`castVoteWithReasonAndParams()`	is used to cast a vote and provide a reason and certain encoded parameters
`castVoteBySig()`	is used to cast a vote using the voter signature
`castVoteWithReason...` `...AndParamsBySig()`	is used to cast a vote, provide a reason, add encoded parameters and use a signature
`_castVote()`	By implementation, all the functions listed above, call the internal function `_castVote`. The function checks if the state of the proposal is valid for casting a vote. The current voting weight of an account is calculated, and the vote is registered to the proposal. The function emits a `VoteCast()` or a `VoteCast()` event, depending on the parameters. Finally, the tally for the proposal is updated.

Table A.8: Governor vote casting functions

`_queueOperations()`	is the function that implements queueing. To do so, it must be overridden. The default function is empty, because the contract can be used without queueing as well
`_executeOperations`	is the contract's internal function for proposal execution. Using a `for` loop the function ensures that all the functions of the target contracts are called.

Table A.9: Governor Queue and Execute Functions

`_cancel()`	is the internal function for proposal cancellation. It sets the proposal state to `Canceled` and emits a `ProposalCanceled()` event

Table A.10: Cancellation functions

Appendix B

Implemented Code

Minimal changes were made to the Wareblock contract and the WarehouseCrowdsale contract. The changes significantly reduce the bytecode size of the contracts.

B.1 WarehouseToken

The following shows the changes to the original WarehouseToken contract.

```
7  import "@openzeppelin/contracts/token/ERC20/extensions/ERC20Votes.
       sol";
8  import "@openzeppelin/contracts/token/ERC20/extensions/
       ERC20Burnable.sol";
9  import "@openzeppelin/contracts/token/ERC20/extensions/ERC20Permit.
       sol";
```

```
47     function _update(address from, address to, uint256 amount)
           internal override(ERC20, ERC20Votes) {
48         super._update(from, to, amount);
49     }
50
51     function nonces(address owner) public view override(ERC20Permit,
           Nonces) returns (uint256){
52         return super.nonces(owner);
53     }
```

B.2 WarehouseGovernor

```
4 import "@openzeppelin/contracts/governance/Governor.sol";
5 import "@openzeppelin/contracts/governance/extensions/
     GovernorCountingSimple.sol";
6 import "@openzeppelin/contracts/governance/extensions/GovernorVotes
     .sol";
7
8 contract WarehouseGovernor is
9     Governor,
10    GovernorCountingSimple,
11    GovernorVotes
12 {
13    constructor(IVotes _token)
14       Governor("WarehouseGovernor")
15       GovernorVotes(_token)
16    {}
17
18    function votingDelay() public pure override returns (uint256) {
19       return 1;
20    }
21
22    function votingPeriod() public pure override returns (uint256) {
23       return 50;
24    }
25
26    function proposalThreshold() public pure override returns (
        uint256) {
27       return 10;
28    }
29
30    function quorum(uint256) public pure override returns (uint256)
        {
31       return 40;
```

```
32      }
33  }
```

B.3 RealEstate

Listing B.1: Added: Importing ERC20 libraries

```
5   import "@openzeppelin/contracts/token/ERC20/IERC20.sol";
6   import "@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol
        ";
```

Listing B.2: Replacement of shares with ERC20 tokens

```
15  IERC20 public token;
```

Listing B.3: Removed: Main property owner role

```
26  address public mainPropertyOwner;
```

Listing B.4: Rmoved: shares

```
32  mapping (address => uint256) public shares;
```

Listing B.5: Replacement of share logic, with ERC20 functionality

```
61  constructor (
62      address _token,
63      uint8 _tax,
64      uint8 _avgBlockTime,
65      address[] memory _stakeholders
66  ) {
67      token = IERC20(_token);
68      ...
69  }
```

Listing B.6: Removed: Share logic, now handled by internal ERC20 functions

```
181 function seizureFrom(address _from, address _to, uint256 _value)
```

```
        public returns (bool success) {
182     require(msg.sender == gov, "Only government can seize tokens
            ");
183
184     // This will fail unless the contract has been approved as a
            spender
185     try token.transferFrom(_from, _to, _value) {
186         emit Seizure(_from, _to, _value);
187         return true;
188     } catch {
189         return false;
190     }
191 }
```

Listing B.7: Main property owner logic removed

```
233     function claimOwnership () public { //claim main property
            ownership
234     require(shares[msg.sender] > (totalSupply /2) && msg.sender
            != mainPropertyOwner,"Error. You do not own more than 50%
            of the property tokens or you are the main owner allready
            ");
235     mainPropertyOwner = msg.sender;
236     emit MainPropertyOwner(mainPropertyOwner);
237 }
```