SAPIENZA
UNIVERSITÀ DI ROMA

DEPARTMENT OF COMPUTER SCIENCE

EcoCity

BLOCKCHAIN AND DISTRIBUTED LEDGER TECHNOLOGIES

Professors:
Claudio Di Ciccio

Students:
Alberto Cotumaccio
Giovanni Montobbio

Academic Year 2022/2023

Contents

1 Preface

1.1 Brief description of the DApp

EcoCity is a decentralized application game born with the idea of making people aware of environmental issues. The main objective that led us to carry out such a project is the current environmental and climatic situation, which we consider an urgent matter and which we have decided to deal with in an ingenious and practical way.

To ensure the accuracy and relevance of our project, we conducted a research on scientific reports, articles, and other reputable sources. This research served as the foundation for our gameplay choices and allowed us to integrate real-world solutions and address these challenges basing decisions on scientific knowledge.

Our DApp offers users the opportunity to own a virtual community and make investments about how to allocate resources to address environmental challenges, thus allowing players to learn more about the environmental impact of their choices and take responsibility for the consequences. At the same time, we aim to raise players' awareness of the potential environmental disasters that can occur by simulating various natural scenarios within the game, illustrating the dire consequences that could arise.

Finally, we offer a concrete solution to support the environmental cause, by donating a small percentage of all transactions made by users in the game to a charitable organization focused on environmental preservation. In this way, every player can make a tangible difference and contribute to the preservation of our planet.

This project was developed as the final project of the course "Blockchain and Distributed Ledger Technologies" of the Master's Degree in Computer Science at the University of Rome La Sapienza.

1.2 Outline of the report

In the following paragraph, we will briefly describe the content of each section of the report:

- **Background**: we will start by defining and introducing the fundamental concepts on which our DApp is based.
- **Presentation of the context**: in this section, we will explain in detail the gameplay, the objectives, the actors involved and we will analyze the decision-making process that led us to choose the blockchain technology.
- **Software Architecture**: in this section, we will focus on the design of the application, describing the software architecture and explaining in detail the various UML diagrams regarding the logic of the DApp.

- **Implementation**: In this section we will go into the technical details of the technologies used. We will describe then how we implemented the front-end, the back-end and the smart contract, describing the different pages of the application. Here, we will also dedicate space to the concept of randomness, on which our application is based.

- **Conclusions**: in this concluding section, we will carry out a final analysis of the DApp, highlighting limitations and problems and any future developments.

1.3 Team responsabilities

The project was designed and implemented entirely by Alberto Cotumaccio and Giovanni Montobbio. In the first phase, we wrote a requirements specification where we collected all the ideas and functionalities of the game. Subsequently, as it was the first approach to the blockchain world for both of us, we opted for a collaborative development, sharing responsibilities and tasks equally, especially for the realization of the backend logic of the DApp, the implementation of the Solidity smart contract, and the creation of UML diagrams. Instead, for the front-end side, we divided the pages to be developed to facilitate the work. This approach certainly required effective communication and continuous collaboration, as well as the ability to work together synergically, even with moments of discussion and shared decision-making to resolve any problems or discrepancies. At the same time, we both learned how to implement a smart contract and how users can interact with it.

2 Background

2.1 What is a Blockchain?

Blockchain [1] is a technology that enables the creating of a distributed, immutable and secure ledger of transactions, mantaining a shared list of records. These records are called blocks, and each encrypted block of code contains the history of every block that came before it with timestamped transaction data down to the second, chaining these blocks together, creating the blockchain.

A blockchain is made up of two primary components: a decentralized network facilitating and verifying transactions, and the immutable ledger that network maintains. Everyone in the network can see this shared transaction ledger, but there is no single point of failure from which records or digital assets can be hacked or corrupted.

Due to the integrated ledger, transactions can only be recorded once, which eliminates the possibility of double spending. Moreover, once a transaction is recorded, it cannot be changed, therefore, transactions are called immutable.

When a transaction is made, these are signed with a public key in order to provide the security of the blockchain, and the validation of the transaction is ensured with a decentralized consensus [2] protocol that adds new blocks to the chain. The consensus protocol used depends on the specific blockchain implementation, but the two most common protocols are Proof of Stake (PoS) [3] and Proof of Work (PoW) [4]. As said before all transactions are public, so are visible to all network participants but cryptography is also used to ensure the privacy of the transactions by mantaining user anonimity.

2.2 Key concepts

Ethereum blockchain Ethereum [5] is a blockchain technology that powers the cryptocurrency Ether (ETH) and Ethereum is based on the consensus algorithm Proof-of-Stake (see section 2.3) and enables the deployment of Smart Contracts within the blockchain . The latter are decentralized pieces of code that are executed directly on-chain and are one of the reasons of the success of Ethereum. Smart contracts allow the develop of decentralized applications (DApp) and are fundamental to the so-called Web 3.0.

Smart contracts Smart contracts [6] are computer programs that run on the blockchain and can automate the process of negotiating, in essence are simply programs stored on a blockchain that run when determined preconditions are met and verified. Once a smart contract runs on the blockchain, it behaves like a self-operating computer program. The blockchain is then updated when the transaction is completed. That means the transaction cannot be changed, and only parties who have been granted

permission can see the results. These smart contracts are established in programming code (e.g., Solidity) and are executed based on the requirements defined within the contract. Smart contracts are created using the Ethereum Virtual Machine (EVM).

There are many advantages in using the smart contracts [6]:

- Speed, efficiency and accuracy: Once a condition is met, the contract is executed immediately. Because smart contracts are digital and automated, there's no paperwork to process and no time spent reconciling errors that often result from manually filling in documents.
- Trust and transparency: since there's no third party involved, and because encrypted records of transactions are shared across participants, there's no need to question whether information has been altered for personal benefit.
- Security: blockchain transaction records are encrypted, which makes them very hard to hack. Moreover, because each record is connected to the previous and subsequent records on a distributed ledger, hackers would have to alter the entire chain to change a single record.
- Savings: smart contracts remove the need for intermediaries to handle transactions and, by extension, their associated time delays and fees.

DApps A DApp [7] has its backend code running on a decentralized peer-to-peer network. Contrast this with an app where the backend code is running on centralized servers. A DApp can have frontend code and user interfaces written in any language to make calls to its backend. Furthermore, its frontend can get hosted on decentralized storage. The main characteristics of DApps are:

- Decentralization: DApps operate on Ethereum, an open public decentralized platform where no one person or group has control
- Determinism: DApps perform the same function irrespective of the environment in which they get executed
- Turing completion: DApps can perform any action given the required resources
- Isolation: DApps are executed in a virtual environment known as Ethereum Virtual Machine so that if the smart contract has a bug, it won't hamper the normal functioning of the blockchain network

2.3 Consensus algorithms

Consensus mechanisms [8] [9] have been incorporated in blockchains as a fault-tolerant mechanism for transaction verification. The consensus is utilized to preserve agreement

among the nodes in the network. When the network expands, the number of nodes increases and it is quite challenging to achieve agreement. Public blockchain requires the participation of users for verification and authentication of the transactions. Since blockchain is a dynamic, self-regulating system, it requires the incorporation of a secure mechanism to ensure the authenticity of the transactions, having participants reach agreement on a consensus. Various types of consensus mechanisms have been proposed that differ in terms of their underlying principles and applications.

- **Proof of work** [4]: miners compete to create new blocks filled with processed transactions. The winner shares the new block with the rest of the network and earns some cryptocurrency. The race is won by the computer which is able to solve a math puzzle fastest. This produces the cryptographic link between the current block and the block that went before. Solving this puzzle is the work in "proof-of-work". The canonical chain is then determined by a fork-choice rule that selects the set of blocks that have had the most work done to mine them. The network is kept secure by the fact that the user would need 51% of the network's computing power to defraud the chain. This would require such huge investments in equipment and energy; you're likely to spend more than you'd gain.

- **Proof of stake** [3]: validators create blocks. One validator is randomly selected in each slot to be the block proposer. Their consensus client requests a bundle of transactions as an "execution payload" from their paired execution client. They wrap this in consensus data to form a block, which they send to other nodes on the Ethereum network. This block production is rewarded with cryptocurrency. In rare cases when multiple possible blocks exist for a single slot, or nodes hear about blocks at different times, the fork choice algorithm picks the block that forms the chain with the greatest weight of attestations (where weight is the number of validators attesting scaled by their balance). A proof-of-stake system is secure crypto-economically because an attacker attempting to take control of the chain must destroy a massive amount of ETH. A system of rewards incentivizes individual stakers to behave honestly, and penalties disincentivize stakers from acting maliciously.

Both of these two algorithms provide strong security, nevertheless, POW algorithms consume much more computing power since this is the only way how new blocks can be mined, instead, POS algorithms, need much investment done up front, which enables prominent players in the system to influence the random selection.

2.4 Types of Blockchain

A first differentiation in the blockchain technology is between the permissionless and permissioned blochains. In a permissionless blockchain anyone can join the network

and can take part in any activity without requesting additional permissions and the users within that network are pseudo-anonymous. In a permissioned blockchain the access to the network is restricted, since a user needs to be accepted before entering. Within the network, the identity of all participants is known.
There are four main types of blockchain networks [10]: public blockchains, private blockchains, consortium blockchains and hybrid blockchains. Each one of these platforms has its benefits, drawbacks and ideal uses.

- **Public**: this is where cryptocurrency like Bitcoin [11] originated and helped to popularize distributed ledger technology (DLT). It removes the problems that come with centralization, including less security and transparency. DLT doesn't store information in any one place, instead distributing it across a peer-to-peer network. Its decentralized nature requires some method for verifying the authenticity of data. That method is a consensus algorithm whereby participants in the blockchain reach agreement on the current state of the ledger and Proof of work (PoW) and proof of stake (PoS) are the two most common consensus methods. Public blockchain is non-restrictive and permissionless, and anyone with internet access can sign on to a blockchain platform to become an authorized node. This user can access current and past records and conduct mining activities, the complex computations used to verify transactions and add them to the ledger. No valid record or transaction can be changed on the network, and anyone can verify the transactions, find bugs or propose changes because the source code is usually open source. One of the advantages of public blockchains is that they are completely independent of organizations, so if the organization that started it ceases to exist the public blockchain will still be able to run, as long as there are computers still connected to it. Another advantage is the network's transparency, as long as the users follow security protocols and methods fastidiously, public blockchains are mostly secure. On the other hand there are also some disadvantages, they can be slow and they don't scale well, the network slows down as more nodes join the network. The most common use case for public blockchains is mining and exchanging cryptocurrencies like Bitcoin. However, it can also be used for creating a fixed record with an auditable chain of custody, such as electronic notarization of affidavits and public records of property ownership.

- **Private**: it is a blockchain network that works in a restrictive environment like a closed network, or that is under the control of a single entity. While it operates like a public blockchain network in the sense that it uses peer-to-peer connections and decentralization, this type of blockchain is on a much smaller scale. Instead of just anyone being able to join and provide computing power, private blockchains typically are operated on a small network inside controlled

by a single identity. They're also known as permissioned blockchains or enterprise blockchains. The controlling organization sets permission levels, security, authorizations and accessibility. Because they're limited in size, private blockchains can be very fast and can process transactions much more quickly than public blockchains. The biggest disadvantage of private blockchains is the claim that they are not true blockchains since the core philosophy of blockchain is decentralization. It's also more difficult to fully achieve trust in the information, since centralized nodes determine what is valid. The small number of nodes can also mean less security. If a few nodes go rogue, the consensus method can be compromised. There is no anonimity on a private blockachain, either. This type can be used in internal voting systems or supply chain management.

- **Hybrid**: a type of blockchain technology that combines elements of both private and public blockchain. It lets organizations set up a private, permission-based system alongside a public permissionless system, allowing them to control who can access specific data stored in the blockchain, and what data will be opened up publicly. Typically, transactions and records in a hybrid blockchain are not made public but can be verified when needed, such as by allowing access through a smart contract. Confidential information is kept inside the network but is still verifiable. Even though a private entity may own the hybrid blockchain, it cannot alter transactions. When a user joins a hybrid blockchain, they have full access to the network. The user's identity is protected from other users, unless they engage in a transaction. Then, their identity is revealed to the other party. One of the big advantages of hybrid blockchain is that, because it works within a closed ecosystem, outside hackers can't mount a 51% attack on the network. It also protects privacy but allows for communication with third parties. Transactions are cheap and fast, and it offers better scalability than a public blockchain network. On the other hand this ype of blockchain isn't completely transparent because information can be shielded. Upgrading can also be a challenge, and there is no incentive for users to participate or contribute to the network.

- **Consortium** [12]: is similar to a hybrid blockchain in that it has private and public blockchain features. But it's different in that multiple organizational members collaborate on a decentralized network. Essentially, a consortium blockchain is a private blockchain with limited access to a particular group, eliminating the risks that come with just one entity controlling the network on a private blockchain. The consensus procedures are controlled by preset nodes. It has a validator node that initiates, receives and validates transactions. Member nodes can receive or initiate transactions. A consortium blockchain tends to be more secure, scalable and efficient than a public blockchain network, like private

and hybrid blockchain, it also offers access controls but is less transparent than public blockchains and can be still compromised if a member node is breached.

2.5 Application domain

The blockchain has a wide range of applications [13], from cryptocurrencies to supply chain management, from electronic voting to decentralized finance (DeFi) applications.

- Financial Sector: Blockchain enables faster and more secure transactions, eliminates intermediaries, and reduces costs. It facilitates seamless cross-border payments, smart contracts, and decentralized lending, opening up new avenues for financial inclusion and innovation.

- Supply Chain Management: Blockchains offer end-to-end visibility and traceability in supply chains. They allow tracking and verification of products at every stage, ensuring authenticity, preventing counterfeiting, and enhancing trust. Blockchain-powered supply chains can streamline logistics, improve inventory management, and enable efficient provenance tracking.

- Healthcare: The healthcare industry faces challenges such as data interoperability, patient privacy, and counterfeit drugs. Blockchain solutions can securely store and share medical records, ensuring data integrity and enabling seamless information exchange between healthcare providers. Additionally, blockchain can help track and authenticate pharmaceutical products, ensuring patient safety.

- Identity Management: Traditional identity management systems are prone to data breaches and identity theft. Blockchain provides a decentralized and tamper-proof platform for identity verification and authentication. Individuals can have control over their personal data and selectively share it, enhancing privacy and reducing the risk of identity fraud.

- Voting Systems: Electoral processes can benefit from blockchain technology by ensuring transparent and tamper-proof voting. Blockchain-based voting systems can eliminate duplicate votes, verify voter identities securely, and provide an auditable trail of all transactions, thus enhancing the integrity and trustworthiness of the democratic process.

- Intellectual Property: Protecting intellectual property rights is crucial in the digital age. Blockchain offers a secure and transparent platform for registering and managing intellectual property assets, such as patents, copyrights, and trademarks. It provides proof of ownership, prevents unauthorized use, and facilitates fair licensing and royalty distribution.

- Energy Sector: With the growing focus on renewable energy and peer-to-peer energy trading, blockchain can play a significant role. Blockchain-based energy platforms enable direct transactions between energy producers and consumers, eliminating intermediaries and reducing costs. It also allows for transparent tracking of energy sources and incentivizes sustainable practices.

3 Presentation of the context

In this section, our objective is to explore the motivations that drove the development of our game. We will offer a thorough overview of the gameplay mechanics, highlighting types of users who can interact with our DApp. Finally, we will analyze the choice to base the project on blockchain from a technical perspective, motivating our choice to overcome the limitations of traditional games.

3.1 Overview of EcoCity Features and Objectives

EcoCity DApp is a role-playing game where users own and manage a community with the aim of making it sustainable over time by tackling seven environmental challenges (table 1). In order to play and interact with the blockchain, each user must connect their own MetaMask wallet. After that, the user will have two options:

1. Create a new community from scratch, with a maximum level of pollution, to which you can assign a name of your choice that will no longer be editable. This will only involve paying a small amount of gas to make the transaction in the blockchain.

2. Purchase, via Ethereum, an already existing community that has been put up for sale by another user of the game, by transferring ownership. In this case, the user will have the option to start from an advanced state. 10 percent of the transaction will go to the wallet of an environmental charity to ensure the DApp has a real impact on the world.

Each player cannot own two communities at the same time, but can decide to put it for sale by freely deciding the ethereum price. Only when it is sold to another player then it will be possible to purchase a new community in the shop or create a new one from scratch for free.

Investment of resources In order to minimize the number of transactions and reduce gas fees, we allow users to make a maximum of one investment every 24 hours. This enhances gameplay strategy and prevents excessive trading. During each investment, players can allocate and distribute their available ECO (in-game token) to 15 different resources, each with a value ranging from 0 to 100. Following each investment, there is a 30 percent chance of an unexpected random event occurring, which may have a negative impact on the community. These events specifically target certain prefixed resources (table 2) and result in damage proportional to the severity of the event and the current value of affected resources. The likelihood of these events happening is determined by the random generation of a number, which is discussed in detail in the section 5.3. At the conclusion of each investment, the community's new

score will be updated based on the allocations made and the occurrence of any natural disasters.

Ranking and classification of communities Within the DApp, there is a top-five ranking based on an overall score of communities, which encourages competitiveness among players. The score of a community is calculated based on the performance of the seven environmental challenges it is facing. There are four types of community classification:

- **Eco-friendly**: they are effectively addressing environmental challenges.
- **InTransition**: they are making progress but still need time.
- **Polluting**: They have started the transition but are still behind.
- **AtRisk**: they are not taking actions to address challenges.

EcoCity represents a unique experience that goes beyond the concept of a simple game, as it offers players an engaging, fun, and educational experience. Our main goal is to raise awareness among players about the importance of sustainability and ecology, encouraging them to become agents of change for a more sustainable future.

We have designed EcoCity to be intuitive and user-friendly so that even those who have never played a role-playing game or who are not experts in sustainability can participate and learn. Thanks to this DApp, players have the opportunity to deepen their knowledge of the environmental impact of their choices, making important decisions on how to allocate resources to counter environmental challenges.

Table 1: Each environmental challenge is countered by investing in 4 different resources

Environmental Challenge	**Resources to invest in**
Deforestation [14]	Trees, Organic fields, Sustainable buildings, Parks
Loss of biodiversity [15]	Trees, Birdhouses, Parks, Green Roofs
Scarcity of natural resources [16]	Renewable energy, Water Cisterns, Sustainable Transport, Sustainable buildings
Waste disposal [17]	Waste Recycling Bins, Composting facilities, Second-hand stores, Green Roofs
Climate change [18]	Renewable energy, Sustainable buildings, Sustainable Transport, Parks
Air pollution [19]	Energy-efficient appliances, Air purifiers, Sustainable Transport, Sustainable buildings
Water pollution [20]	Water Purifiers, Water Cisterns, Organic fields, Parks

Table 2: Each disaster has some resources that limit damage and some damaged

Event	**How to mitigate damages**	**Vulnerable resources**
Storm	Water Cisterns	Water Cisterns
	Trees	Organic fields
	Sustainable buildings	Birdhouses
Drought	Water Cisterns	Organic fields
	Second-hand stores	Trees
	Water purifiers	Parks
	Green roofs	
Flood	Water Cisterns	Organic fields
	Water purifiers	Trees
	Parks	Green roofs
		Renewable energy
Epidemic	Sustainable transports	Second-hand stores
	Air purifiers	Sustainable transports
	Low-energy appliances	Parks
Fire	Water cisterns	Trees
	Sustainable buildings	Organic fields
	Air purifiers	Green roofs
	Renewable energy	Renewable energy
Tornado	Water Cisterns	Trees
	Sustainable buildings	Sustainable transports
	Water purifiers	Parks
		Green roofs
		Sustainable buildings

3.2 Actors involved

Just a few key actors are involved, each playing a crucial role in contributing to the overall ecosystem of the game. Mainly the active players can be categorized into two distinct groups: those without a community and community owners, each has its own functionalities.

Another significant actor within the DApp is the environmental organization although has a passive role, because he is not a real player but he can receive the percentages from the transactions because we have specified his wallet address in the smart contract.

3.3 Why using a blockchain (Koens and Poll)

Blockchain games are revolutionizing the gaming industry through non-fungible tokens and system transparency. They offer new research possibilities, new game mechanics,

and innovative narrative perspectives [21].

Over time, several decision-making models have been proposed in the literature and on the web, often contradicting each other and incomplete. We decided to base our decision by referring to the Koens and Poll model [22], which is one of the best flowcharts on the subject. It is a sequence of binary choices leading to a final state that provides the optimal solution for a specific scenario. The scheme also considers few limitations of the blockchain, such as processing a large number of transactions per second or storing large amounts of transactional data. However, we are not interested in these two cases.

In our project, it only makes sense to use a public permissionless blockchain, where we need to store state and where multiple players are unknown. We do not wish to use a trusted third party for state updates. This type of blockchain allows anyone to participate without requiring permissions or authorizations, providing maximum freedom to users and making our DApp more inclusive and accessible to all.

A very important aspect is transparency, a feature of blockchain that improves the reliability of games. Indeed, since the blockchain permanently and securely records all transactions and changes, users can verify the authenticity of information and actions within the game, eliminating the risk of fraud or data manipulation. In addition, creating a game on blockchain, as in our case, allows critical game rules to be implemented via a smart contract, which can be supervised by the players themselves. This ensures the execution of the rules and provides a secure basis for transactions and property exchanges within the DApp. This justifies the need for no intermediaries (trusted third parties).

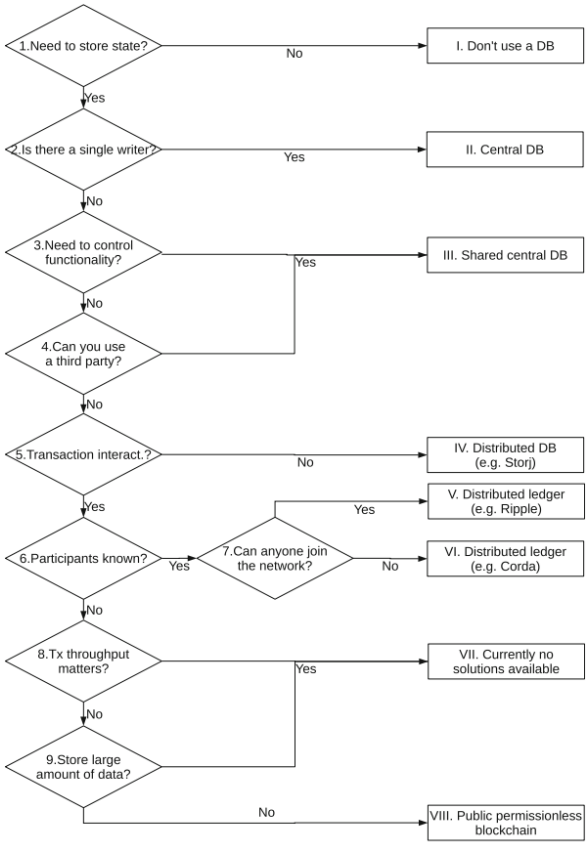

Figure 1: Koens Pool model

4 Software Architecture

Unlike traditional games, blockchain-based games require players to register an address on the relevant blockchain before they can start playing [21]. This address, accessible via a wallet programme such as Metamask in our case, serves as a destination for player-specific virtual resources. In addition, the presence of the smart contract allows the rules of the game to be grouped together, while the game server provides the frontend with the static elements and we entrust it with the generation of the random numbers.

In figure 2 is shown an overview of the tools used during the development of the project. The application architecture consists of a front-end application, implemented with React which uses Web3 to communicate with the blockchain and interact with the smart contract. The players register their address on the blockchain with a Metamask wallet and Ganache is used to quickly fire up a personal Ethereum blockchain and run tests on it.

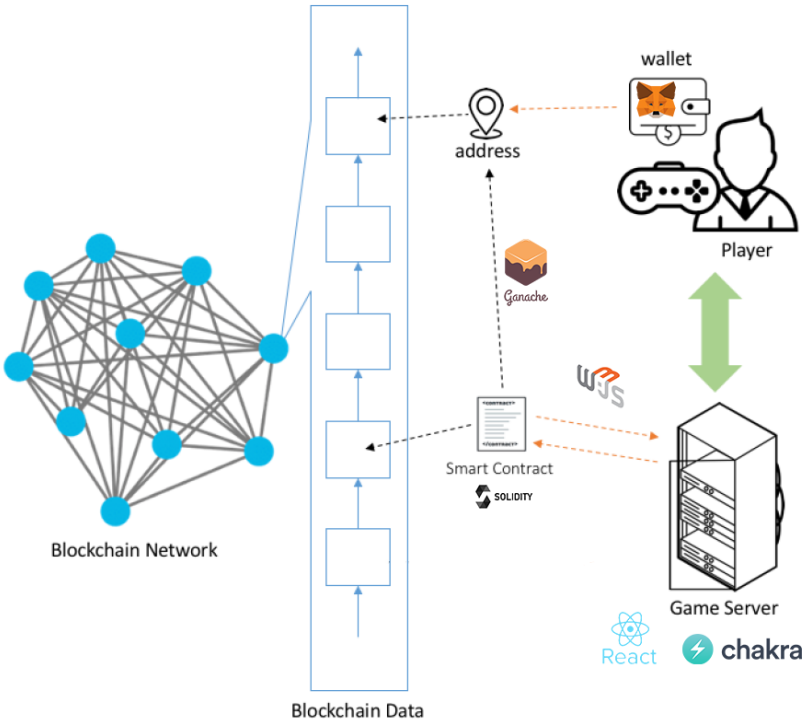

Figure 2: Architecture for EcoCity

Ganache Is a software tool used to create a local blockchain network for testing and developing purposes [23]. Developers may effectively test different scenarios and troubleshoot their blockchain apps by simulating a blockchain network on their local PC with Ganache .

React React is a JavaScript library that provides a component-based architecture, simplifying the creation of dynamic and responsive user interfaces. With React [24], we divided our application into smaller, manageable parts, and this modularity facilitated collaborative development.

Chakra UI For the user interface styling, we utilized Chakra UI [25], a library of pre-designed and customizable components that provided us foundational visual elements. This library also allowed us to employ various modes, such as light and dark themes. It proved to be extremely flexible and intuitive to use, enabling us to build visually appealing and consistent interfaces while keeping our focus on the DApp's logic.

Metamask Metamask [26] is a web browser extension that enables users to interact with the blockchain directly from their browser, eliminating the need to manually input complex addresses or other information. Acting as a bridge between the browser and the Ethereum blockchain, Metamask allows users to manage their Ethereum digital wallets, enabling them to send and receive transactions.

Web3 Web3 [27] is a library that includes a series of modules with specific functionalities for the Ethereum ecosystem. Thanks to Web3, we were able to easily and effectively program the calls to our smart contract's functions and manage transactions with the blockchain.

Remix It is an open-source online tool which allows users to write contracts using Solidity. It is written in Javascript and allows for testing, debugging, and deploying smart contracts [28].

Solidity Solidity [29] is an object-oriented, high-level language for implementing smart contracts. It is influenced by C++, Python, and JavaScript. Solidity is statically typed, supports inheritance, libraries, and complex user-defined types, among other features. With Solidity, users can create contracts for many different uses.

The architecture of the DApp is explained in more detail in the following section with the help of UML diagrams.

4.1 UML Use case diagram

The purpose of a use case diagram is to demonstrate the different ways that a user might interact with a system. In our project we have three different type of users (actors): the unauthenticated user, that is still not authenticated with Metamask, the Metamask user, who is a user that still does not own a community, and a community owner, a Metamask user that owns a community.

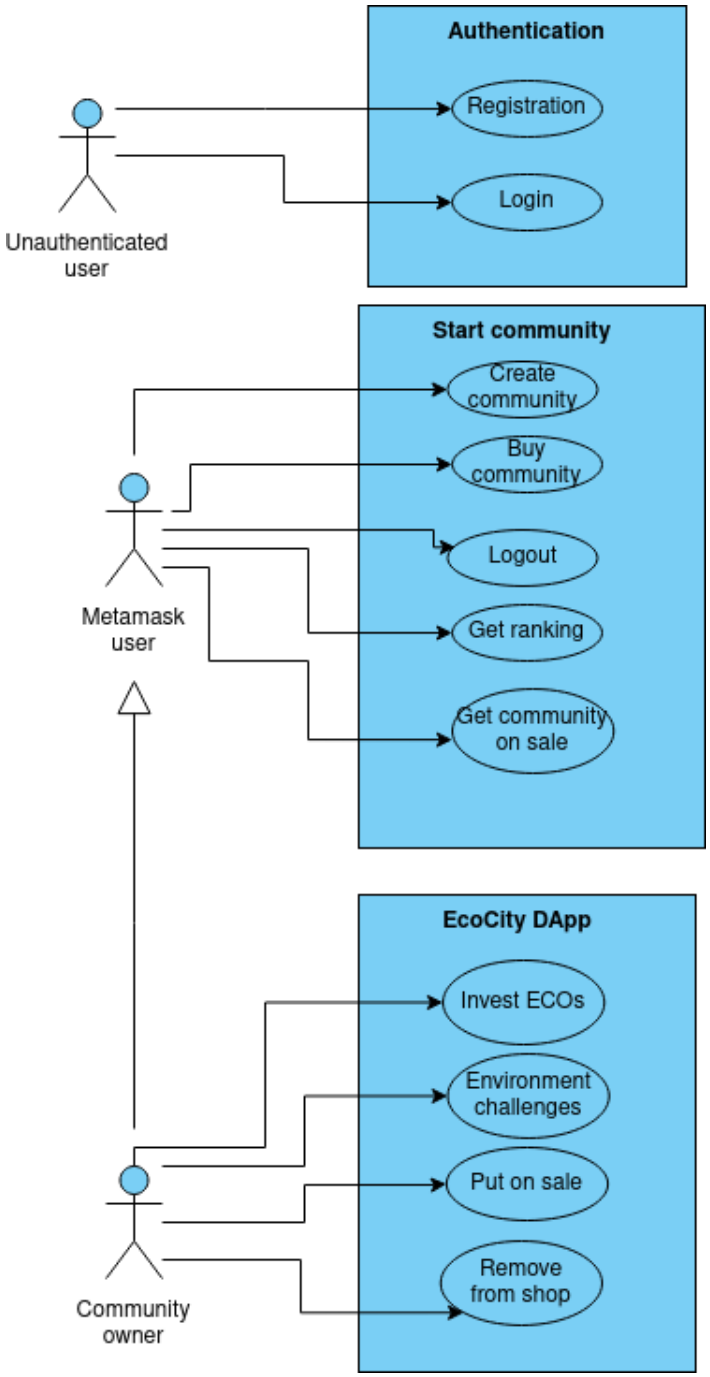

Figure 3: UML use case diagram

4.2 UML Collaboration diagram

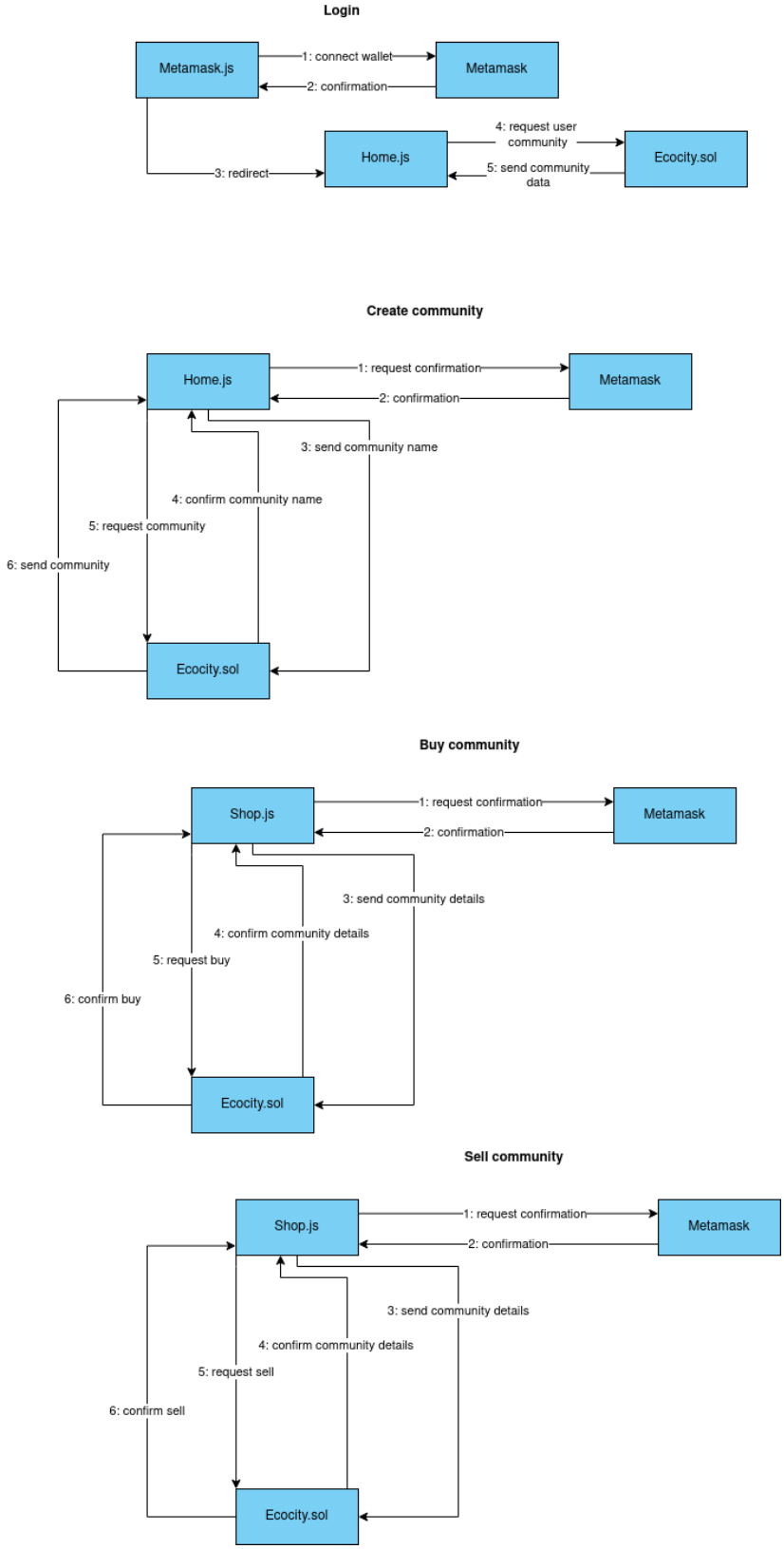

Figure 4: UML collaboration diagram showing the relationship between software objects

4.3 UML Sequence diagram

Sequence diagrams are used to show process interactions arranged in time sequence. They depict the processes and objects involved and the sequence of messages exchanged between the processes and objects needed to carry out the functionality. In our case, we provided the sequence diagram for the four most important steps within our DApp.

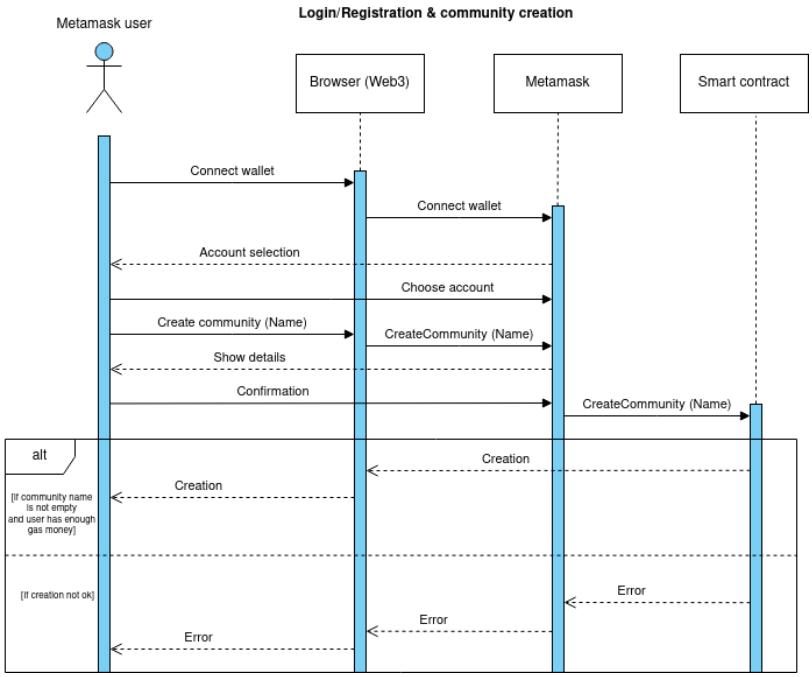

Figure 5: Community creation UML sequence diagram

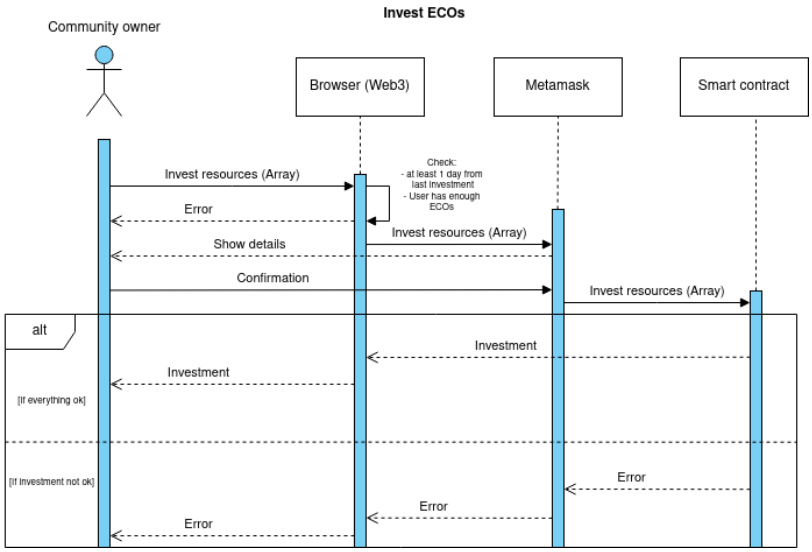

Figure 6: Invest resources UML sequence diagram

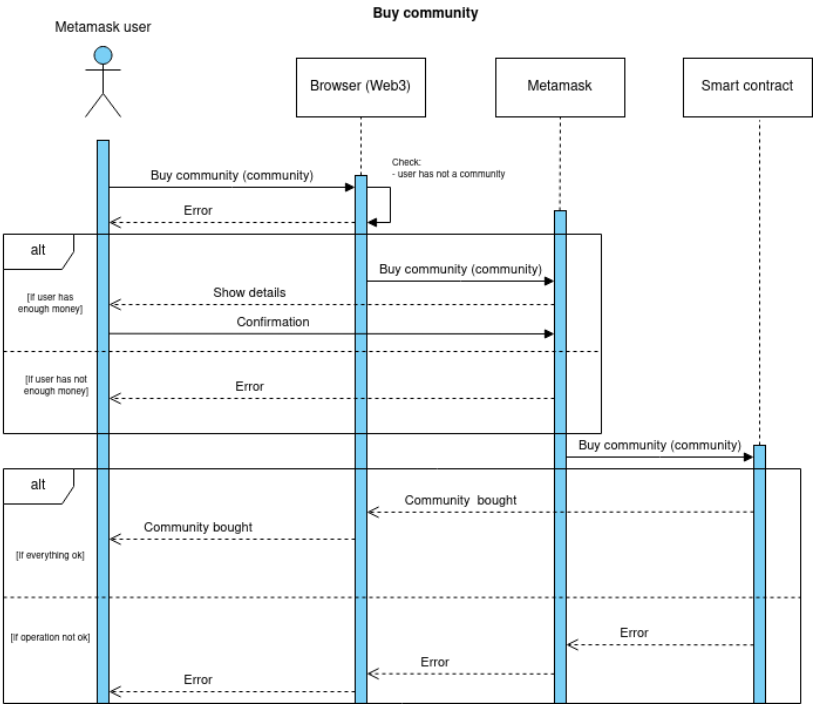

Figure 7: Buy a community UML sequence diagram

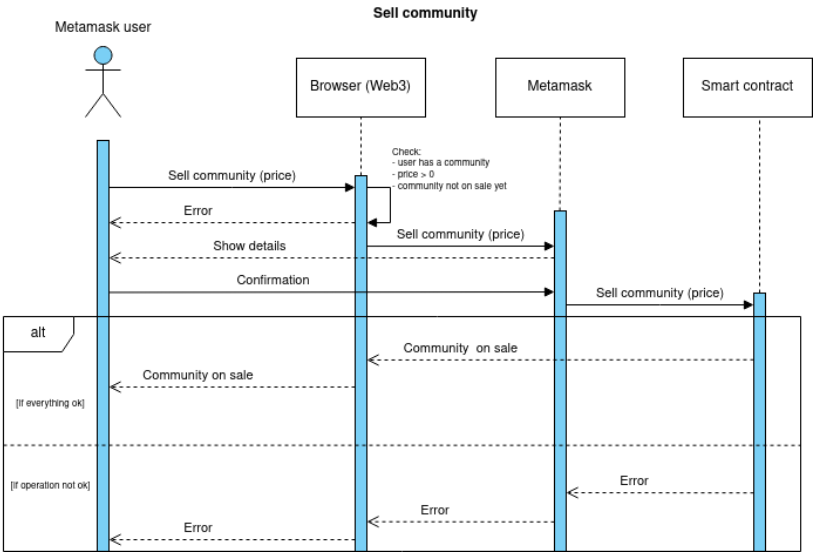

Figure 8: Sell a community UML sequence diagram

4.4 UML Concept Diagram of the smart contract

The concept diagram represents the relations between abstract concepts. The diagram provided below describes the relations of the smart contract.

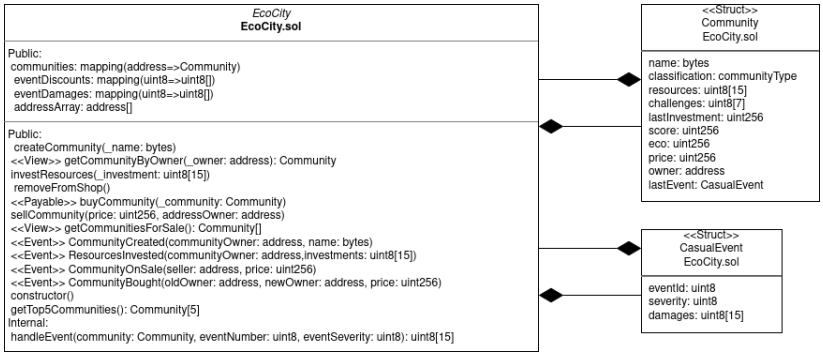

Figure 9: UML concept diagram

5 Implementation

In this section, we will delve into the implementation process with a detailed technical overview. Specifically, we will provide an in-depth description of the front-end, back-end, and smart contract. We will also discuss the concept of randomness, and finally, we will provide an accurate description of the DApp's pages.

5.1 React development

As already mentioned we developed EcoCity using React and Chakra UI libraries, where each page is a separate javascript file. Our main goal was to create an intuitive platform for users.

The core of the implementation resides in the **App.js** file, which represents the nerve center of all navigation. Here, we wrapped the entire application with ChakraProvider to provide the choice of theme. Additionally, using the localStorage.getItem() function, we retrieve the user's selected account address from the browser's local storage and provide it as context to different parts of the application that require it. To ensure smooth navigation within the application, we utilized react-router-dom library. This allowed us to define the different routes and pages of the application, so users can easily move between various sections thanks to that well-structured navigation system. Furthermore, we implemented a logout function to give users the ability to log out, offering flexibility and control over their accounts. Users can choose to disconnect at any time, removing the default account and restoring the login page. Finally we initialize the contract instance, asynchronously obtained using the function initMyContract(), and pass it as input to each page of the project. We will provide a detailed description of this custom function in the next paragraph.

To ensure a modular structure for our project, we delegated the management of the smart contract instance and the connection to the blockchain to the **Contract.js** file. In this file, we imported the ABI (Application Binary Interface) of the EcoCity contract from the EcoCity.json file. The ABI defines the contract's interface and the available methods for interaction. Within the react asynchronous function **initMyContract()**, we create a Web3 object that connects to the local ganache blockchain. This allows us to establish communication between the contract and the local blockchain. Next, we instantiate the contract using the ABI and its address as input parameters. This contract instance will be used in other parts of the application to interact with the contract and perform specific operations.

5.2 Smart contract

To develop the EcoCity.sol smart contract, we included the statement `pragma solidity >=0.4.22 <0.9.0;` to specify the minimum and maximum version of Solidity compatible

with our contract.

Within the EcoCity.sol contract, we defined a struct called **Community** that represents community-related information. The struct includes the following attributes:

- **name**. Is a field of type bytes that offers better storage efficiency compared to the string type. In the frontend, we use the `Web3.utils.hexToUtf8()` function to convert the hexadecimal value of the string into the correct format, allowing its proper display in the user interface.
- **classification**. Is an attribute of type enum that can take values EcoFriendly, InTransition, Polluting, AtRisk, representing the community's classification.
- **resources**. Is an array of 15 values of type uint8 that represent the level of environmental resources. Each array cell corresponds to the value, ranging from 0 to 100, of a particular resource.
- **challenges**. Is an array of 7 values of type uint8 indicating the progress of each environmental challenge.
- **lastInvestment**. It is an attribute of type uint256 that keeps track of the last investment of the community. This value is updated using the timestamp of the current block, then in the react front-end we get it in milliseconds formatting using `Date(lastInvestment * 1000)`.
- **score**. Is an attribute of type uint256 representing the community's total score.
- **eco**. Is an attribute of type uint256 representing the number of available points to invest and distribute across the 15 different resources. Initially, this value is set to 10.
- **price**. Represents the wei price of the community. If the community is not for sale, the value will be 0.
- **owner**. Is an attribute of type "address payable" indicating the address of the community's owner. It is declared as payable so that the address can receive Ethereum transactions if the community is sold.
- **lastEvent**. Is an attribute of type **CasualEvent** which is a second struct containing the event's ID, its severity, and an array of 15 values of type uint8 representing the calculated damage for each resource.

We utilized various data structures to manage information related to communities and events. Specifically, we employed a mapping called **communities** to associate each wallet address with a Community data structure, allowing each wallet to possess

at most one community. Additionally, we maintained an **addressArray** array that keeps track of registered community addresses.

To manage events, we utilized two mappings:

- **eventDiscounts** to associate each event ID with an array of resource IDs that mitigate the event.
- **eventDamages** to associate each event ID with an array of resource IDs that are affected by the event.

These two mappings are initialized in the contract constructor, following the specifications provided in Table 2.

Now we list and describe the functions of the smart contract:

createCommunity The function allows users to create a new community. It requires the community name as input of type bytes. In the front-end, the user enters a string, which is converted to hexadecimal format using the `Web3.utils.utf8ToHex()` function. In the function we included checks with the require clause to ensure that the calling user has not already created a community and that the provided name is valid. Subsequently, all the initial parameters of the community are set, with the classification set to "atRisk" and the resources and challenges set to 0. Finally, the caller's address is added to the array of existing community addresses, and an event is emitted to signal the creation of the community.

getCommunityByOwner The function is invoked whenever the user accesses the home and environmental sections to load it's community data. This function returns the Community data structure associated with the specified input address. Before returning the community, the function includes a check with the require clause to ensure that the owner's address actually has an associated community.

investResources The function allows users to invest resources in their community once a day. To enforce this time restriction, we used a modifier. The function requires an input array of uint8 representing the specific investments for each resource, along with three random numbers: randomNumber, eventNumber, and eventSeverity, which will be discussed in the next paragraph.

Before making the investment, checks are performed to verify that the user is not investing more ECO than he has available. The resources are then updated, taking into account the maximum limit of 100 percent for each resource. Subsequently, the timestamp of the last investment is updated.

The function also handles random events if randomNumber is less than or equal to 30, invoking the internal function handleEvent and updating the last event in the community. Then, the overall score of the community is calculated based on the

available resources. This score determines the community's classification (EcoFriendly, InTransition, Polluting, or AtRisk), and the amount of available ECO for the next investment is updated.

Finally, the community's challenges are calculated according to the specifications outlined in Table 1, and an event is emitted to indicate the user's successful investment of resources.

handleEvent This is an internal function that handles an event within a community. The function takes the community, event number, and severity as inputs. Within the function, two arrays containing specific discounts and damages for the corresponding event are retrieved. Next, the total discount is calculated based on the specific discounts of the event and the available community resources that counteract the event. The discounted amount is calculated as 10 percent of the value of the resources involved in the discounts. Then, damages are calculated for each resource involved in the event. The severity level of the event influences the extent of the damages. A percentage value (25, 50, or 75 percent) is calculated based on the severity and applied to the resources involved to determine the amount of damages. Subsequently, the calculated damages are reduced by the discount. Finally, the damages are subtracted from the community's resources and stored in the output array that contains the damages suffered by the community's resources.

removeFromShop This is a function that requires no input and allows the owner of a community to remove it from the shop. When called, the function accesses the Community data structure associated with the caller's address. We used a require clause to check if the community has a price different than zero, ensuring that it is currently for sale. If the community is for sale, the function sets the community's price to zero, effectively removing it from the store and indicating that it is no longer available for purchase.

buyCommunity The function allows purchasing a specified community provided as input. Initially, it verifies that the caller is not already the owner of the community, as it is not possible to buy one's own community. It also checks if the community is currently available for sale, ensuring it has a non-zero price. If all the requirements are met, the function proceeds with the purchase process. It replaces the current owner's address with the new owner's address in the system's address list, effectively transferring ownership. Additionally, the function transfers the purchase funds from the buyer to the previous owner while retaining a 10 percent commission that is allocated to the charitable organization. After the ownership and fund transfer, the function updates the new owner and the community's price, indicating it is no longer available for purchase. It updates the community mapping by removing the entry for

the previous owner and adding a new entry for the new owner. Finally, an event is emitted to indicate that a community has been purchased.

sellCommunity This function aims to allow the owner of a community to put it up for sale in the shop. When this function is called, the owner specifies a desired ethereum selling price as input, which is converted in wei trought the function `Web3.utils.toWei`. We then verify if the community is already for sale, if the specified price is greater than zero, and if the caller of the function is indeed the owner of the community. If all these conditions are met, the price of the community is set to the desired value, indicating that the community is officially for sale. Additionally, an event is emitted to notify the successful listing of the community for sale.

getCommunitiesForSale This view function aims to provide a list of communities available for sale and is invoked whenever a user enters the shop section. The function starts by counting how many communities are currently for sale: it iterates through all the communities using a for loop and checks if the community's price is greater than zero. Each time it finds a community for sale, it increments a counter. Next, we create an array of communities with a size equal to the number of communities found for sale. This array will store the information about the communities available for purchase. Finally, another for loop is executed to populate the array with the communities for sale, which is then returned as output.

getTop5Communities This view function aims to return the top five communities with the highest score, and it is invoked every time the user enters the ranking section. When called, the function executes a loop that iterates through the array of community addresses. During each iteration, the current address is used to access the information of the corresponding community. If the community has a score greater than zero, the correct position in the final array is determined based on its score. The algorithm used to find the correct position requires a nested loop that is only executed when the current community has a score higher than any of the communities in the array. This way, the communities are sorted based on their scores, with those having the highest score positioned at the top of the array. The loop continues until five communities with scores greater than zero are found or until all community addresses are exhausted.

5.3 Randomness

In EcoCity, the generation of random numbers is a crucial aspect to ensure the unpredictability of natural events that may occur within communities after an investment is made. We want to ensure that it is resistant to malicious attacks, such as attempts to predict or manipulate the generated numbers. We have analyzed various approaches and methodologies used to ensure the randomness and unpredictability of events within

the game context. We have relied on a wide range of sources to understand the challenges and solutions related to secure and reliable random number generation within a DApp.

Keccak256 Our initial approach was to perform pseudorandom number generation within the smart contract. To do this, we can use the cryptographic hash function keccak256 [30], which is a variant of the SHA-3 hash function. One solution involves calculating a hash based on a combination of factors such as the current timestamp and the hash of the previous block, and then applying a series of mathematical operations to scale the generated value within the desired range. However, it is important to note that this approach presents some challenges. Since the smart contract executes on the blockchain, which is a deterministic environment, generating random numbers can be difficult, and the randomness achieved through this technique is limited [31]. This is because transactions on the blockchain must be reproducible, and the keccak256 function is deterministic. There may be a risk of manipulation by malicious actors who could deduce the random numbers to gain unfair advantages or influence the game. While this approach would certainly offer a high degree of transparency, achieving true randomness within a smart contract is a complex challenge.

ChainLink A second approach involves using an external oracle like Chainlink [32]. Decentralized oracles act as a bridge between the blockchain and external data sources, allowing for obtaining random numbers from reliable and unpredictable sources. This methodology provides a high level of security and unpredictability in generating random numbers, effectively mitigating risks [33]. Chainlink is known for its reliability and offers a wide variety of external data sources that can be used for this purpose. However, after careful evaluation of our needs and the costs associated with using the Chainlink external oracle, we have decided not to adopt this approach. Our decision was based on several factors, including technical complexity and the additional cost that the implementation would entail.

Math.random() A third possible approach, which we have adopted, involves generating random numbers on the backend side of the React framework. This allowed us to separate responsibilities and reduce the complexity within the smart contract. Thus, the smart contract can focus on its core functionalities without having to handle random number generation. We utilized the built-in JavaScript library "Math.random()".

In terms of implementation, we created one function responsible for generating three distinct numbers:

- **randomNumber**, has a range between 1 and 100 and is used to determine if a random event will occur within the community. Specifically, if the value falls between 1 and 30, no event will take place.

- **eventNumber**, can take values between 1 and 6 and represents the identifier of one of the six possible natural events.
- **eventSeverity**, has a range between 1 and 3 and represents the severity of the event.

After generating these three numbers, they are passed as arguments to the investResources(...) function of the smart contract. Once invoked, this function handles the players' resource investments and takes into account the randomly generated numbers to determine the occurrence and severity of any events.

5.4 Ecocity DApp

5.4.1 Login

The initial page of the EcoCity DApp is dedicated to login via MetaMask. It features a "Connect Wallet" button that allows the user to interact with the browser extension and choose one Ethereum digital wallet. The connected account is then set as the default account for future interactions to simplify the process. Additionally, the account is saved in the browser's local storage, enabling the user to maintain access. If the user does not have the MetaMask extension, an explanatory error message will be displayed. Otherwise, after successful login, he will be redirected to the HomePage.

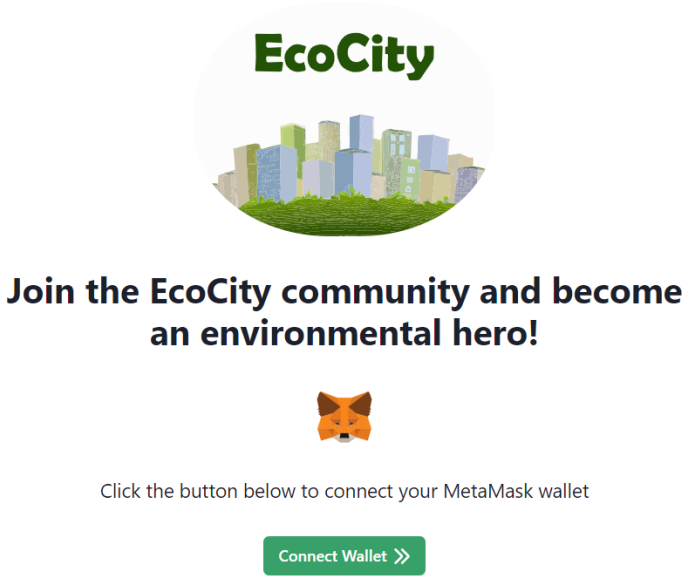

Figure 10: Login Page

5.4.2 Homepage

In the HomePage, it is necessary to distinguish between two cases: whether the user owns a community or not. The elements to be displayed depend on the IF condition

that verifies the call to the getCommunityByOwner method in the smart contract.

In the case where the logged-in user is not the owner of a community, a Box will be shown containing a form where the user can enter the name of the community he wants to create. Once the transaction is finalized using MetaMask, the outcome of the operation will be displayed on the screen using the "Toast" component of ChakraUI. Specifically, if the transaction is successful, the page will be reloaded to display the screen of the newly created community.

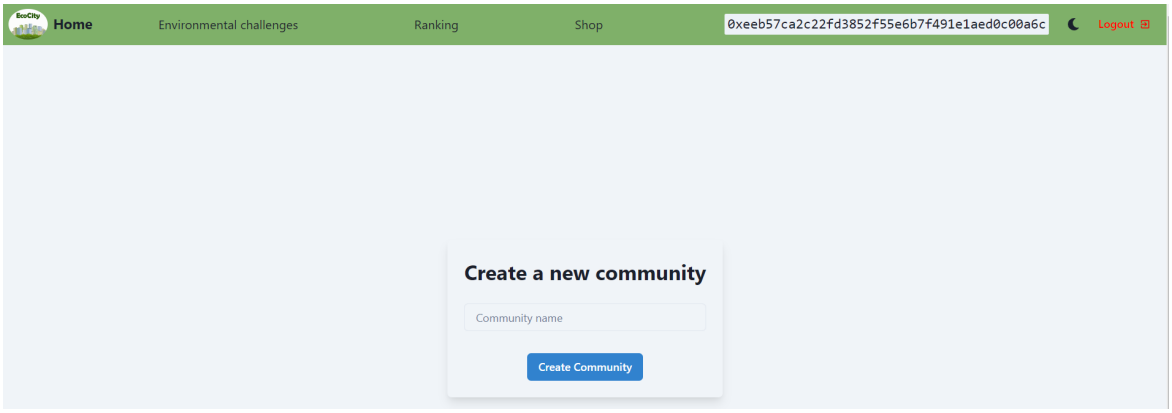

Figure 11: HomePage without owning a community

As mentioned earlier, if the player owns a community, all their information is displayed on the HomePage. In the upper center part of the screen, a dynamic timer has been implemented formatted in hours, minutes, and seconds. This timer is initialized to 24 hours after executing an investment, so that the player can check how much time is left before making the next investment. The main information about the community is shown at the top of the screen, while on the right, information about the last natural event, if any, that occurred within the community after the last investment, is displayed along with its severity. In the center, there is the "Invest" button, which, when clicked, initiates the investment procedure by generating random numbers and preparing the dialogue with the investment function of the smart contract. However, this button is clickable only after the timer (24 hours) has expired. At the bottom, there is a grid containing 15 sustainable resources. Each resource features a stylized image, an info button to learn about the benefits of the resource, a circular progress bar representing the current value of the resource, and a box to increase its level in the next investment without exceeding the total available eco points. Additionally, if a natural event has occurred, is also indicated the decrease in the level of the corresponding resources.

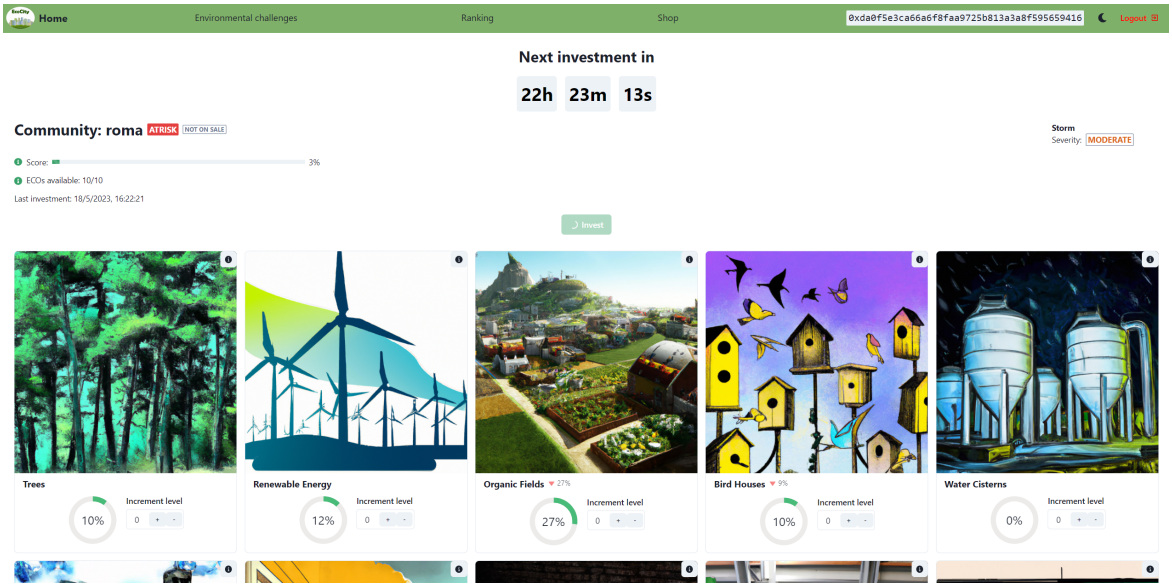

Figure 12: HomePage

5.4.3 Environmental challenges

As in the HomePage, in this section as well, the elements to be displayed depend on the output of the getCommunityByOwner function of the smart contract. If the user owns a community, seven progress bars are shown, each with a value ranging from 0 to 100, representing the seven environmental challenges. Additionally, for each challenge, there is a button that, when clicked, allows reading an accurate scientific description explaining the reason for its presence and how it can be effectively addressed.

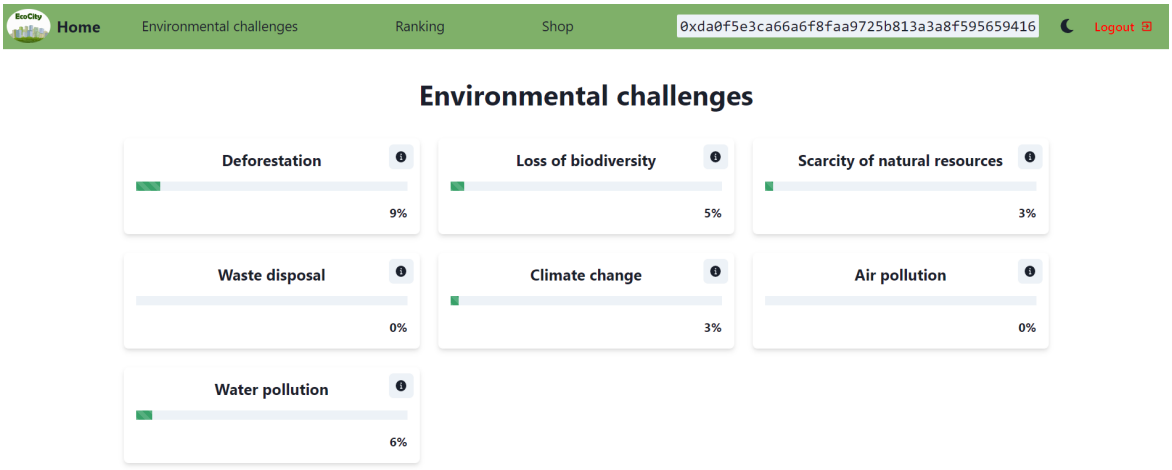

Figure 13: Environmental challenges page

5.4.4 Ranking

In this section, the getTop5Communities function of the smart contract is called to display the ranking of the top EcoCity communities, along with their corresponding

scores. As mentioned earlier, communities with a score of 0 are not considered in the ranking.

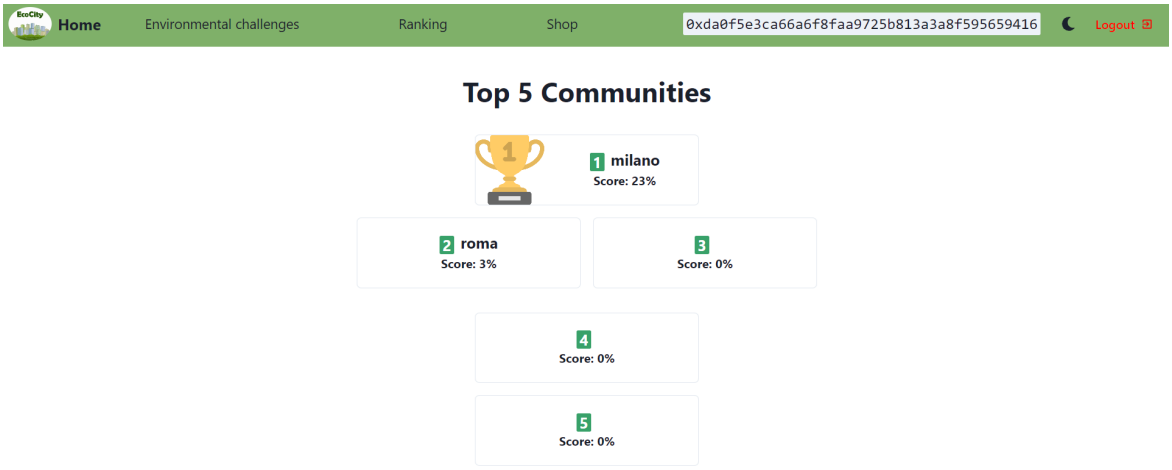

Figure 14: Ranking page

5.4.5 Shop

On this page, there is a form where users can put on sale their communitis at a chosen price in Ethereum. At the same time, there is the option to remove the community from the sales list. In both cases, a dialogue with the smart contract is established. Additionally, there is a grid that lists all the communities currently available for sale in the game. For each community, the following data is displayed: owner's address, classification, price, name, and the percentage of each of the seven environmental challenges. By clicking on the "Buy" button, the smart contract function is invoked to complete the community acquisition.

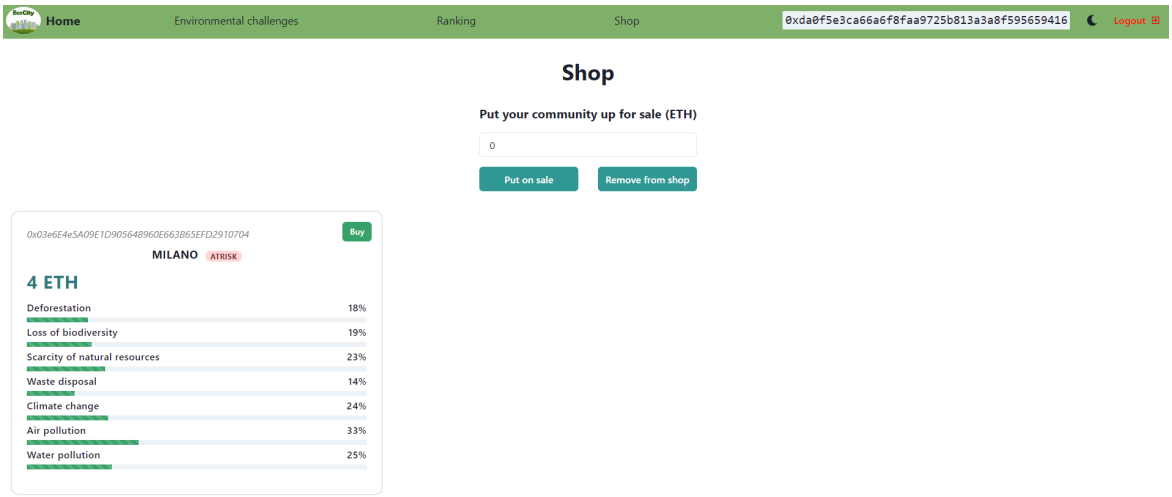

Figure 15: Shop page

6 Conclusions

EcoCity is a decentralized game that aims to raise people's awareness of environmental issues and provide them with tools to tackle environmental challenges in the real world. The main objective of the game is to educate players about the environmental impact of their choices and encourage them to become agents of change for a more sustainable future. While developing EcoCity, we conducted in-depth research based on scientific reports and reliable sources to ensure the accuracy and relevance of the project. This research has allowed us to integrate real-world solutions and scientific knowledge bases to address environmental challenges in the game. However, like any project, EcoCity also has some peculiarities. One of the limitations is that, although the game is free, it requires gas expenses from the player every time he makes a decision about his community, such as when he invests resources or when he decides to put it up for sale.

As for future developments, we have several possibilities in mind to improve and expand EcoCity. A possible future development could be the introduction of virtual animals as extensions that can be purchased in the shop, which would allow players to better understand the importance of biodiversity. Additionally, we may consider adding new environmental challenges to provide players with a greater variety of scenarios to tackle and solve. Finally, we could explore the possibility of further involving environmental organizations and expanding donations to support a wider range of environmental initiatives.

References

[1] PCMag. Blockchain: The Invisible Technology That's Changing the World.

[2] Ethereum. Consensus Mechanisms.

[3] Ethereum. Proof of Stake (PoS).

[4] Ethereum. Proof of Work (PoW).

[5] Vitalik Buterin. Ethereum: A next-generation smart contract and decentralized application platform. *Ethereum.org*, 2014.

[6] IBM. Smart Contracts.

[7] Ethereum. Decentralized Applications (DApps).

[8] Natalia Chaudhry and Muhammad Murtaza Yousaf. Consensus Algorithms in Blockchain: Comparative Analysis, Challenges and Opportunities. *University of Punjab*, 2018.

[9] Bahareh Lashkari and Petr Musilek. A Comprehensive Review of Blockchain Consensus Mechanisms. *IEEE Access*, 2020.

[10] TechTarget. What Are the 4 Different Types of Blockchain Technology?

[11] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. *Bitcoin.org*, 2008.

[12] Wei Yao, Junyi Ye, Renita Murimi, and Guiling Wang. A Survey on Consortium Blockchain Consensus Mechanisms. *New Jersey Institute of Technology University of Dallas*, 2021.

[13] IBM. Blockchain Use Cases.

[14] M. C. Hansen, P. V. Potapov, R. Moore, M. Hancher, S. A. Turubanova, A. Tyukavina, ..., and J. R. Townshend. High-resolution global maps of 21st-century forest cover change. *Science*, 2013.

[15] Secretariat of the Convention on Biological Diversity. *Global Biodiversity Outlook 4*. Montreal, Canada, 2014.

[16] United Nations Environment Programme (UNEP). *Global Resources Outlook 2019: Natural resources for the future we want*. Nairobi, Kenya, 2019.

[17] Daniel Hoornweg and Perinaz Bhada-Tata. *What a Waste: A Global Review of Solid Waste Management*. Washington, DC, 2012.

[18] Intergovernmental Panel on Climate Change (IPCC). *Climate Change 2014: Synthesis Report. Contribution of Working Groups I, II and III to the Fifth Assessment Report of the Intergovernmental Panel on Climate Change*. Geneva, Switzerland, 2014.

[19] J. Lelieveld, J. S. Evans, M. Fnais, D. Giannadaki, and A. Pozzer. The contribution of outdoor air pollution sources to premature mortality on a global scale. *Nature*, 2015.

[20] United Nations Environment Programme (UNEP). *Global Environment Outlook 6: Healthy Planet, Healthy People*. Nairobi, Kenya, 2022.

[21] Tian Min, Hanyi Wang, Yaoze Guo, and Wei Cai. Blockchain Games: A Survey. June 2019.

[22] Koens and Poll. What Blockchain Alternative Do You Need? *Radboud University, Nijmegen, The Netherlands*, 2018.

[23] Truffle Suite. Ganache.

[24] React. React.

[25] ChakraUI. ChakraUI.

[26] MetaMask. MetaMask.

[27] Web3.js. Web3.js.

[28] Remix. Remix.

[29] Solidity. Solidity Documentation.

[30] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. Keccak: The New SHA-3 Encryption Standard. *Journal of Cryptology*, 2011.

[31] Peng Qian, Jianting He, Lingling Lu, Siwei Wu, Zhipeng Lu, Lei Wu, Yajin Zhou, and Qinming He. Demystifying Random Number in Ethereum Smart Contract: Taxonomy, Vulnerability Identification, and Attack Detection. *IEEE Transactions on Software Engineering*, 2023.

[32] Sergey Nazarov. Chainlink: A Decentralized Oracle Network. In *Proceedings of the International Conference on Financial Cryptography and Data Security*, 2019.

[33] Mudabbir Kaleem and Weidong Shi. Demystifying Pythia: A Survey of ChainLink Oracles Usage on Ethereum. In *Proceedings of the 25th International Conference on Financial Cryptography and Data Security's 1st Workshop on Decentralized Finance (FC DeFi 2021)*, Houston, Texas, USA, 2021. University of Houston.