SAPIENZA
UNIVERSITÀ DI ROMA

Department of Computer Science

4Ever

Blockchain and Distributed Ledger Technologies

Professor:

Claudio Di Ciccio

Students:

Andrea Massignan

Cesare Corsi

0. Index

1. Introduction

1.1 Team Members

The project was developed by a team of three:

- **Andrea Massignan**: Full Stack Developer.
- **Cesare Corsi**: Backend Developer and Smart Contract Developer.
- **Ketevan Sikharulidze**: Tester and Report Lead Writer.

1.2 Background Of 4Ever

4Ever emerged from the prospective of us video-games lovers and blockchain enthusiasts. Inspired by MMORPGs and blockchain's potential, we envisioned a decentralized app to redefine in-game economies.

The idea stemmed from beloved online games like World of Warcraft, sparking exploration into enhancing gaming engagement through asset ownership. With blockchain's decentralized nature and NFTs, we saw an opportunity to empower players with true ownership.

By merging MMORPG fun with blockchain security, 4Ever aims to create a fair gaming environment where players control their assets. The vision is to establish a decentralized gaming ecosystem where ownership is transparent, secure, and player-centric, democratizing gaming.

2. Blockchain and Decentralized Applications (DApps)

2.1 Introduction to Blockchain Technology

The first Blockchain that was created was Bitcoin **[1]** in 2009 by Satoshi Nakamoto (presumably a fake name). He was trying to invent digital cash. Digital cash is a file and a file can be copied, meaning we can create more Bitcoin (although this is a very hard process).

The **Blockchain** serves as a collectively maintained and unchangeable ledger of transactions. Its key components include:

- **Nodes**: Participants in the blockchain network who can engage in transactions through externally owned accounts. Miners, a subset of nodes, propose new blocks through a consensus mechanism.

- **Transactions**: Integral units of the ledger representing asset transfers between accounts.

- **Blocks**: Structures within the blockchain containing transactions along with additional fields such as headers and hashes referencing previous blocks.

Ethereum [2] is a blockchain technology underlying the cryptocurrency Ether (ETH). It operates on the Proof-of-Stake consensus algorithm and facilitates Smart Contract deployment within the blockchain. Smart Contracts are decentralized pieces of code executed directly on-chain, playing a crucial role in Ethereum's success and enabling decentralized applications (DApps) known as Web 3.0.

Smart Contracts are computer programs on the blockchain automating negotiation processes. Stored on the blockchain, they execute when predefined conditions are met, behaving as self-operating programs. Once executed, transactions are immutable, visible only to authorized parties. Smart Contracts, typically written in languages like **Solidity [3]**, are executed using the Ethereum Virtual Machine (EVM).

Advantages of **Smart Contracts** include:

- **Speed, efficiency, and accuracy**: Instant execution upon meeting conditions, eliminating manual processing and errors.
- **Trust and transparency**: Transactions are transparent and immutable, ensuring integrity without third-party involvement.
- **Security**: Encrypted records on a distributed ledger make hacking difficult, as altering one record requires changing the entire chain.
- **Savings**: Elimination of intermediaries reduces transaction delays and fees associated with traditional processes.

2.2 Overview of Decentralized Applications (DApps)

A Decentralized Application (DApp) operates its backend code on a decentralized peer-to-peer network, distinguishing it from traditional applications where the backend runs on centralized servers. Unlike traditional apps, a DApp's frontend code and user interfaces can be written in any language, allowing seamless interaction with its backend. Additionally, its frontend can be hosted on decentralized storage systems.

The key characteristics of DApps are as follows:

- **Decentralization:** DApps leverage platforms like Ethereum, an open public decentralized network where control is not centralized with any individual or group.
- **Determinism:** DApps consistently perform the same function regardless of the execution environment, ensuring predictable behavior.
- **Turing Completeness:** DApps possess the capability to execute any action provided with the necessary resources, offering extensive computational capabilities.

- **Isolation:** DApps execute within a virtual environment known as the Ethereum Virtual Machine (EVM), ensuring that any bugs or errors within a smart contract do not disrupt the normal operation of the blockchain network.

2.3 Consensus

Consensus mechanisms are vital components of blockchain systems, ensuring transaction verification and maintaining agreement among network nodes. However, achieving consensus becomes increasingly challenging with the growth in the number of nodes.

Various types of consensus mechanisms have been proposed, each with distinct principles and applications. Two commonly employed mechanisms are:

- **Proof of Work (PoW):** Miners compete to create new blocks and solve mathematical puzzles quickly. The winning miner shares the new block with the network and earns cryptocurrency. PoW ensures network security by requiring an attacker to possess 51% of the network's computing power to manipulate the chain. The canonical chain is determined by a fork-choice rule that selects blocks based on the most work done to mine them.
- **Proof of Stake (PoS):** Validators are responsible for creating blocks, with one validator randomly selected as the block proposer. The proposer gathers a bundle of transactions and forms a block, which is then distributed to other network nodes. Block production is rewarded with cryptocurrency. In PoS systems, the fork choice algorithm selects the block with the highest weight of attestations when multiple blocks exist for a single slot. PoS ensures security as controlling the chain requires the destruction of a significant amount of cryptocurrency. Rewards incentivize honest behavior, while penalties discourage malicious actions.

Although both PoW and PoS mechanisms offer robust security, they differ in several aspects:

- PoW algorithms necessitate substantial computing power, as mining new blocks relies solely on computational effort.
- PoS algorithms require an upfront investment, allowing influential participants to have a greater influence on the random selection process.

2.4 Role of Non-Fungible Tokens (NFTs) in asset ownership

Non-Fungible Tokens (NFTs) are transforming asset ownership in gaming, providing unique advantages such as individual ownership representation, immutable records, interoperability, and enhanced monetization.

In contrast to fungible cryptocurrencies, NFTs are distinct tokens representing specific assets. Their blockchain records are immutable, ensuring the integrity of ownership and safeguarding against tampering or counterfeiting. Furthermore, NFTs are interoperable across platforms, increasing asset liquidity and utility.

Creators leverage NFTs to monetize digital creations by tokenizing them and selling them directly to collectors in decentralized marketplaces. The value of NFTs is determined by market demand and community sentiment, fostering a dynamic ecosystem of digital asset exchange.

2.5 Advantages of using Blockchain for gaming applications

The integration of blockchain technology into gaming applications offers a myriad of benefits that enhance the gaming experience significantly. These advantages include:

- **Ownership Assurance:** Features like non-fungible tokens (NFTs) ensure players have genuine ownership of in-game assets. This instills confidence in players' ability to trade, sell, and possess virtual items within and beyond the game environment.
- **Secure and Transparent Transactions:** Blockchain's transparent and secure transaction system minimizes the risk of fraud or manipulation. Each transaction is securely recorded on an immutable ledger, providing verifiability to players and developers alike.
- **Improved Player Trust and Engagement:** Blockchain's transparency and security foster trust and engagement among players. With assurance in the security of their in-game assets, players are more likely to participate actively and invest in the game.

3. Overview of 4Ever

3.1 Concept and Objectives

4Ever aims at simulating online gaming elements with a decentralized application (DApp) technology, modifying the gaming experience by leveraging blockchain's transparency and security.

Drawing inspiration from MMORPGs systems, such as World of Warcraft, 4Ever entrust players with complete control over in-game assets and experiences.

In particular we aim to achieve the following objectives:

1) **In-Game Economy Management:** 4Ever has a system for securely buying and selling virtual assets via blockchain, ensuring transparency and fairness in all transactions.

2) **Asset Ownership via NFTs:** By leveraging Non-Fungible Tokens (NFTs), 4Ever provides players with true ownership of virtual assets, preventing duplication or manipulation of in-game items.

3) **Simulated Reward System through Quests:** Players participate in quests within the game, earning rewards that are securely recorded on the blockchain. This fosters community engagement and incentivize player interaction.

3.2 Blockchain Please

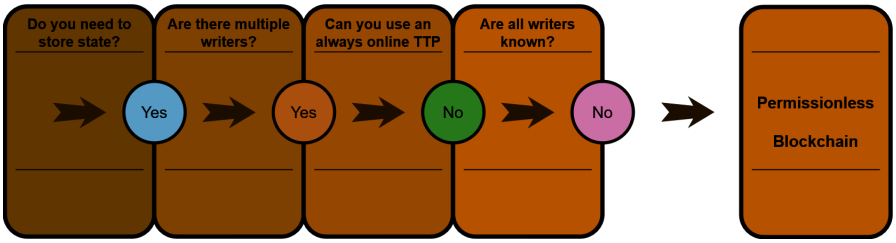

Figura 3.1: Blockchain

3.3 Key features of the 4Ever DApp

NFT Structure: Each NFT within 4Ever includes attributes such as an image (URL), description, owner information, and rarity category. Future enhancements may involve the integration of 3D models to enhance the gaming experience.

Quest System: The quest system in 4Ever generates quests that are supposed to be sponsored by gaming companies who offer the NFT as the reward, ranging from varying levels of difficulty and rarity of the reward itself. Players participate by paying a fee to the company, and upon successful completion, they receive unique NFTs. The value of these NFTs is technically determined by the combined participation fees.

Market System: 4Ever includes a marketplace where players can trade NFTs with other players. The marketplace is designed to facilitate secure and transparent transactions, for now only allowing to buy and sell NFTs at a fixed price.

3.4 Advantages

The use and integration of blockchain technology in 4Ever offers several advantages, including:

- **Ownership Assurance:** Players have genuine ownership of in-game assets, preventing duplication or manipulation of items.

- **Secure Transactions:** Transactions are securely recorded on the blockchain, ensuring transparency and integrity.

4. Technical Architecture of 4Ever

4.1 Client and Server Architecture

The 4Ever DApp is built on a client-server architecture, with the client-side comprising the user interface and the server-side comprising the backend and smart contracts.

The client-side is responsible for interacting with the user, while the server-side handles the processing and storage of data while communicating with the blockchain network.

Specifically the client-side is built using **Vite [4]** and **React [5]**, while the server-side is built using **NodeJS [6]** and **Express [7]**.

4.2 Overview of the underlying Blockchain Technology

To simulate the blockchain environment and the interaction between the application and the smart contract we used the **Ganache [8]** blockchain. This software enables to create a small, local and personal blockchain based on **Ethereum** for the development, testing and deployment of our DApp.

We also used the **Truffle [9]** framework, that allowed us to compile and debug our smart contract

4.3 Smart Contracts Implementation

Our design choises led to a single smart contract implementation that handle all the functionalities of the 4Ever DApp. The contract manages the three fundamental parts of our application: Market, Quests and NFTs.
In the first part of the contract we have defined three structure:

- `NFT`: represents the exchangeable assets. It has three main part: a *tokenId*, an *owner* and a *url*, the first element represents a unique identifier in the context of the market, the second one represent the owner of the NFT and the last one describes the location of the asset in the company databases.
- `User`: represents the member of the market. For each user who has joined the marketplace we have stored a list of all the NFTs that the user has purchased or won.
- `Quest`: represents the single quest published by a company. This structure stores all market members participating the quest, the winner of the quest itslef and the company that published it.

Apart from the functions, the contract keeps track of all the users who has joined the marketplace, all the NFTs on sale in the market and all the quests published by different companies.
Twelve functions have been implemented:

- `joinMarketplace`: allows a user to join the marketplace, if he is not already a member.
- `isUserMember`: checks if a user is alredy a member.
- `getSellNFTs`: returns all NFTs for sale.
- `sellNFT`: adds to the market a NFT. The NFT is added on the market array and a flag is setted in the user's NFT to mark it as on sale.
- `unsellNFT`: removes from the market a NFT. This function "revert" the procedure of the previous one.
- `buyNFT`: allows a user to buy a NFT from the market. First of all removes the NFT from the available ones and then change the ownership of the NFT transfering the coins from the ol owner to the new one.

- `joinQuest`: allows a user to join a quest. The user spend a fixed amount of money to partecipate to a quest. This procedure is irreversible and its done when a certain threshold of minimum participants is exceeded. In the meantime the participant is registered on a off-chain database.
- `endQuest`: end a particular quest. Set the winner of the quest via a random selection and create a new NFT which will be inserted in the winner's NFTs' list.
- `getQuestSeed`: returns the seed of a quest. The seed is calculated and used to determine the winner of the quest.
- `getQuestParticipantsNumber`: returns the participants number of a quest.
- `getNFTs`: returns all the NFTs possessed by an user.
- `calculateRarity`: calculates the rarity of a NFT based on the participants number.

4.4 Smart contract's UML Class Diagram

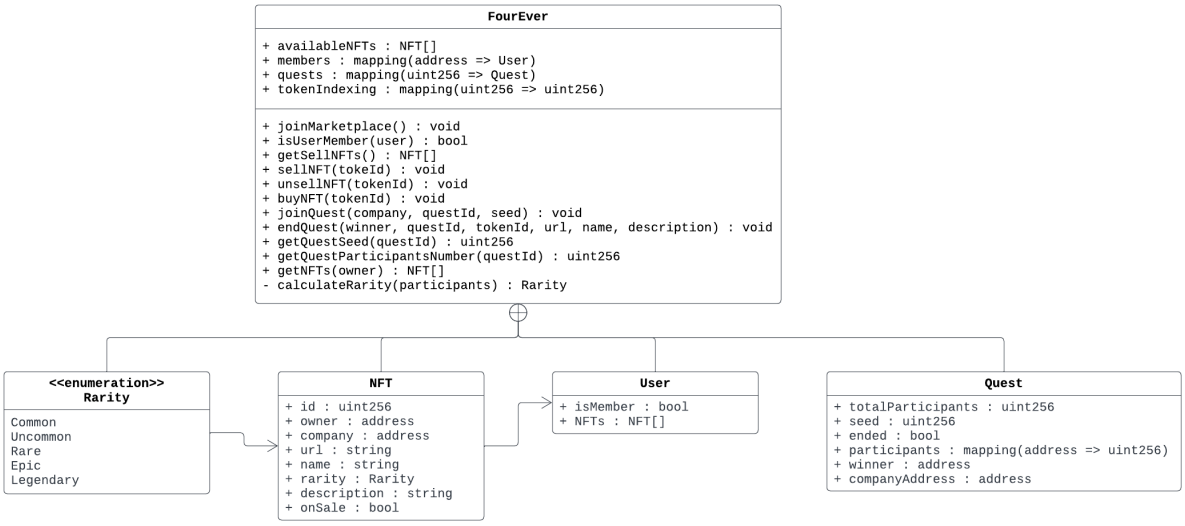

Figura 4.1: UML diagram of the smart contract class

4.5 Integration of NFTs for asset ownership

In our application an NFT is essentially a way to describe and identify uniquely one assets that belong to a particular game. In reality the single asset should be provided by the company who publishes the quest. The true asset should be stored in the company's database and the NFT its one way to associate a user to his asset.

When it comes to implement the asset generation, we used a Stable Diffusion model to create an image based on a description: when a user wins a quest an image is generated and it is stored in a directory inside the project to simulate the company database.

The *url* inside a NFT refers to the path in the directory but in the real word can be used to identify the path or the information to access the real database. Obviously we simulate all of this in a very simple way without generating useless overhead.

5. In-Game Economy Management

5.1 Asset Management and Ownership

Assets in 4Ever are managed and owned through the use of Non-Fungible Tokens (NFTs) as it was previously mentioned.

What we tried to create here is the representation of what should be a typical asset in an online game, meaning that the NFTs are supposed to represent items, weapons, armors, and so on.

Since the NFTs are also provided by the companies that sponsor the quests, they are also supposed to represent the rarity of the item by the number of users that partecipated in the quest.

Typically when a users first join this platform starts from zero, meaning that they don't have any NFTs in their wallet. The only way to get them is by participating in quests, and the only way to participate in quests is by paying a fee to the company that sponsors the quest.

After that the user receives complete ownership of the NFT, and they can decide to keep it and use it in game, or sell it.

5.2 Introduction to Quests

This is the sequential diagram representing the interaction between the user and the smart contract when the user wants to participate in a quest, and all the steps that are involved in this process.

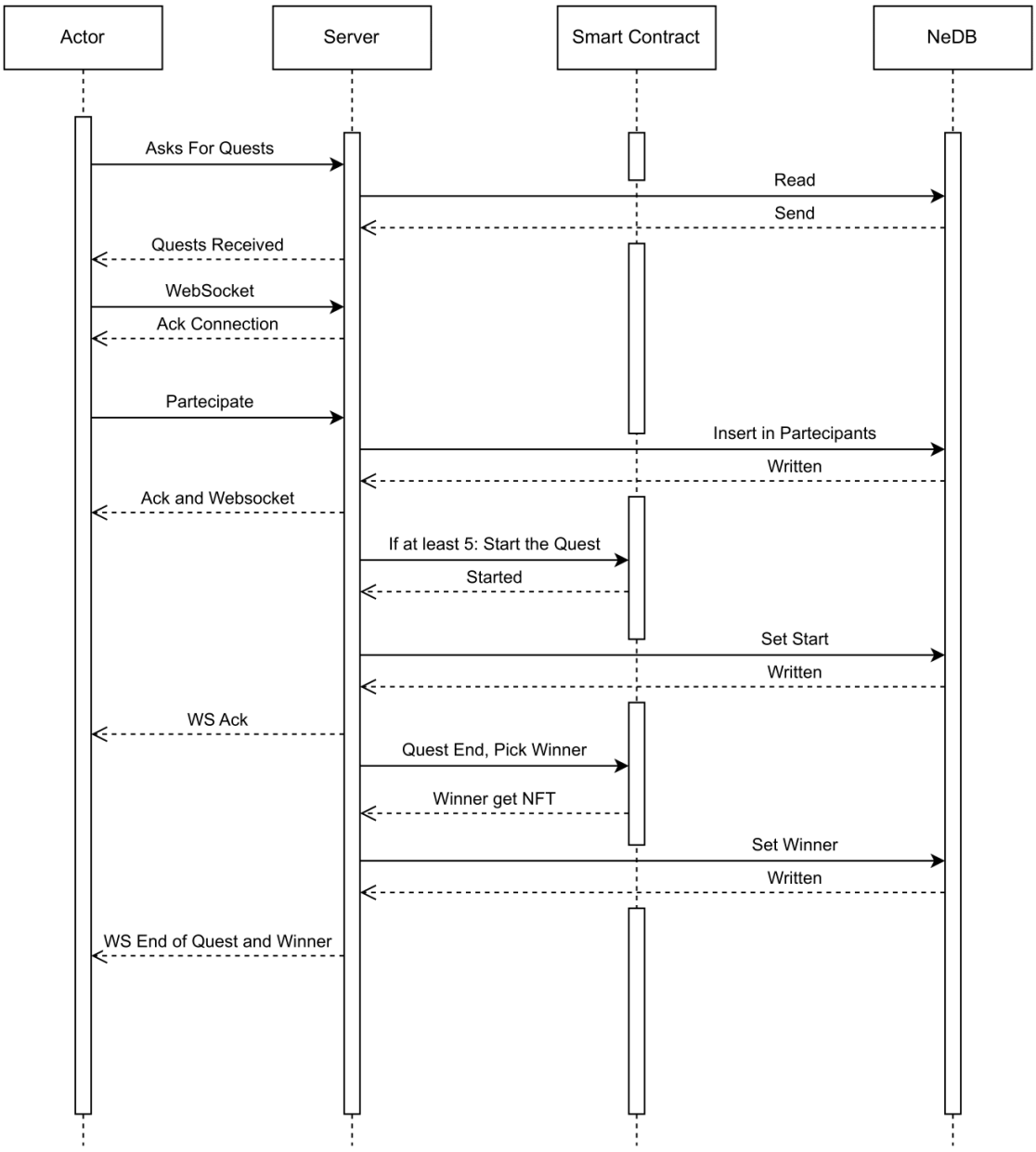

Figura 5.1: Quests

5.3 Design and Implementation of Quests

Since we lack the integration with actual gaming companies, the quest system is simulated.

The server reads a list of quests from a JSON file and if there are new quests, the are added in the **NeDB** database.

A quest is structured as follows:

- **name:** The name of the quest.
- **description:** A brief description of the quest.
- **startDate:** The date when the quest starts.
- **expirationDate:** The date when the quest ends.
- **participants:** An array of the users that partecipated in the quest.
- **usersThreshold:** The threshold of users that need to partecipate in the quest in order to start it (**min 5**).
- **questRegistered:** If the quest is registered in the blockchain and ready to be started.
- **questEnded:** If the quest is ended.
- **winner:** The user that won the quest.
- **sponsor:** The name of the company that sponsored the quest.
- **companyaddress:** The address of the company on the blockchain.
- **_id:** The unique identifier of the quest.

On the smart contract side only the following methods are implemented to reduce the complexity of the contract:

- **Participants Number**
- **Seed** → For random winner selection
- `bool` **ended** → To check if the quest is ended

- A mapping of the partecipants to calculate the final quote of the NFT
- **Winner address**
- **Company address**

5.4 Reward Distribution Mechanisms

The reward distribution mechanism is based on the number of users that partecipated in the quest. When the quest start each user that wants to partecipate in the quest has to pay a fee to the company that sponsored the quest.

The company in return will provide to the user that has won an NFT, that represents the reward of the quest.

5.5 Market System

This is the sequential diagram representing the interaction between the user and the smart contract when the user wants to buy or sell an NFT, and all the steps that are involved in this process.

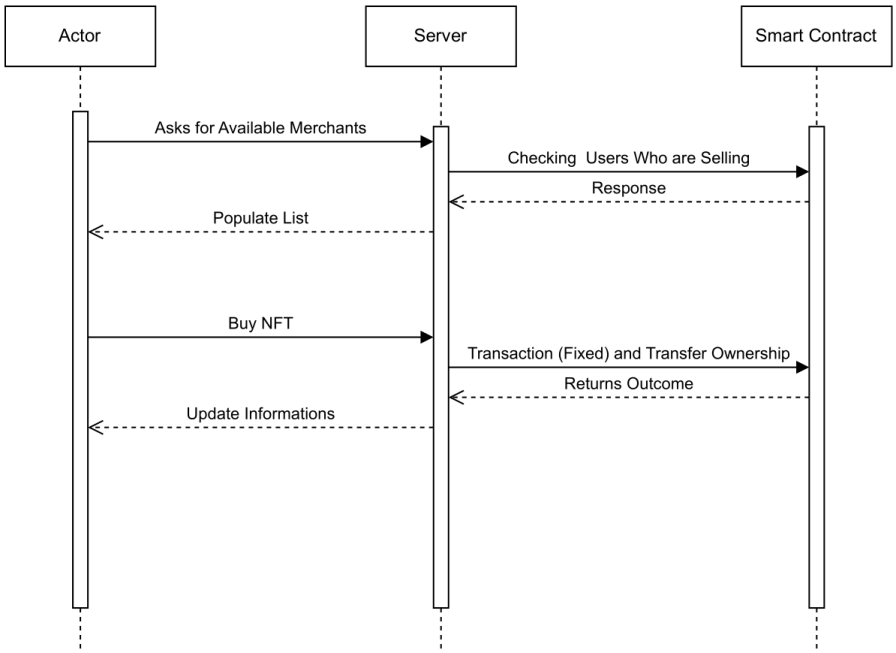

Figura 5.2: Market

6. User Experience and Interface Design

6.1 User Experience Principles

The user is presented with a single page application that is designed to be as simple as possible.

The user can log in with their Metamask wallet and access the platform, initially the user is required to join the environment by paying a one time fee, as it is illustrated in the following sequential diagram.

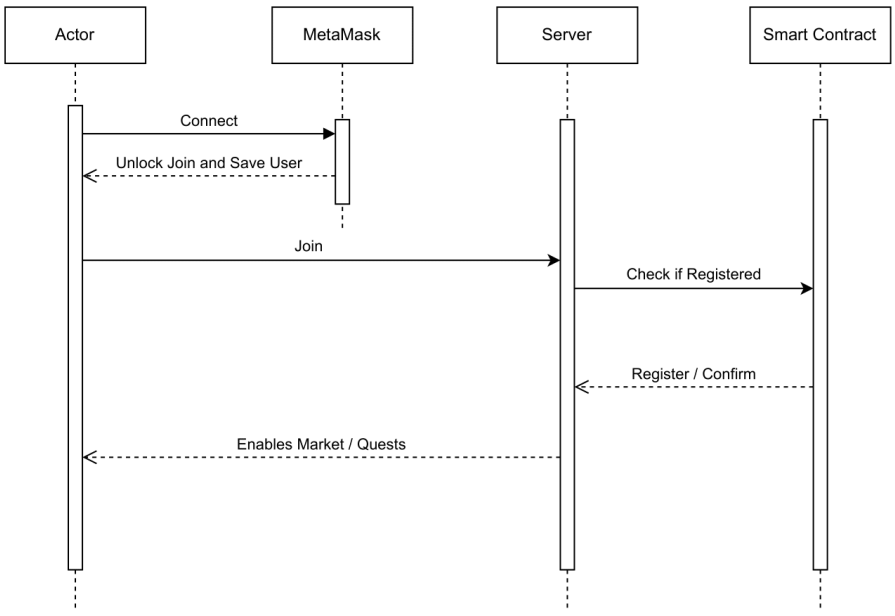

Figura 6.1: User Join

6.2 Interface Design Principles

The interface is designed to be as linear and captivating as possible, the user is presented with a single page that changes dynamically based on the user's actions.

For the styling we used **ChakraUI [10]** to try and keep the design simple but modern, and for the animations we used **Framer Motion** to give the user a more dynamic experience.

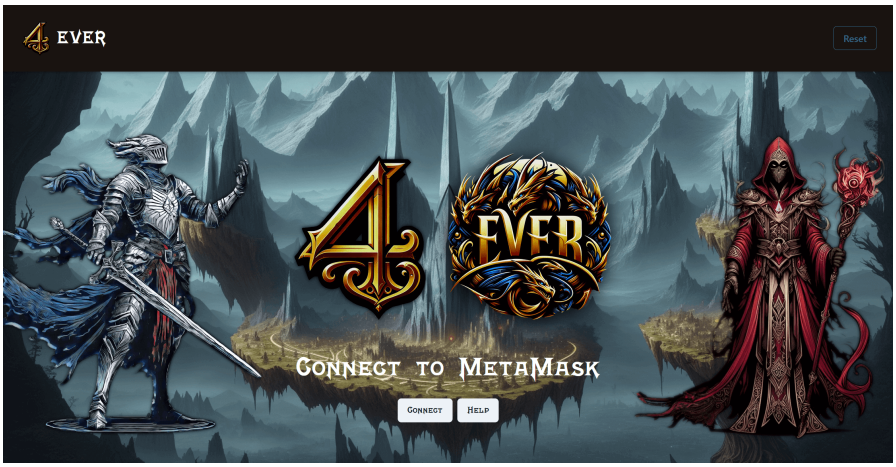

Figura 6.2: User Interface Before

Figura 6.3: User Interface After

7. Future Directions and Current Limitations

The 4Ever DApp has several limitations, including:

- **Actual Companies Integration:** The quest system is currently simulated and not integrated with actual gaming companies. Future enhancements may involve the integration of real companies offering quests and rewards. For example an approach that could be used is to implement the storage of the NFTs with a decentralized database like ***IPFS*** **[11]**.

- **Marketplace Limitations:** The marketplace currently only allows fixed-price transactions. In the future, users should be able to set their price according to the rarity of the NFT they possess, also there could be integrated an auction system for NFT trading.

- **Metamask [12] Requirement:** Users are required to use Metamask for accessing the paltform and reading their wallet. Future enhancements may involve the integration of other wallets.

- **Gameplay Limitations:** The DApp currently does not include an actual simulated gaming environment. This is a potential area for future development.

8. Conclusion

Lastly we wanted to underline how actual integration of blockchain technology in the gaming industry is still in its early stages, and there is a lot of potential for growth.

However due to the complexity of the technology and the lack of knowledge of the general public, it is still a long way before we can see the full potential of the blockchain in the gaming field (Like how **Ubisoft [13]** tried, and is currently trying, for example).

This is a practice that not many companies are willing to take, due to the high risk and the knowledge complexity of the technology, but we believe that in the future it could turn to be interesting.

9. References

[1] Bitcoin Whitepaper: `https://bitcoin.org/bitcoin.pdf`

[2] Ethereum Whitepaper: `https://ethereum.org/en/whitepaper/`

[3] Solidity Documentation: `https://docs.soliditylang.org/en/v0.8.24/`

[4] Vite: `https://vitejs.dev/`

[5] React: `https://reactjs.org/`

[6] NodeJS: `https://nodejs.org/en/`

[7] Express: `https://expressjs.com/`

[8] Ganache: `https://trufflesuite.com/ganache/`

[9] Truffle: `https://trufflesuite.com/truffle/`

[10] ChakraUI: `https://chakra-ui.com/`

[11] IPFS: `https://ipfs.tech/`

[11] Metamask: `https://metamask.io/`

[12] Ubisoft and NFTs: `https://quartz.ubisoft.com`