SAPIENZA
UNIVERSITÀ DI ROMA

eTreeum

eTreeum was created to help raise awareness on the environmental impact of blockchain technologies. Ideally, users of this app will help plant trees in the real world by playing with crypto-trees. Users are able to start with a free seed, take care of it and then sell it for cryptocurrencies.

Facoltà di Scienze Matematiche Fisiche e Naturali

Corso di Laurea Magistrale in Computer Science

Candidate

1. Giuseppe Masi: software architecture, back-end/front-end communications
2. Fabiana Migliorini: Back-end - Solidity development
3. Rocco Pisciuneri: Front-end - HTML, CSS and Javascript development

ID number Masi 1962771, Migliorini 1700205, Pisciuneri 1962772

Academic Year 2021/2022

Contents

Chapter 1

Background

1.1 Introduction

The blockchain is a sequence of linked blocks, each one containing a list of records and connected to the previous one in chronological order, resulting in a chain of blocks. Each block contains the cryptographic hash of the previous block, a timestamp and transaction data. This iterative process confirms the integrity of the previous block, all the way back to the initial block.

Moreover, the blockchain does not allow anyone to modify the stored data because once recorded, the data in any given block cannot be altered retroactively without altering all subsequent blocks. To assure the integrity of a block and the data contained in it, the block is digitally signed.

Typically, it is managed by a peer-to-peer network where the nodes agree about the communication protocol and how to validate new blocks. So, it results in a shared data structure that is "immutable".

1.2 History

The most famous application of such technology is as a publicly distributed ledger, in which the blocks hold batches of valid transactions. In this setting, the blockchain is described as a value-exchange protocol.

The first reliable implementation of such blockchain is by Satoshi Nakamoto in 2008, when he started his public ledger of the crypto-currency Bitcoin. Bitcoin is the first digital currency to solve the double-spending problem without the need for a trusted central authority.

Moreover, the use of a blockchain not only confirms that each unit of value was transferred only once but also removes the characteristics of infinite reproducibility from a digital asset.

1.3 Advantages

The use of a blockchain as a data structure to store information implies some direct advantages over traditional centralized databases.

1. **Full decentralization**: each node can read and write to the blockchain in a secure way. So, there is no single authority that controls the data structure.

2. **Extreme fault tolerance**: every node in a decentralized system has a copy of the whole blockchain, so there is no central point of failure.

3. **Independent verification**: everyone can verify the transactions, without the need of a third party.

1.4 Building blocks

The building blocks of such technology are:

1. the hardware infrastructure

2. the networking support: node discovery, information propagation and verification

3. the consensus mechanism: i.e. Proof-of-Work or Proof-of-Stake

4. the data: transactions grouped in blocks

5. the application: smart contracts (if applicable)

1.4.1 Consensus mechanism

Mining nodes are the ones in charge of validating transactions, adding them to the block they are building, and then broadcasting the complete block to other nodes.

In PoW the mining nodes try to solve a complex mathematical problem in order to get a reward. Once a solution is found, the miner validates a new block which is added to the blockchain. After that, the other nodes check that the solution found is correct. However, the mining operation in a blockchain adopting the proof-of-work consensus mechanism requires a significant amount of energy.

In PoS, the mechanism to decide the new blocks to be validated is based on the fact that the nodes of the network will put their own crypto-currencies into play (stake). In general, the more stakes a participant makes available and the longer these crypto-currencies live, the higher the probability that the latter becomes a validator.

Fork Sometimes separate blocks can be produced concurrently, creating a temporary fork.

Any blockchain has a specified algorithm for scoring different versions of the history so that one with a higher score can be selected over others: they keep only the highest-scoring version of the database known to them. Whenever a peer receives a higher-scoring version they extend or overwrite their own version and retransmit the improvement to their peers.

For example, Bitcoin uses a proof-of-work system, where the chain with the most cumulative proof-of-work (so, the longest one) is considered the valid one by the network.

1.4.2 Programmable application

Blockchain-based smart contracts can be partially or fully executed or enforced without human interaction. One of the main objectives of a smart contract is automated escrow. A key feature of smart contracts is that they do not need a

trusted third party (such as a trustee) to act as an intermediary between contracting entities since the blockchain network executes the contract on its own. This may reduce friction between entities when transferring value and could subsequently open the door to a higher level of transaction automation.

In the case of Ethereum, a smart contract is code that runs on the EVM. It can accept and store both ether and data. Then, using the logic programmed into the contract, it can distribute that ether to other accounts or even other smart contracts.

1.5 Security

The security of the blockchain relies on the use of public-key cryptography. The public key is actually an address on the blockchain. Value tokens sent across the network are recorded as belonging to that address. By using its private key, the owner can access their digital assets or the means to otherwise interact with the various capabilities that blockchains now support.

1.6 Types of blockchain

Permissionless vs. Permissioned In a permissionless blockchain, any user is allowed to join the network becoming a node of it. In this setting, the user joins pseudo-anonymously and there is no restriction on its rights with respect to the blockchain network.

Conversely, a permissioned blockchain restricts access to the network to certain nodes and may also restrict the rights of those nodes on that network. The identity of a user allowed to join the permissioned blockchain is known to the other users of that network.

A permissionless blockchain tends to be more secure than a permissioned one, since there are (usually) more nodes to validate transactions, and it would be more difficult for malicious users to perform an attack on the network. However, a permissionless blockchain also tends to have long transaction processing times due to a large number of nodes and the large size of the transactions.

On the other hand, a permissioned blockchain tends to be more efficient, since there are fewer nodes resulting in less processing time per transaction.

1. **Public**: permissionless blockchain in which anyone is allowed to join, and it is completely decentralized. Each node of the network has equal rights to access the blockchain, create new blocks of data, and validate blocks of data. Popular public blockchains are Bitcoin and Ethereum.

2. **Private**: permissioned blockchain under the control of an organization. There is a central authority able to determine who can be a node and its relative rights to perform actions on the network.

3. **Consortium**: permissioned blockchain controlled by a group of organizations, A consortium blockchain results in a higher level of security with respect to a private blockachain, since it enjoys more decentralization.

4. **Hybrid**: blockchain under the control of a single organization, but when it is required to perform certain transaction validations the supervision is carried out by the public blockchain.

Chapter 2

Presentation of the context

2.1 Aim of the DApp

2.1.1 Motivation

In a Proof-of-Work based blockchain system, the amount of energy consumption needed for every single transaction is significant. In the (not so) long run, this technology will be unsustainable for our planet. In a short time, the effects of pollution on the climate will be irreversible.

For this reason, we created the **eTreeum** game, which aims on the one hand to mitigate the environmental impact of the blockchains and on the other to raise awareness on the matter by being fun and enjoyable.

While the users are using the blockchain technology to play with our DApp, a fee on each exchange-value transaction will be sent to an organization to plant a tree in the real-world.

2.1.2 Gameplay

New users start for free obtaining an initial seed in their crypto-garden. The user must take care of his plants by giving them water and sun periodically and with the right amount.

In the game there are different species of plants, each one with its own extinction risk based on real-world information. Moreover, each species has a different amount of water and sun needed each week in order to grow healthy.

A plant cannot get too much water and sun altogether, but the user should give sun and water a little at a time to reach the needed amount in a week. If the user follows the need of the plant properly, the latter will grow to reach the next stage.

The available stages are:

1. Seed
2. Bush
3. Adult
4. Majestic
5. Secular

The species in the game are distributed according to their extinction risk: the lower it is, the higher the probability that a new plant will be of that species. So, the number of plants with a high extinction risk will be rare in the game, thus making them more valuable.

In addition to the initial free seed, the players can buy how many seeds they want from the game by paying in crypto-currencies: the proceeds of this transaction are sent to the real-world organization to plant trees.

2.1.3 Play-to-earn

The owner can add his plants to the shop when the *Adult* stage is reached, deciding its price and earning crypto-currencies (ETH) when it is sold. The economic principle of the shop is based on the free market, but the DApp imposes a surplus commission to the buyer: this commission will go to the organization which plants trees in the real world.

Of course, the more the plant is valuable (based on the extinction risk and the stage) the more the owner will be able to sell it at a higher price.

In the end, the players are encouraged by the Play-to-earn mechanism to attend their plants or invest in a plant for sale to grow it even more and re-sell it for a higher value.

This results in a direct and indirect contribution of the players to the real-world tree planting.

2.1.4 Ranking

Every plant in the game has a value associated to it, based on its extinction risk and on its current stage.

Each player has a score given by the sum of the values of his owned plants, introducing competition among the players to be the number one player: in the home page the three highest-score players are shown, and the first place is crowned the *King of the forest*.

2.2 Why using a blockchain

As said in the first chapter, the use of a blockchain removes the characteristics of infinite reproducibility from a digital asset. This feature lets us define the so-called *Non-Fungible Tokens*, NFT.

An NFT is a non-interchangeable unit of data stored on a blockchain, uniquely identifiable. The blockchain ledger provides the public certificate of authenticity and proof of ownership. NFTs differ from standard *Tokens* (like crypto-currencies) because they are not reproducible and equivalent to each other, such as a piece of art.

In our application, the crypto-plants are implemented as NFTs and the users are able to trade these digital assets for crypto-currencies. The very nature of NFTs, granting the uniqueness of the asset, should encourage players to buy and trade as many trees as possible, like in a physical collectables game.

2.3 Type to use in production

The adequate type of blockchain for the exposed situation is a Permissionless Blockchain.

In fact, first of all we need to store at least the digital assets each user owns to guarantee the certification of the possession by the blockchain itself.

Furthermore, each user interacting with the application performs actions that need to be saved on the blockchain and generally speaking not all writers (aka players) are known, as every real-world person should be able to register himself on the application and play with the game: that is actually what we want to encourage, as to raise more awareness as possible.

Lastly, an always-online Trusted Third Party would imply all the famous centralization problems that the blockchain overcomes. In particular, in order to develop our application with an always-online TTP we would need a central server (and database) always available to store and react to the user actions, and also the users would need to trust a central authority to ensure their ownerships and to supervision their tradings, which could discourage them to play the game in the first place.

Chapter 3

Software architecture

Figure 3.1 shows the general idea of the software architecture of a DApp.

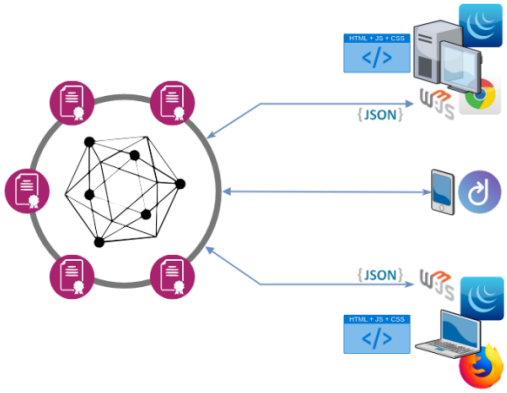

Figure 3.1. Software architecture developed.

3.1 Front-end

The technologies involved in the front-end development are HTML5, CSS3 and Javascript. These three languages have been used to define the structure and the style of the web interface of the application. In particular, Javascript let us build the web-pages dynamically, based on the personal history of each user with the DApp, stored in the back-end.

3.2 Middle-ware

In order to connect the presentation layer of the application with the back-end, we used `web3.js`. It is a collection of libraries that allows you to interact with a local or remote Ethereum node using `HTTP`, `IPC` or `WebSocket`.

In fact, it provides a way to call the smart contract methods asynchronously. If the smart contracts method called is constant (PURE or VIEW) its return value can be obtained from the CALL `web3` method. Otherwise, if the method is non-constant (aka it modifies the state of the smart contract) we can call it with the `web3` method SEND and then exploit EVENTS to get the new state, by subscribing to them.

3.3 Back-end

We use `Ganache` to fire up a personal Ethereum blockchain and test and deploy our smart contract written in the `Solidity` language.

3.4 UML diagrams

In this section UML concept, use cases, component, and collaboration diagrams are shown.

3.4.1 Concept diagram

A conceptual diagram is a visual representation of the relationships between the abstract concepts (i.e. the entities). We use it here to to show the structure and content of the smart contract.

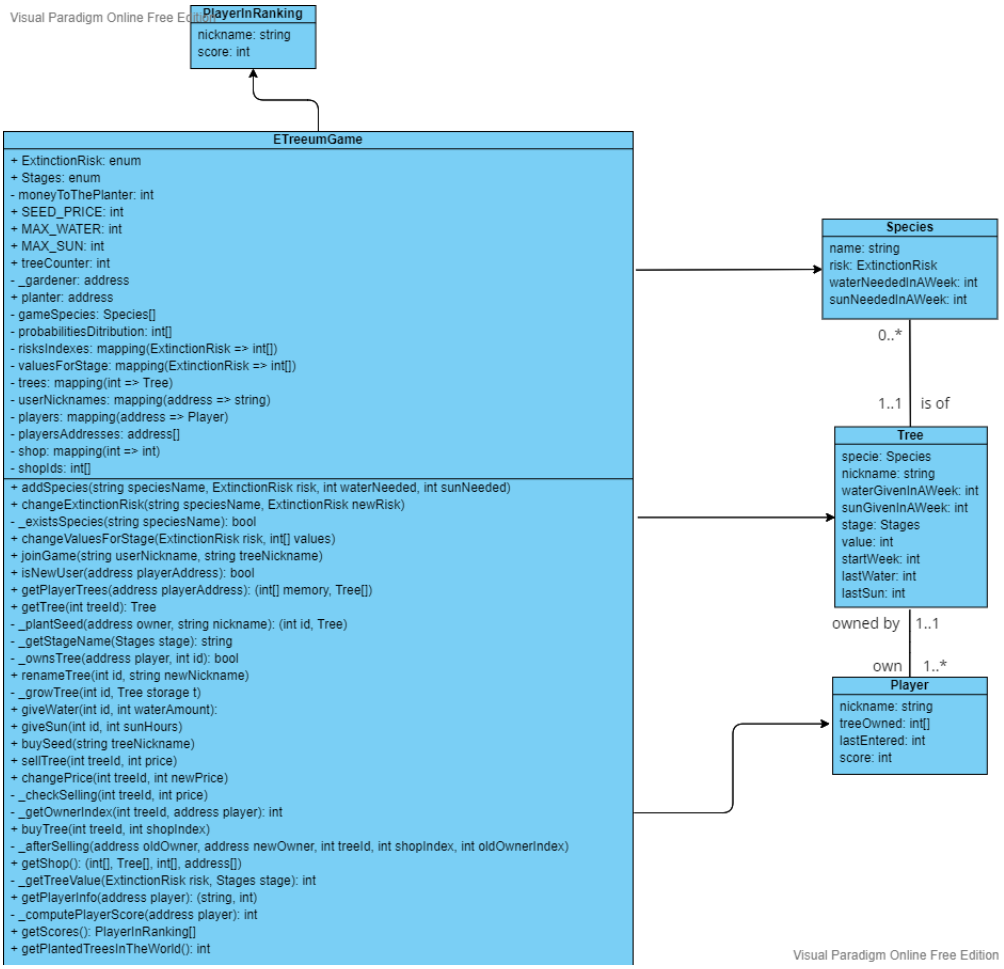

Figure 3.2. Concept diagram of the DApp.

3.4.2 Use cases diagram

A use case diagram is a graphical depiction of a user's possible interactions with a system.

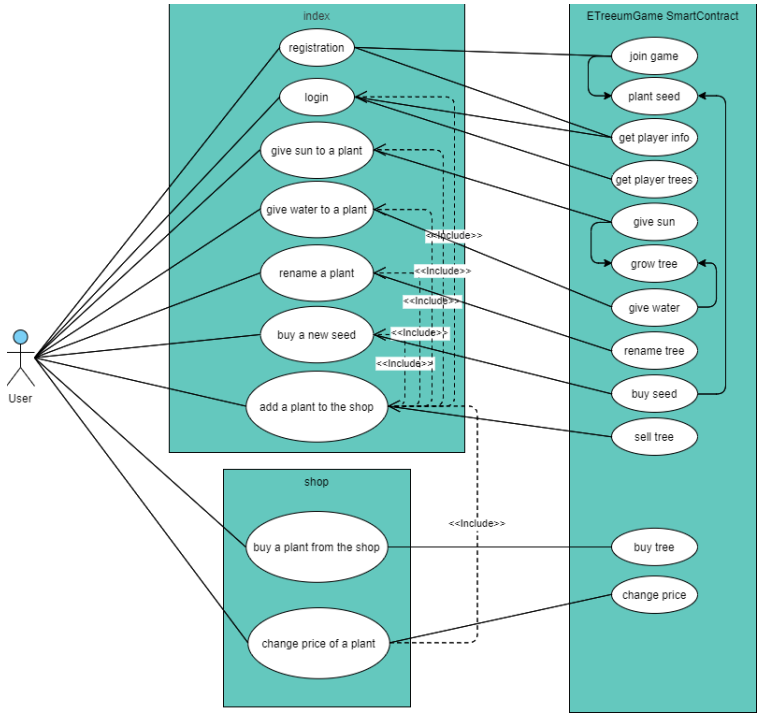

Figure 3.3. Use cases diagram of the DApp.

3.4.3 Component diagram

A component diagram depicts how components are wired together to form larger components or software systems.

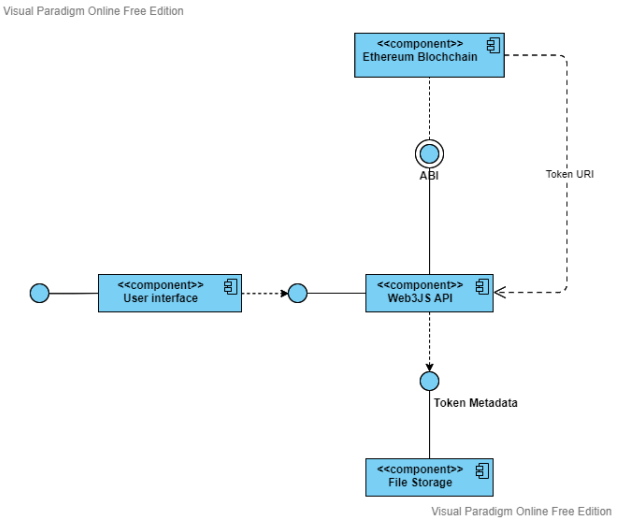

Figure 3.4. Component diagram of the DApp.

3.4.4 Sequence diagrams

A sequence diagram shows the objects arranged in time sequence. It represents the objects involved in the scenario and the sequence of messages exchanged between them in order to perform the functionality of the scenario. We show here the most significant ones for our DApp.

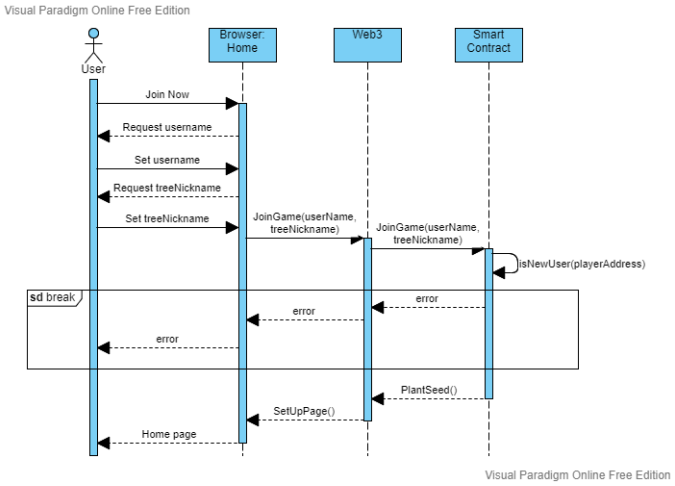

Figure 3.5. Sequence diagram showing the steps for the registration of a new user.

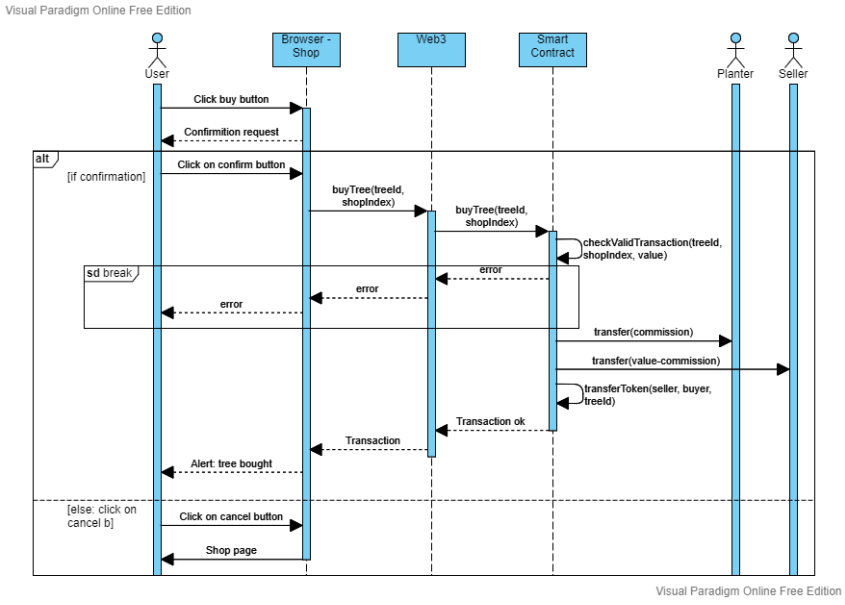

Figure 3.6. Sequence diagram showing the steps for the user buying a tree from the shop.

Chapter 4

Implementation

4.1 Front-end

The front-end of the DApp was implemented as a web interface divided into two pages: the Home and the Shop.

4.1.1 Home-page

The home page renders accordingly to the fact that the connected user is already registered in the game or not, by allowing him to register in the latter case or by showing him his dashboard.

4.1.2 Registration

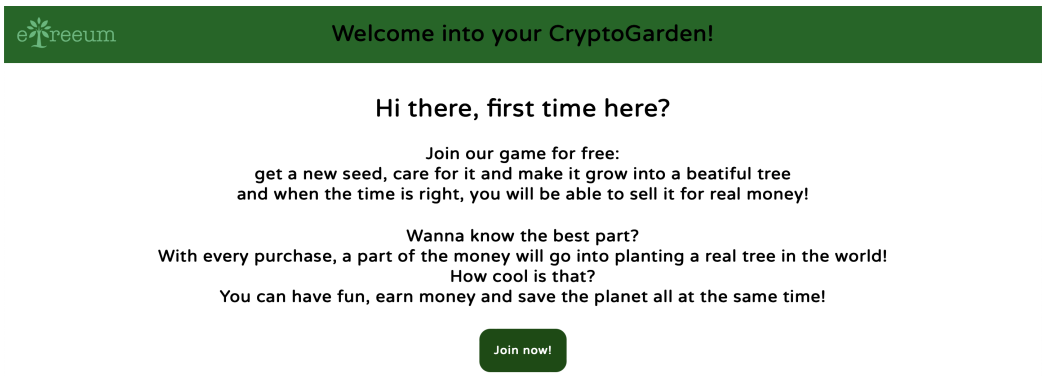

Figure 4.1. First registration step for a user in the DApp.

To register in the game, the user:

1. Clicks on the button *Join now!*
2. Chooses his userName in the game;
3. Chooses the name of the free tree received by the game and clicks on the confirmation button

Login

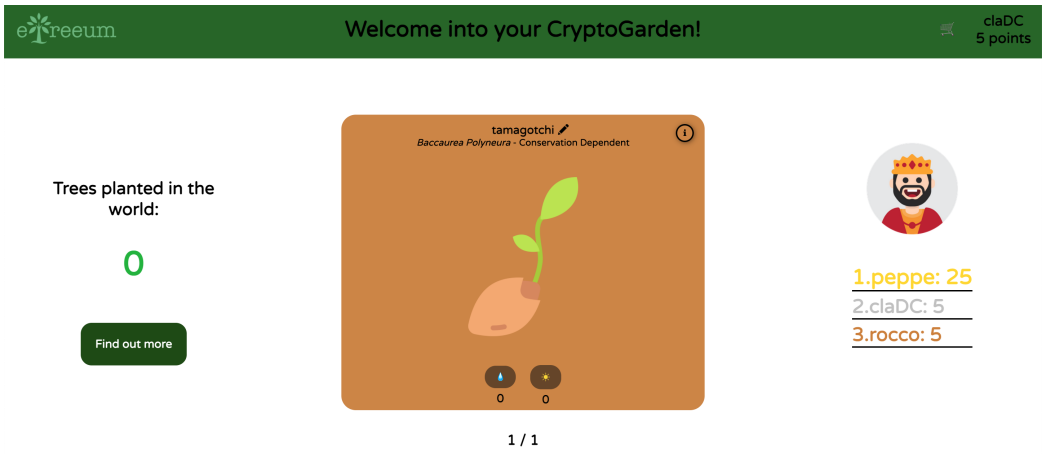

Figure 4.2. Dashboard of a registered player.

From the dashboard, the user is able to:

- Check on his trees and care for them
- Rename one of his tree
- Gets real world facts about the species he owns
- Sell a tree he owns
- Go to the shop or buy a new seed from the game
- See the current ranking of the game
- See how many trees were planted thanks to the game

4.1.3 Shop-page

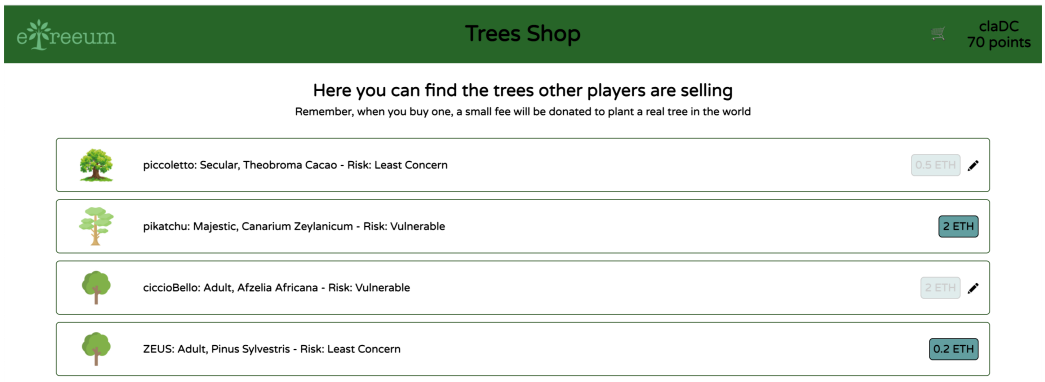

Figure 4.3. Shop page

In the shop, the user can:

- Look at all the trees in the shop
- Buy a tree from another user
- Change the price value of an owned tree in the shop
- Go back to the home-page or buy a new seed from the game

NOTE: Users can access the Shop-page only if they are registered in the game.

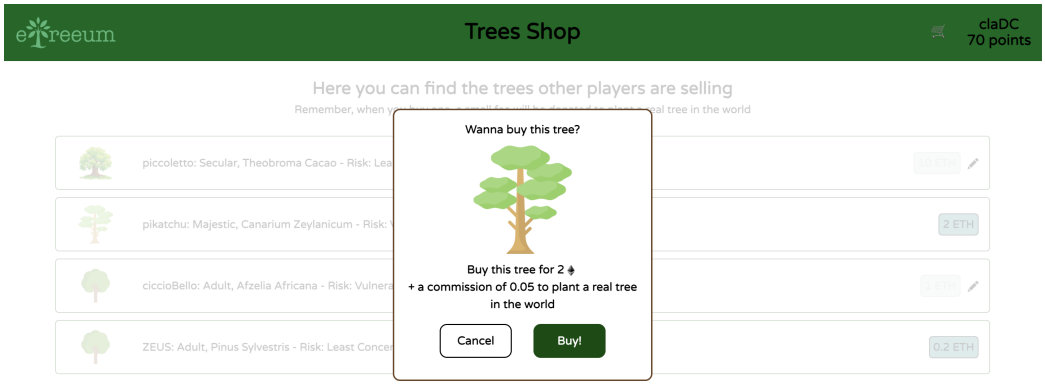

Figure 4.4. Confirmation request before buying a tree in the shop.

4.2 Back-end

The back-end is written in Solidity, *pragma version* >=0.8.0 <0.9.0.

It implements the NFT pattern, ERC-271[2], by extending the ERC721URIStorage[4] class from *OpenZeppelin*[5]: in this way, a tree token is also associated with some off-chain metadata (such as an image and some facts on the real-world species) referenced but a tokenUri stored in the blockchain.

As we needed a token able to change and evolve in time, when the user cares for the trees and it reaches a new stage, every time a tree grows its tokenUri is changed accordingly as to reference the metadata associated with the new stage.

```
/**
    @notice The object representing a species in the game
*/
struct Species {
    string name;
    ExtinctionRisk risk;
    uint16 waterNeededInAWeek;
    uint8 sunNeededInAWeek;
}

/**
    @notice The object representing a tree in the game
*/
struct Tree {
    Species specie;
    string nickname;
    uint16 waterGivenInAWeek;
    uint8 sunGivenInAWeek;
    Stages stage;
    uint8 value;
    uint256 startWeek;
    uint256 lastWater;
    uint256 lastSun;
}
```

Figure 4.5. Structs in the contract which represent a species and a tree respectively

Chapter 5

Conclusions

In conclusion, we presented a game based on the blockchain whose hope is to help raise awareness on the world environmental crisis and actually do something to mitigate it, by engaging its user with some gamification mechanism (i.e. encouraging the player to be the first in the ranking).

5.1 Known issues and limitations

Storing data on-chain can get really expensive, and ideally we need to associate an image to each stage of each species of tree. As even uploading a 1MB image could break one's bank account, this is unfortunately not an option. This means that the only viable way is to store metadata off-chain and then reference it on-chain, unfortunately sacrificing part of the security and transparency offered by the blockchain itself.

5.2 Future works

The game could be further extended by adding new game-play mechanics, such as:

1. Possible auctions when selling a tree;
2. A user could personalize his plants with some accessories;
3. A plant may be able to get sick and downgrade if not attended properly, to encourage more interactions and real learning from the players

5.3 References

1. Blockchain https://en.wikipedia.org/wiki/Blockchain
2. Standard NFT ERC-721, https://ethereum.org/it/developers/docs/standards/tokens/erc-721/
3. Figure 3.1 - Slide 09, page 5 of the course.
4. OpenZeppelin ERC-721, https://docs.openzeppelin.com/contracts/4.x/api/token/erc721
5. OpenZeppelin, https://docs.openzeppelin.com/