# P2P Finalproject - TRY

Geremia Pompei (MAT. 638432)

June 1, 2022

## 1 Introduction

For the final project I implemented a DAPP using smart contracts of final term.

### 1.1 Smart contracts

In addition to the old `ERC721.sol` and `Lottery.sol` I add a new contract called `TRY.sol` that is able to create lotteries and emit events. In this way when a manager wants to open a lottery other users can be notified waiting for the event `LotteryCreated`. Smart contracts are deployed with *Truffle* on a local blockchan network that is simulated with *Ganache*.

### 1.2 DAPP

To implement the DAPP I used the library *express.js* to create a simple server in *node.js*. The server is able to provide to the browser only static files that are related to the client side of the app. The web app that I created is written using *Vue.js* framework for manipulate in a better way the js code and *Bootstrap* about css part. Another library that I imported to interact with smart contracts is *Web3* while to put and retrieve NFT images from IPFS I used *Web3.storage* service in REST API version.

## 2 Running DAPP

### 2.1 Prerequisites

- node.js and npm
- Ganache
- truffle (installable with the command `npm install -g truffle`)
- Metamask wallet (you should connect wallet to Ganache network and import an ethereum account of Ganache)


Figure 1: Main menu

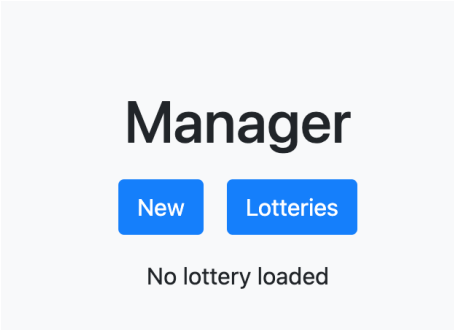


Figure 2: Manager menu without loaded lottery

## 2.2 Execution

- open and run *Ganache*
- go to `smart_contracts` directory and run `truffle migrate --reset`
- go to `server` directory and run `npm install` and `npm start`
- open browser and go on `http://localhost:3000`

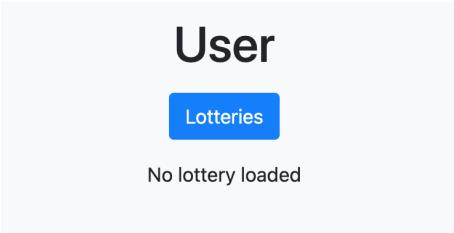


Figure 3: User menu without loaded lottery

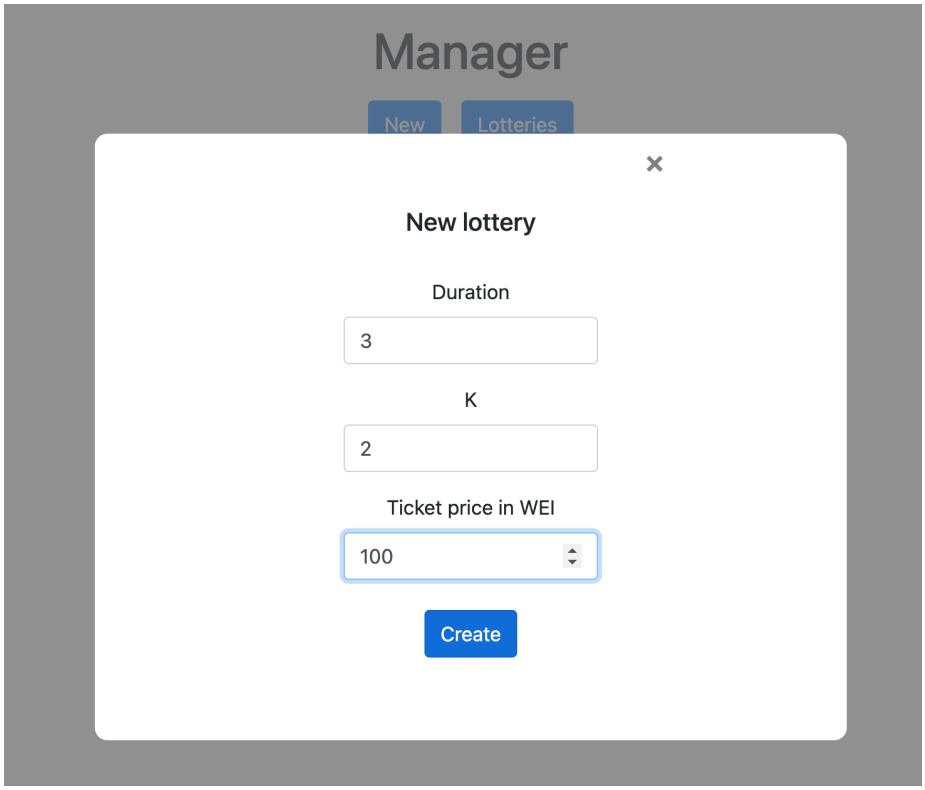


Figure 4: Form to create new lottery

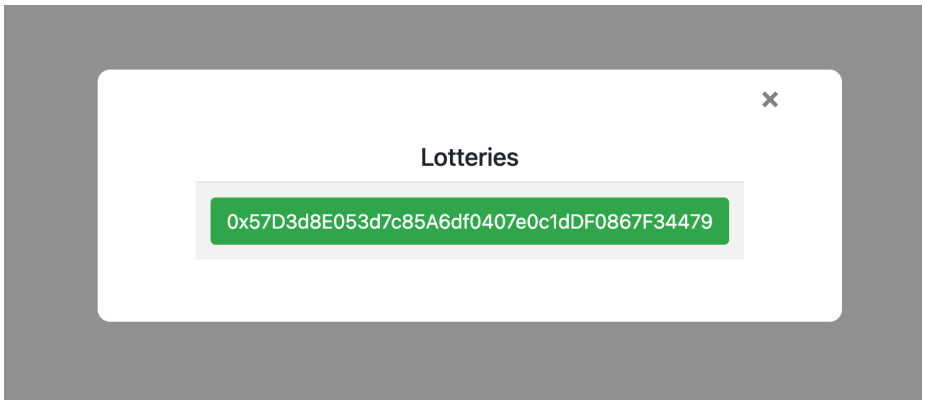


Figure 5: List of lotteries to load

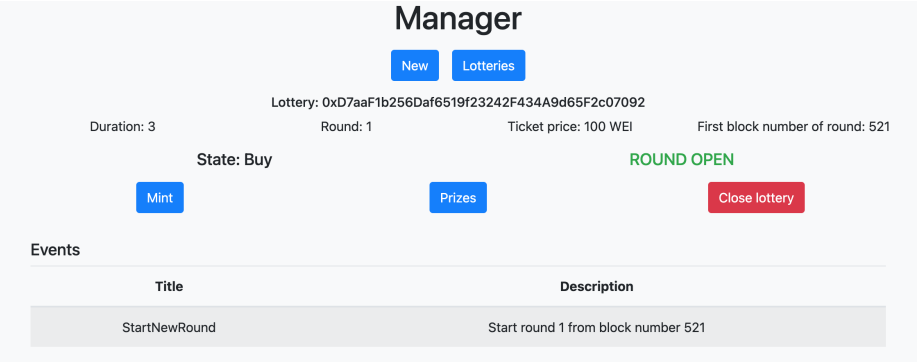


Figure 6: Manager menu in the initial state with a loaded lottery

## 2.3 Interfaces and use

### 2.3.1 Menu and roles

The Figure 1 represents the first page of web app where can be choose the role to play lottery. If I am a manager I press the relative button and go on manager menu represented on Figure 2 while I am user I press relative button and go on page represented to Figure 3. Manager can create a new lottery pressing the New button and compiling the form of Figure 4 while both roles can load an existing lottery to play with this. The list of lotteries it's showed to Figure 5.

### 2.3.2 Manager operations on loaded lottery

Loaded a lottery a manager can manage it with the interface visible to Figure 6. Throw it he can mine a new NFT and associate it to a class of prizes (the form to do this is visible to Figure 7), view prizes and close lottery. In the Figure are shown options available on the initial state that is Buy state when users can buy tickets. When this state finish will appear a new button to draw numbers and pressed this will appear another button to give prizes to players if they win the lottery. In the end after given prizes will appear a new button to start a new round of the lottery.

### 2.3.3 User operations on loaded lottery

A user can interact with a lottery using the interface visible in Figure 8. Throw this interface users can buy tickets and play numbers (on Figure 9 there is the form used to play numbers). They can also visualize lottery prizes and won prizes related to the loaded lottery. Both managers and users can visualize to the bottom part of their interface the events related to the loaded lottery.

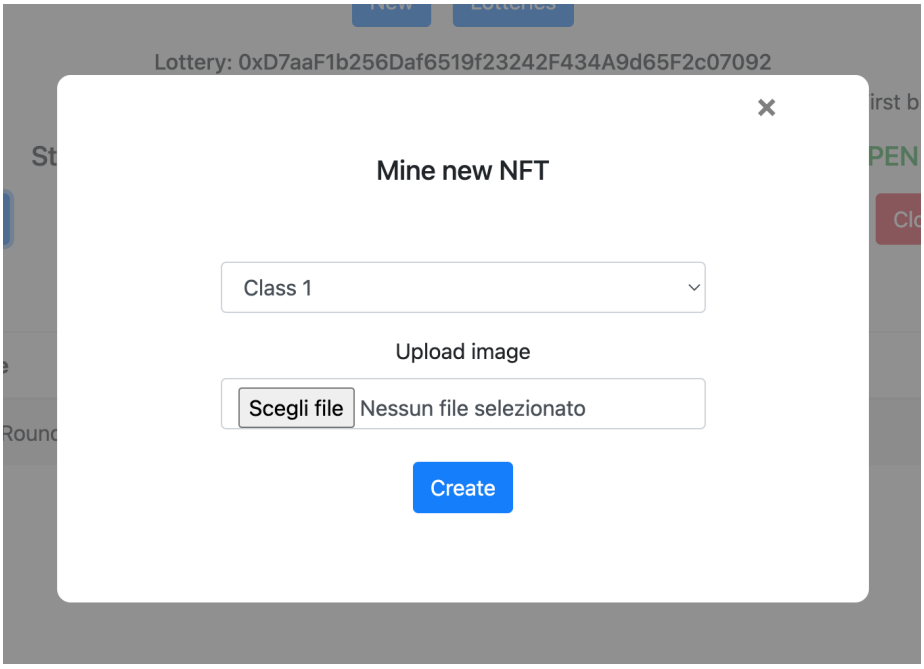


Figure 7: Form to mine a new NFT and associate it to a class of prizes

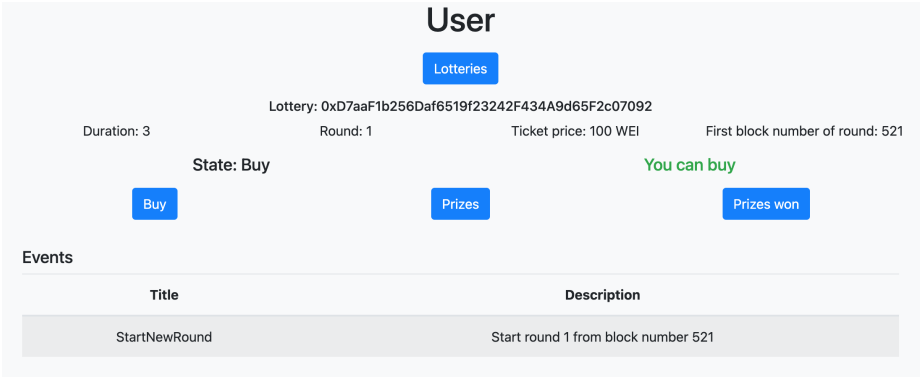


Figure 8: User menu in the initial state with a loaded lottery

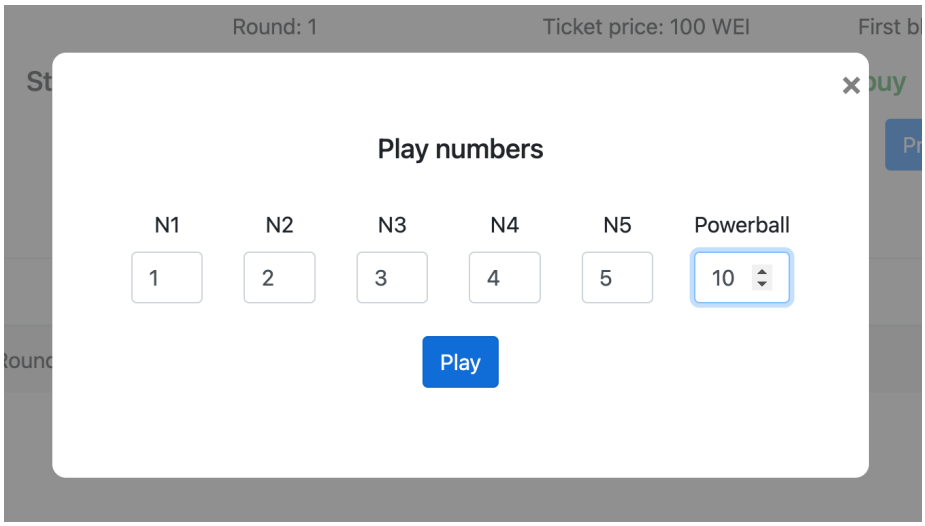


Figure 9: Form buy ticket and to play numbers

# 3 Particular features

**Events** In this project I exploit the websocket protocol to interact with the blockchain throw web3 because in this way each time that an event is emitted the interface can catch it soon to modify his state.

**IPFS** I used the `Web3.storage` API service to upload images related to an NFT on IPFS. In particular when a manager mint a new NFT he choose the class to connect to him and upload an image. When the operation is confirmed first the image is uploaded on IPFS and then it's retrieved the CID and used to call the method `mint` on lottery contract. In this way the created NFT is associated to the unique CID of the image inside IPFS.