TRY dApp: a dApp for the NFT Lottery

Federico Bernacca

1 Introduction

The following implementation choices were made for the development of the project.

The application was implemented using the language **Python** with the library Web3.py and the microframework Flask.
Smart contracts were compiled using the framework Truffle, and the blockchain created via Ganache.

2 Design choices

Once the Web application is executed, the contracts **NFT.sol** and **Lottery.sol** are deployed, at which point filters are created for contract events, so that there is the ability to receive notifications when certain events occur on the blockchain.

2.1 Users

The concept of a user in session has been implemented, i.e., a user, identified by his address on the blockchain, must log in to the service to perform certain operations.

By default, the first address identifies the lottery manager, who will therefore be able to perform different operations than the base user (to mint tokens, to open lottery, round, ...).
The basic user, on the other hand, will be able to register for the lottery, to buy a ticket, and he will receive a notification when the following events occur:

- Lottery opening.
- Opening of the round.
- Drawing of the winning ticket.
- Awarding of prizes.
- Closing of the lottery.

Each logged-in user, has his own personal page where his or information is located, as shown below

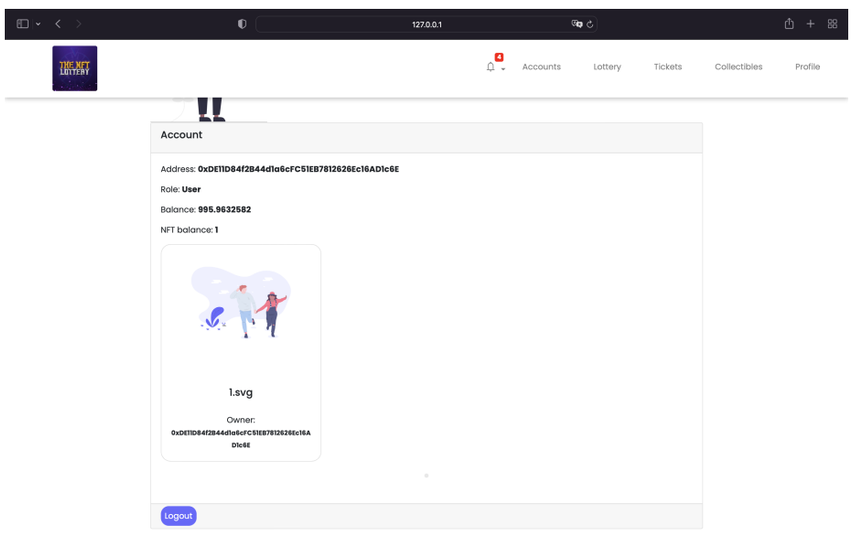

Figure 1: user profile

2.2 Notifications

Notifications are related to blockchain events and are shown on the screen in realtime, without the need to refresh the web page.

The implementation involves defining a route that when called queries the blockhain to catch any events, and returns them in a json object.
This route is called by a script embedded in the web pages every N seconds and eventually updates the GUI.
From the documentation asynchronous-filter-polling it seems that there used to be a watch method, but they removed it, and the two solutions it shows are just a while in a separate thread or asynchronous calls.
For this reason I did not have the possibility to register a callback.

Below it is possible to see the notfications related to the opening of the lottery and the round, for a user interested in the lottery.

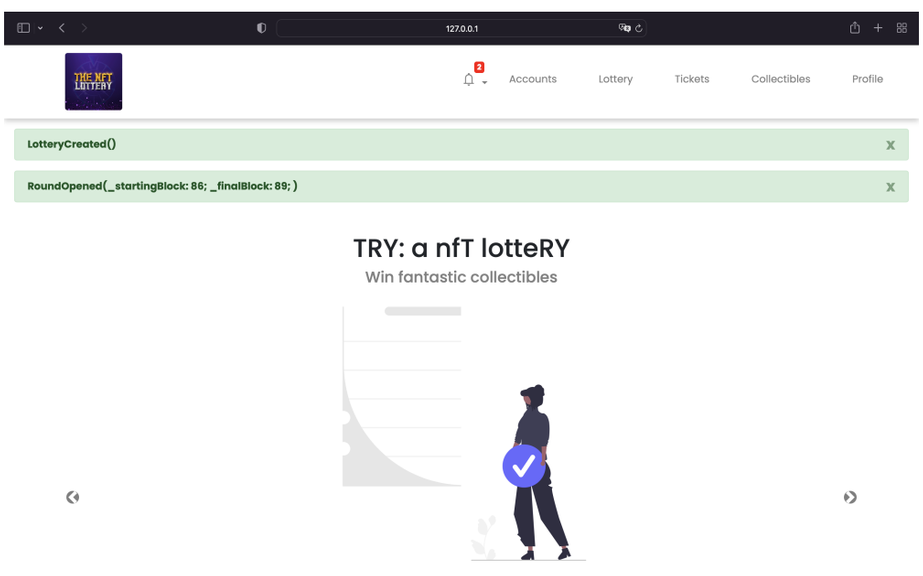

Figure 2: User notifications

2.3 Lottery functions

The class **LotteryProcessor** has been implemented, which contains all the methods for registering to events, calling functions, and executing transactions related to the smart contract **Lottery.sol**.
In particular, the return from a transaction, in case of a negative outcome (contract requirement not met) does not contain the error message, but only the failure status.
To provide a better user experience, auxiliary functions have been defined within the smart contract that return the state of the system, these functions are then called by the code to provide a more informative error message to the user.

2.4 Other

Additional explorable pages are present:

- **Accounts**: where all the addresses of the accounts on the blockchain are present.
- **Collectibles**: page showing all collectibles present, 100 per page, searchable by name or address owner.
- **Tickets**: page showing all tickets present to the lottery operator, instead to a normal user showing only his tickets, in a round.