URBE

This project is a web application for a charity auction organized by UrbE, a micromobility company

Requirements

- Develop the auction platform with Django.
- Set up the platform to use the default sqlite database for user data that registers and other data needed for development.
- Use the Redis database for everything related to auction bids.
- At the end of each auction, in addition to storing the information in the relational database, a JSON file must be generated containing all the details of the auction and references to the winner. Then hash this JSON and write it in a transaction on the Ethereum (Goerli) blockchain.

Useful links

Here are the various project links:

- GitHub Backend/Frontend repo
- GitHub Hardhat repo
- GitHub TheGraph repo

Code

The web app was written using Solidity/Hardhat (framework based on JavaScript) for the NFTs smart contract and the smart contract that manages the auctions, TheGraph was used as an event indexer, Django as a backend that communicates through REST API with the frontend which was written in NextJS/TailwindCSS.

SQLite was used as a database that manages user data and Redis as a database to manage the data of the various auctions.

Solidity / Hardhat project structure

The Hardhat part of the project consists of the following elements:

- the *contracts* folder contains the smart contract that manages the auctions and the smart contract that represent the NFTs.
- the *deploy* folder contains the scripts to deploy each smart contract and a script to update the files on the frontend that take the ABI and address of the deployed smart contracts.
- the *scripts* folder contains a script to mint and list NFTs.
- the *test* folder contains the unit tests.
- the *utils* folder contains the scripts to verify contracts and to move blocks.
- *config* files.

TheGraph project structure

The TheGraph part of the project consists of the following elements:

- the *abis* folder contains the ABI file of the UrbE Auction’s smart contract.
- the *generated* folder.
- the *src* folder contains the file where we tell TheGraph how to map and work with smart contracts.
- the *tests* folder.
- *schema.graphql* where we tell TheGraph how to work with events. Uses GraphQL.
- *subgraph.yaml* where we tell subgraph how to combine files together.

NextJS / TailwindCSS project structure

The front-end part of the project consists of the following elements:

- the *components* folder contains some components to add to the pages.
- the *constants* folder contains files that are written when a smart contract is deployed to a new address.
- the *pages* folder contains the pages that we can see on the website.
- the *public* folder contains some static files.
- the *styles* folder contains CSS files.
- the utils folder contains a file that via REST API asks the backend for some user data and another file that sets some global variables for the app.
- *config* files.

Django project structure

The back-end part of the project consists of the following elements:

- the *UrbE* folder contains some config files.
- the *accounts* folder contains all the files to manage signin, signup and all user data.
- the *auction* folder contains all the files to manage the various auctions.
- the *static* folder contains some static files.
- *config* files.