Josip Di Benedetto changed readme yesterday

This branch is up to date with `main` .

Contribute Sync fork

BlCh_2023 / **frontend** / Top

Name	Name	Last commit da...
..		
.idea	experimental integration ...	2 months ago
cache	added navbar and functio...	yesterday
public	added navbar and functio...	yesterday
readme_screenshots	modified readme	yesterday
src	modified readme	yesterday
.env	added function to mint nf...	2 weeks ago
.gitignore	configured hardhat	3 weeks ago
README.md	changed readme	yesterday
config-overrides.js	added navbar and functio...	yesterday
hardhat.config.js	modified eventdetails	2 weeks ago
package-lock.json	modified readme	yesterday

Name	Name	Last commit da...
package.json	added navbar and functio...	yesterday

README.md

Deprecated packages

Used older version of etherjs ("ethers": "^5.7.2") because new version is unstable. Specifically, the defaultAbiCoder of the ether utils does not work in new versions. Source Stackoverflow. The defaultAbiCoder is needed to retrieve the return value of the non view function in solidity. Other packages are up to date.

How to use

In the project directory, you can run `npm install` and start the app:

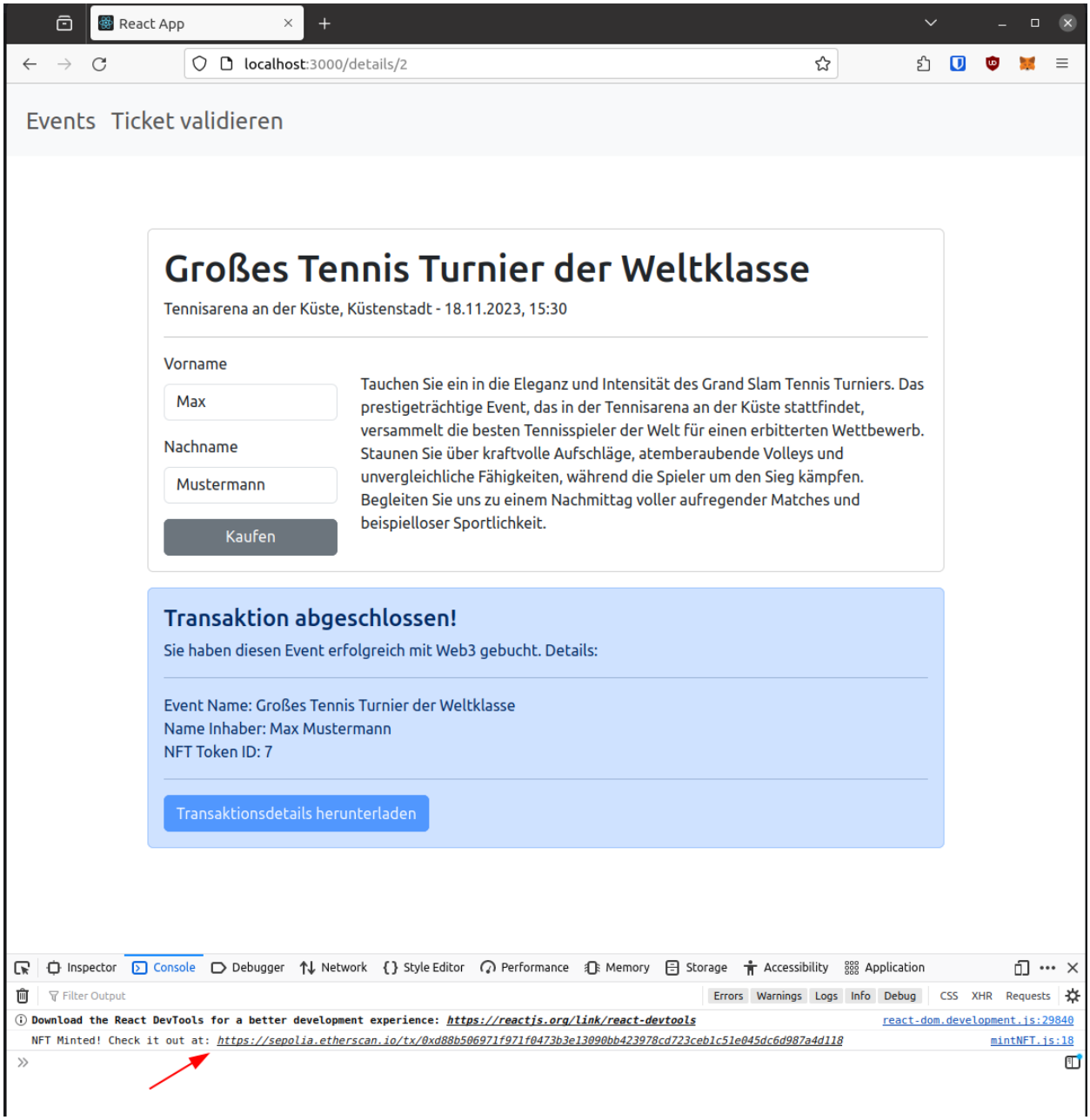
```
npm run start
```

Runs the app in the development mode.
Open http://localhost:3000 to view it in your browser.

Process of buying and validating a ticket

Buying a ticket

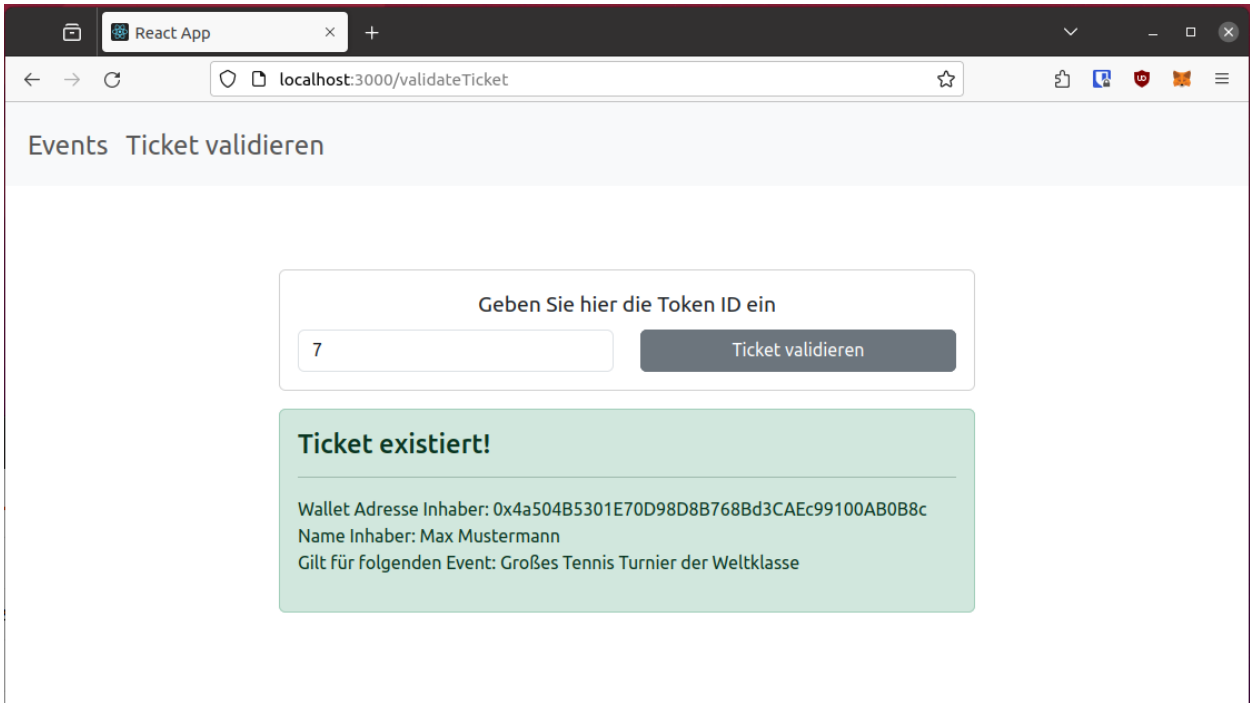
This app requires the Metamask extension to be installed in your browser. You need a wallet account on the Sepolia Testnet, and some SepoliaETH. When you run the app, you can go to `http://localhost:3000/` to see all events. Click on the "details" button to get to the events page. When buying a ticket, you will be notified in the frontend when the minting process is done. However, you won't be notified when the minted data has been written to the blockchain. After pressing the "buy" button in the event details page, open the browser console. When the minting process is done, the link for the etherscan page will be displayed.



Currently, you only pay the gas fees, since there is no implementation on setting prices for an event. The buyer needs to save the Etherscan link or note the TokenID

Validating a ticket

When you see in etherscan that your data has been written to the blockchain, the ticket can be validated. For that, go to the `http://localhost:3000/validateTicket` and enter the TokenID. Click on "Ticket validieren" button.



When the user comes to the event, he has to present his wallet address and token ID. If the ticket exists, and the displayed wallet address matches the users wallet address, then he has a valid ticket for the event. This is the basic idea behind the system.

Smart Contract

The smart contract has four functions:

> function mintNFT(address recipient, uint256 eventId, string memory ownerName) public returns (uint256)

This function is responsible to create the NFT when the user buys an event. The recipient address is retrieved directly from metamask and passed to the mint function. The eventId and ownerName are also passed to this function, so that this data can be mapped to the token ID.

Based on this mapping, we can validate the ticket. Following functions are defined to retrieve the ownername, eventId and tokenId:

> function getOwnerByTokenId(uint256 tokenId) public view returns (address)

> function getEventIdByTokenId(uint256 tokenId) public view returns (uint256)

> function getOwnerNameByTokenId(uint256 tokenId) public view returns (string memory)

Compile and deploy smart contract

For the deployment of the smart contract on the test chain, hardhat is used. If you change the smart contract, run following command in the root directory of this project.

```
npx hardhat compile
```

To deploy the contract, run this:

```
npx hardhat run src/scripts/deploy
```

When deploying, the contract address will be displayed. Copy and paste that contract address in the src/scripts/mintNFT.js, by replacing the value of the CONTRACT_ADDRESS constant.

Then start the application.

Do not change other parameters like Alchemy API-URL or private key. Alchemy is the Provider for the ETH Blockchain used in this project.

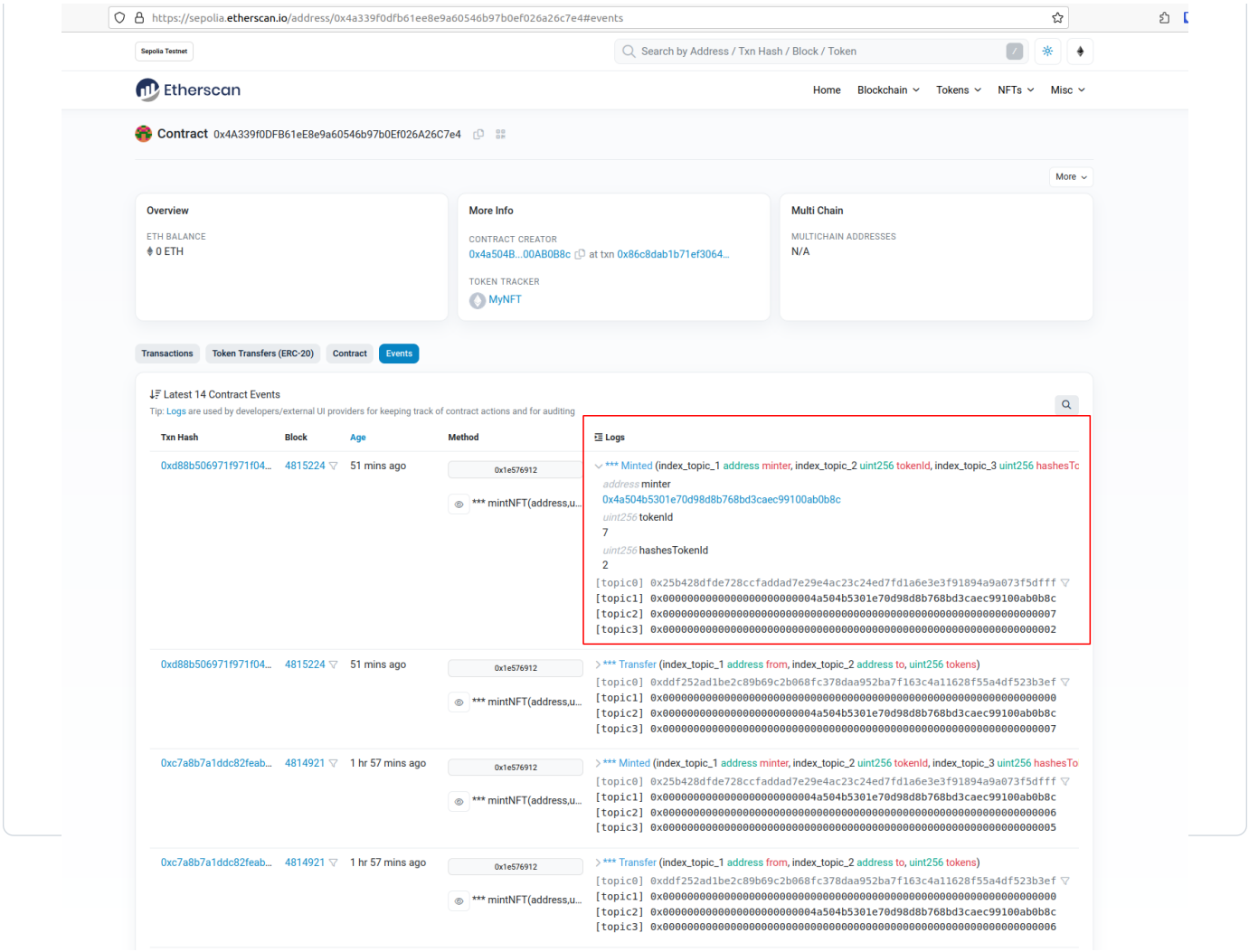
Audit trail

The smart contract emits events:

> event Minted(address indexed recipient, uint256 indexed tokenId, uint256 indexed eventId);

Every time an NFT is minted, the event is being emitted with the recipient address, tokenId and eventId. This information can be viewed by anyone on Etherscan.

Simply paste the contract address in Etherscan and click on the "Events" Button:



Problems

In the latest version of React, Webpack 5 is used. The problem is that Webpack 5 does not include Node code modules anymore, so they cannot be used in the frontend. This prevents me from using modules like dotenv to store data like the public key in an .env file, because the dotenv package relies on the path core module.