Exercises 3 - Volcano Token and Truffle

Make an ERC721 contract with Open Zeppelin

ERC721 Recap:

The ERC721 contract is a standard for non-fungible tokens. Non-fungible tokens allow us to tokenize unique assets - digital and physical.

In ERC721, tokens can be minted and burned, and each token should have a unique ID.

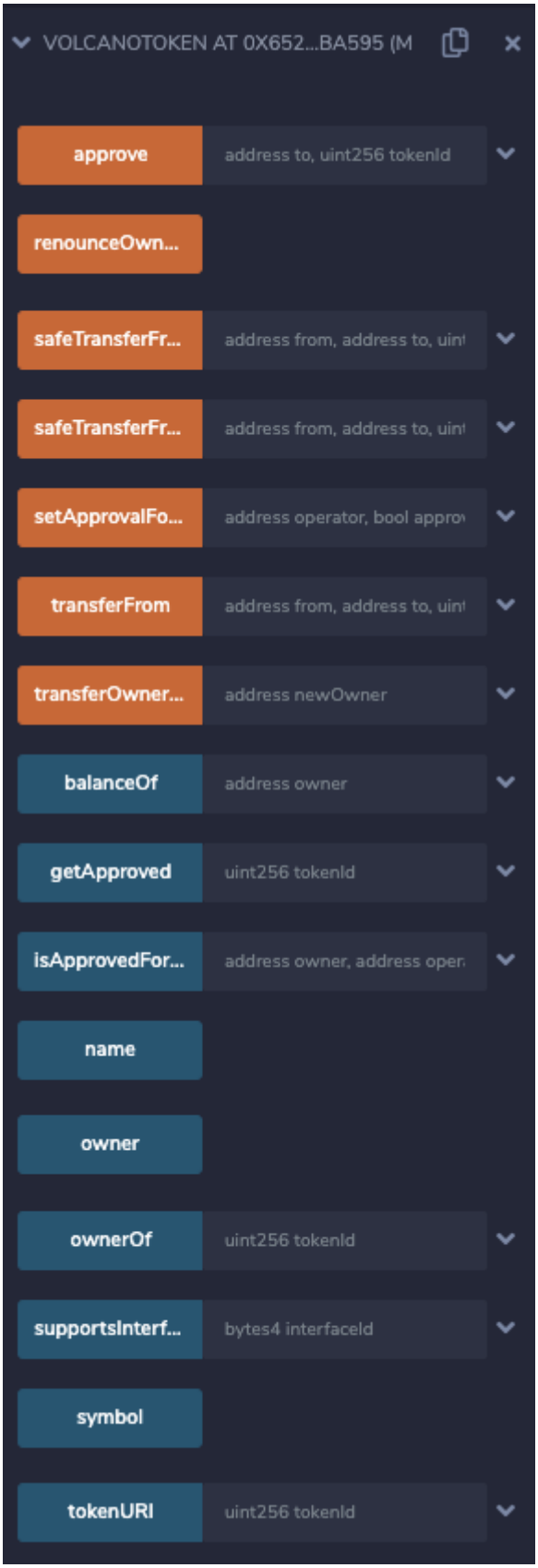
1. In Remix or Truffle, create a new file called `VolcanoToken.sol`.
2. Import Open Zeppelin's ERC721 and Ownable contract.
   Have a look at the ERC721 standard and Open Zeppelin's implementation of this.
   https://eips.ethereum.org/EIPS/eip-721 (https://eips.ethereum.org/EIPS/eip-721)
   https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/token/ERC721/ERC721.sol
   (https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/token/ERC721/ERC721.sol)
3. Create a `contract` called `VolcanoToken` that inherits `ERC721` and `Ownable`.
4. After the contract is deployed, the tokens can be minted, burned and transferred to users. To identify the token we need a token ID. Declare a `uint256` for the token ID.
5. Make a struct for each token's metadata. The struct should have a timestamp, the token ID and a `string` for the tokenURI.

Token URI:

A token URI provides additional metadata about a token. Each URI should be unique to the token.
In many cases, IPFS is used to store this file.

6. Create a public record for token ownership with a `mapping` of the users `address` to an array of structs.

7. Before we start writing our functions, compile and deploy the contract. Don’t forget to select the correct contract from the drop down before deploying. Have a look at the inherited ERC721 public functions available.



Internal ERC721 functions:

Similar to ERC20, there are internal ERC721 functions available to the contract it's defined in.

Useful methods are `_safeMint` and `_burn` ,

8. Make a function that mints a token to a user with the token ID. You can use the struct from part 5 to store the metadata for the token, for example
   `StructName memory newTokenData = StructName(...)`

   Use `block.timestamp` for timestamp and any `string` for the tokenURI.

   Store the struct to the user's record.

   Increment the token ID.
9. Make a function that burns tokens, taking the token ID.
10. Make an internal function that loops over the array of structs and removes the burned tokenID. Call the function inside your burn function.
11. Add a require statement to ensure that only the owner of the token can burn the token.

    Hint: Have a look at the public ERC721 functions, there is one that returns the owner when given a token ID input.
12. We need to remove the token from the mapping. Make an function that deletes the token from the mapping. You can make this an internal function, which can then be called within the burn function.

Truffle Exercises

13. Replace the provided VolcanoCoin contract with your own, compile and deploy it to ganache.
14. Fix the unit test in TruffleIntroduction so that it passes
15. Add 2 more unit tests to show
    - that account 0xcB7C09fEF1a308143D9bf328F2C33f33FaA46bC2 has a zero balance
    - that the owner has a balance of 10000 tokens.
16. Add your VolcanoToken contract source file to the contracts directory and include it in the migration
17. Compile and deploy your VolcanoToken contract to ganache.

18. Add further tests for the functions in our contract
19. Show that a token that has been burnt cannot be spent by its former owner.
20. How can you test an internal function in your contract with Truffle ?

Resources

https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/token/ERC721/ERC721.sol
(https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/token/ERC721/ERC721.sol)