WEEKEND PROJECT 2 REPORT – GROUP 6

ENCODE SOLIDITY BOOTCAMP Q3 2023

This week's project goal was to write scripts (*typescript*) that would allow the group to interact with a deployed Ballot contract (*Ballot.sol*), giving voting rights, casting votes, delegating votes, and querying results.
Instead of breaking each interaction (function call) into a separate script file we opted to only have one file with all the functions and have a conditional statement to ensure that each function (including the deploy function) could be called from the terminal with arguments if needed.
As an example, the following shows how to deploy the contract and how to vote on the second proposal (Batman):

yarn run ts-node --files ./scripts/interactBallot.ts deploy Spiderman Batman Superman

yarn run ts-node --files ./scripts/interactBallot.ts vote 2

–> Contract deployment

The contract was deployed from the script file, passing in 4 arguments, the *deploy* function and the names for 3 *proposals* (Spiderman, Batman and Superman).

Contract address: 0x6E08F69f938c9478eD6701A55F95959421519527
Transaction hash: 0xcc4f80a530c79f2a42414b4a00e93deb9992ed134745f98b32b71430264b226e

To start the interaction with the Ballot contract using the script file we had to update the variable *contractAddress* with the newly created contract address. From here the first interactions come from the contract creator's account (who becomes *chairperson*) since he is the only one who can give voting rights to other accounts.

–> giveRightToVote(address to)

This function gives the right to vote to another account by receiving its address.

Transaction hash: 0xe07d0f2318b4ac5e4ce887678e111ded8318b9c5bb141c98193bcdf0a926f45d

When another account that is not the *chairman* tries to give voting rights, the transaction is reverted with the error: *"Only chairperson can give right to vote"*.

If we try to give voting rights to an account that has already voted, the transaction is reverted with the error: *"The voter already voted"*.

–> delegate(address to)

This function allows an account to pass their voting power to another account.

Transaction hash: 0x395c357491695609998326bd15fede5e1b136f421ebd09977e69f83873580cce

For the delegation to work the account that is inheriting the voting power needs to have been given a right to vote, otherwise the transaction will fail with no error message.

The following transaction is a vote cast by the delegated account from the previous transaction shown:

Transaction hash: 0x6c803c4db284dda9f1bad49ad9a91a8eb8b7bb8994d925526dde55e24f1bed3e

If the account delegating doesn't have voting right, the transaction will fail with the error: *"You have no right to vote"*.

If the account delegating has already voted, the transaction will fail with the error: *"You already voted"*.

If the account delegating tries to delegate to itself the transaction will fail with the error: *"Self-delegation is disallowed"*.

If we try to delegate to someone who has already delegated their voting rights to another account, the transaction is reverted with the error: *"Found loop in delegation"*.

–> vote(uint proposal)

This function sets the vote on the proposal's array index passed as argument. Regarding our script we changed it so that the vote will be cast on the proposal's number and not the index. To vote on proposal number 1 (index 0) we call the function with the argument set to 1.

Transaction hash: 0x08563366da991df5f4fbd09df1fd164f222e32aceaaf427f4805c376e1120e68

If the account casting the vote has no right to vote, the transaction will revert with the error: *"Has no right to vote"*.

If the account casting the vote has already voted, the transaction will revert with the error: *"Already voted."*

If the vote is set to an index outside of the array's range, the transaction will revert.

–> winningProposal()

This represents a query function that will return the index of the proposal with the most votes. In the case of our script, it will output the proposal's number and not the index.

In the case that no vote has been cast, when this function is called it will return the first proposal as the winner.

–> winnerName()

This represents a query function that will return the name of the proposal with the most votes.

In the case that no vote has been cast, when this function is called it will return the first proposal's name as the winner.