

## Documentación Práctica final de módulo Emilio Pérez Arjona

A continuación se realizan anotaciones sobre las diferentes partes en las que se divide la práctica y se añaden capturas de pantalla.

### **Smart Contract.**

Consiste en rellenar el código de una DAO. Se da una plantilla con varias funciones y modifiers y el objetivo es rellenar el contenido de las funciones. Hay comentarios explicando los requisitos de cada función.

En este apartado he utilizado prácticamente el mismo código visto ya en las clases.

Lo novedoso fue el uso de un bucle en la función “createProposal” para rellenar los mappings.

```
//iterar sobre el array de opciones para rellenar los mappings. Pista usar _options.length
for(uint8 i = 0; i < _options.length; i++){
    //inicializar las opciones
    proposal.optionsText[i] = _options[i];
    //inicializar los votos
    proposal.optionsVotes[i] = 0;
}
```

Así como la inicialización de un nuevo objeto “MultipleChoiceProposal” en la función “getProposalInfo”.

```
function getProposalInfo(uint256 _proposalId) doesProposalExist(_proposalId) public view returns(MultipleChoiceProposalInfo memory){
    //obtener la propuesta cuyo id es el parametro recibido
    MultipleChoiceProposal storage selectedProposal = proposals[_proposalId];
    //inicializar el objeto MultipleChoiceProposalInfo. Pista no es necesario usar storage, mejor usar memory
    MultipleChoiceProposalInfo memory proposalInfo;
    //inicializar el array optionsText dentro del struct
    proposalInfo.optionsText = new string[](selectedProposal.optionsNumber);//Explica por que no existia ya cuando se creó la propuesta:
    //inicializar el array optionsVotes dentro del struct
    proposalInfo.optionsVotes = new uint8[](selectedProposal.optionsNumber);

    //iterar sobre las opciones. Pista usar el campo optionsNumber como tope del bucle
    for(uint8 i = 0; i < proposals[_proposalId].optionsNumber; i++){
        //coger el optionsText
        proposalInfo.optionsText[i] = proposals[_proposalId].optionsText[i];
        //coger el optionsVotes
        proposalInfo.optionsVotes[i] = proposals[_proposalId].optionsVotes[i];
    }
    //devolver el objeto MultipleChoiceProposalInfo
    return proposalInfo;
}
```

Se añade un control a “voteProposal” ya que actualmente no se impedía que alguien pudiera votar las veces que quiera aún ya habiendo votado (de esto me dí cuenta más adelante, probando el front, vi que podía clicar las veces que quisiera y se enviaba la votación).

```
function voteProposal(uint256 _proposalId, uint8 _optionCode) public doesProposalExist(_proposalId) isProposalActive(_proposalId){
    // Verificar si la dirección ya votó
    if (hasAddressVoted(_proposalId, msg.sender)) {
        revert("You have already voted on this proposal");
    }
    //sumar el voto a su opcion
    proposals[_proposalId].optionsVotes[_optionCode]++;
    //incluir el address como que ya voto
    proposals[_proposalId].voters[msg.sender] = true;
    //emitir el evento ProposalVoted
    emit ProposalVoted(_proposalId, msg.sender);
}
```

```
eth_getTransactionCount
eth_estimateGas
Contract call:      KeepCodingDAO#voteProposal
From:               0x8626f6940e2eb28930efb4cef49b2d1f2c9c1199
To:                 0x5fbdb2315678afecb367f032d93f642f64180aa3
Value:              0 ETH

Error: reverted with reason string 'You have already voted on this proposal'
```

## Test

Están divididos en dos partes. Primero hay unos test ya desarrollados que sirven para comprobar que la lógica implementada en las funciones está desarrollada correctamente. Si ejecutando los tests sale todo bien es que el desarrollo es correcto. Después hay una segunda parte en la que se pide desarrollar tests que comprueben que cuando las condiciones para ejecutar una función no se cumplen salte el error correspondiente. Tal y como hemos hecho en clase.

En esta parte fue de mucha ayuda el código que se aporta ya para los tests ya que, además de la sintaxis para el “deployDAOFixture” también pude fijarme en cómo simular que el tiempo se incrementa a la hora de desarrollar el test “Should revert when the proposal is not active”

```
it("Should revert when the proposal is not active", async function(){
    const {dao} = await loadFixture(deployDAOFixture)
    await time.increase(30 * 60)
    await expect(dao.voteProposal(1, vote)).to.be.revertedWithCustomError(
        dao,
        'ProposalDeadlineExceeded'
    )
})
```

Finalmente, se ejecutan y pasan todos los tests.

```
DAO Test Suite
  Create Proposal
    ✓ Should crete proposal (473ms)
  Get Proposal Info
    ✓ Should get the proposal properly
    ✓ Should revert when the proposal does not exist
  Vote Proposal
    ✓ Should vote the proposal properly
    ✓ Should revert when the proposal does not exist
    ✓ Should revert when the proposal is not active
  Execute Proposal
    ✓ Should execute the proposal properly
    ✓ Should revert when the proposal does not exist
    ✓ Should revert when the proposal cannot be executed

9 passing (495ms)
```

### Front

Por último, hay que desarrollar el front. Como en la parte de los smart contract, existe una parte de código ya hecho, el HTML, y hay que rellenar parte del JS para inicializar los contratos y construir las llamadas a la blockchain.

Aquí me quedé algo atascado probando la funcionalidad.

Creación de la proposal, bien.

```
eth_sendRawTransaction
Contract call: KeepCodingDAO#createProposal
Transaction: 0x19ddb2fb114df66acfc7c1cc9264d899498805fa5356292234edb5163a2a8e79
From: 0x8626f6940e2eb28930efb4cef49b2d1f2c9c1199
To: 0x5fbdb2315678afecb367f032d93f642f64180aa3
Value: 0 ETH
Gas used: 266732 of 266732
Block #2: 0x7d6b7df00e5d195550ba39bb7dc356b59de6b7c4bd48b02b57271a14871879c4
```

Traer la info de la proposal, bien, pero en consola JS, "Undefined"

# CREATE PROPOSAL

# GET PROPOSAL INFO

```
getButton.addEventListener("click", async () => {  
  const response = await getProposalInfo(  
    getInput.value  
  )  
  console.log(response)  
})
```

```
Console was cleared VM1799:1  
< undefined  
undefined app.js:78  
>
```

Sin embargo, por la consola del Visual Studio Code, la operación parece que se realiza. Es más, cuando introduzco un Id que no existe, se revierte con el error que hemos planteado en el ejercicio. Y eso sí que sale por consola.

```
eth_chainId  
eth_call  
Contract call: KeepCodingDAO#getProposalInfo  
From: 0xf39fd6e51aad88f6f4ce6ab8827279cfff92266  
To: 0x5fbdb2315678afecb367f032d93f642f64180aa3
```

[illegible]

```
console.log("Fetching Proposal ID: %s", _proposalId);
console.log("Options Number: %s", selectedProposal.optionsNumber);
for(uint8 i = 0; i < selectedProposal.optionsNumber; i++) {
    console.log("Option %s: %s - Votes: %s", i, selectedProposal.optionsText[i], selectedProposal.optionsVotes[i]);
}
```

```
eth_call
Contract call:      KeepCodingDAO#getProposalInfo
From:               0xf39fd6e51aad88f6f4ce6ab8827279cfff92266
To:                 0x5fbdb2315678afecb367f032d93f642f64180aa3

console.log:
  Fetching Proposal ID: 1
  Options Number: 3
  Option 0: a - Votes: 0
  Option 1: b - Votes: 0
  Option 2: c - Votes: 0
```

La votación de la propuesta, parece que también funciona bien, ya que por la consola del VSC se ve correcto.

```
VOTE PROPOSAL
```

1	1	VOTE PROPOSAL
---	---	---------------

```
eth_chainId  
eth_sendRawTransaction  
Contract call:  
Transaction:    KeepCodingDA0#voteProposal  
                0x79cfd1ce2ec028d4b887399e7449d894333a170659ade74ed875ed869d80c32  
From:           0x8626f6940e2eb28930efb4cef49b2d1f2c9c1199  
To:             0x5fbdb2315678afecb367f032d93f642f64180aa3  
Value:          0 ETH  
Gas used:       72907 of 72907  
Block #3:       0xd480202bf8af592ef45e41bc8eae75d441f31e99952188c093de3d4f3913706f
```

Si haces click repetidamente el control funciona.

[illegible]

La ejecución del contrato, da error tal como se espera, porque aún no pasaron los 30 minutos. Pero sí es cierto, que el error que da es “unrecognized” cuando debería ser del tipo “ProposalCannotBeExecuted”.

## EXECUTE PROPOSAL

1	EXECUTE PROPOSAL
---	------------------

```

x ▶ Uncaught (in promise) Error: execution reverted (unknown custom      ethers.umd.min.js:1
error) (action="estimateGas",
data="0xe98dbc5a00000000000000000000000000000000000000000000000000000001",
reason=null, transaction={ "data":
"0xd61b519000000000000000000000000000000000000000000000000000000001", "from":
"0x8626f6940E2eb28930eFb4CeF49B2d1F2C9C1199", "to":
"0x5FbDB2315678afecb367f032d93F642f64180aa3" }, invocation=null, revert=null,
code=CALL_EXCEPTION, version=6.9.0)
    at makeError (ethers.umd.min.js:1:2851)
    at getBuiltinCallException (ethers.umd.min.js:1:227926)
    at AbiCoder.getBuiltinCallException (ethers.umd.min.js:1:229802)
    at JsonRpcProvider.getRpcError (ethers.umd.min.js:1:351600)
    at ethers.umd.min.js:1:345277

```

[illegible]

De todas formas, en los tests, pasaba la prueba cuando aumentamos el tiempo. Aquí da error, solo que no el que nosotros esperamos.

Con esto, la práctica estaría completada.

**Extra, despliegue en Amoy tesnet.**

Modificar el `hardhat.config` para añadir la network Amoy (punto de acceso y clave privada)

```
require("@omicfoundation/hardhat-toolbox");
require("@omicfoundation/hardhat-verify");

const AMOY_ENDPOINT = "https://polygon-amoy.g.alchemy.com/v2/_3UvND3xyxsHKYHV_zZ6MCJCYMsPiUSP";
const PRIVATE_KEY = "AQUÍ TU CLAVE PRIVADA PARA TESNET";
const POLYGONSACN_APIKEY = "AQUÍ TU API KEY";

/** @type import('hardhat/config').HardhatUserConfig */
module.exports = {
  solidity: "0.8.28",
  networks: {
    amoy: {
      url: AMOY_ENDPOINT,
      accounts: [PRIVATE_KEY]
    },
  },
  etherscan: {
    apiKey: POLYGONSACN_APIKEY
  }
};
```

Creación de un nuevo módulo de despliegue (para no modificar el de la práctica), llamado TesnetDAO.js

```
JS TesnetDAO.js x DAO.sol JS app.js hardhat.config.js JS DAO.js .../m
ignition > modules > JS TesnetDAO.js > [?] <unknown>
1 // This setup uses Hardhat Ignition to manage smart contract deployments.
2 // Learn more about it at https://hardhat.org/ignition
3
4 const { buildModule } = require("@nomicfoundation/hardhat-ignition/modules");
5
6 module.exports = buildModule("TesnetDAOModuleV2", (m) => {
7
8     const dao = m.contract(
9         "KeepCodingDAO",
10        []
11    )
12
13    return {dao}
14 })
```



## Modificar el Front en app.js (Dejo comentadas las líneas 40 y 47)

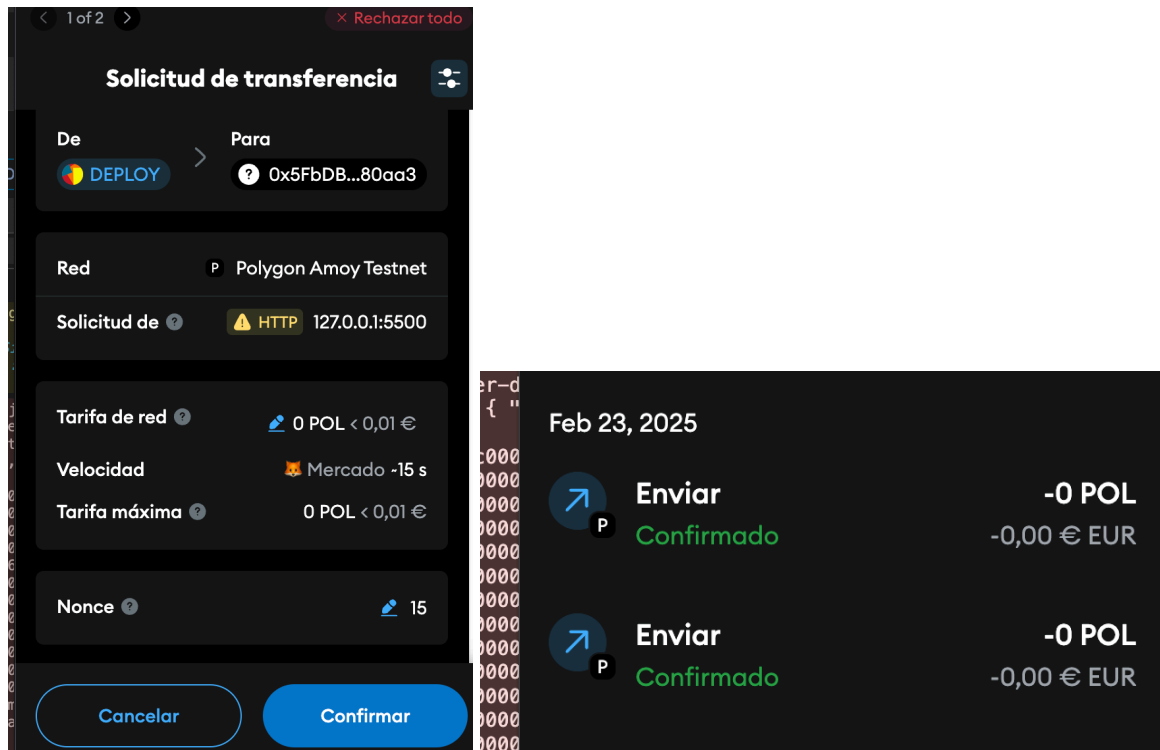
```
1 const HARDHAT_URL = "http://127.0.0.1:8545";
2 const DAO_ADDRESS_LOCALHOST = "0x5fbdb2315678afecb367f032d93f642f64180aa3"
3 const DAO_ADDRESS_TESTNET = ""
4 const SIGNER_PRIVATEKEY = "0xdf57089febbacf7ba0bc227dafbffa9fc08a93fdc68e1e42411a14efcf23656e"
5
6 const initializeMetamask = async () => {
7   let signer = null;
8
9   let provider;
10  if (window.ethereum == null) {
11
12    // If MetaMask is not installed, we use the default provider,
13    // which is backed by a variety of third-party services (such
14    // as INFURA). They do not have private keys installed,
15    // so they only have read-only access
16    console.log("MetaMask not installed; using read-only defaults")
17    provider = ethers.getDefaultProvider()
18
19  } else {
20
21    // Connect to the MetaMask EIP-1193 object. This is a standard
22    // protocol that allows Ethers access to make all read-only
23    // requests through MetaMask.
24    provider = new ethers.BrowserProvider(window.ethereum)
25
26    // It also provides an opportunity to request access to write
27    // operations, which will be performed by the private key
28    // that MetaMask manages for the user.
29    signer = await provider.getSigner();
30
31    return {provider, signer}
32  }
33 }
34
35 const initializeContract = async () => {
36   const response = await fetch('../..artifacts/contracts/DAO.sol/KeepCodingDAO.json')
37   const data = await response.json()
38
39   //provider
40   //const provider = new ethers.JsonRpcProvider(HARDHAT_URL)
41
42   const {provider, signer} = await initializeMetamask() //Alternativa para conexión con Metamask, utilizado para la tesnet
43
44   //read contract
45   const readContract = new ethers.Contract(DAO_ADDRESS_LOCALHOST, data.abi, provider)
46
47   //signer
48   //const signer = new ethers.Wallet(SIGNER_PRIVATEKEY, provider)
49
50   //write contract
51   const writeContract = readContract.connect(signer)
52   //return provider, signer, write contract, read contract
53   return {provider, signer, writeContract, readContract}
54 }
```

## Despliegue en la tesnet

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS COMMENTS
Private Key: 0xf214f2b2cd398c886f846317254e0f6b08106433832737d97a22a48e01628897
Account #11: 0x71b63f3384f5f09895986a0802f2426c5788 (10000 ETH)
Private Key: 0x701b615bb0f09de65240bc28bd21bb0996645a3d5767b12bc2bd6f192c82
Account #12: 0xf4B8Ba9d68888445f87357272f282c5651694a (10000 ETH)
Private Key: 0xa267530f14916280290e0f313ee7af160627f2a80ce289751d80a043f64495701
Account #13: 0x1C83b277809094e10f157d48c24c7c264073C9C (10000 ETH)
Private Key: 0x47c99a0e532427807c28aff1f1267659318cc3f7f0baa8b23d08652942d4
Account #14: 0xf301846486A083f673ab319CCE4f1a57c7807 (10000 ETH)
Private Key: 0xc526ee95bf440bf485a158b0884d9d1238099f0612e9f33d080bb789009aaa
Account #15: 0xcd38766CD06A472114f452C558C635964ce71 (10000 ETH)
Private Key: 0x8160f546ba0d6d521a8369ca086c5d299e46670292085c8f5ee9cc28e84f1f61
Account #16: 0x25468cd3c84621e970201854d14322a277fCEC30 (10000 ETH)
Private Key: 0xe06c44ac0307f658ba7630baa07164820d4310a071c77d61banc7c37642484a0
Account #17: 0x0dA57477F065F806b54c465e087040c51B107E (10000 ETH)
Private Key: 0x689af8efaf8c651a91ad2876825273a72f69f6581a7ac4b061667b5a93a037fd
Account #18: 0xd02F04581271e238368238F933705c84308f4408 (10000 ETH)
Private Key: 0xede9be838da4a475276426328d5e5262ccf3ba408bfac583680fa6c4c20b4ee0
Account #19: 0x8620f6940c26b28930ef94cf4982d172C9C1199 (10000 ETH)
Private Key: 0xdf57089febbacf7ba0bc227dafbffa9fc08a93fdc68e1e42411a14efcf23656e
WARNING: These accounts, and their private keys, are publicly known

emilio@MacBook-Pro-de-Eva PRACTICA_ENL110 % npx hardhat ignition deploy ./ignition/modules/TesnetDAO.js --network amoy
Confirm deploy to network amoy (80802)? - yes
Hardhat Ignition
Resuming existing deployment from ./ignition/deployments/chain-80802
Deploying [ TesnetDAOModuleV2 ]
Warning - previously executed futures are not in the module:
- DAOModuleV2#KeepCodingDAO
Batch #1
Executed TesnetDAOModuleV2#KeepCodingDAO
[ TesnetDAOModuleV2 ] successfully deployed
Deployed Addresses
DAOModuleV2#KeepCodingDAO - 0x8a903ad25a5c4680439e693184093b0227564397
TesnetDAOModuleV2#KeepCodingDAO - 0xc584449Cef9389e24195C8b6ccE4987A3F67436b
% Emilio@MacBook-Pro-de-Eva PRACTICA_ENL110 %
```

## Creación de la propuesta



Se ha podido desplegar y ejecutar una función del contrato sin problemas.

También he intentado verificar el contrato

Para la verificación usé la API KEY que nos dejaste en las clases, la he borrado del repo para que no suba a git.

```
emilio@MacBook-Pro-de-Eva PRACTICA_EMILIO % npm install --save-dev @nomiclabs/hardhat-etherscan
npm warn deprecated @nomiclabs/hardhat-etherscan@3.1.8: The @nomiclabs/hardhat-etherscan package is deprecated, please use @nomicfoundation/hardhat-verify instead
added 8 packages, and audited 584 packages in 5s
97 packages are looking for funding
  run `npm fund` for details
30 vulnerabilities (11 low, 2 high, 17 critical)
To address issues that do not require attention, run:
  npm audit fix
Some issues need review, and may require choosing
a different dependency.
Run `npm audit` for details.
emilio@MacBook-Pro-de-Eva PRACTICA_EMILIO %
```

```
emilio@MacBook-Pro-de-Eva PRACTICA_EMILIO % npx hardhat verify --network amoy 0xC59A449Cef9309e24195C8b6fcE4987A3F67436b
[INFO] Sourcify Verification Skipped: Sourcify verification is currently disabled. To enable it, add the following entry to your Hardhat configuration:
sourcify: {
  enabled: true
}
Or set 'enabled' to false to hide this message.
For more information, visit https://hardhat.org/hardhat-runner/plugins/nomicfoundation-hardhat-verify#verifying-on-sourcify
Successfully submitted source code for contract
contracts/DAO.sol:KeepCodingDAO at 0xC59A449Cef9309e24195C8b6fcE4987A3F67436b
for verification on the block explorer. Waiting for verification result...
Successfully verified contract KeepCodingDAO on the block explorer.
https://amoy.polygonscan.com/address/0xC59A449Cef9309e24195C8b6fcE4987A3F67436b#code
```

Contrato verificado en polygonscan

Contract

0xC59A449Cef9309e24195C8b6fcE4987A3F67436b

Source Code

Overview

POL BALANCE  
0 POL

More Info

CONTRACT CREATOR  
0x73BdBfcc...271A6E76e at txn 0xdbcc8e1f320...

Multichain Info

N/A

Transactions

Token Transfers (ERC-20)

Contract

Events

Code

Read Contract

Write Contract

Contract Source Code Verified (Exact Match)

Contract Name:KeepCodingDAO

Optimization Enabled:No with 200 runs

Compiler Version:v0.8.28+commit.7893614a

Other Settings:paris EvmVersion

Contract Source Code (Solidity Standard Json-Input format)

File 1 of 4 : DAO.sol

1 // SPDX-License-Identifier: MIT

2 pragma solidity ^0.8.28;

3

El repo está subido a git sin claves privadas (tan solo la clave privada que usas tú, del nodo simulado).