Introducción a la tecnología blockchain y smart contracts

Práctica 4

Javier Mulero Martín y Jorge del Valle Vázquez

19 de mayo de 2022

En esta práctica vamos a continuar estudiando las funcionalidades que nos ofrece Solidity, como contratos abstractos y las distintas opciones de visibilidad y modificadores.

1. Asigna direcciones a los siguientes roles.

Role	Address
`Authority`	0x5B38Da6a701c568545dCfcB03FcB875f56beddC4
`President1`	0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2
`President2`	0x4B20993Bc481177ec7E8f571ceCaE8A9e22C02db
`Voter1`	0x78731D3Ca6b7E34aC0F824c42a7cC18A495cabaB
`Voter2`	0x617F2E2fD72FD9D5503197092aC168c91465E7f2
`Voter3`	0x17F6AD8Ef982297579C203069C1DbfFE4348c372
`Voter4`	0x5c6B0f7Bf3E7ce046039Bd8FABdfD3f9F5021678
`Voter5`	0x03C6FcED478cBbC9a4FAB34eF9f40767739D1Ff7

2. Despliega el contrato `Elections` con la dirección `authority`. La votación se creará con 3 partidos.

Hacemos *deploy* con la dirección de `Authority` con `nPartidos` $= 3$.

3. Crea dos sedes electorales mediante la función `createPollingStation` y direcciones de `President1` y `President2` respectivamente. La primera se le corresponderá a Madrid mientras que la segunda a Teruel.

Con la dirección de `Authority`, llamamos a `createPollingStation` para cerar dos sedes:

- La primera con `idRegion` $= 28$ (Madrid), y la dirección de `President1`.
  Esta llamada nos devuelve la dirección del contrato creado `DhondtPollingStation` para Madrid: 0x5C9eb5D6a6C2c1B3EFc52255C0b356f116f6f66D
- La segunda con `idRegion` $= 44$ (Teruel), y la dirección de `President2`.
  Esta llamada nos devuelve la dirección del contrato creado `DhondtPollingStation` para Teruel: 0xb8f43EC36718ecCb339B75B727736ba14F174d77

Una vez tenemos las direcciones de los contratos de las sedes de Madrid y Teruel, los desplegamos seleccionando el contrato `DhondtPollingStation` y pulsamos *At Address* con cada una de las direcciones devueltas.

4. Llama con cada uno de los presidentes a la función `openVoting`. Seleccionamos la dirección de cada presidente, y llamamos a `openVoting` en cada contrato con su correspondiente presidente (solo puede ejecutar el presidente correspondiente)l.

5. Emite un voto con cada una de las siguientes direcciones mediante la función `castVote`:

- `Voter1`: Partido 1 en Madrid.
- `Voter2`: Partido 0 en Teruel.
- `Voter3`: Partido 2 en Madrid.
- `Voter4`: Partido 1 en Madrid.
- `President1`: Partido 1 en Madrid.
- `Voter5`: Partido 0 en Teruel.
- `President2`: Partido 1 en Teruel.

Observación: en nuestra implementación, comprobamos si un votante ya ha votado. Si no lo ha hecho, puede votar correctamente. Si ya ha votado, no revertimos la ejecución y "pierde" el coste de la transacción por intentar "hacer trampas".

6. Llama a la función `getResults` con la dirección de la autoridad administrativa. ¿Qué mensaje se obtiene al lanzar la ejecución?

El mensaje que se obtiene es "No han acabado todas las sedes", ya que los presidentes no han ejecutado la función `closeVoting`.

7. Cierra las votaciones con ambos presidentes con la función `closeVoting`.

Ejecutamos con cada correspondiente presidente en su sede la función `closeVoting` (solo la pueden ejecutar ellos).

8. Llama a la función `getResults` con la dirección de uno de los presidentes. ¿Qué mensaje se obtiene?

Se obtiene "Solo puede ejecutar la autorida", ya que `getResults` tiene el modificador `onlyAuthority`, con lo que solo puede ser ejecutada por `Authority`.

9. Llama a la función `getResults` con la dirección de la autoridad administrativa. ¿Qué resultados se obtienen?

Partido	Resultado
Partido 0	10
Partido 1	8
Partido 2	1

Anexos

A. DhondtElectionRegion.sol

```
1  // SPDX-License-Identifier: GPL-3.0
2
3  pragma solidity >=0.7.0 <0.9.0;
4
5  contract DhondtElectionRegion {
6
7      uint immutable regionId; // immutable pq se le da valor en el
           constructor solo, y no cambia
8      mapping(uint => uint) private weights;
9      uint[] results;
10
11     constructor(uint numPartidos, uint idRegion){
12         savedRegionInfo();
13         regionId = idRegion;
14         results = new uint[](numPartidos); // inicializar results
15     }
16
17     function savedRegionInfo() private {
18         weights[28] = 1; // Madrid
19         weights[8] = 1; // Barcelona
20         weights[41] = 1; // Sevilla
21         weights[44] = 5; // Teruel
22         weights[42] = 5; // Soria
23         weights[49] = 4; // Zamora
24         weights[9] = 4; // Burgos
25         weights[29] = 2; // Malaga
26     }
27
28     function registerVote(uint partido) internal returns (bool) {
29         if (partido < results.length) {
30             results[partido] += weights[regionId];
31             return true;
32         }
33         return false;
34     }
35 }
```

B. PollingStation.sol

```
1  // SPDX-License-Identifier: GPL-3.0
2
3  pragma solidity >=0.7.0 <0.9.0;
4
```

```
5  abstract contract PollingStation {
6
7      bool public votingFinished;
8      bool private votingOpen;
9      address immutable presidente; // Presidente
10
11     modifier onlyPresi {
12         require(msg.sender == presidente, "Solo puede ejecutar el
               presidente");
13         _;
14     }
15
16     modifier onlyOpen {
17         require(votingOpen, "La votacion no esta abierta");
18         _;
19     }
20
21     constructor(address presi) {
22         presidente = presi;
23     }
24
25     function openVoting() external onlyPresi {
26         votingOpen = true;
27     }
28
29     function closeVoting() external onlyPresi {
30         votingOpen = false;
31         votingFinished = true;
32     }
33
34     function castVote(uint partido) virtual external; // pure?
35
36     function getResults() virtual external returns (uint[] memory); //
           pure?
37 }
```

C. DhondtPollingStation.sol

```
1  // SPDX-License-Identifier: GPL-3.0
2
3  pragma solidity >=0.7.0 <0.9.0;
4
5  import "./PollingStation.sol";
6  import "./DhondtElectionRegion.sol";
7
8  contract DhondtPollingStation is PollingStation, DhondtElectionRegion {
9
10     constructor (address presi, uint numPartidos, uint idRegion)
11         PollingStation(presi) DhondtElectionRegion(numPartidos, idRegion){
```

```
    }

    function castVote(uint partido) override external onlyOpen {
        require(registerVote(partido), "El partido al que quiere votar no
            existe");
    }

    function getResults() override external view returns (uint[] memory){
        require(votingFinished, "La votacion no ha sido cerrada aun");
        return results;
    }
}
```

D. Election.sol

```
// SPDX-License-Identifier: GPL-3.0

pragma solidity >=0.7.0 <0.9.0;

import "./DhondtPollingStation.sol";

contract Election {
    mapping(uint => DhondtPollingStation) sedes;
    uint[] private keys;
    mapping(address => bool) votantes; // true si ha votado, false cc
    address autoridad;
    uint numPartidos;

    modifier onlyAuthority{
        require(msg.sender == autoridad, "Solo puede ejecutar la autoridad
            ");
        _;
    }

    modifier freshId(uint regionId) {
        require(address(sedes[regionId]) == address(0x0), "La region ya
            esta registrada ");
        _;
    }

    modifier validId(uint regionId) {
        require(address(sedes[regionId]) != address(0x0), "La region no ha
             sido registrada");
        _;
    }


    constructor(uint nPartidos) {
        numPartidos = nPartidos;
```

```
32          autoridad = msg.sender;
33      }
34
35      function createPollingStation(uint idRegion, address presi) external
            freshId(idRegion) onlyAuthority returns(address) {
36          sedes[idRegion] = new DhondtPollingStation(presi, numPartidos,
                idRegion);
37          keys.push(idRegion);
38          return address(sedes[idRegion]);
39      }
40
41      function castVote(uint idRegion, uint partido) external validId(
            idRegion) {
42          if (!votantes[msg.sender]){
43              votantes[msg.sender] = true;
44              sedes[idRegion].castVote(partido);
45          }
46      }
47
48      function getResults() external onlyAuthority view returns (uint[]
            memory){
49          uint[] memory res = new uint[](numPartidos);
50          for (uint i = 0; i < keys.length; ++i){
51              require(sedes[keys[i]].votingFinished(), "No han acabado todas
                    las sedes");
52              uint[] memory resultRegion = sedes[keys[i]].getResults();
53              for (uint j = 0; j < numPartidos; ++j){
54                  res[j] += resultRegion[j];
55              }
56          }
57          return res;
58      }
59
60      // function getResults2() external onlyAuthority view returns (uint
            [] memory){
61      //     uint[] memory res = new uint[](numPartidos);
62      //     for (uint i = 0; i < keys.length; ++i){
63      //         require(sedes[keys[i]].votingFinished(), "No han acabado
            todas las sedes");
64      //     }
65      //     for (uint i = 0; i < keys.length; ++i){
66      //         uint[] memory resultRegion = sedes[keys[i]].getResults();
67      //         for (uint j = 0; j < numPartidos; ++j){
68      //             res[j] += resultRegion[j];
69      //         }
70      //     }
71      //     return res;
72      // }
73  }
```