PRACTICA 4 BLOCKCHAIN

Alejandro Ramírez y David Seijas

March 2022

1 Código

1.1 DhontElectionRegion

```
contract DhontElectionRegion{

    mapping(uint => uint) private weights;
    uint immutable regionId;
    uint[] internal results;

    constructor(uint _regionId, uint _nParties){
        regionId = _regionId;
        results = new uint[](_nParties);
        savedRegionInfo();
    }

    function savedRegionInfo() private{
        weights[28] = 1; // Madrid
        weights[8] = 1; // Barcelona
        weights[41] = 1; // Sevilla
        weights[44] = 5; // Teruel
        weights[42] = 5; // Soria
        weights[49] = 4; // Zamora
        weights[9] = 4; // Burgos
        weights[29] = 2; // Malaga
    }

    function registerVote(uint _party) internal returns(bool){
        if(_party >= 0 && _party < results.length){
            results[_party] += weights[regionId];
            return true;
        }
        else return false;
    }
}
```

1.2 PollingStation

```
abstract contract PollingStation{

```

```
3       bool public votingFinish;
4       bool private votingOpen;
5       address immutable presidentDesk;
6
7       modifier onlyPresident{
8           require(msg.sender == presidentDesk, "No eres el presidente
                de la mesa. No tienes permitido esta accion.");
9           _;
10      }
11
12      modifier openVote{
13          require(votingOpen, "La votacion no esta abierta. No puedes
                votar.");
14          _;
15      }
16
17      constructor(address _presidentDesk){
18          presidentDesk = _presidentDesk;
19          votingFinish = false;
20          votingOpen = false;
21      }
22
23      function openVoting() external onlyPresident{
24          votingOpen = true;
25      }
26
27      function closeVoting() external onlyPresident{
28          votingOpen = false;
29          votingFinish = true;
30      }
31
32      function castVote(uint _party) external virtual;
33      function getResults() external view virtual returns(uint[]
            memory);
34  }
```

1.3 DhondtPollingStation

```
1   contract DhontPollingStation is PollingStation, DhontElectionRegion
        {
2
3       constructor(address _presidentDesk, uint _nParties, uint
            _regionId)
4       DhontElectionRegion(_regionId, _nParties)
5       PollingStation(_presidentDesk){}
6
7       function castVote(uint _party) override external openVote{
8           require(registerVote(_party), "El partido votado no es
                valido.");
9       }
10
11      function getResults() override external view returns(uint[]
            memory){
12          return results;
13      }
14  }
```

1.4 Election

```
1  contract Election{
2
3      mapping(uint => DhontPollingStation) mappingSedes;
4      uint[] regions;
5      uint immutable nParties;
6      mapping(address => bool) votants;
7      address owner;
8
9      modifier onlyAuthority {
10         require(msg.sender == owner, "Solo la autoridad
               administrativa tiene permiso para realizar esta accion.
               ");
11         _;
12     }
13
14     modifier freshId(uint regionId) {
15         require(address(mappingSedes[regionId]) == address(0));
16         _;
17     }
18
19     modifier validId(uint regionId) {
20         require(address(mappingSedes[regionId]) !=  address(0));
21         _;
22     }
23
24     constructor(uint _nParties){
25         owner = msg.sender;
26         nParties = _nParties;
27     }
28
29     function createPollingStation(uint _regionId, address
           _presidentDesk) external freshId(_regionId) onlyAuthority
           returns(address){
30         DhontPollingStation ps = new DhontPollingStation(
               _presidentDesk, nParties, _regionId);
31         mappingSedes[_regionId] = ps;
32         regions.push(_regionId);
33         return address(ps);
34     }
35
36     function castVote(uint _regionId, uint _party) external validId
           (_regionId){
37         require(!votants[msg.sender], "Solo puedes votar 1 vez");
               //No castigamos a los tramposos. Utilizando if else sin
                revert les supondr a coste extra por intentar
               enga ar.
38         votants[msg.sender] = true;
39         mappingSedes[_regionId].castVote(_party);
40     }
41
42     function getResults() external view onlyAuthority returns(uint
           [] memory){
43         uint[] memory results = new uint[](nParties);
44         for(uint i = 0; i < regions.length; ++i){
45             if(!mappingSedes[regions[i]].votingFinish()){
```

```
46                  revert("No todas las sedes han acabado su votacion.
                        ");
47              }
48              else{
49                  uint[] memory res = mappingSedes[regions[i]].
                        getResults();
50                  for(uint j = 0; j < res.length; ++j){
51                      results[j] += res[j];
52                  }
53              }
54          }
55          return results;
56      }
57  }
```

2 Prueba de uso

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.

   Dirección contrato Election: 0xd9145CCE52D386f254917e481eB44e9943F39138

3. Crea dos sedes electorales (polling station) mediante la función `createPollingStation` y direcciones de `President1` y `President2` respectivamente. La primera sede corresponderá a Madrid mientras que la segunda a Teruel.

   Contrato Sede Madrid: 0x5C9eb5D6a6C2c1B3EFc52255C0b356f116f6f66D
   Contrato Sede Teruel: 0xb8f43EC36718ecCb339B75B727736ba14F174d77

4. Llama con cada uno de los presidentes a la función `openVoting`.

   Transaction Cost de ambas llamadas: 43503

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.

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

   The transaction has been reverted to the initial state. Reason provided by the contract: "No todas las sedes han acabado su votacion.". Debug the transaction to get more information.

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

   Transaction Cost de ambas llamadas: 26692

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

   The transaction has been reverted to the initial state. Reason provided by the contract: "Solo la autoridad administrativa tiene permiso para realizar esta accion.". Debug the transaction to get more information.

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

Transacción:
from 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4
to Election.getResults() 0xd9145CCE52D386f254917e481eB44e9943F39138
execution cost 75808 gas (Cost only applies when called by a contract)
hash 0xd25267f6cdca78f8bd337a32f570fa8bc2232809bfd5e1488120a59821640c0a
input 0x471...7f97c
decoded input {}
decoded output { "0": "uint256[]: 10,8,1" }
*hash*logs []