PRACTICA 5 BLOCKCHAIN

Alejandro Ramírez y David Seijas

March 2022

1 Código

```
// SPDX-License-Identifier: GPL-3.0
//ALEJANDRO RAMÍREZ Y DAVID SEIJAS
pragma solidity >=0.7.0 <0.9.0;

interface ERC721simplified {
    // EVENTS
    event Transfer(address indexed _from, address indexed _to,
        uint256 indexed _tokenId);
    event Approval(address indexed _owner, address indexed
        _approved, uint256 indexed _tokenId);

    // APPROVAL FUNCTIONS
    function approve(address _approved, uint256 _tokenId) external
        payable;

    // TRANSFER FUNCTION
    function transferFrom(address _from, address _to, uint256
        _tokenId) external payable;

    // VIEW FUNCTIONS (GETTERS)
    function balanceOf(address _owner) external view returns (
        uint256);
    function ownerOf(uint256 _tokenId) external view returns (
        address);
    function getApproved(uint256 _tokenId) external view returns (
        address);
}

library ArrayUtils {
    function contains(string[] storage arr, string memory val)
        external view returns(bool) {
        for (uint i = 0; i < arr.length; i++)
            if (keccak256(abi.encodePacked(arr[i])) == keccak256(
                abi.encodePacked(val)))
                return true;
        return false;
    }

    function increment(uint[] storage arr, uint8 per) external {
        for (uint i = 0; i < arr.length; i++)
            arr[i] += arr[i]*per;
```

```
33      }
34
35      function sum(uint[] storage arr) external view returns(uint) {
36          uint suma = 0;
37          for (uint i = 0; i < arr.length; i++)
38              suma += arr[i];
39          return suma;
40      }
41  }
42
43  contract MonsterTokens is ERC721simplified{
44
45      struct Weapons {
46          string[] names; // name of the weapon
47          uint[] firePowers; // capacity of the weapon
48      }
49
50      struct Character {
51          string name; // character name
52          Weapons weapons; // weapons assigned to this character
53          address ownerToken; //propietario del personaje (token)
54          address approvedOwnerToken; //direccion autorizada por
                Owner
55      }
56
57      mapping(uint => Character) characters;
58      address immutable owner;
59      uint contToken = 10000;
60      mapping(address => uint) n_tokens;
61
62      constructor(){
63          owner = msg.sender;
64      }
65
66      modifier onlyOwner{
67          require(msg.sender == owner, "No eres el propietario, no
                tienes estos permisos");
68          _;
69      }
70
71      modifier onlyPropietary(uint _tokenId){
72          require(characters[_tokenId].ownerToken == msg.sender, "No
                eres propietario del token");
73          _;
74      }
75
76      modifier onlyPropOrApproved(uint _tokenId){
77          require(characters[_tokenId].ownerToken == msg.sender ||
                characters[_tokenId].approvedOwnerToken == msg.sender,
                "No eres propietario del token ni estas autorizado");
78          _;
79      }
80
81      function createMonsterToken(string memory _name, address
            _ownerToken) external onlyOwner returns(uint) {
82          contToken += 1;
83          characters[contToken] = Character(_name, Weapons(new string
```

```
            [](0), new uint[](0)), _ownerToken, address(0));
84          n_tokens[_ownerToken] += 1;
85          return contToken;
86      }
87
88      function addWeapon(uint _tokenId, string memory _nameWeapon,
            uint _powerWeapon) external onlyPropietary(_tokenId) {
89          require(!ArrayUtils.contains(characters[_tokenId].weapons.
                names, _nameWeapon),"Ya tienes este arma");
90          characters[_tokenId].weapons.names.push(_nameWeapon);
91          characters[_tokenId].weapons.firePowers.push(_powerWeapon);
92      }
93
94      function incrementFirePower(uint _tokenId, uint8 _per) public {
95          ArrayUtils.increment(characters[_tokenId].weapons.
                firePowers, _per);
96      }
97
98      function collectProfits() public onlyOwner{
99          payable(msg.sender).transfer(address(this).balance);
100     }
101
102     function approve(address _approved, uint256 _tokenId) override
            external onlyPropietary(_tokenId) payable{
103         require(msg.value >= ArrayUtils.sum(characters[_tokenId].
                weapons.firePowers), "Necesitas mas Weis para ejecutar
                esta funcion");
104         characters[_tokenId].approvedOwnerToken = _approved;
105         if(_approved != address(0)) //No mandamos approval en caso
                de haber revocado al aprobado anterior
106             emit Approval(msg.sender, _approved, _tokenId);
107     }
108
109     function transferFrom(address from, address to, uint256
            _tokenId) override external onlyPropOrApproved(_tokenId)
            payable{
110         require(msg.value >= ArrayUtils.sum(characters[_tokenId].
                weapons.firePowers), "Necesitas mas Weis para ejecutar
                esta funcion");
111         characters[_tokenId].ownerToken = to;
112         characters[_tokenId].approvedOwnerToken = address(0);
113         n_tokens[from] -= 1;
114         n_tokens[to] += 1;
115         emit Transfer(from, to, _tokenId);
116     }
117
118     function balanceOf(address _owner) override external view
            returns(uint256){
119         return n_tokens[_owner];
120     }
121
122     function ownerOf(uint256 tokenId) override external view
            returns(address){
123         require(characters[tokenId].ownerToken != address(0), "No
                existe el propietario de este token");
124         return characters[tokenId].ownerToken;
125     }
```

```
126
127     function getApproved(uint256 tokenId) override external view
            returns(address){
128         require(tokenId >= 10001 && tokenId <= contToken, "tokenId
                invalido");
129         return characters[tokenId].approvedOwnerToken;
130     }
131 }
```

2 Prueba de uso

1. Asigna tres direcciones distintas con los siguientes roles:

Role	Address
GameMaster	0x5B38Da6a701c568545dCfcB03FcB875f56beddC4
TokenOwner1	0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2
TokenOwner2	0x4B20993Bc481177ec7E8f571ceCaE8A9e22C02db

2. Despliega el contrato `MonsterToken` con la dirección `GameMaster`.

   Dirección contrato: 0xd8b934580fcE35a11B58C6D73aDeE468a2833fa8

3. Crea dos tokens, cada uno propiedad de un TokenOwner. Añade dos armas a cada token con distintos valores.

   Transacción createMonsterToken 1:
   transaction cost 103534
   decoded input {
   "string _name": "David",
   "address _ownerToken": "0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2"
   }
   decoded output { "0": "uint256: 10001" }
   val 0 wei
   Transacción createMonsterToken 2:
   transaction cost 103522
   decoded input {
   "string _name": "Alex",
   "address _ownerToken": "0x4B20993Bc481177ec7E8f571ceCaE8A9e22C02db"
   }
   decoded output { "0": "uint256: 10002" }
   val 0 wei

   Insertamos en 10001: pistola con 1 y bazoca con 2.

Insertamos en 10002: revolver con 2 y ballesta con 1

4. Realiza una transferencia de uno de los tokens a `GameMaster`.

Transferencia de 10002 a GameMaster
transaction cost 67082
decoded input {
"address _from": "0x4B20993Bc481177ec7E8f571ceCaE8A9e22C02db",
"address _to": "0x5B38Da6a701c568545dCfcB03FcB875f56beddC4",
"uint256 _tokenId": "10002"
}
decoded output {}
log {
"from": "0x358AA13c52544ECCEF6B0ADD0f801012ADAD5eE3",
"topic": "0xddf252ad1be2c89b69c2b068fc378daa952ba7f163c4a11628f55a4df523b3ef"
"event": "Transfer"
"args": {
"0": "0x4B20993Bc481177ec7E8f571ceCaE8A9e22C02db",
"1": "0x5B38Da6a701c568545dCfcB03FcB875f56beddC4"
"2": "10002"
"_from": "0x4B20993Bc481177ec7E8f571ceCaE8A9e22C02db",
"_to": "0x5B38Da6a701c568545dCfcB03FcB875f56beddC4"
"_tokenId": "10002"
}
}
val 3 wei

5. Autoriza a `GameMaster` para poder operar el otro token.

Aprobación de 10001 a GameMaster
transaction cost 60301
decoded input {
"address_approved": "0x5B38Da6a701c568545dCfcB03FcB875f56beddC4",
"uint256_tokenId": "10001"
}
decoded output {}
log {
"from": "0x358AA13c52544ECCEF6B0ADD0f801012ADAD5eE3",
"topic": "0x8c5be1e5ebec7d5bd14f71427d1e84f3dd0314c0f7b2291e5b200ac8c7c3b925"
"event": "Approval"
"args": {
"0": "0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2",
"1": "0x5B38Da6a701c568545dCfcB03FcB875f56beddC4"

"2": "10001"
"_owner": "0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2",
"_approved": "0x5B38Da6a701c568545dCfcB03FcB875f56beddC4"
"_tokenId": "10001"
}
}
val 10 wei

6. Llama a las funciones `balanceOf`, `ownerOf` y `getApproved` para verificar que las operaciones anteriores se han realizado correctamente.

balanceOf	**Output**	**Gas**
0x5B38Da6a701c568545dCfcB03FcB875f56beddC4	1	24295
0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2	1	24295
0x4B20993Bc481177ec7E8f571ceCaE8A9e22C02db	0	24295

ownerOf	**Address**	**Gas**
10001	0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2	26248
10002	0x5B38Da6a701c568545dCfcB03FcB875f56beddC4	24345

getApproved	**Output**	**Gas**
10001	0x5B38Da6a701c568545dCfcB03FcB875f56beddC4	26248
10002	0x0000000000000000000000000000000000000000	26248