```
// SPDX-License-Identifier: GPL-3.0
pragma solidity ^0.8.0;

contract CodeCriptoDao {
    mapping(address => uint) public balances;
    address[] public miembros;
    uint public totalMiembros;
    uint public totalBalance;

    struct Propuesta {
        string descripcion;
        address destinatario;
        uint monto;
        uint votos;
        bool activa;
    }
    Propuesta[] public propuestas;
    mapping(address => mapping (uint => bool)) public propuestasVotadas;
    event EjecutaPropuestaOffChain(address destinatario, string descripcion);

    constructor(address[] memory _miembros) {
        miembros = _miembros;
        totalMiembros = _miembros.length;
    }

    function deposito() public payable {
        balances[msg.sender] += msg.value;
        totalBalance += msg.value;
    }

    function retirar(uint _monto) public {
```

```
        require(balances[msg.sender] >= _monto, "No posees balance");
        balances[msg.sender] -= _monto;
        totalBalance -= _monto;
        payable(msg.sender).transfer(_monto);
    }

    function transferir(address _destinatario, uint _monto) public {
        require(balances[msg.sender] >= _monto, "No posees balance");
        balances[msg.sender] -= _monto;
        balances[_destinatario] += _monto;
    }

    function createPropuesta(string memory _descripcion, address _destinatario, uint _monto)
public {
        Propuesta memory nuevaPropuesta;
        nuevaPropuesta.descripcion = _descripcion;
        nuevaPropuesta.destinatario = _destinatario;
        nuevaPropuesta.monto = _monto;
        nuevaPropuesta.votos = 0;
        nuevaPropuesta.activa = true;

        propuestas.push(nuevaPropuesta);
    }

    function votar(uint _idProposta) public {
        uint idProposta = _idProposta - 1;
        Propuesta storage propuesta = propuestas[idProposta];
        require(propuesta.activa, "La propuesta no esta activa");
        require(noHasVotado(idProposta), "Ya ha votado");
        propuestasVotadas[msg.sender][idProposta] = true;
        propuesta.votos += 1;
```

```
    }

    function ejecutarLaPropuesta(uint _idProposta) public {
        uint idProposta = _idProposta - 1;
        Propuesta storage propuesta = propuestas[idProposta];

        require(propuesta.activa, "La propuesta no esta activa");
        require(propuesta.votos > totalMiembros / 2, "No hay suficientes votos");
        require(propuesta.monto <= totalBalance, "No hay suficiente balance");

        payable(propuesta.destinatario).transfer(propuesta.monto);
        totalBalance -= propuesta.monto;
        propuesta.monto = 0;
        propuesta.activa = false;

        emit EjecutaPropuestaOffChain(propuesta.destinatario, propuesta.descripcion);
    }

    function noHasVotado(uint _indiceDeProposta) internal view returns (bool) {
        return propuestasVotadas[msg.sender][_indiceDeProposta] == false;
    }

    function numeroDePropuestas() public view returns(uint) {
        return propuestas.length;
    }
}
```