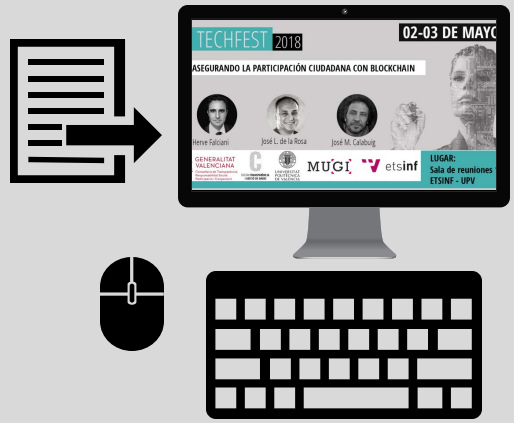
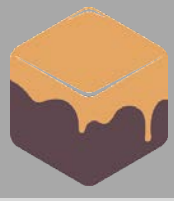
Hervé Falciani

Pep Lluís de la Rosa

Jose M. Calabuig

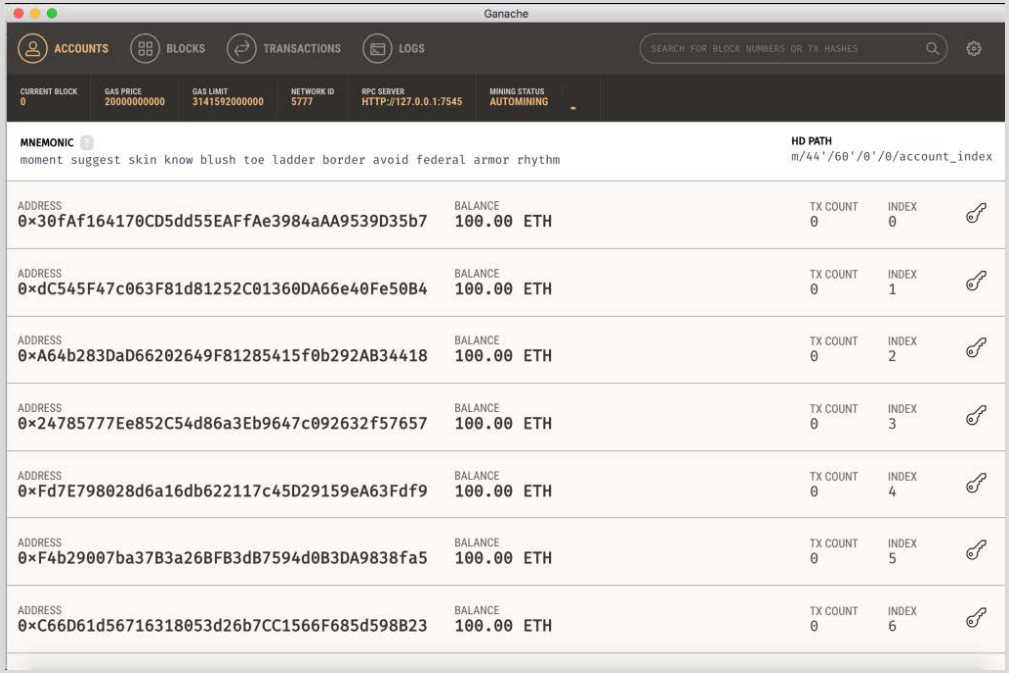


Ganache sin terminal



http://truffleframework.com/ganache/

El puerto por defecto es el 7545



Contratos desde Remix

ethereum

https://remix.ethereum.org

Remix - Solidity IDE

Jose Manuel

Es seguro | https://remix.ethereum.org/#optimize=false&version=builtin

Aplicaciones | Visualizaciones R | Whitepapers | https://www.analyti... | Blockchain | Remix - Solidity IDE | Practicas UPV

browser/ballot.sol

Compile | Run | Settings | Analysis | Debugger | Support

browser

config

```
pragma solidity ^0.4.11;

/// @title Votación con voto delegado
contract Ballot {
    // Declara un nuevo tipo de dato complejo, que será
    // usado para almacenar variables.
    // Representará a un único votante.
    struct Voter {
        uint weight; // el peso del voto se acumula mediante la delegación de votos
        bool voted;  // true si esa persona ya ha votado
        address delegate; // persona a la que se delega el voto
        uint vote;   // índice de la propuesta votada
    }

    // Representa una única propuesta.
    struct Proposal {
        bytes32 name;   // nombre corto (hasta 32 bytes)
        uint voteCount; // número de votos acumulados
    }

    address public chairperson;

    // Declara una variable de estado que
    // almacena una estructura de datos `Voter` para cada posible dirección.
    mapping(address => Voter) public voters;

    // Una matriz dinámica de estructuras de datos de tipo `Proposal`.
    Proposal[] public proposals;

    // Crea una nueva votación para elegir uno de los `proposalNames`.
    function Ballot(bytes32[] proposalNames) {
        chairperson = msg.sender;
        voters[chairperson].weight = 1;

        // Para cada nombre propuesto
        // crea un nuevo objeto de tipo Proposal y lo añade
        // al final del array.
        for (uint i = 0; i < proposalNames.length; i++) {
            // `Proposal({...})` crea una nuevo objeto de tipo Proposal
            // de forma temporal y se añade al final de `proposals`
            // mediante `proposals.push(...)`.
            proposals.push(Proposal({
                name: proposalNames[i],
                voteCount: 0
            }));
        }
```

Environment: Web3 Provider Custom (1524993994747)

Account: 0x2c7...b8d64 (99.999999999987925)

Gas limit: 3000000

Value: 0 wei

Create

Load contract from Address | At Address

0 pending transactions

0 contract Instances

[2] only remix transactions, script | Search transactions | Listen on network

En la pestaña **Run** selecciona **Web3 Provider** con el puerto

Mejorando el código

ethereum


```
pragma solidity ^0.4.11;

  // Votación con voto delegado
contract Ballot {

  // Único votante.
  struct Voter {
      uint weight; // el peso del voto se acumula mediante la delegación de votos
      bool voted;  // true si esa persona ya ha votado
      address delegate; // persona a la que se delega el voto
      uint vote;   // índice de la propuesta votada
  }

  // Única propuesta.
  struct Proposal {
      bytes32 name;   // nombre corto (hasta 32 bytes)
      uint voteCount; // número de votos acumulados
  }
```

Mejorando el código

ethereum


```
address public chairperson;

// Variable de estado que almacena una estructura de datos
// `Voter` para cada posible dirección.
mapping(address => Voter) public voters;

// Matriz dinámica de estructuras de datos de tipo `Proposal`.
Proposal[] public proposals;
```

Mejorando el código

ethereum


```
// Creamos una nueva votación para elegir uno de los `proposalNames`.

function Ballot(bytes32[] proposalNames) {
    chairperson = msg.sender;
    voters[chairperson].weight = 1;

  // Para cada nombre propuesto crea un nuevo objeto de tipo Proposal
  // y lo añadimos al final del array.
    for (uint i = 0; i < proposalNames.length; i++) {

        // `Proposal({...})` crea una nuevo objeto de tipo Proposal
        // de forma temporal y se añade al final de `proposals`
        // mediante `proposals.push(...)`.
        proposals.push(Proposal({
            name: proposalNames[i],
            voteCount: 0
        }));
    }
}
```

Mejorando el código

ethereum


```
// Proporciona a `voter` el derecho a votar en esta votación.
// Sólo puede ser ejecutado por `chairperson`.
function giveRightToVote(address voter) {

    // Si el argumento de `require` da como resultado `false`,
    // finaliza la ejecución y revierte todos los cambios
    // producidos en el estado y los balances de Ether.
    // A veces es buena idea usar esto por si las funciones
    // están siendo ejecutadas de forma incorrecta.

    //Pero ten en cuenta
    // que de esta forma se consumirá todo el gas enviado
    // (está previsto que esto cambie en el futuro).

    require((msg.sender == chairperson) && !voters[voter].voted);
    voters[voter].weight = 1;
}
```

Mejorando el código

ethereum


```
// Delega tu voto a `to`.
function delegate(address to) {
    // Asignación por referencia
    Voter sender = voters[msg.sender];
    require(!sender.voted);

    // No se permite la delegación a uno mismo.
    require(to != msg.sender);

    // Propaga la delegación en tanto que `to` también delegue.
    // Por norma general, los bucles son muy peligrosos porque si tienen muchas iteraciones
    // puede darse el caso de que empleen más gas del disponible en un bloque.
    // En este caso, eso implica que la delegación no será ejecutada pero en otros casos
    //puede suponer que un contrato se quede completamente bloqueado.
    while (voters[to].delegate != address(0)) {
        to = voters[to].delegate;

        // Encontramos un bucle en la delegación. No está permitido.
        require(to != msg.sender);
    }
```

Mejorando el código

ethereum


```
    // Dado que `sender` es una referencia, esto
    // modifica `voters[msg.sender].voted`
    sender.voted = true;
    sender.delegate = to;
    Voter delegate = voters[to];
    if (delegate.voted) {

        // Si la persona en la que se ha delegado el voto ya ha votado,
        // se añade directamente al número de votos.
        proposals[delegate.vote].voteCount += sender.weight;
    } else {

        // Si la persona en la que se ha delegado el voto
        // todavía no ha votado, se añade al peso de su voto.
        delegate.weight += sender.weight;
    }
}
```

Mejorando el código

ethereum


```
// Da tu voto (incluyendo los votos que te han delegado)
// a la propuesta `proposals[proposal].name`.
function vote(uint proposal) {
    Voter sender = voters[msg.sender];
    require(!sender.voted);
    sender.voted = true;
    sender.vote = proposal;

    // Si `proposal` está fuera del rango de la matriz,
    // esto lanzará automáticamente una excepción y
    // se revocarán todos los cambios
    proposals[proposal].voteCount += sender.weight;
}
```

Mejorando el código

ethereum


```
// Calcula la propuesta ganadora teniendo en cuenta
// todos los votos realizados.
function winningProposal() constant
        returns (uint winningProposal)
{
    uint winningVoteCount = 0;
    for (uint p = 0; p < proposals.length; p++) {
        if (proposals[p].voteCount > winningVoteCount) {
            winningVoteCount = proposals[p].voteCount;
            winningProposal = p;
        }
    }
}
```

Mejorando el código

ethereum


```
  // Llama a la función winningProposal() para obtener
  // el índice de la propuesta ganadora y así luego devolver el nombre.
  function winnerName() constant
          returns (bytes32 winnerName)
  {
      winnerName = proposals[winningProposal()].name;
  }
}
```

okay

THANKYOU

MADφ
MATHEMATICS & APPLICATIONS TO DATA & PHYSICS