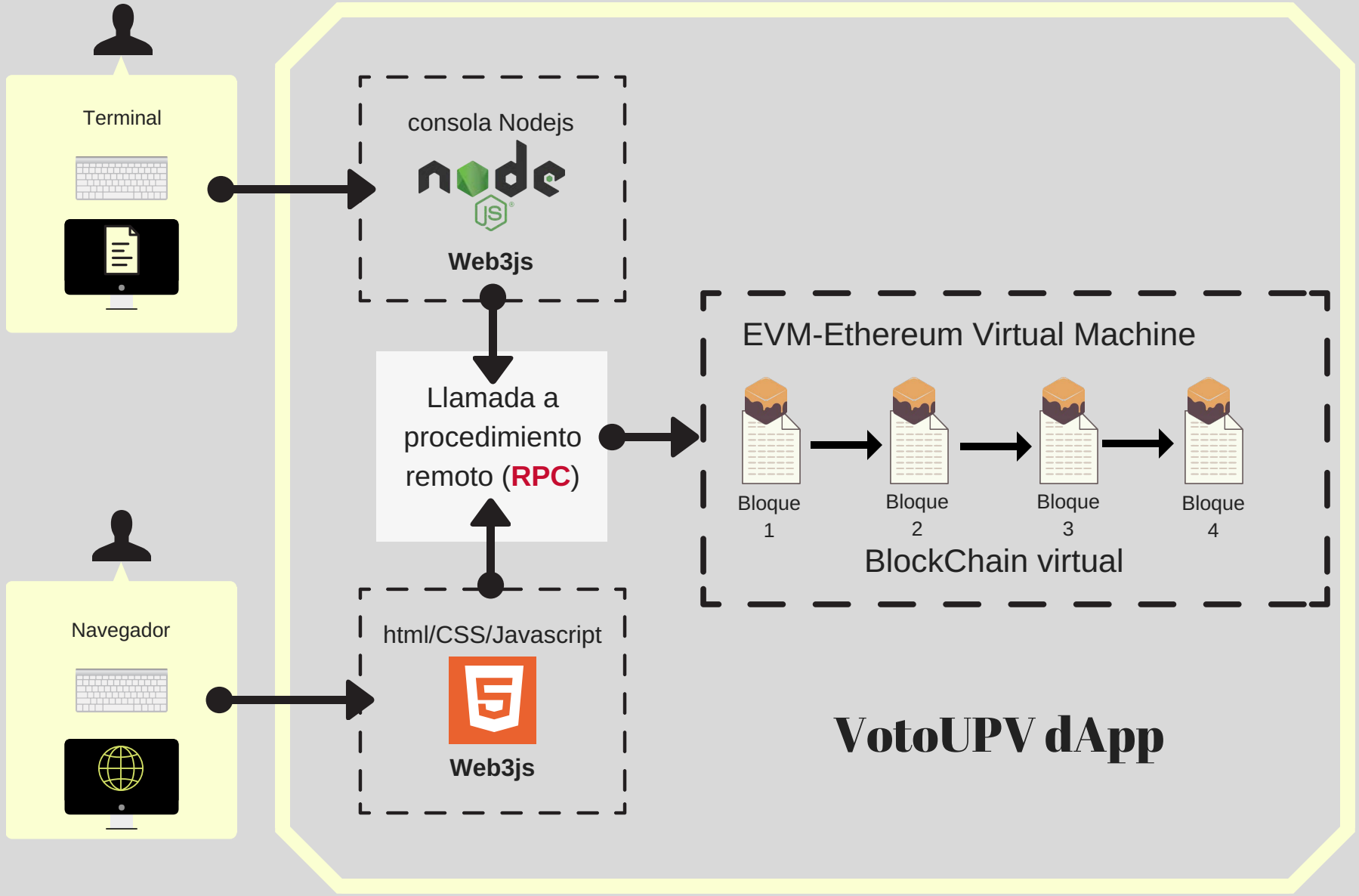
Hervé Falciani

Pep Lluís de la Rosa

Jose M. Calabuig

Esquema de la dApp: VotoUPV



Instalación

jmcalabu.blogs.upv.es

Grupo MadPhy

Taller TechFest 2018

Instalación Linux | Instalación Windows | Instalación Mac Os

```
sudo apt-get update
curl -sL https://deb.nodesource.com/setup_7.x -o
nodesource_setup.sh
sudo bash nodesource_setup.sh
sudo apt-get install nodejs
node --version
npm --version mkdir -p tallerUPV
cd tallerUPV
npm install ganache-cli web3@0.20.1 solc
node_modules/.bin/ganache-cli
```

Instalación Linux | Instalación Windows | Instalación Mac Os

En primer lugar instala Homebrew: https://brew.sh/.
Este paquete te ayudará a instalar otros paquetes directamente desde el terminal.

```
brew update brew install nodejs
node --version
npm --version
mkdir -p tallerUPV
cd tallerUPV
npm install ganache-cli web3@0.20.1 solc
node_modules/.bin/ganache-cli
```

Instalación Linux | Instalación Windows | Instalación Mac Os

1. Instalar **Visual Studio Community Edition**. Es necesario como mínimo tener todo lo relacionado con Visual C++.
2. Instalar **Windows SDK for Windows.**
3. Instalar **Python 2.7**. Asegúrate que está en el PATH (**sino pincha aquí**).
4. Instalar **git** añadiéndolo de nuevo al PATH.
5. Instalar **OpenSSL**. Asegúrate de elegir el paquete adecuado (según tu arquitectura) y elige la instalación completa. No cambies la ruta predefinida para la instalación.
6. Descarga e instala **node v8.1.2.**
7. Ejecuta ahora el comando npm install ganache-cli web3@0.20.1 solc

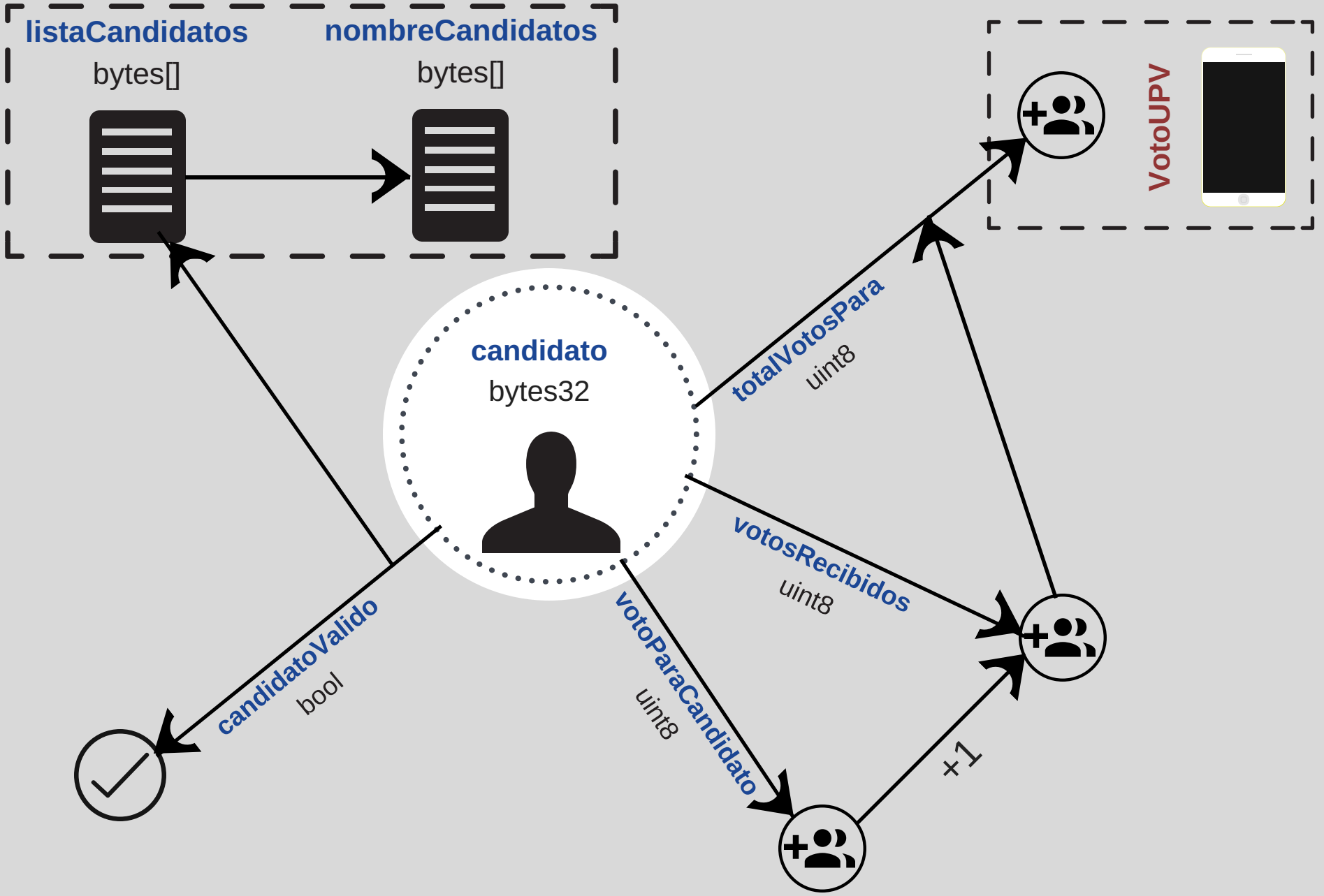
```
Ganache CLI v6.0.3 (ganache-core: 2.0.2)
Available Accounts
==================
(0) 0x5c252a0c0475f9711b56ab160a1999729eccce97
(1) 0x353d310bed379b2d1df3b727645e200997016ba3
(2) 0xa3ddc09b5e49d654a43e161cae3f865261cabd23
(3) 0xa8a188c6d97ec8cf905cc1dd1cd318e887249ec5
(4) 0xc0aa5f8b79db71335dacc7cd116f357d7ecd2798
(5) 0xda695959ff85f0581ca924e549567390a0034058
(6) 0xd4ee63452555a87048dcfe2a039208d113323790
(7) 0xc60c8a7b752d38e35e0359e25a2e0f6692b10d14
(8) 0xba7ec95286334e8634e89760fab8d2ec1226bf42
(9) 0x208e02303fe29be3698732e92ca32b88d80a2d36

Private Keys
==================
(0) a6de9563d3db157ed9926a993559dc177be74a23fd88ff5776
(1) 17f71d31360fbafbc90cad906723430e9694daed3c24e1e9e1
(2) ad2b90ce116945c11eaf081f60976d5d1d52f721e659887fce
(3) 68e2288df55cbc3a13a2953508c8e0457e1e71cd8ae62f0c78
(4) 9753b05bd606e2ffc65a190420524f2efc8b16edb8489e734a
(5) 6e8e8c468cf75fd4de0406a1a32819036b9fa64163e8be5bb6
(6) c287c82e2040d271b9a4e071190715d40c0b861eb248d5a671
(7) cec41ef9ccf6cb3007c759bf3fce8ca485239af1092065aa52
(8) c890580206f0bbea67542246d09ab4bef7eeaa22c3448dcb72
(9) eb8841a5ae34ff3f4248586e73fcb274a7f5dd2dc07b352d2c

HD Wallet
==================
Mnemonic:   cancel better shock lady capable main crun
Base HD Path: m/44'/60'/0'/0/{account_index}

Listening on localhost:8545
```

Solidity contract: VotoUPV.sol

Contrato inteligente: VotoUPV.sol

```
pragma solidity ^0.4.18;
contract VotoUPV {
  mapping (bytes32 => uint8) public votosRecibidos;
   bytes32[] public listaCandidatos;
  function VotoUPV(bytes32[] nombreCandidatos) public {
              listaCandidatos = nombreCandidatos;  }
  function totalVotosPara(bytes32 candidato) view public returns (uint8) {
             require(candidatoValido(candidato));
              return votosRecibidos[candidato];  }
  function votoParaCandidato(bytes32 candidato) public {
              require(candidatoValido(candidato));
              votosRecibidos[candidato] += 1;  }
  function candidatoValido(bytes32 candidato) view public returns (bool) {
              for(uint i = 0; i < listaCandidatos.length; i++) {
                      if (listaCandidatos[i] == candidato) {
                           return true;}}
                      return false;}}
```

Solidity contract: VotoUPV.sol



Compilando web3js + node

Arrancamos node en otro terminal
Cargamos el código en la variable de cadena: code
Compilamos con solc.compile almacenando el resultado en **compiledCode**

web3js es una librería que permite interactuar con blockchain vía RPC. Así podremos desplegar el contrato e interactuar con blockchain

La librería **solc** permite compilar el código

Compilando el contrato

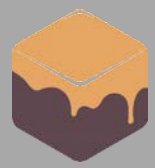
```
node
> Web3 = require('web3')
> web3 = new Web3(new Web3.providers.HttpProvider("http://localhost:8545"));
> web3.eth.accounts
> code = fs.readFileSync('VotoUPV.sol').toString()
> solc = require('solc')
> compiledCode = solc.compile(code)
```

bytecode
es el número de código en el que se desplegará en el blockchain

Si escribimos en el terminal compiledCode veremos dos campos importantes:
compiledCode.contracts[':VotoUPV'].bytecode
compiledCode.contracts[':VotoUPV'].interface

interface
es la plantilla del contrato (llamado **abi**) que informa al usuario de las variables disponibles en el contrato

Desplegando



Desplegando el contrato

```
> abiDefinition = JSON.parse(compiledCode.contracts[':VotoUPV'].interface)
> VotoUPVContract = web3.eth.contract(abiDefinition)
> byteCode = compiledCode.contracts[':VotoUPV'].bytecode
> deployedContract = VotoUPVContract.new(['Herve','Pep Lluis','Jose'],{data: byteCode, from: web3.eth.accounts[0],
gas: 4700000})
> deployedContract.address
'0xeb85d153f6da07c2a9caa64032fc32a8e8b9067f' <- ESTA SERÁ LA DIRECCIÓN DEL CONTRATO
> contractInstance = VotoUPVContract.at(deployedContract.address)
```

VotoUPVContract es el nuevo objeto que se despliega en el blockchain mediante **VotoUPVContract.new**

data es el byteCode

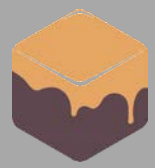
El despliegue se almacena en **deployedContract** y tiene tres parámetros

from es el propietario en nuestro caso sólo usamos la primera

gas el dinero que cuesta interactuar en la red

contractInstance utiliza la dirección **deployedContract.address** para interactuar en el blockchain

Desplegando



Desplegando el contrato

```
> abiDefinition = JSON.parse(compiledCode.contracts[':VotoUPV'].interface)
> VotoUPVContract = web3.eth.contract(abiDefinition)
> byteCode = compiledCode.contracts[':VotoUPV'].bytecode
> deployedContract = VotoUPVContract.new(['Herve','Pep Lluis','Jose'],{data: byteCode, from: web3.eth.accounts[0],
gas: 4700000})
> deployedContract.address
'0xeb85d153f6da07c2a9caa64032fc32a8e8b9067f' <- ESTA SERÁ LA DIRECCIÓN DEL CONTRATO
> contractInstance = VotoUPVContract.at(deployedContract.address)
```

VotoUPVContract es el nuevo objeto que se despliega en el blockchain mediante **VotoUPVContract.new**

data es el byteCode

El despliegue se almacena en **deployedContract** y tiene tres parámetros

from es el propietario en nuestro caso sólo usamos la primera

gas el dinero que cuesta interactuar en la red

contractInstance utiliza la dirección **deployedContract.address** para interactuar en el blockchain

Probando desde el terminal

Probando el contrato...a mano

```
> contractInstance.totalVotosPara.call('Herve')
{ [String: '0'] s: 1, e: 0, c: [ 0 ] }
> contractInstance.votoParaCandidato('Herve', {from: web3.eth.accounts[0]})
'0xdedc7ae544c3dde74ab5a0b07422c5a51b5240603d31074f5b75c0ebc786bf53'
> contractInstance.votoParaCandidato('Herve', {from: web3.eth.accounts[0]})
'0x02c054d238038d68b65d55770fabfca592a5cf6590229ab91bbe7cd72da46de9'
> contractInstance.votoParaCandidato('Herve', {from: web3.eth.accounts[0]})
'0x3da069a09577514f2baaa11bc3015a16edf26aad28dffbcd126bde2e71f2b76f'
> contractInstance.totalVotosPara.call('Herve').toLocaleString()
'3'
```

Cada vez votamos por un candidato obtenemos una **identificación de la transacción**:

0xdedc7ae544c3dde74ab5a0b07422c5a51b5240603d31074f5b75c0ebc786bf53

Este identificador de transacción es la prueba de ésta se ha producido esta transacción y puede consultarlo en cualquier momento en el futuro.

Esta transacción es **inmutable**

Probando desde la web

... y desde la web: index.html

```
<!DOCTYPE html><html>
<head>
   <title> DApp Asegurando la participación ciudadana</title>
   <link href='https://fonts.googleapis.com/css?family=Open Sans:400,700' rel='stylesheet' type='text/css'>
   <link href='https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css' rel='stylesheet'
type='text/css'>
</head>
<body class="container">
   <h1>DApp Asegurando la participación ciudadana con BlockChain. TechFest 2018 UPV</h1>
   <div class="table-responsive">  <table class="table table-bordered">
      <thead>  <tr>     <th>Candidato</th>     <th>Número de votos</th>       </tr>   </thead>
      <tbody>    <tr>     <td>Herve</td>       <td id="candidate-1"></td>    </tr>
                  <tr>     <td>Pep Lluis</td>   <td id="candidate-2"></td>    </tr>
                  <tr>     <td>Jose</td>        <td id="candidate-3"></td>    </tr>
   </tbody>  </table>
   </div> <input type="text" id="candidate" />
   <a href="#" onclick="votoParaCandidato()" class="btn btn-primary">Pincha para votar</a>
</body>
<script src="https://cdn.rawgit.com/ethereum/web3.js/develop/dist/web3.js"></script>
<script src="https://code.jquery.com/jquery-3.1.1.slim.min.js"></script>
<script src="./index.js"></script>
</html>
```

index.html
llama a
index.js
que interactúa
con el blockchain

Probando desde la web

...index.js

```
web3 = new Web3(new Web3.providers.HttpProvider("http://localhost:8545"));
abi = JSON.parse('
   [{"constant":true,"inputs":[{"name":"","type":"uint256"}],"name":"listaCandidatos","outputs":
   [{"name":"","type":"bytes32"}],"payable":false,"stateMutability":"view","type":"function"},
     {"constant":false,"inputs":
   [{"name":"candidato","type":"bytes32"}],"name":"votoParaCandidato","outputs":
   [],"payable":false,"stateMutability":"nonpayable","type":"function"},{"constant":true,"inputs":
   [{"name":"candidato","type":"bytes32"}],"name":"totalVotosPara","outputs":
   [{"name":"","type":"uint8"}],"payable":false,"stateMutability":"view","type":"function"},{"constant":true,"inputs":
   [{"name":"candidato","type":"bytes32"}],"name":"candidatoValido","outputs":
   [{"name":"","type":"bool"}],"payable":false,"stateMutability":"view","type":"function"},{"constant":true,"inputs":
   [{"name":"","type":"bytes32"}],"name":"votosRecibidos","outputs":
   [{"name":"","type":"uint8"}],"payable":false,"stateMutability":"view","type":"function"},{"inputs":
   [{"name":"nombreCandidatos","type":"bytes32[]"}],"payable":false,"stateMutability":
       "nonpayable","type":"constructor"}]')

VotoUPVContract = web3.eth.contract(abi);
contractInstance = VotoUPVContract.at('0xeb85d153f6da07c2a9caa64032fc32a8e8b9067f');
candidatos = {"Herve": "candidate-1", "Pep Lluis": "candidate-2", "Jose": "candidate-3"}

function votoParaCandidato() {
   nombreCandidato = $("#candidato").val();
   contractInstance.votoParaCandidato(nombreCandidato,
   {from: web3.eth.accounts[0]}, function() {
   let div_id = candidatos[nombreCandidato];
   $("#" + div_id).html(contractInstance.totalVotosPara.call(nombreCandidato).toString());});

$(document).ready(function() {
   nombreCandidatos = Object.keys(candidatos);
   for (var i = 0; i < nombreCandidatos.length; i++) {
   let name = nombreCandidatos[i]
   let val = contractInstance.totalVotosPara.call(name).toString()
   $("#"+candidatos[name]).html(val); } });
```

Probando desde la web

DApp Asegurando la participación ciudadana con BlockChain. TechFest 2018 UPV

Candidato	Número de votos
Herve	15
Pep Lluis	5
Jose	2

Pincha para votar

okay

THANKYOU

MADφ
MATHEMATICS & APPLICATIONS TO DATA & PHYSICS