ETSIT UPM

dit UPM

Blockchain: Desarrollo de Aplicaciones

Acceso a los Nodos

BCDA 2018

Versión: 2018-10-17

Índice

- ~~Repasar ideas blockchain, ethereum.~~
- ~~Caso de estudio: Contrato Contador MUY sencillo.~~
- ~~Desarrollar en el nodo:~~
  - ~~Lanzar nodo, crear cuentas, desarrollar contratos, usar los contratos, ...~~
- Acceder al nodo desde fuera:
  - HTTP JSON RPC, WS RPC, IPC RPC,
  - Web3js, Web3j...
- Desarrollar una aplicación node.
- Desarrollar un aplicación web de cliente (SPA).
  - Repaso: html, css, js, promesas, async/await, sagas, webpack,
- Truffle:
  - Compilar, despliegue y migraciones, desarrollar scripts y clientes web, pruebas, ...
- Drizzle:
  - Integración React-Redux,
  - Repaso: React, Redux,

Comunicación App-Nodo

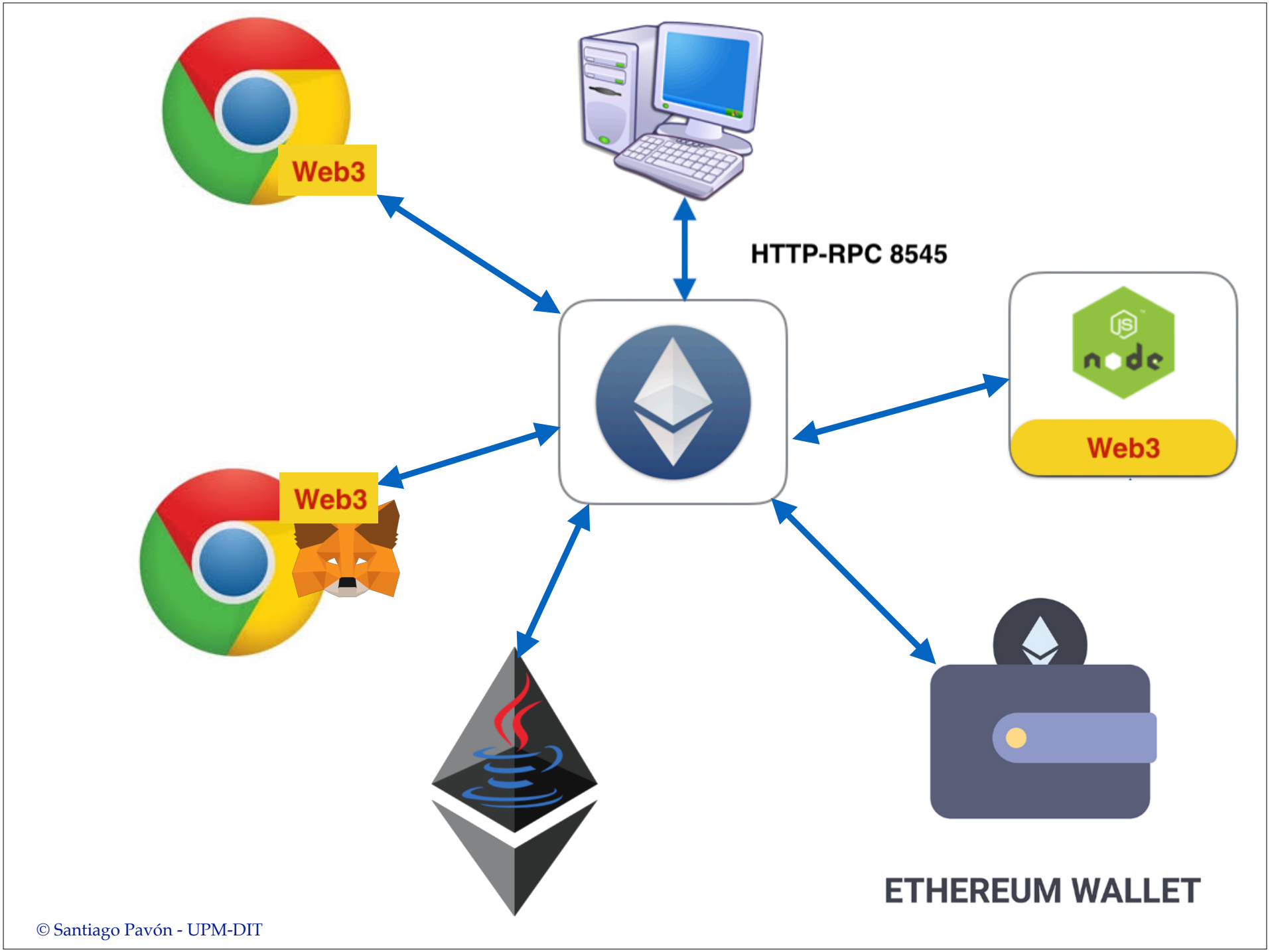
- Una aplicación externa debe comunicarse con un nodo de la red Ethereum para realizar transacciones, usar los contratos inteligentes, consultar la cadena de bloques, etc...

- Para comunicare con un nodo: JSON RPC
  - https://github.com/ethereum/wiki/wiki/JSON-RPC
  - Protocolo de llamadas a procedimientos remotos ligero, sin estado, intercambiando JSON.
  - HTTP, WS, IPC, ...
    - Soporte full duplex (ws, ipc) requerido para soportar subscripciones/notificaciones a eventos.

- Comunicarse usando la librería web3.js
  - Para comunicarse con el nodo ethereum desde una aplicación JavaScript usando este lenguaje.
    - https://github.com/ethereum/wiki/wiki/JavaScript-API
  - Existen librerías para otros lenguajes web3j para java, ...

Web3 JavaScript app API

- Documentación:
  - Versión web3.js 0.2x.x
    - https://github.com/ethereum/wiki/wiki/JavaScript-API
  - Versión web3.js 1.0
    - https://web3js.readthedocs.io/en/1.0/index.html
    - Cambios en muchas funciones, añade promesas y mantiene callbacks, emisores de eventos para seguir los diferentes estados por los que pasa una acción, ...
- La librería JavaScript Web3.js ofrece una interface más amigable para acceder a los nodos.
  - Internamente usa JSON RPC.



Geth

- Opciones de línea de comandos para configurar acceso RPC.

```
$ geth --help
API AND CONSOLE OPTIONS:
  --rpc                    Enable the HTTP-RPC server
  --rpcaddr value          HTTP-RPC server listening interface
                           (default: "localhost")
  --rpcport value          HTTP-RPC server listening port (default: 8545)
  --rpcapi value           API's offered over the HTTP-RPC interface
  --ws                     Enable the WS-RPC server
  --wsaddr value           WS-RPC server listening interface
                           (default: "localhost")
  --wsport value           WS-RPC server listening port (default: 8546)
  --wsapi value            API's offered over the WS-RPC interface
  --wsorigins value        Origins from which to accept websockets requests
  --ipcdisable             Disable the IPC-RPC server
  --ipcpath                Filename for IPC socket/pipe within the datadir
                           (explicit paths escape it)
  --rpccorsdomain value    Comma separated list of domains from which
                           to accept
                           cross origin requests (browser enforced)
  --rpcvhosts value        Comma separated list of virtual hostnames from
                           which to accept requests (server enforced).
                           Accepts '*' wildcard. (default: "localhost")
```

- Ejemplo:
  - Activar el servidor HTTP JSON-RPC atendiendo en el puerto 8545 desde el interface localhost:

```
$ geth --rpc --rpcaddr localhost --rpcport 8545
```

  - Si el acceso se va a realizar desde un navegador Web, es necesario usar la opcion --rpccorsdomain para indicar los dominios desde los que se puede acceder. Por defecto solo se permite el acceso desde el mismo origen.

```
$ geth --rpc --rpccorsdomain "http://localhost:3000"
```

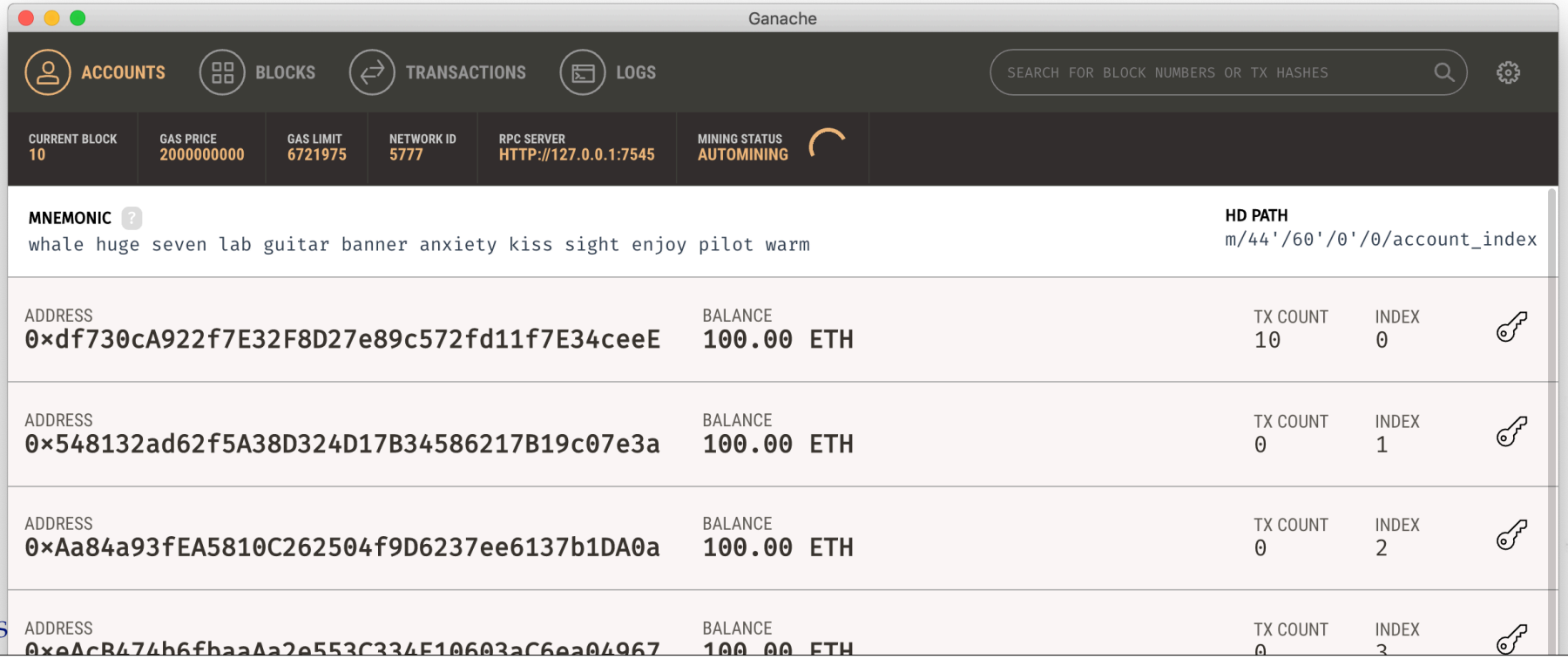
  - Usar la opción --rpcapi para indicar a que APIs se pueden hacer peticiones.
    - Las APIs existentes son admin, debug, eth, miner, net, personal, shh, txpool, web3.

- NOTA: La configuración del acceso RPC también puede hacerse desde la consola.
  - Ejemplo:

```
- admin.startRPC("localhost", 8545)
```

Ganache

- Red ligera para desarrollo
  - Interface gráfica, proporciona cuentas con dinero, mina instantáneamente un bloque para cada transacción, ...
- También ganache-cli
  - Util para especificar cuanto tiempo se tarda en minar un bloque, ...



Ejemplos: HTTP JSON RPC

Ejemplo: Dirección Base

- Lanzar un nodo geth desde un terminal:
  - permitiendo acceso HTTP JSON RPC
  - por defecto escucha en localhost:8545

```
$ geth --testnet --rpc
```

- Desde otro terminal usamos **curl** para hacer peticiones HTTP JSON.

- Consultar dirección base:

```
$ curl -X POST -H "Content-Type: application/json"
       --data '{"jsonrpc":"2.0",
                "method":"eth_coinbase",
                "id":1}' localhost:8545
{"jsonrpc":"2.0",
 "id":1,
 "result":"0x795ff0852b2f79bf4652f29cf65bd192d3252aa2"}
```

Formato de los parámetros

- Los valores se pasan en el JSON codificados en hexadecimal.
  - Los valores numéricos se representan en hexadecimal precedidos de "0x".
    - "0x0" es 0
    - "0x400" es 1024
  - Otros tipos de valores se tratan como una cadena de bytes, representados en hexadecimal, precedidos de "0x" y usando dos dígitos hexadecimales por byte.
    - "0x" es ""
    - "0x686f6c61" es "hola"
      - *Comprobar que los ejemplos hex anteriores están bien:*

```
$ geth --testnet console 2> /dev/ttys001
> web3.toHex("hola")
"0x686f6c61"
> web3.toHex(1024)
"0x400"
```

- Algunos métodos (eth_getBalance, eth_getCode, eth_call, ...) tienen un parámetro para indicar un bloque de la cadena.
  - Este parámetro puede ser el número de bloque en hexadecimal, o los strings "earliest", "latest" o "pending".

Ejemplo: Consultar Valor del Contador

- Consultar el estado actual del contador que hemos desplegado:
  - Primero veamos cómo se codifica la llamada a **valor()** calculando el hash desde su signatura.

```
> web3.sha3("valor()").substring(0,10)
"0xecbac7cf"
```

  - Otra forma de calcularlo es usando el objeto **contractInstance** que creamos en el ejemplo del tema de uso de la consola Geth:

```
> contractInstance.valor.getData()
'0xecbac7cf'
```

  - También podemos obtenerlo del fichero Contador.signatures que se creo al compilar el contrato Contador.sol usando la opción --hashes de solc (o de Remix, ...).

```
119fbbd4: incr()
ecbac7cf: valor()
```

- La dirección donde se desplegó el contrato Contador era:

```
"0xe5c52e5fa9bd9cd8e78010ef53131e754bffa144"
```

- La petición HTTP JSON RPC sería:

```
$ curl -X POST -H "Content-Type: application/json"
  --data '{
  "jsonrpc":"2.0",
  "method": "eth_call",
  "params": [{
     "from": "0x795ff0852b2f79bf4652f29cf65bd192d3252aa2",
     "to": "0xe5c52e5fa9bd9cd8e78010ef53131e754bffa144",
     "data": "0xecbac7cf"},
    "latest"],
  "id": 2}' localhost:8545
{"jsonrpc":"2.0",
 "id":2,
 "result":"0x000000000000000000000000000000000000000000000000000000000000000d"}
```

  - El campo params contiene los parámetros de eth_call: el campo data con el método que queremos ejecutar, y el bloque de la cadena donde consultar.
  - El resultado se devuelve en result: 13 *("0xd" es 13 en decimal).*

Métodos con Parámetros

- El campo **data** del ejemplo anterior solo contenía la codificación del nombre del método **valor()**.
- Si se quiere llamar a un método que tenga parámetros, hay que concatenar en el campo data la codificación hexadecimal de los parámetros, rellenando con ceros para ajustar a 32 bytes.
- Por ejemplo, suponga que existiese el método add(uint256)
  - Para llamarlo pasando 5 como parámetro, se necesita codificarlo así:

```
> web3.sha3("add(uint256)").substring(0, 10)
"0x1003e2d2"
> web3.toHex(5)
"0x5"
```

    - Otra forma de codificar la llamada:

```
> contractInstance.add.getData(5)
"0x1003e2d2000000000000000000000000000000000000000000000000000000000000005"
```

      - El parámetro se rellena con ceros para que ocupe 32 bytes.
  - Y en el campo **data** habría que poner:

```
"0x39020680000000000000000000000000000000000000000000000000000000000000005"
```

Ejemplo: Incrementar el Contador

- En este ejemplo estamos creando una transacción para modificar el estado del contrato.
- Desbloqueamos la cuenta.

```
> var primaryAddress = eth.accounts[0];
> personal.unlockAccount(primaryAddress, "1234", 3600)
```

- La codificación del método incr() es "0x119fbbd4".
  - Se calcula igual que en el ejemplo anterior

```
> web3.sha3("valor()").substring(0,10)
"0x119fbbd4"
> contractInstance.valor.getData()
'0x119fbbd4'
```

  - Mirarlo en el fichero Contador.signatures.

- Hacemos la petición HTTP JSON:

```
$ curl -X POST -H "Content-Type: application/json"
  --data '{
   "jsonrpc": "2.0",
   "method": "eth_sendTransaction",
   "id": 3,
  "params": [{
      "gas": 200000,
      "from": "0x795ff0852b2f79bf4652f29cf65bd192d3252aa2",
      "to": "0xe5c52e5fa9bd9cd8e78010ef53131e754bffa144",
      "data": "0x119fbbd4"}]}' localhost:8545
{"jsonrpc":"2.0",
 "id":3,
 "result":"0xb648c9a455f256fb06d55299230bf9af6832f0aeb513a94dabccdb7fe3a241a3"}
```

- Devuelve el hash de la transacción creada.

Más Ejemplos JSON RPC

- Verlos en la página:

  https://github.com/ethereum/wiki/wiki/JSON-RPC

Ejemplos: App con Nodejs

Web3 JavaScript app API

- Documentación:
  - Versión web3.js 0.2x.x
    - https://github.com/ethereum/wiki/wiki/JavaScript-API
  - Versión web3.js 1.0
    - https://web3js.readthedocs.io/en/1.0/index.html
    - Cambios en muchas funciones, añade promesas y mantiene callbacks, emisores de eventos para seguir los diferentes estados por los que pasa una acción, ...
- La librería JavaScript Web3.js ofrece una interface más amigable para acceder a los nodos.
  - Internamente usa JSON RPC.

- Primero hay que obtener web3.js para poder usarla en nuestros programas.

```
$ npm install web3@0.20.x
```

- Lanzamos node en modo interprete para ejecutar los ejemplos paso a paso, o le indicamos que ejecute directamente un programa contenido en un fichero:

```
$ node
$ node <fichero.js>
```

- Hay que crear una instancia de web3 asignándola el proveedor a usar para conectarse con el nodo Ethereum.
  - Hay que tener cuidado de no modificar el proveedor si este ya ha sido asignado.
  - Para ello se puede comprobar si ya existe una instancia de web3.
  - Por ejemplo, si estamos usando MIST, o MetaMask, ya existirá una instancia web3 con un proveedor asignado.

- El código típico para hacer esto es:

```
var Web3 = require("web3");  // Cargar paquete web3

if (typeof web3 !== 'undefined') {  // Ya existe instancia web3
   web3 = new Web3(web3.currentProvider);  // Usamos el mismo Provider
} else {  // Usar Provider que especifiquemos.
   web3 = new Web3(new Web3.providers.HttpProvider("http://localhost:7545"));
}
```

Usando el nodo Ganache

- Código para desplegar un Contador

```
var primaryAddress = web3.eth.accounts[0];

var abi = [{"constant":false,, ... ETC ..."name":"Tic","type":"event"}];

var code = "0x6080604 ... ETC ... fc0029";

var MyContract = web3.eth.contract(abi);

var contractInstance = MyContract.new({
    from: primaryAddress,
    data: code,
    gas: 500000},
    function(err, contract) {
        if (err) {
            console.log("Error =", err);
            process.exit(1)
        }
        if (contract.address) {
            console.log("Contrato desplegado en",contract.address);
            process.exit(0)
        }

  });
```

- Ejecución del ejemplo 1-desplegar.js:

```
$ node 1-desplegar.js
Contrato desplegado en 0xb25be7ca2686c860249dcc98beb828e82da18ce0
```

```
var Web3 = require("web3");  // Cargar paquete web3

if (typeof web3 !== 'undefined') {  // Ya existe instancia web3
   web3 = new Web3(web3.currentProvider);  // Usamos el mismo Provider
} else {  // Usar Provider: GANACHE
   web3 = new Web3(new Web3.providers.HttpProvider("http://localhost:7545"));
}

var primaryAddress = web3.eth.accounts[0];

var abi = [{"constant":false,"inputs":[],"name":"incr","outputs":
[],"payable":false,"stateMutability":"nonpayable","type":"function"},{"constant":true,"inputs":
[],"name":"valor","outputs":
[{"name":"","type":"uint8"}],"payable":false,"stateMutability":"view","type":"function"},
{"payable":false,"stateMutability":"nonpayable","type":"fallback"},{"anonymous":false,"inputs":
[{"indexed":false,"name":"msg","type":"string"},
{"indexed":false,"name":"out","type":"uint8"}],"name":"Tic","type":"event"}];
var code =
"0x6080604052600080600061010a81548160ff021916908360ff160217905550348015610 02b57600080fd5b5061019b8061003b60003
96000f3006080604052600436106100 4c576000357c0100000000000000000000000000000000000000000000000000000000900463ffffff
fff168063119fbbd41461005e578063ecbac7cf14610075575b34801561005857600080fd5b50600080fd5b34801561006a57600080fd5b5
06100736100a6565b005b34801561008157600080fd5b5061008a61015d565b60405180826 0ff1660ff16815260200191505060405180910
390f35b60008081819054906101000a900460ff16809291906001019190610100 0a81548160ff021916908360ff160217905550507f27873
3a8e0534f74d81486a11876429bb0d35a6968fa576ec403ad7aecfa2e6e60008090549061010 00a900460ff16604051808060200183 60ff1
660ff16815260200182810382526 00b8152602001807f41637475616c697a61646f00000000000000000000000000000000000000008152
5060200192505050604051809103 90a1565b60008090549061010 00a900460ff16815600a165627a7a723058206f3f3f1b3275d1f83c6ec
ff99a4a971f0af10f7532a6c651cb8d6c013b9a89fc0029";

var MyContract = web3.eth.contract(abi);

var contractInstance = MyContract.new({
          from: primaryAddress,
    data: code,
    gas: 500000},
    function(err, contract) {
        if (err) {
            console.log("Error =", err);
            process.exit(1)
        }
        if (contract.address) {
            console.log("Contrato desplegado en",contract.address);
            process.exit(0)
        }
    });
```

1-desplegar.js

- Consultar el valor del Contador:

```
var abi = [{"constant":false, ..... "type":"event"}];

var addr = "0xb25be7ca2686c860249dcc98beb828e82da18ce0";

var MyContract = web3.eth.contract(abi);

var contractInstance = MyContract.at(addr);

var valor = contractInstance.valor();

console.log("Valor =", valor);
console.log("Valor =", valor.toNumber());
```

  - Nota: Como JavaScript no maneja números my muy grandes, se usa la librería BigNumber para manejar los valores numéricos.

- Ejecución del ejemplo 2-consultar.js

```
$ node 2-consultar.js
Valor = BigNumber { s: 1, e: 0, c: [ 0 ] }
Valor = 0
```

```
var Web3 = require("web3");  // Cargar paquete web3

if (typeof web3 !== 'undefined') {  // Ya existe instancia web3
   web3 = new Web3(web3.currentProvider);  // Usamos el mismo Provider
} else {  // Usar Provider: GANACHE
   web3 = new Web3(new Web3.providers.HttpProvider("http://localhost:7545"));
}

var abi = [{"constant":false,"inputs":[],"name":"incr","outputs":
[],"payable":false,"stateMutability":"nonpayable","type":"function"},
{"constant":true,"inputs":[],"name":"valor","outputs":
[{"name":"","type":"uint8"}],"payable":false,"stateMutability":"view","type":"function
"},{"payable":false,"stateMutability":"nonpayable","type":"fallback"},
{"anonymous":false,"inputs":[{"indexed":false,"name":"msg","type":"string"},
{"indexed":false,"name":"out","type":"uint8"}],"name":"Tic","type":"event"}];

var addr = "0xb25be7ca2686c860249dcc98beb828e82da18ce0";

var MyContract = web3.eth.contract(abi);

var contractInstance = MyContract.at(addr);

var valor = contractInstance.valor();

console.log("Valor =", valor);
console.log("Valor =", valor.toNumber());
```

2-consultar.js

- Código para incrementar el valor del contador:

```
var primaryAddress = web3.eth.accounts[0];

var abi = [{"constant":false, ... "type":"event"}];
var addr = "0xb25be7ca2686c860249dcc98beb828e82da18ce0";

var MyContract = web3.eth.contract(abi);
var contractInstance = MyContract.at(addr);

var valor1 = contractInstance.valor();
console.log("Valor Inicial =", valor1.toNumber());

contractInstance.incr.sendTransaction({
    from: primaryAddress,
    gas: 200000},
    function(err, result) {
        if (err) {
            console.log("Error =", err);
            process.exit(1)
        }
        var valor2 = contractInstance.valor();
        console.log("Valor final =", valor2.toNumber());
        process.exit(0)
    });
```

- Ejecución del ejemplo 3-incrementar.js

```
$ node 3-incrementar.js
Valor Inicial = 2
Valor final = 3
```

```
var Web3 = require("web3");  // Cargar paquete web3

if (typeof web3 !== 'undefined') {  // Ya existe instancia web3
   web3 = new Web3(web3.currentProvider);  // Usamos el mismo Provider
} else {  // Usar Provider: GANACHE
   web3 = new Web3(new Web3.providers.HttpProvider("http://localhost:7545"));
}

var primaryAddress = web3.eth.accounts[0];

var abi = [{"constant":false,"inputs":[],"name":"incr","outputs":
[],"payable":false,"stateMutability":"nonpayable","type":"function"},{"constant":true,"inputs":[],"name":"valor","outputs":
[{"name":"","type":"uint8"}],"payable":false,"stateMutability":"view","type":"function"},
{"payable":false,"stateMutability":"nonpayable","type":"fallback"},{"anonymous":false,"inputs":
[{"indexed":false,"name":"msg","type":"string"},{"indexed":false,"name":"out","type":"uint8"}],"name":"Tic","type":"event"}];
var addr = "0xb25be7ca2686c860249dcc98beb828e82da18ce0";

var MyContract = web3.eth.contract(abi);
var contractInstance = MyContract.at(addr);

var valor1 = contractInstance.valor();
console.log("Valor Inicial =", valor1.toNumber());

contractInstance.incr.sendTransaction({
    from: primaryAddress,
    gas: 200000},
    function(err, result) {
        if (err) {
            console.log("Error =", err);
            process.exit(1)
        }
        var valor2 = contractInstance.valor();
        console.log("Valor final =", valor2.toNumber());
        process.exit(0)
    });
```

3-incrementar.js

- Observar eventos Tic:

```
var abi = [{"constant":false, ... "type":"event"}];

var addr = "0xb25be7ca2686c860249dcc98beb828e82da18ce0";

var MyContract = web3.eth.contract(abi);
var contractInstance = MyContract.at(addr);

contractInstance.Tic(function(err, data) {
    console.log("Se ha producido un evento Tic:");
    if (err){
        console.log(err);
    } else {
        var msg = data.args.msg;
        var out = data.args.out;
        console.log(" * Msg =", msg);
        console.log(" * Out =", out.valueOf());
    }
});
```

- Ejecución del ejemplo 4-observar.js

```
$ node 4-observar.js
Se ha producido un evento Tic:
 * Msg = Actualizado
 * Out = 3
Se ha producido un evento Tic:
 * Msg = Actualizado
 * Out = 4
Se ha producido un evento Tic:
 * Msg = Actualizado
 * Out = 5
Se ha producido un evento Tic:
 * Msg = Actualizado
 * Out = 6
Se ha producido un evento Tic:
 * Msg = Actualizado
 * Out = 7
```

```
var Web3 = require("web3");  // Cargar paquete web3

if (typeof web3 !== 'undefined') {  // Ya existe instancia web3
   web3 = new Web3(web3.currentProvider);  // Usamos el mismo Provider
} else {  // Usar Provider: GANACHE
   web3 = new Web3(new Web3.providers.HttpProvider("http://localhost:
7545"));
}

var abi = [{"constant":false,"inputs":[],"name":"incr","outputs":
[],"payable":false,"stateMutability":"nonpayable","type":"function"},
{"constant":true,"inputs":[],"name":"valor","outputs":
[{"name":"","type":"uint8"}],"payable":false,"stateMutability":"view",
"type":"function"},
{"payable":false,"stateMutability":"nonpayable","type":"fallback"},
{"anonymous":false,"inputs":
[{"indexed":false,"name":"msg","type":"string"},
{"indexed":false,"name":"out","type":"uint8"}],"name":"Tic","type":"ev
ent"}];

var addr = "0xb25be7ca2686c860249dcc98beb828e82da18ce0";

var MyContract = web3.eth.contract(abi);

var contractInstance = MyContract.at(addr);

contractInstance.Tic(function(err, data) {
    console.log("Se ha producido un evento Tic:");
    if (err){
        console.log(err);
    } else {
        var msg = data.args.msg;
        var out = data.args.out;
        console.log(" * Msg =", msg);
        console.log(" * Out =", out.valueOf());
    }
});
```

4-observar.js

Ejemplos: Cliente Web

Clientes Web sin Blockchain

- Ejemplo: 1.1-SinContrato/index.html
  - Una página html con css y javascript embebido.
  - Muestra el valor de un contador y un botón para incrementarlo.
  - No hay nada de Blockchain.
- Ejemplo: 1.2-SinContrato/*
  - El ejemplo 1.1 separando el css y el js en ficheros independientes, style.css y app.js.

```
<!DOCTYPE html>
<html lang="es">
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <style>
       #valor {
          font-size: x-large;
          color: red;
       }
    </style>
    <title>Contador 1.1</title>
  </head>
  <body>
    <h1>Contador 1.1</h1>

    <p>
      Valor actual = <span id="valor">???</span>
    </p>

    <button type="button" id="cincr">Incrementar</button>

    <script src="https://ajax.googleapis.com/ajax/libs/jquery/2.2.4/jquery.min.js"></script>

    <script>
      var App = {

        valor: 0,

        init: function() {
          App.bindEvents();
          App.refreshContador();
        },

        bindEvents: function() {
           $(document).on('click', '#cincr', App.handleIncr);
        },

        handleIncr: function(event) {
          event.preventDefault();
          App.valor++;
          App.refreshContador();
        },

        refreshContador: function() {
          $('#valor').text(App.valor);
        }
      };

      // Ejecutar cuando se ha terminado de cargar la pagina.
      $(function() {
         $(window).load(function() {
           App.init();
         });
      });

    </script>

  </body>
</html>
```

1.1-SinContrato/index.html

```html
<!DOCTYPE html>
<html lang="es">
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet" href="style.css">
    <title>Contador 1.2</title>
  </head>
  <body>
    <h1>Contador 1.2</h1>

    <p>
      Valor actual = <span id="valor">???</span>
    </p>

    <button type="button" id="cincr">Incrementar</button>

    <script src="https://ajax.googleapis.com/ajax/libs/jquery/2.2.4/jquery.min.js"></script>

    <script src="app.js"></script>

  </body>
</html>
```

1.2-SinContrato/index.html

```css
#valor {
      font-size: x-large;
      color: red;
}
```

1.2-SinContrato/style.css

```
var App = {

  valor: 0,

  init: function() {
    App.bindEvents();
    App.refreshContador();
  },

  bindEvents: function() {
     $(document).on('click', '#cincr', App.handleIncr);
  },

  handleIncr: function(event) {
    event.preventDefault();
    App.valor++;
    App.refreshContador();
  },

  refreshContador: function() {
    $('#valor').text(App.valor);
  }
};


// Ejecutar cuando se ha terminado de cargar la pagina.
$(function() {
   $(window).load(function() {
     App.init();
   });
});
```

1.2-SinContrato/app.js

Crear Servidor Web con Nodejs

- Usar **Nodejs** para crear un servidor web de páginas estáticas que sirva los ejemplos que estamos haciendo.
- Ejecutar:

```
$ sudo npm install -g express-generator
$ express --ejs 1.3-node_server
$ cd 1.3-node_server
$ npm install
```

- Añadir en **node_server/app.js** la línea:

```
app.use(express.static(path.join(__dirname,
                                 '../1.2-SinContrato')));
```

  - Después de la línea app.use(express.static(path.join(__dirname, 'public')));
  - También se pueden copiar los ficheros a servir en el directorio public.
- Lanzar el servidor ejecutando:

```
$ npm start
```

- Lanzar un navegador y visitar la página **http://localhost:3000**

Cliente Web con Web3

- Ejemplo: 1.4-ConContrato.js
  - Se ha añadido uso del contrato Contador.
  - Se crea el objeto web3, se instancia el contrato ya desplegado anteriormente, se observan eventos, ...
  - Debería modificarse el servidor web 1.2 para que sirva las pagina estáticas de este ejemplo.
  - No se ha metido un tratamiento de errores exhaustivo.
    - Añadir callbacks para detectarlos, ...

```
App = {
    Contador: null,
    contador: null,

    init: function() {
        console.log("Inicializando App.");

        App.initWeb3();
        App.initContractAbstracts();
        App.initContractInstance();
        App.bindEvents();
        App.refreshContador();
    },

    initWeb3: function() {
        console.log("Inicializando web3.");

        // Si hay inyectada una instancia de web3:
        if (typeof web3 !== 'undefined') {
            web3 = new Web3(web3.currentProvider);
        } else {
            // Uso Ganache porque no hay una instancia de web3 inyectada.
            web3 = new Web3(new Web3.providers.HttpProvider('http://localhost:7545'));
        }
    },
```

1.4-ConContrato/app.js (1/4)

```
    initContractAbstracts: function() {
        console.log("Inicializando abstracción del contrato.");

        var abi = [{"constant":false,"inputs":[],"name":"incr","outputs":[],
            "payable":false,"stateMutability":"nonpayable","type":"function"},
            {"constant":true,"inputs":[],"name":"valor","outputs":[{"name":""
            ,"type":"uint8"}],"payable":false,"stateMutability":"view","type":"function"},
            {"payable":false,"stateMutability":"nonpayable","type":"fallback"},
            {"anonymous":false,"inputs":[{"indexed":false,"name":"msg","type":"string"},
            {"indexed":false,"name":"out","type":"uint8"}],"name":"Tic","type":"event"}];

        App.Contador = web3.eth.contract(abi);
    },

    initContractInstance: function() {
        console.log("Obtener instancia desplegada del contador.");

        var addr = "0x9d9d2b8e4e067a8522caf47af41c67843a8f17c9";

        App.contador = App.Contador.at(addr);

        console.log("Configurar Vigilancia de los eventos del contador.");
        App.contador.Tic((err, data) => {
            console.log("Se ha producido un evento Tic:");
            if (err){
                console.log(err);
            } else {
                var msg = data.args.msg;
                var out = data.args.out;
                console.log(" * Msg =", msg);
                console.log(" * Out =", out.valueOf());

                $('#valor').text(out.valueOf());
            }
        });
    },
```

1.4-ConContrato/app.js (2/4)

```
    bindEvents: function() {
        console.log("Configurando manejador de eventos del boton.");

        $(document).on('click', '#cincr', App.handleIncr);
    },

    handleIncr: function(event) {
        console.log("Se ha hecho Click en el botón.");

        event.preventDefault();

        web3.eth.getAccounts(function(error, accounts) {
            if (error) {
                console.log(error);
                return;
            }

            const account = accounts[0];

            console.log("Cuenta =", account);

            // Ejecutar incr como una transacción desde la cuenta account.
            App.contador.incr.sendTransaction({
                from: account,
                gas: 200000});
        });
    },
```

1.4-ConContrato/app.js (3/4)

```
    refreshContador: function() {
        console.log("Refrescando el valor mostrado del contador.");

        var valor = App.contador.valor.call()

        console.log("Valor =", valor);

        $('#valor').text(valor ? valor.valueOf() : "XXX");
    }

};

// Ejecutar cuando se ha terminado de cargar la pagina.
$(function() {
  $(window).load(function() {
    App.init();
  });
});
```

1.4-ConContrato/app.js (4/4)