Desarrollo de dapps en Ethereum

Para este laboratorio necesitaremos las siguientes herramientas:

- Atom (O cualquier editor de texto)
  https://atom.io/

- Ganache (TestRPC): Esta herramienta simula una blockchain y ofrece un API RPC de Ethereum
  http://truffleframework.com/ganache/
  https://github.com/trufflesuite/ganache-cli

- Nodejs y npm: Uno de los principales entornos de desarrollo para aplicaciones en Javascript y su gestor de paquetes
  https://nodejs.org/es/
  https://www.npmjs.com/

- Truffle: El entorno de desarrollo más sencillo para desarrollo de aplicaciones en Ethereum
  http://truffleframework.com/

- Remix: Herramienta on-line para el desarrollo de smart contracts que ofrece interesantes consejos y alertas para debugging
  https://remix.ethereum.org

- Documentación de solidity
  http://solidity.readthedocs.io/en/v0.4.21/

- Documentación de librería web3
  0.2x.x https://github.com/ethereum/wiki/wiki/JavaScript-API
  1.0 http://web3js.readthedocs.io/en/1.0/index.html

- Web de Infura (consensys): Este servicio permite conectarse a las redes de ethereum sin necesidad de tener el cliente corriendo
  https://infura.io/

- Metamask: Este plugin para navegador permite acceder a servicios blockchain de manera transparente:
  https://metamask.io/

El laboratorio se desarrollará en 6 pasos:

1. Instalación de las herramientas necesarias
2. Creación de Smart contract básicos
3. Creación de aplicación javascript para interactuar con el smart contract
4. Desarrollo de test básico y explicación de su utilidad
5. Despliegue de la aplicación en local
6. Despliegue de la aplicación en Ropsten (Truffle y Remix)

Resumen de instrucciones del laboratorio (para ubuntu):

1. Instalación de las herramientas necesarias

Partiremos del punto de que se tiene la maquina virtual con el editor de texto favorito de cada uno instalado.

a. Primero instalamos Node.js y npm mediante NVM

```
$ curl -sL https://deb.nodesource.com/setup_6.x | sudo -E bash -
sudo apt-get install nodejs
```

Probamos que todo está correcto

```
$ node -v
```

Comprobamos que npm también se ha instalado

```
$ Npm -v
```

b. Instalamos Ganache

```
$ sudo npm install -g ganache-cli
```

c. Instalamos Truffle azt

```
$ npm install -g truffle
```

2. Contrato básico

```
pragma solidity ^0.4.21;

contract myFirstContract {
     //Declaracion de variables
     address public owner;
     uint256 private totalSupply;
     mapping (address=>uint256) internal balanceOf;

     event Transfered(address from, address indexed to,uint256
amount);

     //Constructor del contrato
     constructor (uint256 _totalSupply) {
     owner = msg.sender;
     balanceOf[msg.sender] = _totalSupply;
     totalSupply = _totalSupply;
     }

     //Modificadores
     modifier onlyOwner() {
     if(msg.sender != owner){
          throw;
     }
     _;
     }

     //funciones del contrato
     function send(address _to, uint256 amount) public {
     require(balanceOf[msg.sender] >= amount);
     require(balanceOf[_to] + amount >= balanceOf[_to]);

     balanceOf[msg.sender] -= amount;
     balanceOf[_to] += amount;

     emit Transfered(msg.sender, _to, amount);
     }

     function getBalance(address _user) view public onlyOwner
returns(uint256) {
     return balanceOf[_user];
     }

}
```

3. Configuración de Truffle.js para Ropsten

```
$ npm install truffle-hdwallet-provider --save
```

Se modifica el fichero truffle.js

```
var HDWalletProvider = require("truffle-hdwallet-provider");

var infura_apikey = "XXXXXX";
var mnemonic = "twelve words you can find in
metamask/settings/reveal seed words blabla";

module.exports = {
  networks: {
    development: {
      host: "localhost",
      port: 8545,
      network_id: "*" // Match any network id
    },
    ropsten: {
      provider: new HDWalletProvider(mnemonic,
"https://ropsten.infura.io/"+infura_apikey),
      network_id: 3
    }
  }
};
```

Se ejecuta el comando

```
$ truffle migrate --network ropsten
```

Recordar que se necesita cuenta en INFURA para generar el apikey y metamask para simplificar el mnemonico.

4. Aplicación Javascript

Perdonadme que en el desarrollo del meetup no funcionase la aplicación para conectarse con web3, al final descubrí el fallo, se había instalado la version 1.0 de web3 en lugar de la 0.2 que es la que tenía preparada y por eso no conectó. Os dejo

todo el código de ejemplo subido en mi github:

https://github.com/nefera606/BlockchainForDevelopers

5. Oráculos

Como primera aproximación a los oráculos, ya que muchos lo preguntasteis, os dejo esta guia breve creada por Alex Beregszaszi de como montar un oráculo muy sencillo.

https://github.com/axic/tinyoracle