



FAUCET

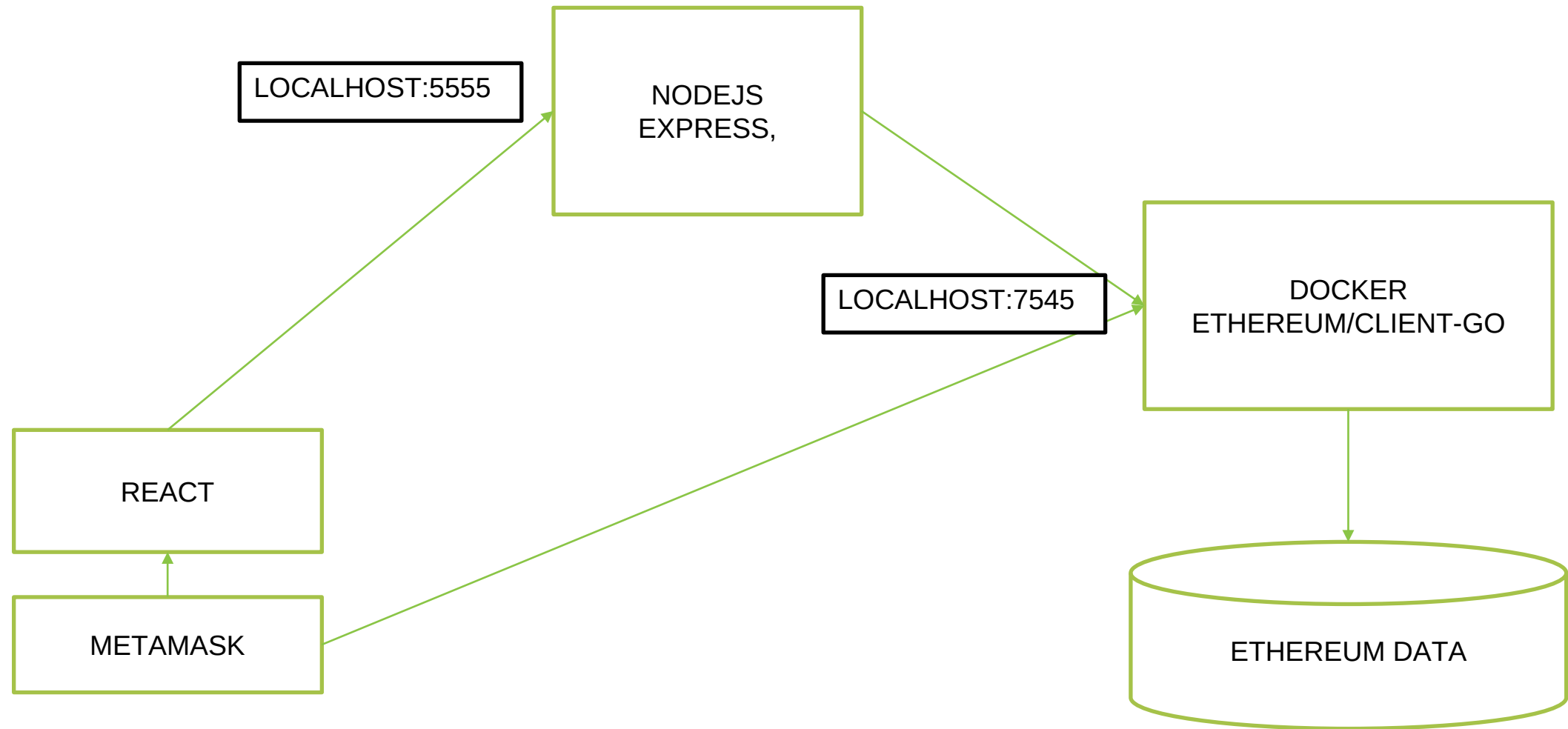
Proyectos fin de introducción a la programación

OBJETIVOS

FAUCET

1. Se trata de hacer un faucet de una red privada de Ethereum
 1. Tendremos que instalar un nodo Ethereum en local con el networkid 222333444. Usaremos Docker para hacerlo
 2. Configuraremos esta red en el metamask
 3. Haremos una aplicación REACT que permita conectar la cuenta donde queramos nos ingresen los ETH. Es un solo componente REACT
 4. Haremos un servidor web nodejs que firme la transacción de envío de x eth desde la cuenta del faucet a la cuenta conectada.

ARQUITECTURA



DOCKER

```
{
  "config": {
    "chainId": 8888,
    "homesteadBlock": 0,
    "eip150Block": 0,
    "eip155Block": 0,
    "eip158Block": 0,
    "byzantiumBlock": 0,
    "constantinopleBlock": 0,
    "petersburgBlock": 0,
    "ethash": {}
  },
  "difficulty": "1",
  "gasLimit": "12000000",
  "alloc": {
    "0xff21E724B7D483fc93708855AbE6ee4f1eD97BF3": {
      "balance": "10000000000000000000000000000000"
    }
  }
}
```

genesis.json

Create a account

```
docker run --rm -it -v ${PWD}/data/keystore:/data  
ethereum/client-go account new --keystore /data
```

--rm el contenedor desaparece después de ejecutarse la orden

-v\${PWD}/data:/data (mapeo del directorio del host al directorio data del container)

-it sesión interactiva. Nos va a pedir el comando una palabra de paso

Ethereum/client-go image del cliente Ethereum

account comando que se ejecuta y que sirve para la gestión de cuentas

new parámetro del comando

--keystore /data -> directorio del container que contendrá la cuenta. Esta mapeada al \${PWD}/data

```
{
  "address": "405ed0af3ff15f9315b26e4db1e60994b81bde99",
  "crypto": {
    "cipher": "aes-128-ctr",
    "ciphertext": "fb07591477f655874fad00b27a5fcf4a44378ff42bf8a5c9206620e875f7a6a6",
    "cipherparams": {
      "iv": "0b5f26975d8df78e898285a60e387c07"
    },
    "kdf": "scrypt",
    "kdfparams": {
      "dklen": 32,
      "n": 262144,
      "p": 1,
      "r": 8,
      "salt": "0da594f49270350599c50a9780497411de2a10cf397b10a04c248dad8daa1465"
    },
    "mac": "f50541d5a34d7e660e1c077faf67344f4997cd3ac6aaedbb99c5cb20920ecd57"
  },
  "id": "bc70517f-2177-4a0e-b38e-d04aad305fe2",
  "version": 3
}
```

Fichero json que nos ha creado el comando anterior. Contiene los datos de la clave publica y privada

```
const ethers = require("ethers")
const fs = require("fs")

const datos = fs.readFileSync("./data/UTC--2022-12-19T09-43-19.618801095Z--405ed0af3ff15f9315b26e4db1e60994b81bde99").toString("utf-8")

const wallet = ethers.Wallet.fromEncryptedJsonSync(datos, "1234")

console.log('privada', wallet.privateKey)
console.log('address', wallet.address)
console.log('public', wallet.publicKey)
```

Librería para leer el wallet

Fichero creado con el wallet encriptado

Orden para obtener el wallet

A partir del wallet puedo obtener las claves y el address

```
privada 0x6b4e09e8d9e2b45c86efdd002291f969ddd bfd4650714db656357f862c39011f
address 0x405Ed0Af3Ff15f9315B26E4db1e60994b81bDe99
public 0x0459cc416a6ec27bd4c9b6bb26f6eb95b88e57fd8dd2e81b503d24b22809
b0f982cbc50c173591a18b94e8a7bf5666cf6ffeafaedc71912b692b36362aeccd0431
```

```
{
  "config": {
    "chainId": 8888,
    "homesteadBlock": 0,
    "eip150Block": 0,
    "eip155Block": 0,
    "eip158Block": 0,
    "byzantiumBlock": 0,
    "constantinopleBlock": 0,
    "petersburgBlock": 0,
    "ethash": {}
  },
  "difficulty": "1",
  "gasLimit": "12000000",
  "alloc": {
    "0xff21E724B7D483fc93708855AbE6ee4f1eD97BF3": {
      "balance": "100000000000000000000000000000000"
    }
  }
}
```

El address del wallet que hemos creado



```
docker run -d \  
  -v ${PWD}/data:/data \  
  -v ${PWD}/genesis.json:/genesis.json \  
  --name eth-node \  
  ethereum/client-go \  
  init --datadir data /genesis.json
```

The diagram consists of four green rectangular boxes on the right side of the slide, each containing white text. Four green arrows originate from these boxes and point to specific parts of the Docker command in the code block on the left. The first arrow points from 'Mapeo de datos' to the volume mapping '-v \${PWD}/data:/data'. The second arrow points from 'Mapeo de genesis.json' to the volume mapping '-v \${PWD}/genesis.json:/genesis.json'. The third arrow points from 'Nombre contenedor' to the '--name eth-node' flag. The fourth arrow points from 'Comando init para inicializar el nodo' to the 'init' command at the end of the command line.

Mapeo de datos

Mapeo de genesis.json

Nombre contenedor

Comando init para
inicializar el nodo

```
docker run -d -p 8545:8545 -p:30303:30303 \  
-v ${PWD}/data:/data \  
--name eth-node-01 \  
    ethereum/client-go \  
--datadir data \  
--http.api personal,eth,net,web3 \  
--http --http.addr 0.0.0.0 \  
--http.port 8545 \  
--mine --miner.etherbase 0xff21E724B7D483fc93708855AbE6ee4f1eD97BF3 \  
--miner.threads=1
```

Lanzamos la red

Con `docker logs eth-node-01` vemos los logs del container

Si lo lanzamos sin `-d` la salida la vemos por

back

Conectarse a un provider

```
const Web3 = require("web3")  
const fs = require("fs")  
  
const web3 = new Web3("http://localhost:8545")
```

Obtener el balance de una cuenta

```
web3.eth.getBalance("0xda682C875E6C792D9a7086627d5EC618E7C5E805")  
  .then(saldo => { console.log(saldo) })
```

Obtener un Wallet encriptado

```
const json = JSON.parse(fs.readFileSync("../nodo/keystore/UTC--2022-12-19T18-46-24.804058039Z--da682c875e6c792d9a7086627d5ec618e7c5e805").toString("utf-8"))

const account = web3.eth.accounts.decrypt(json, "1234");
```

Firmar una trasaction con un wallet y enviarla al provider

```
const tx = {
  to: "0xff21E724B7D483fc93708855AbE6ee4f1eD97BF3",
  from: account.address,
  gas: 30000,
  value: web3.utils.toWei("0.033", 'ether')
}

account.signTransaction(tx).then(tx1 => {
  console.log(tx1.rawTransaction)
  web3.eth.sendSignedTransaction(tx1.rawTransaction).then(i => {
    console.log(i)
  })
})
```

Firmar y enviar una transacción a un provider

```
app.get("/balance/:cuenta", async (req, res) => {  
  const balance = await web3.eth.getBalance(req.params.cuenta)  
  res.send({ balance: balance })  
});
```

repositorios

Los repos del video

[curso-faucet-6](#)
[curso-faucet-server-6](#)

Preparación

[curso-faucet-server](#)
[curso-faucet](#)