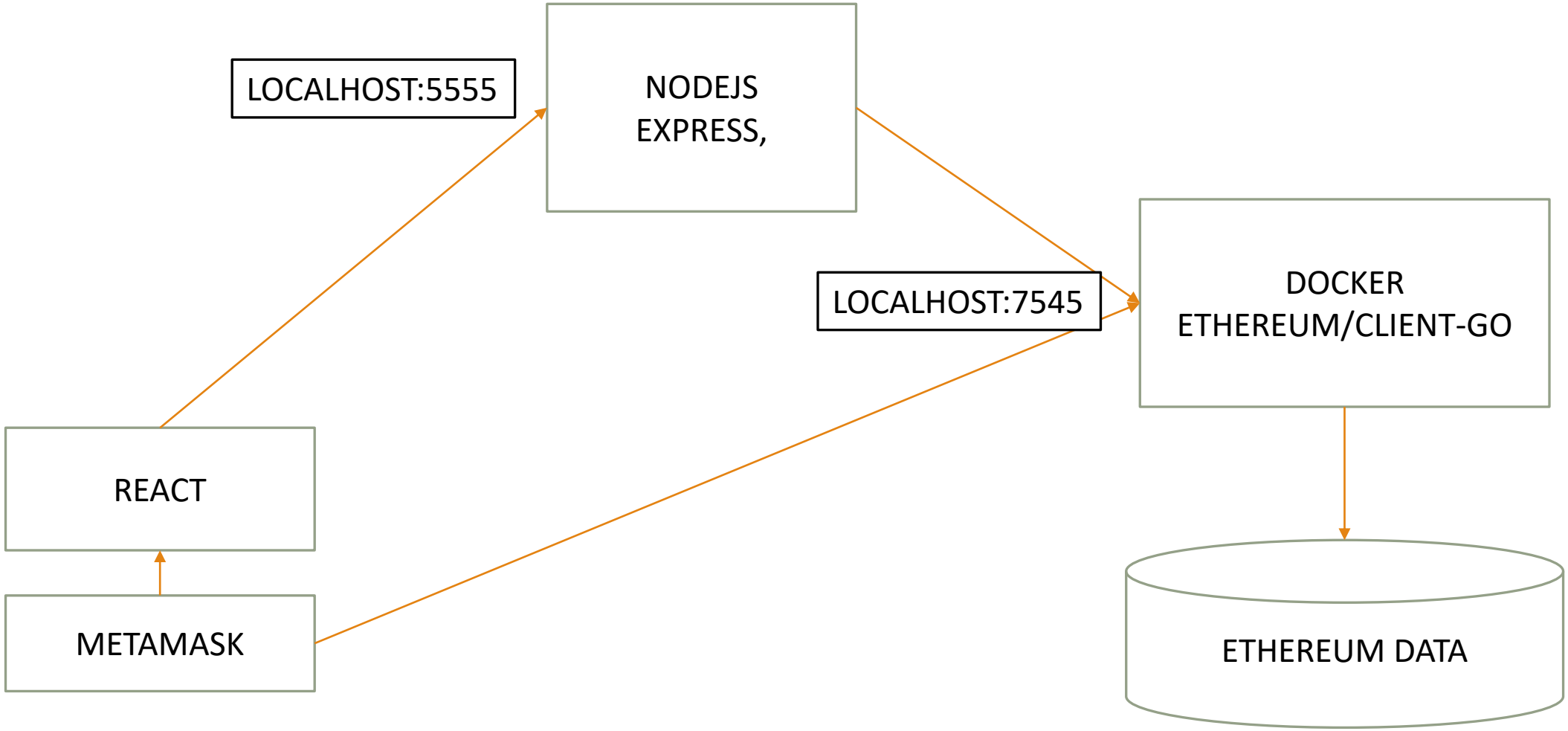
FAUCET

PROYECTOS FIN DE INTRODUCCIÓN A LA PROGRAMACIÓN

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.


LOCALHOST:5555
NODEJS
EXPRESS,
LOCALHOST:7545
DOCKER
ETHEREUM/CLIENT-GO
REACT
METAMASK
ETHEREUM DATA

repositorios

Los repos del video

curso-faucet-6
curso-faucet-server-6

Preparación

curso-faucet-server
curso-faucet

```
{
    "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": "100000000000000000000000000"
        }
      }
    }
```

genesis.json

PRIVATE_KEY=0x0813457ea9dc2968d1d2b03e6eef34ebe3ab189b69ff639e7371f0029d57a778
ADDRESS=0xff21E724B7D483fc93708855AbE6ee4f1eD97BF3

Private key y address

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

Mapeo de datos

Mapeo de genesis.json

Nombre container

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 console

Librería web3 para acceso a Ethereum
https://web3js.readthedocs.io/en/v1.7.1/getting-started.html

Uso del metamask en el navegador
https://docs.metamask.io/guide/ethereum-provider.html#methods

Express como servidor web
https://expressjs.com/

Cors middleware
https://expressjs.com/en/resources/middleware/cors.html

Dotenv
https://www.npmjs.com/package/dotenv

```
                                    1 ETHER = 1,000,000,000,000,000,000 WEI = 1 (EXA)WEI
                1 (MILLI)ETHER = 0.001 ETHER = 1,000,000,000,000,000 WEI = 1 (PETA)WEI
             1 (MICRO)ETHER = 0.000001 ETHER = 1,000,000,000,000 WEI = 1 (TERA)WEI
          1 (NANO)ETHER = 0.000000001 ETHER = 1,000,000,000 WEI = 1 (GIGA)WEI
       1 (PICO)ETHER = 0.000000000001 ETHER = 1,000,000 WEI = 1 (MEGA)WEI
  1 (FEMTO)ETHER = 0.000000000000001 ETHER = 1,000 WEI = 1 (KILO)WEI
1 (ATTO)ETHER = 0.000000000000000001 ETHER = 1 WEI
```

Firmar y enviar una transacción a un provider

```
const Web3 = require("web3")

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

const tx = await web3.eth.accounts.signTransaction({
        to: req.params.cuenta,
        from: process.env.ADDRESS,
        value: 10E18,
        gas: 2000000
    }, process.env.PRIVATE_KEY)
    // enviar la tx al provider
    const txSended = await web3.eth.sendSignedTransaction(
        tx.rawTransaction
    )
```

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 })
});
```