Minicurso Introdução às Tecnologias Blockchain: Práticas

Prof. Rogério Aparecido Gonçalves *Universidade Tecnológica Federal do Paraná (UTFPR)*

Blockchain é uma tecnologia nova e considerada revolucionária e disruptiva, sendo até mesmo comparada, quanto ao impacto, ao surgimento da Internet. Neste minicurso serão apresentados conceitos e alguns fundamentos básicos relacionadas à Tecnologia Blockchain. Neste material complementar são apresentadas a práticas relacionadas ao desenvolvimento com *Ethereum*.

Sumário

1 Prática: Instalando o Cliente *Ethereum*: `Geth`

A proposta desta prática é vermos o funcionamento do *software* cliente da rede *Ethereum*, o `Geth`, transformando a máquina em um nó da rede *Ethereum*.

1.1 Instalando o `Geth`

Nesta etapa, você instalará um cliente padrão de camada de execução (`geth`). Baixe e execute o a última versão `64-bit` estável do **Geth installer** para seu Sistema Operacional do site *Geth downloads page*.

O `geth` pode ser instalado em sistemas derivados do `Debian` e `Ubuntu` com o pacote `ethereum`:

```
$ sudo apt-get install -y software-properties-common
$ sudo add-apt-repository -y ppa:ethereum/ethereum
$ sudo apt-get update
$ sudo apt-get install -y ethereum
```

Em outros Sistemas como o `Manjaro`:

```
[rag@nitro-ryzen ~]$ sudo pacaur -Ss ethereum
community/go-ethereum 1.10.25-1 [instalado]
    Official Go implementation of the Ethereum protocol
[rag@nitro-ryzen ~]$ sudo pacaur -S go-ethereum
[rag@nitro-ryzen ~]$ pacaur -S go-ethereum
resolvendo dependencias...
procurando pacotes conflitantes...

Pacotes (1) go-ethereum-1.10.25-1

Tamanho total instalado: 197,38 MiB
Alteração no tamanho: 0,00 MiB

:: Continuar a instalação? [S/n]
```

Instruções para outros Sistemas Operacionais podem ser encontradas no site oficial da documentação do `Ethereum`, artigo *Installing Geth*.

Verifique se a versão mais nova já não foi instalada. O link v1.11.5 lista as versões.

Na minha máquina está intalada a versão `1.11.5-stable-a38f4108`, você pode verificar a sua com `geth --version` ou `geth version`:

```
[rogerio@ryzen-nitro execution]$ geth --version
geth version 1.11.5-stable-a38f4108
[rogerio@ryzen-nitro execution]$ geth version
Geth
Version: 1.11.5-stable
Git Commit: a38f4108571d1a144dc3cf3faf8990430d109bc4
Git Commit Date: 20230321
Architecture: amd64
Go Version: go1.20.2
Operating System: linux
GOPATH=
GOROOT=
[rogerio@ryzen-nitro execution]$
```

1.2 *Executando o `Geth`*

Executando o `Geth` diretamente ele irá sincronizar com a rede principal do `Ethereum`, a `mainnet`. Será criado o diretório `~/.ethereum` com os dados da rede.

```
[rag@nitro-ryzen ~]$ geth
INFO [10-20|21:07:12.911] Starting Geth on Ethereum mainnet...
INFO [10-20|21:07:12.912] Bumping default cache on mainnet provided=1024 updated=4096
INFO [10-20|21:07:12.914] Maximum peer count ETH=50 LES=0 total=50
INFO [10-20|21:07:12.915] Smartcard socket not found, disabling err="stat
    /run/pcscd/pcscd.comm: no such file or directory"
INFO [10-20|21:07:12.920] Set global gas cap cap=50,000,000
INFO [10-20|21:07:12.922] Allocated trie memory caches clean=614.00MiB dirty=1024.00MiB
INFO [10-20|21:07:12.923] Allocated cache and file handles
    database=/home/rag/.ethereum/geth/chaindata cache=2.00GiB handles=262,144
INFO [10-20|21:07:12.946] Opened ancient database
    database=/home/rag/.ethereum/geth/chaindata/ancient/chain readonly=false
INFO [10-20|21:07:12.950]
INFO [10-20|21:07:12.950]
    ---------------------------------------------------------------------------
INFO [10-20|21:07:12.950] Chain ID: 1 (mainnet)
INFO [10-20|21:07:12.950] Consensus: Beacon (proof-of-stake), merged from Ethash
    (proof-of-work)
INFO [10-20|21:07:12.950]
INFO [10-20|21:07:12.950] Pre-Merge hard forks:
INFO [10-20|21:07:12.950] - Homestead: 1150000
    (https://github.com/ethereum/execution-specs/blob/master/network-upgrades/mainnet-upgrades/homest
INFO [10-20|21:07:12.950] - DAO Fork: 1920000
    (https://github.com/ethereum/execution-specs/blob/master/network-upgrades/mainnet-upgrades/dao-fo
INFO [10-20|21:07:12.950] - Tangerine Whistle (EIP 150): 2463000
    (https://github.com/ethereum/execution-specs/blob/master/network-upgrades/mainnet-upgrades/tanger
INFO [10-20|21:07:12.950] - Spurious Dragon/1 (EIP 155): 2675000
    (https://github.com/ethereum/execution-specs/blob/master/network-upgrades/mainnet-upgrades/spuric
INFO [10-20|21:07:12.950] - Spurious Dragon/2 (EIP 158): 2675000
    (https://github.com/ethereum/execution-specs/blob/master/network-upgrades/mainnet-upgrades/spuric
INFO [10-20|21:07:12.950] - Byzantium: 4370000
    (https://github.com/ethereum/execution-specs/blob/master/network-upgrades/mainnet-upgrades/byzant
INFO [10-20|21:07:12.950] - Constantinople: 7280000
    (https://github.com/ethereum/execution-specs/blob/master/network-upgrades/mainnet-upgrades/consta
INFO [10-20|21:07:12.950] - Petersburg: 7280000
    (https://github.com/ethereum/execution-specs/blob/master/network-upgrades/mainnet-upgrades/peters
INFO [10-20|21:07:12.950] - Istanbul: 9069000
    (https://github.com/ethereum/execution-specs/blob/master/network-upgrades/mainnet-upgrades/istanb
INFO [10-20|21:07:12.950] - Muir Glacier: 9200000
    (https://github.com/ethereum/execution-specs/blob/master/network-upgrades/mainnet-upgrades/muir-g
INFO [10-20|21:07:12.950] - Berlin: 12244000
    (https://github.com/ethereum/execution-specs/blob/master/network-upgrades/mainnet-upgrades/berlin
INFO [10-20|21:07:12.950] - London: 12965000
    (https://github.com/ethereum/execution-specs/blob/master/network-upgrades/mainnet-upgrades/londor
INFO [10-20|21:07:12.950] - Arrow Glacier: 13773000
    (https://github.com/ethereum/execution-specs/blob/master/network-upgrades/mainnet-upgrades/arrow-
INFO [10-20|21:07:12.950] - Gray Glacier: 15050000
    (https://github.com/ethereum/execution-specs/blob/master/network-upgrades/mainnet-upgrades/gray-g
```

```
INFO [10-20|21:07:12.950]
INFO [10-20|21:07:12.950] Merge configured:
INFO [10-20|21:07:12.950] - Hard-fork specification:
    https://github.com/ethereum/execution-specs/blob/master/network-upgrades/mainnet-upgrades/paris.m
INFO [10-20|21:07:12.950] - Network known to be merged: true
INFO [10-20|21:07:12.950] - Total terminal difficulty: 58750000000000000000
INFO [10-20|21:07:12.950] - Merge netsplit block: <nil>
INFO [10-20|21:07:12.950]
    ------------------------------------------------------------------------------------
INFO [10-20|21:07:12.950]
INFO [10-20|21:07:12.952] Disk storage enabled for ethash caches
    dir=/home/rag/.ethereum/geth/ethash count=3
INFO [10-20|21:07:12.952] Disk storage enabled for ethash DAGs dir=/home/rag/.ethash
    count=2
INFO [10-20|21:07:12.952] Initialising Ethereum protocol network=1 dbversion=8
INFO [10-20|21:07:12.963] Loaded most recent local header number=0 hash=d4e567..cb8fa3
    td=17,179,869,184 age=53y6mo3w
INFO [10-20|21:07:12.963] Loaded most recent local full block number=0
    hash=d4e567..cb8fa3 td=17,179,869,184 age=53y6mo3w
INFO [10-20|21:07:12.963] Loaded most recent local fast block number=0
    hash=d4e567..cb8fa3 td=17,179,869,184 age=53y6mo3w
INFO [10-20|21:07:12.964] Loaded local transaction journal transactions=0 dropped=0
INFO [10-20|21:07:12.964] Regenerated local transaction journal transactions=0
    accounts=0
INFO [10-20|21:07:12.965] Chain post-merge, sync via beacon client
INFO [10-20|21:07:12.965] Gasprice oracle is ignoring threshold set threshold=2
WARN [10-20|21:07:12.965] Engine API enabled protocol=eth
INFO [10-20|21:07:12.966] Starting peer-to-peer node
    instance=Geth/v1.10.25-stable-69568c55/linux-amd64/go1.19.1
INFO [10-20|21:07:12.991] New local node record seq=1,665,519,113,919
    id=da440578e33a2ce7 ip=127.0.0.1 udp=30303 tcp=30303
INFO [10-20|21:07:12.992] Started P2P networking
    self=enode://9ae8fcdad4a7243d1bd2308a159c5800ec170e588862be110152627c9ed3fa67376ef8c7526d7a56e9bb
INFO [10-20|21:07:12.993] IPC endpoint opened url=/home/rag/.ethereum/geth.ipc
INFO [10-20|21:07:12.993] Loaded JWT secret file
    path=/home/rag/.ethereum/geth/jwtsecret crc32=0xdeccafe4
INFO [10-20|21:07:12.994] WebSocket enabled url=ws://127.0.0.1:8551
INFO [10-20|21:07:12.994] HTTP server started endpoint=127.0.0.1:8551 auth=true
    prefix= cors=localhost vhosts=localhost
INFO [10-20|21:07:16.251] New local node record seq=1,665,519,113,920
    id=da440578e33a2ce7 ip=187.95.110.26 udp=2770 tcp=30303
INFO [10-20|21:07:22.992] Looking for peers peercount=0 tried=2 static=0
INFO [10-20|21:07:32.994] Looking for peers peercount=0 tried=3 static=0
INFO [10-20|21:07:43.205] Looking for peers peercount=0 tried=9 static=0
WARN [10-20|21:07:47.967] Post-merge network, but no beacon client seen. Please launch
    one to follow the chain!
INFO [10-20|21:07:53.281] Looking for peers peercount=0 tried=13 static=0
INFO [10-20|21:08:03.346] Looking for peers peercount=0 tried=9 static=0
```

O que vai levar um certo tempo para fazer toda a sincronização da rede completa.

1.3 Redes de Teste

Vimos as redes *Ethereum* disponíveis, uma lista completa pode ser encontrada em https://ethereum.org/en/developers/docs/networks/.

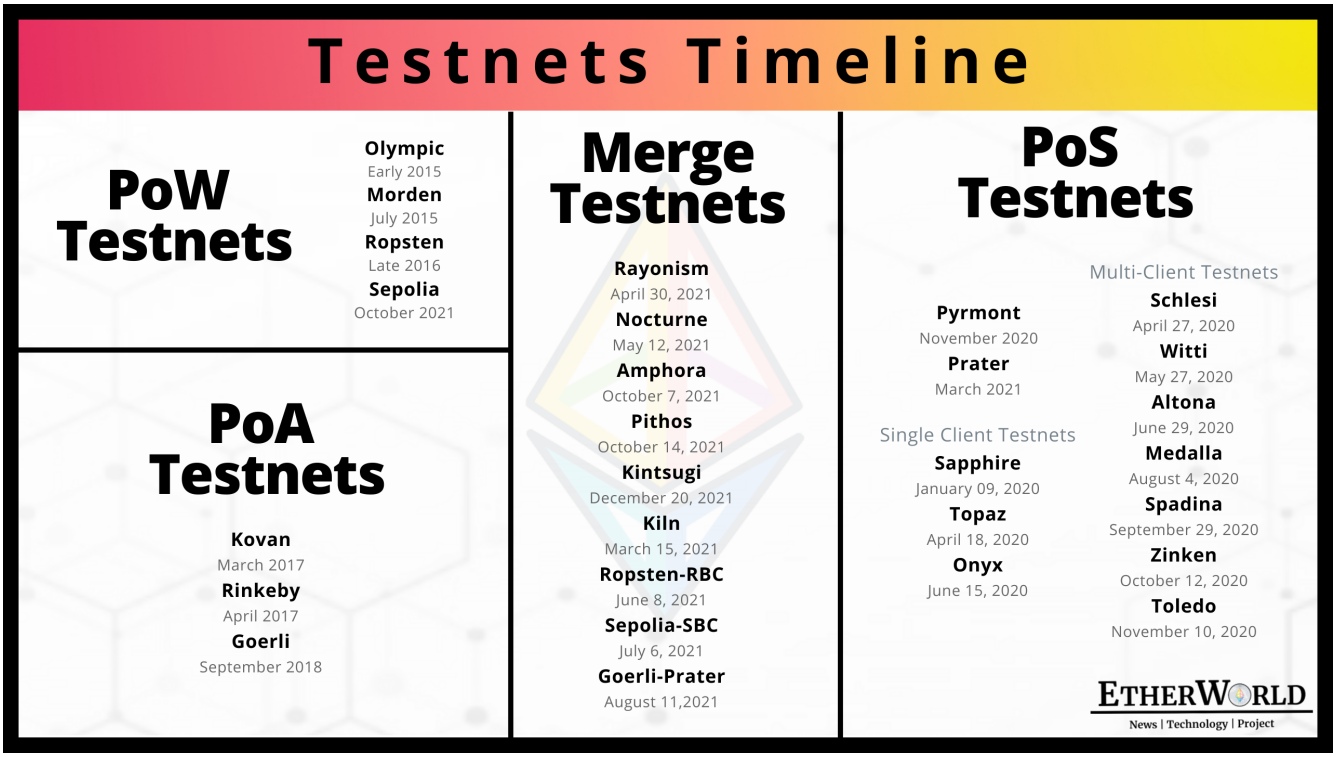


Figura 1: *Timeline* das Redes de Teste Fonte: Etherwold

A evolução das Redes de Teste pode ser vista nesse artigo *The Evolution of Ethereum Testnets*.

Atualizar as redes de teste.

Rede	Id	Parâmetro	Descrição
mainnet	1	`--mainnet`	Rede principal
testnet		`--testnet`	O livro indica que para as redes de teste deve ser usado o parâmetro `--testnet` para acessar a rede `ropsten` por padrão ou fornecer o nome da rede, como `--testnet rinkeby`. Na versão atual os parâmetros são os seguintes.
ropsten		`--ropsten`	Ropsten network: pre-configured Proof of Work test network
rinkeby		`--rinkeby`	Rinkeby network: pre-configured Proof of Authority test network
goerli		`--goerli`	Görli network: pre-configured Proof of Authority test network
kiln		`--kiln`	Kiln network: pre-configured proof-of-work to proof-of-stake test network
sepolia		`--sepolia`	Sepolia network: pre-configured proof-of-work test network

A evolução das Redes de Teste pode ser vista nesse artigo The Evolution of Ethereum Testnets.

1.4 Executando o `Geth` com uma Rede de Teste

Ao executar o `geth` com algum das redes de teste será criado um diretório dentro do diretório padrão `~/.ethereum/` para cada rede de teste. Por exemplo, `~/.ethereum/goerli` para `--goerli` e `~/.ethereum/sepolia` para o `--sepolia`.

Iremos executar o cliente com a rede `Sepolia` que é recomendada como `testnet` padrão, uma vez que `Goerli` será descontinuada em 2023.

A rede `Sepolia` usa um conjunto validador permissionado. É uma rede nova, o que significa que seu estado e seu histórico são pequenos, o que deixa sua sincronização mais rápida e exige menos armazenamento. Características importantes para usuários que querem rapidamente montar um nó da rede e interagir com a rede diretamente.

```
[rogerio@ryzen-nitro execution]$ geth --sepolia --syncmode full --http --http.addr
    127.0.0.1 --http.port 8559 --http.api "eth,net,web3,personal,engine,admin"
    --keystore ~/.ethereum/sepolia/keystore
INFO [04-16|21:27:45.640] Starting Geth on Sepolia testnet...
INFO [04-16|21:27:45.641] Maximum peer count ETH=50 LES=0 total=50
INFO [04-16|21:27:45.643] Smartcard socket not found, disabling err="stat
    /run/pcscd/pcscd.comm: no such file or directory"
INFO [04-16|21:27:45.646] Set global gas cap cap=50,000,000
INFO [04-16|21:27:45.649] Allocated trie memory caches clean=154.00MiB dirty=256.00MiB
INFO [04-16|21:27:45.649] Using leveldb as the backing database
INFO [04-16|21:27:45.649] Allocated cache and file handles
    database=/home/rogerio/.ethereum/sepolia/geth/chaindata cache=512.00MiB
    handles=262,144
INFO [04-16|21:27:45.660] Using LevelDB as the backing database
INFO [04-16|21:27:45.663] Opened ancient database
    database=/home/rogerio/.ethereum/sepolia/geth/chaindata/ancient/chain
    readonly=false
INFO [04-16|21:27:45.664] Disk storage enabled for ethash caches
    dir=/home/rogerio/.ethereum/sepolia/geth/ethash count=3
INFO [04-16|21:27:45.664] Disk storage enabled for ethash DAGs
    dir=/home/rogerio/.ethash count=2
INFO [04-16|21:27:45.664] Initialising Ethereum protocol network=11,155,111 dbversion=8
INFO [04-16|21:27:45.667]
INFO [04-16|21:27:45.667]
    ---------------------------------------------------------------------------------------------------
INFO [04-16|21:27:45.667] Chain ID: 11155111 (sepolia)
INFO [04-16|21:27:45.667] Consensus: Beacon (proof-of-stake), merged from Ethash
    (proof-of-work)
INFO [04-16|21:27:45.667]
INFO [04-16|21:27:45.667] Pre-Merge hard forks (block based):
INFO [04-16|21:27:45.667] - Homestead: #0
    (https://github.com/ethereum/execution-specs/blob/master/network-upgrades/mainnet-upgrades/homest
INFO [04-16|21:27:45.667] - Tangerine Whistle (EIP 150): #0
    (https://github.com/ethereum/execution-specs/blob/master/network-upgrades/mainnet-upgrades/tanger
INFO [04-16|21:27:45.667] - Spurious Dragon/1 (EIP 155): #0
    (https://github.com/ethereum/execution-specs/blob/master/network-upgrades/mainnet-upgrades/spuric
INFO [04-16|21:27:45.667] - Spurious Dragon/2 (EIP 158): #0
    (https://github.com/ethereum/execution-specs/blob/master/network-upgrades/mainnet-upgrades/spuric
INFO [04-16|21:27:45.667] - Byzantium: #0
    (https://github.com/ethereum/execution-specs/blob/master/network-upgrades/mainnet-upgrades/byzant
INFO [04-16|21:27:45.667] - Constantinople: #0
    (https://github.com/ethereum/execution-specs/blob/master/network-upgrades/mainnet-upgrades/consta
INFO [04-16|21:27:45.667] - Petersburg: #0
```

```
      (https://github.com/ethereum/execution-specs/blob/master/network-upgrades/mainnet-upgrades/peters
27 INFO [04-16|21:27:45.667] - Istanbul: #0
      (https://github.com/ethereum/execution-specs/blob/master/network-upgrades/mainnet-upgrades/istanb
28 INFO [04-16|21:27:45.667] - Muir Glacier: #0
      (https://github.com/ethereum/execution-specs/blob/master/network-upgrades/mainnet-upgrades/muir-g
29 INFO [04-16|21:27:45.667] - Berlin: #0
      (https://github.com/ethereum/execution-specs/blob/master/network-upgrades/mainnet-upgrades/berlin
30 INFO [04-16|21:27:45.667] - London: #0
      (https://github.com/ethereum/execution-specs/blob/master/network-upgrades/mainnet-upgrades/londor
31 INFO [04-16|21:27:45.667]
32 INFO [04-16|21:27:45.667] Merge configured:
33 INFO [04-16|21:27:45.667] - Hard-fork specification:
      https://github.com/ethereum/execution-specs/blob/master/network-upgrades/mainnet-upgrades/paris.m
34 INFO [04-16|21:27:45.667] - Network known to be merged: true
35 INFO [04-16|21:27:45.667] - Total terminal difficulty: 17000000000000000
36 INFO [04-16|21:27:45.667] - Merge netsplit block: #1735371
37 INFO [04-16|21:27:45.667]
38 INFO [04-16|21:27:45.667] Post-Merge hard forks (timestamp based):
39 INFO [04-16|21:27:45.667] - Shanghai: @1677557088
      (https://github.com/ethereum/execution-specs/blob/master/network-upgrades/mainnet-upgrades/shangh
40 INFO [04-16|21:27:45.667]
41 INFO [04-16|21:27:45.667]
      ----------------------------------------------------------------------------------------------------
42 INFO [04-16|21:27:45.667]
43 INFO [04-16|21:27:45.668] Loaded most recent local block number=0 hash=25a5cc..3e6dd9
      td=131,072 age=1y6mo2w
44 INFO [04-16|21:27:45.668] Loaded local transaction journal transactions=0 dropped=0
45 INFO [04-16|21:27:45.668] Regenerated local transaction journal transactions=0
      accounts=0
46 INFO [04-16|21:27:45.668] Chain post-merge, sync via beacon client
47 INFO [04-16|21:27:45.669] Gasprice oracle is ignoring threshold set threshold=2
48 WARN [04-16|21:27:45.669] Engine API enabled protocol=eth
49 INFO [04-16|21:27:45.670] Starting peer-to-peer node
      instance=Geth/v1.11.5-stable-a38f4108/linux-amd64/go1.20.2
50 INFO [04-16|21:27:45.681] New local node record seq=1,681,600,532,512
      id=05b2a1f369961544 ip=127.0.0.1 udp=30303 tcp=30303
51 INFO [04-16|21:27:45.682] Started P2P networking
      self=enode://69520d55cffdc600a5a46f15c8255ad6831dbe9cb11f73b6aef0847a5e048b99043950daede9c4a374a3
52 INFO [04-16|21:27:45.682] IPC endpoint opened
      url=/home/rogerio/.ethereum/sepolia/geth.ipc
53 INFO [04-16|21:27:45.683] Loaded JWT secret file
      path=/home/rogerio/.ethereum/sepolia/geth/jwtsecret crc32=0xa99ff4fd
54 INFO [04-16|21:27:45.683] HTTP server started endpoint=127.0.0.1:8559 auth=false
      prefix= cors= vhosts=localhost
55 INFO [04-16|21:27:45.686] WebSocket enabled url=ws://127.0.0.1:8551
56 INFO [04-16|21:27:45.686] HTTP server started endpoint=127.0.0.1:8551 auth=true
      prefix= cors=localhost vhosts=localhost
57 WARN [04-16|21:27:55.489] System clock seems off by -25.06956443s, which can prevent
      network connectivity
58 WARN [04-16|21:27:55.490] Please enable network time synchronisation in system
      settings.
59 WARN [04-16|21:28:20.672] Post-merge network, but no beacon client seen. Please launch
      one to follow the chain!
```

Note que está sendo gerada uma mensagem de aviso:

```
WARN [04-16|21:28:20.672] Post-merge network, but no beacon client seen. Please launch
    one to follow the chain!
```

Pesquisando na Internet por *"Post-merge network, but no beacon client seen. Please launch one to follow the chain!"*, encontramos essa solução: https://github.com/ethereum/go-ethereum/issues/25791.

A documentação do *Ethereum* sobre *Consensus Clients*, mostra que `geth` deve ser iniciado, com conexão `RPC` autenticada usando um arquivo `jwtsecret`. Por padrão esse arquivo está em `~/.ethereum/geth/jwtsecret`. Para as redes de teste o caminho terá o diretório da respectiva rede, como por exemplo `~/.ethereum/sepolia/geth/jwtsecret` para a rede `Sepolia`.

```
[rogerio@ryzen-nitro execution]$ geth --sepolia --syncmode full --http --http.addr
    127.0.0.1 --http.port 8559 --http.api "eth,net,web3,personal,engine,admin"
    --keystore ~/.ethereum/sepolia/keystore --authrpc.addr localhost --authrpc.port
    8551 --authrpc.vhosts localhost --authrpc.jwtsecret
    ~/.ethereum/sepolia/geth/jwtsecret
INFO [04-16|21:27:45.640] Starting Geth on Sepolia testnet...
```

- Note que estou executando na rede de testes `--sepolia`, opção na minha versão do `geth` é diferente do livro. No livro ele diz para usar o parâmetro `--testnet` que por padrão usa a rede de testes `Ropsten`, indicando `--testnet sepolia`. Na minha instalação não suporta mais o parâmetro `--testnet`, tem o `--mainnet` e cada rede de teste disponível tem o seu parâmetro correspondente, como `--sepolia`.

1.5 Criando Contas na Rede

O comando `geth account new` cria uma nova conta na rede principal. Indicaremos que a conta é para ser criada na rede de teste `--sepolia`.

```
[rogerio@ryzen-nitro execution]$ geth --sepolia account new
INFO [04-16|21:50:49.343] Maximum peer count ETH=50 LES=0 total=50
INFO [04-16|21:50:49.344] Smartcard socket not found, disabling err="stat
    /run/pcscd/pcscd.comm: no such file or directory"
Your new account is locked with a password. Please give a password. Do not forget this
    password.
Password:
Repeat password:

Your new key was generated

Public address of the key: 0xa9e98368B44b371ceC7d205F9fE2b074b6134C95
Path of the secret key file:
    /home/rogerio/.ethereum/sepolia/keystore/UTC--2023-04-17T00-51-24.036052785Z--a9e98368b44b371cec7

- You can share your public address with anyone. Others need it to interact with you.
- You must NEVER share the secret key with anyone! The key controls access to your
    funds!
- You must BACKUP your key file! Without the key, it's impossible to access account
    funds!
- You must REMEMBER your password! Without the password, it's impossible to decrypt
    the key!
```

A conta com a chave pública `0xa9e98368B44b371ceC7d205F9fE2b074b6134C95` foi criada. Iremos criar uma segunda conta.

```
[rogerio@ryzen-nitro execution]$ geth --sepolia account new
INFO [04-16|21:52:58.948] Maximum peer count ETH=50 LES=0 total=50
INFO [04-16|21:52:58.948] Smartcard socket not found, disabling err="stat
    /run/pcscd/pcscd.comm: no such file or directory"
Your new account is locked with a password. Please give a password. Do not forget this
    password.
Password:
Repeat password:

Your new key was generated

Public address of the key: 0xc061b852A26BEdeC5Bd457b88c031c46a622f4ab
Path of the secret key file:
    /home/rogerio/.ethereum/sepolia/keystore/UTC--2023-04-17T00-53-06.379873395Z--c061b852a26bedec5bd

- You can share your public address with anyone. Others need it to interact with you.
- You must NEVER share the secret key with anyone! The key controls access to your
    funds!
- You must BACKUP your key file! Without the key, it's impossible to access account
    funds!
- You must REMEMBER your password! Without the password, it's impossible to decrypt
    the key!

[rogerio@ryzen-nitro execution]$
```

É para termos as duas contas `0xa9e98368B44b371ceC7d205F9fE2b074b6134C95` e `0xc061b852A26BEdeC5Bd457b88c03` criadas com seus arquivos em `/home/rogerio/.ethereum/sepolia/keystore/UTC--2023-04-17T00-53-06.379873395Z-`

1.6 Listando as Contas

As contas existentes ou que foram criadas podem ser listadas com o comando `geth account list`. Utilizaremos novamente o parâmetro que indica a rede de teste `--sepolia`.

```
[rogerio@ryzen-nitro execution]$ geth --sepolia account list
INFO [04-16|21:56:11.129] Maximum peer count ETH=50 LES=0 total=50
INFO [04-16|21:56:11.130] Smartcard socket not found, disabling err="stat
    /run/pcscd/pcscd.comm: no such file or directory"
INFO [04-16|21:56:11.130] Set global gas cap cap=50,000,000
Account #0: {a9e98368b44b371cec7d205f9fe2b074b6134c95}
    keystore:///home/rogerio/.ethereum/sepolia/keystore/UTC--2023-04-17T00-51-24.036052785Z--a9e98368
Account #1: {c061b852a26bedec5bd457b88c031c46a622f4ab}
    keystore:///home/rogerio/.ethereum/sepolia/keystore/UTC--2023-04-17T00-53-06.379873395Z--c061b852
[rogerio@ryzen-nitro execution]$
```

A documentação do `geth`, bem como comandos e parâmetros podem ser acessados em https://geth.ethereum.org/docs.

Executando com opção de responder a comandos via `RPC`. A documentação desta parte está disponível em https://geth.ethereum.org/docs/rpc/server.

```
[rogerio@ryzen-nitro execution]$ geth --sepolia --syncmode full --http --http.addr
```

```
    127.0.0.1 --http.port 8559 --http.api "eth,net,web3,personal,engine,admin"
    --keystore ~/.ethereum/sepolia/keystore --authrpc.addr localhost --authrpc.port
    8551 --authrpc.vhosts localhost --authrpc.jwtsecret
    ~/.ethereum/sepolia/geth/jwtsecret --nodiscover --maxpeers 15
```

1.7 Clientes de Consenso

Para terminar a configuração é necessário instalar algum cliente de consenso. Existem atualmente cinco clientes de consenso que podem ser executado em conjunto com o `Geth`:

- Lighthouse: escrito em `Rust`.
- Nimbus: escrito em `Nim`.
- Prysm: escrito em `Go`.
- Teku: escrito em `Java`.
- Lodestar: escrito em `Typescript`.

Por ser escrito na linguagem `Go`, assim como `geth`, testaremos o `Prysm`. O `Prysm` é uma implementação da especificação do consenso `proof-of-stake` do `Ethereum`.

A configuração do `Prism` pode ser feita conforme descrito no material https://docs.prylabs.network/docs/install/install-with-script.

O material ensina a usar o `Prysm` para executar um nó *Ethereum*, portanto resolver o problema apresentado após a atualização do `Merge` e opcionalmente como um validador *(validator)*.

1.8 Instalando o cliente de consenso `Prysm`[1]

Para a instalação do `Prysm`, crie no diretório `~/.ethereum/<rede>`, duas subpastas: `consensus` e `execution`. Acesse o diretório `consensus` e execute o comando para baixar o cliente `Prysm` e transformá-lo em executável:

```
1 $ mkdir prysm && cd prysm
2 $ curl https://raw.githubusercontent.com/prysmaticlabs/prysm/master/prysm.sh --output
    prysm.sh && chmod +x prysm.sh
```

1.9 Gerando um arquivo JWT Secret

- A conexão HTTP entre seu nó beacon e seu nó de execução precisa ser autenticada usando um *token* `JWT`. Existem diversas formas de gerar este *token*:
  - Usando um gerado *on line* como este. Copie e cole o valor gerado dentro do arquivo `jwt.hex`.
  - Usando `OpenSSL` para criar o *token* via comando: `openssl rand -hex 32 | tr -d "\n"> "jwt.hex"`.

[1] *Step 2: Install Prysm:* https://docs.prylabs.network/docs/install/install-with-script#step-2-install-prysm

- Usar o que foi gerado pelo cliente de execução `geth`: `~/.ethereum/sepolia/geth/jwtsecret`.
- Usar o próprio `Prysm` para gerar o `jwt.hex`:

```
## Optional. This command is necessary only if you've previously configured
    USE_PRYSM_VERSION
USE_PRYSM_VERSION=v4.0.0

## Required.
./prysm.sh beacon-chain generate-auth-secret
```

Nesta opção o `Prysm` irá mostrar o caminho onde o arquivo `jwt.hex` foi gerado.

1.10 *Executando um Cliente de Execução*[2]

Nesta etapa, executaremos o cliente de camada de execução (`geth`), se ainda não instalou veja os passos de instalação na Seção 1.1 e o nó *beacon* do `Prysm` que se conectará ao cliente em execução.

Note que iremos adicionar o parâmetro ao comando de execução com o arquivo com o *token* `--authrpc.jwtsecret ~/.ethereum/sepolia/geth/jwtsecret`.

```

O comando de execução do `geth` atualizado:

```bash
[rogerio@ryzen-nitro execution]$ geth --sepolia --syncmode full --http --http.addr
    127.0.0.1 --http.port 8559 --http.api "eth,net,web3,personal,engine,admin"
    --keystore ~/.ethereum/sepolia/keystore --authrpc.addr localhost --authrpc.port
    8551 --authrpc.vhosts localhost --authrpc.jwtsecret
    ~/.ethereum/sepolia/geth/jwtsecret --nodiscover --maxpeers 15
```

Veja as Opções de linha de comando do Geth para a definição de parâmetros.

Dependendo das opções a Sincronização pode levar um longo tempo, de horas até dias. Enquanto sincroniza, pode ir fazendo o próximo passo.

Parabéns

Você está agora executando um **nó de execução** na camada de execução da *Ethereum*.

1.11 *Executando um nó beacon usando Prysm*[3]

Altere o comando para iniciar um nó *beacon* que conecta no seu nó de execução local, necessário colocar o *hash* de uma das contas criadas. Por padrão, aqui estou utilizando a primeira conta criada, meu caso a `0xa9e98368b44b371cec7d205f9fe2b074b6134c95`.

[2] *Step 3: Run an execution client:* https://docs.prylabs.network/docs/install/install-with-script#step-3-run-an-execution-client

[3] *Step 4: Run a beacon node using Prysm:* https://docs.prylabs.network/docs/install/install-with-script#step-4-run-a-beacon-node-using-prysm
```

```
./prysm.sh beacon-chain --execution-endpoint=http://localhost:8551
    --jwt-secret=~/.ethereum/geth/jwtsecret --suggested-fee-recipient=<<hash da conta
    principal>>
```

Alterei o comando padrão para o conter o *hash* de uma das minhas contas e para o tipo de rede de teste `--sepolia`. A rede `Sepolia` precisa de um estado inicial (*genesis state*) de onde começar a sincronização. O arquivo pode ser baixado de https://github.com/eth-clients/merge-testnets/blob/main/sepolia/genesis.ssz e deve ser colocado no diretório `sepolia/consensus/prysm`.

```
[rogerio@ryzen-nitro prysm]$ wget
    https://github.com/eth-clients/merge-testnets/raw/main/sepolia/genesis.ssz -O
    genesis.ssz
```

Então o comando pode ser utilizado para iniciar um nó *beacon* que conecta seu nó de execução local.

```
[rogerio@ryzen-nitro prysm]$ ./prysm.sh beacon-chain
    --execution-endpoint=http://localhost:8551 --sepolia
    --jwt-secret=~/.ethereum/sepolia/geth/jwtsecret --genesis-state=genesis.ssz
    --suggested-fee-recipient=0xa9e98368b44b371cec7d205f9fe2b074b6134c95
```

No terminal onde o cliente de execução apareceu um novo `warning`: `"WARN [04-19|13:07:28.462] Served miner_start reqid=14 duration="233.485us"err="etherbase missing: etherbase must be explicitly specified"`.

1.12 *Executando o Console JavaScript*

O console `Javascript` pode também ser conectado ao nó `Geth` usando `IPC`. Quando o `Geth` é iniciado, um arquivo `geth.ipc` é criado automaticamente e salvo no diretório de dados. Este arquivo ou um caminho customizado para um arquivo IPC pode ser passado para o `Geth` usando o parâmetro `attach`:

```
[rogerio@ryzen-nitro ~]$ geth attach /home/rogerio/.ethereum/sepolia/geth.ipc
Welcome to the Geth JavaScript console!

instance: Geth/v1.11.5-stable-a38f4108/linux-amd64/go1.20.2
at block: 0 (Sun Oct 03 2021 10:24:41 GMT-0300 (-03))
 datadir: /home/rogerio/.ethereum/sepolia
 modules: admin:1.0 debug:1.0 engine:1.0 eth:1.0 ethash:1.0 miner:1.0 net:1.0 rpc:1.0
     txpool:1.0 web3:1.0

To exit, press ctrl-d or type exit
>
```

1.13 *Verificação do Funcionamento da Rede*

Para verificar o funcionamento da rede, utilize o comando `net.listing`:

```
[rogerio@ryzen-nitro ~]$ geth attach /home/rogerio/.ethereum/sepolia/geth.ipc
Welcome to the Geth JavaScript console!

```

```
instance: Geth/v1.11.5-stable-a38f4108/linux-amd64/go1.20.2
at block: 0 (Sun Oct 03 2021 10:24:41 GMT-0300 (-03))
 datadir: /home/rogerio/.ethereum/sepolia
 modules: admin:1.0 debug:1.0 engine:1.0 eth:1.0 ethash:1.0 miner:1.0 net:1.0 rpc:1.0
     txpool:1.0 web3:1.0

To exit, press ctrl-d or type exit
> net.listening
true
>
```

A mesma verificação pode ser feita via API JSON RPC no terminal do sistema:

```
curl -X POST --insecure --header "Content-Type: application/json" --data
    '{"jsonrpc":"2.0", "method":"net_listening","params":[], "id":64}' --location
    http://localhost:8559
{"jsonrpc":"2.0","id":64,"result":true}
```

1.14 Listando as Contas pelo Console

A lista de contas pode ser recuperada através dos comandos no console `eth.accounts` e por RPC `{"method": "eth_accounts", "params": []}`. Em versões anteriores do `geth` era possível utilizar `personal.listAccounts`, mas `personal` foi depreciado.

```
[rogerio@ryzen-nitro ~]$ geth attach /home/rogerio/.ethereum/sepolia/geth.ipc
Welcome to the Geth JavaScript console!

instance: Geth/v1.11.5-stable-a38f4108/linux-amd64/go1.20.2
at block: 0 (Sun Oct 03 2021 10:24:41 GMT-0300 (-03))
 datadir: /home/rogerio/.ethereum/sepolia
 modules: admin:1.0 debug:1.0 engine:1.0 eth:1.0 ethash:1.0 miner:1.0 net:1.0 rpc:1.0
     txpool:1.0 web3:1.0

To exit, press ctrl-d or type exit
> net.listening
true
> personal.listAccounts
ReferenceError: personal is not defined
        at <eval>:1:1(0)

> eth.accounts
["0xa9e98368b44b371cec7d205f9fe2b074b6134c95",
    "0xc061b852a26bedec5bd457b88c031c46a622f4ab"]
```

O mesmo resultado pode ser obtido via comando `curl` no terminal para listar as contas usando RPC:

```
[rogerio@ryzen-nitro execution]$ curl -X POST --insecure --header "Content-Type:
    application/json" --data '{"jsonrpc":"2.0","method":"eth_accounts","params":[],
    "id":64}' --location http://localhost:8559
{"jsonrpc":"2.0","id":64,"result":["0xa9e98368b44b371cec7d205f9fe2b074b6134c95","0xc061b852a26bedec5
```

Outros comandos podem ser executados da mesma maneira via Console JavaScript ou invocação `RPC` através do `curl`. Uma lista completa de comandos da API `RPC`estão disponíveis em JSON-RPC API.

Como por exemplo, recuperar a versão do cliente `geth`:

```
[[rogerio@ryzen-nitro execution]$ curl -X POST --insecure --header "Content-Type:
    application/json" --data
    '{"jsonrpc":"2.0","method":"web3_clientVersion","params":[],"id":67}' --location
    http://localhost:8559
{"jsonrpc":"2.0","id":67,"result":"Geth/v1.11.5-stable-a38f4108/linux-amd64/go1.20.2"}
```

Ou a versão da Rede:

```
[rogerio@ryzen-nitro execution]$ curl -X POST --header "Content-Type:
    application/json" --data
    '{"jsonrpc":"2.0","method":"net_version","params":[],"id":67}' --location
    http://localhost:8559
{"jsonrpc":"2.0","id":67,"result":"11155111"}
```

O valor 11155111 em `result` indica que estamos executando sobre a rede `Sepolia`. Uma lista completa com os `IDs` das redes está disponível em https://chainlist.org. Os mais comuns são 1: *Ethereum Mainnet*, 2: *Morden testnet* (depreciado), 3: *Ropsten testnet*, 4: *Rinkeby testnet*, 5: *Goerli testnet* (será depreciada em 2023).

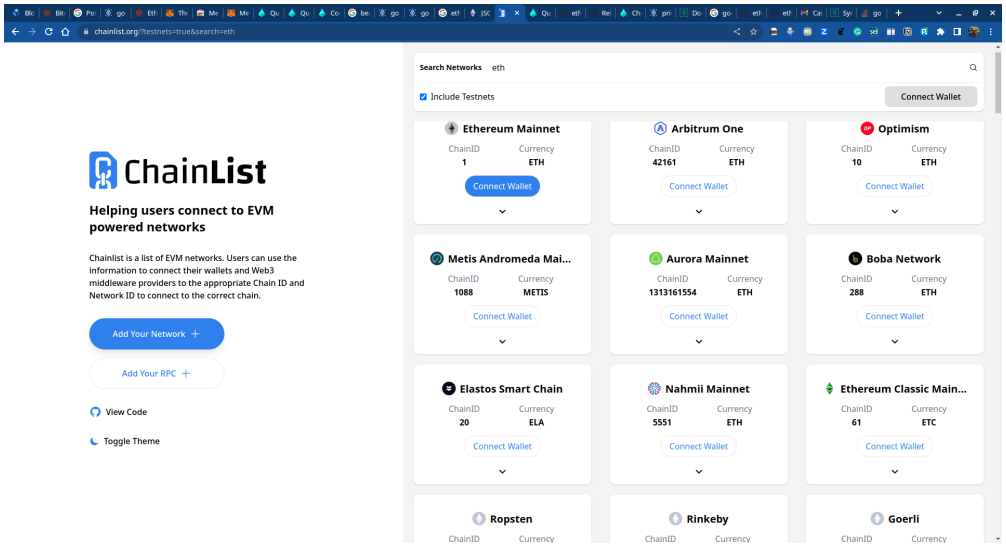


Figura 2: Lista de IDs Fonte: chainlist

1.15 Teste de transferência de valores entre carteiras

Corrigir os endereços.

Podemos testar uma transferência de valores. As duas formas de se conseguir valores é minerando ou solicitando valores para *Faucets* da rede, como por exemplo o https://faucet.sepolia.dev/.

Os valores de cada conta/carteira pode ser verificado com o comando `eth.getBalance(hash)`,

passando como parâmetro a cadeia *hash* de identificação da conta ou indicando na lista de contas `eth.getBalance(eth.accounts[0])`.

```
[rogerio@ryzen-nitro ~]$ geth attach /home/rogerio/.ethereum/sepolia/geth.ipc
Welcome to the Geth JavaScript console!

instance: Geth/v1.11.5-stable-a38f4108/linux-amd64/go1.20.2
at block: 0 (Sun Oct 03 2021 10:24:41 GMT-0300 (-03))
 datadir: /home/rogerio/.ethereum/sepolia
 modules: admin:1.0 debug:1.0 engine:1.0 eth:1.0 ethash:1.0 miner:1.0 net:1.0 rpc:1.0
     txpool:1.0 web3:1.0

To exit, press ctrl-d or type exit
> net.listening

true
> eth.accounts
["0xa9e98368b44b371cec7d205f9fe2b074b6134c95",
    "0xc061b852a26bedec5bd457b88c031c46a622f4ab"]
> eth.getBalance("0xa9e98368b44b371cec7d205f9fe2b074b6134c95")
0
> eth.getBalance("0xc061b852a26bedec5bd457b88c031c46a622f4ab")
0
> eth.getBalance("0xa9e98368b44b371cec7d205f9fe2b074b6134c95")
0
```

Como os saldos estão zerados, vamos iniciar a mineração com o comando `miner.start()`.

```
> miner.start()
Error: etherbase missing: etherbase must be explicitly specified
       at web3.js:6365:9(39)
       at send (web3.js:5099:62(29))
       at <eval>:1:12(2)

> web3.fromWei(eth.getBalance("0xa9e98368b44b371cec7d205f9fe2b074b6134c95"), "ether")
0
> web3.fromWei(eth.getBalance("0xc061b852a26bedec5bd457b88c031c46a622f4ab"), "ether")
0
> web3.fromWei(eth.getBalance(eth.coinbase), "ether")
Error: etherbase must be explicitly specified
       at web3.js:6365:9(39)
       at get (web3.js:6265:66(14))
       at <eval>:1:33(5)

> eth.coinbase
Error: etherbase must be explicitly specified
       at web3.js:6365:9(39)
       at get (web3.js:6265:66(14))
       at <eval>:1:5(1)

> eth.coinbase.panic: Error: etherbase must be explicitly specified at
    web3.js:6365:9(39)

goroutine 82 [running]:
github.com/dop251/goja.(*baseJsFuncObject)._call(...)
       github.com/dop251/goja@v0.0.0-20230122112309-96b1610dd4f7/func.go:396
```

O erro **"Error: etherbase must be explicitly specified"** ocorre pois é necessário fornecer o *hash* da carteira também para a execução do cliente `geth`.

```
[rogerio@ryzen-nitro execution]$ geth --sepolia --syncmode full --http --http.addr
    127.0.0.1 --http.port 8559 --http.api "eth,net,web3,personal,engine,admin"
    --keystore ~/.ethereum/sepolia/keystore --authrpc.addr localhost --authrpc.port
    8551 --authrpc.vhosts localhost --authrpc.jwtsecret
    ~/.ethereum/sepolia/geth/jwtsecret --nodiscover --maxpeers 15
    --miner.etherbase=0xa9e98368b44b371cec7d205f9fe2b074b6134c95
```

Fornecendo a carteira como `--miner.etherbase=0xa9e98368b44b371cec7d205f9fe2b074b6134c95`, agora podemos iniciar a mineração.

```
> miner.start()
null
> miner.stop()
null
> eth.getBalance("0xa9e98368b44b371cec7d205f9fe2b074b6134c95")
5.004e+21
> eth.getBalance("0xa9e98368b44b371cec7d205f9fe2b074b6134c95")
5.004e+21
> eth.accounts
["0xa9e98368b44b371cec7d205f9fe2b074b6134c95",
    "0xc061b852a26bedec5bd457b88c031c46a622f4ab"]
> eth.getBalance("0xc061b852a26bedec5bd457b88c031c46a622f4ab")
0
> web3.fromWei(eth.getBalance("0xa9e98368b44b371cec7d205f9fe2b074b6134c95"), "ether")
5004
> eth.getBalance(eth.accounts[0])
5.004e+21
```

Para enviar 100 *ethers* da primeira para a segunda carteira:

```
> eth.sendTransaction({from: "0xa9e98368b44b371cec7d205f9fe2b074b6134c95", to:
    "0xc061b852a26bedec5bd457b88c031c46a622f4ab", value: 100})
Error: authentication needed: password or unlock
        at web3.js:6365:9(39)
        at send (web3.js:5099:62(29))
        at <eval>:1:20(9)

> eth.sendTransaction({from: "0xa9e98368b44b371cec7d205f9fe2b074b6134c95", to:
    "0xc061b852a26bedec5bd457b88c031c46a622f4ab", value:web3.toWei(100, "ether")})
Error: authentication needed: password or unlock
        at web3.js:6365:9(39)
        at send (web3.js:5099:62(29))
        at <eval>:1:20(13)

> personal.unlockAccount(eth.accounts[0])
ReferenceError: personal is not defined
        at <eval>:1:1(0)
```

O erro **"Error: authentication needed: password or unlock"** ocorre por que precisamos

autorizar a transação. Em versões anteriores era possível desbloquear as contas via *console* `JavaScript`, conforme tentamos no Código acima `personal.unlockAccount(eth.accounts[0])`. Verificando a documentação o `personal` foi depreciado e não pode ser mais utilizado.

Na versão corrente é preciso utilizar o `clef` para fazer a autenticação em um *terminal* separado. Então em um outro *terminal* inicie a instância do `clef` com o comando:

```
[rogerio@ryzen-nitro sepolia]$ clef --chainid 1115511 --keystore
    ~/.ethereum/sepolia/keystore --configdir ~/.ethereum/sepolia/clef --http

WARNING!

Clef is an account management tool. It may, like any software, contain bugs.

Please take care to
- backup your keystore files,
- verify that the keystore(s) can be opened with your password.

Clef is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY;
without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR
PURPOSE. See the GNU General Public License for more details.

Enter 'ok' to proceed:
> ok

INFO [04-19|14:00:58.719] Using CLI as UI-channel
INFO [04-19|14:00:59.171] Loaded 4byte database embeds=268,621 locals=0
    local=./4byte-custom.json
WARN [04-19|14:00:59.171] Failed to open master, rules disabled err="failed stat on
    /home/rogerio/.ethereum/sepolia/clef/masterseed.json: stat
    /home/rogerio/.ethereum/sepolia/clef/masterseed.json: no such file or directory"
INFO [04-19|14:00:59.172] Starting signer chainid=1,115,511
    keystore=/home/rogerio/.ethereum/sepolia/keystore light-kdf=false advanced=false
INFO [04-19|14:00:59.184] Smartcard socket file missing, disabling err="stat
    /run/pcscd/pcscd.comm: no such file or directory"
INFO [04-19|14:00:59.184] Audit logs configured file=audit.log
INFO [04-19|14:00:59.186] HTTP endpoint opened url=http://127.0.0.1:8550/
INFO [04-19|14:00:59.186] IPC endpoint opened
    url=/home/rogerio/.ethereum/sepolia/clef/clef.ipc

------- Signer info -------
* extapi_ipc : /home/rogerio/.ethereum/sepolia/clef/clef.ipc
* intapi_version : 7.0.1
* extapi_version : 6.1.0
* extapi_http : http://127.0.0.1:8550/

------- Available accounts -------
0. 0xa9e98368B44b371ceC7d205F9fE2b074b6134C95 at
    keystore:///home/rogerio/.ethereum/sepolia/keystore/UTC--2023-04-17T00-51-24.036052785Z--a9e98368
1. 0xc061b852A26BEdeC5Bd457b88c031c46a622f4ab at
    keystore:///home/rogerio/.ethereum/sepolia/keystore/UTC--2023-04-17T00-53-06.379873395Z--c061b852
```

O `clef` iniciará e listará as contas disponíveis que irá controlar.

Volte ao terminal onde iniciou o `geth` e indique que as autentificações serão via `clef` com o parâmetro `--signer=/home/rogerio/.ethereum/sepolia/clef/clef.ipc` passando o caminho dado

pelo `clef`.

```
[rogerio@ryzen-nitro execution]$ geth --sepolia --syncmode full --http --http.addr
    127.0.0.1 --http.port 8559 --http.api "eth,net,web3,personal,engine,admin"
    --keystore ~/.ethereum/sepolia/keystore --authrpc.addr localhost --authrpc.port
    8551 --authrpc.vhosts localhost --authrpc.jwtsecret
    ~/.ethereum/sepolia/geth/jwtsecret --nodiscover --maxpeers 15
    --miner.etherbase=0xa9e98368b44b371cec7d205f9fe2b074b6134c95
    --signer=/home/rogerio/.ethereum/sepolia/clef/clef.ipc
```

Retorne ao *console* `JavaScript`, e execute novamente as transações, cada uma deverá ser autorizada no console do `clef`.

```
> eth.sendTransaction({from: "0xa9e98368b44b371cec7d205f9fe2b074b6134c95", to:
    "0xc061b852a26bedec5bd457b88c031c46a622f4ab", value: 100})
```

No console do `clef` é possível autorizar e ver a transação assinada com `hash: 0xb579cc595601e4aca546ce4e4` que é o mesmo `id` devolvido no console JavaScript.

```
-------------------------------------------
Request context:
       NA -> ipc -> NA

Additional HTTP header data, provided by the external caller:
       User-Agent: ""
       Origin: ""
Approve? [y/N]:
> y
--------- Transaction request-------------
to: 0xc061b852A26BEdeC5Bd457b88c031c46a622f4ab
from: 0xa9e98368B44b371ceC7d205F9fE2b074b6134C95 [chksum ok]
value: 100 wei
gas: 0x5208 (21000)
maxFeePerGas: 1000000014 wei
maxPriorityFeePerGas: 1000000000 wei
nonce: 0x0 (0)
chainid: 0xaa36a7
Accesslist

Request context:
       NA -> ipc -> NA

Additional HTTP header data, provided by the external caller:
       User-Agent: ""
       Origin: ""
-------------------------------------------
Approve? [y/N]:
> y
## Account password

Please enter the password for account 0xa9e98368B44b371ceC7d205F9fE2b074b6134C95
>
-----------------------
Transaction signed:
 {
```

```
37     "type": "0x2",
38     "nonce": "0x0",
39     "gasPrice": null,
40     "maxPriorityFeePerGas": "0x3b9aca00",
41     "maxFeePerGas": "0x3b9aca0e",
42     "gas": "0x5208",
43     "value": "0x64",
44     "input": "0x",
45     "v": "0x0",
46     "r": "0xcfaf8674159cf0618393d32bc4e421969acf3bc1c7dfc5bc13224025cc5a6e5e",
47     "s": "0x44cbc80e7123b73846caf72eb13312b8ac8a2dc323173a4b58491b1112c2298c",
48     "to": "0xc061b852a26bedec5bd457b88c031c46a622f4ab",
49     "chainId": "0xaa36a7",
50     "accessList": [],
51     "hash": "0x298bb20795a584c9bd038d47323f592e072b66eace172833d27acaad8278281a"
52   }
```

Com a autorização feita no *console* do `clef` será mostrado as informações da transação assinada, como o `hash: 0x298bb20795a584c9bd038d47323f592e072b66eace172833d27acaad8278281a`. Observe que no *console* `JavaScript` aparece o mesmo *hash* da transação para a transação que solicitamos:

```
1 > eth.sendTransaction({from: "0xa9e98368b44b371cec7d205f9fe2b074b6134c95", to:
      "0xc061b852a26bedec5bd457b88c031c46a622f4ab", value: 100})
2 "0x298bb20795a584c9bd038d47323f592e072b66eace172833d27acaad8278281a"
3 >
```

Se buscarmos pelo recibo da transação ou verificarmos se o saldo da conta 1 foi alterado, veremos que a transação no foi efetivada ainda, pois a consulta devolve `null` e o saldo da conta ainda é zero.

```
1 > eth.sendTransaction({from: "0xa9e98368b44b371cec7d205f9fe2b074b6134c95", to:
      "0xc061b852a26bedec5bd457b88c031c46a622f4ab", value: 100})
2 "0x298bb20795a584c9bd038d47323f592e072b66eace172833d27acaad8278281a"
3 >
      eth.getTransactionReceipt("0x298bb20795a584c9bd038d47323f592e072b66eace172833d27acaad8278281a")
4 null
5 > eth.getBalance(eth.accounts[1])
6 0
7 > eth.getBalance(eth.accounts[0])
8 5.004e+21
```

É preciso minerar para a transação ser efetivada:

```
1
2 > miner.start()
3 null
4 > eth.getBalance(eth.accounts[1])
5 100
6 > miner.stop()
7 null
8 > eth.getBalance(eth.accounts[0])
9 5.085999999999998529e+21
10 > eth.getBalance(eth.accounts[1])
11 100
```

```
>
```

Agora sim, é possível recuper o recibo da transação. Desde do *fork* para a versão `Byzantium`, a *Ethereum* fornece uma forma de se verficar se a transação deu certo ou falhou, o campo *status* no recibo indica a situação (0 - falhou e 1 - foi executada com sucesso).

```
>
    eth.getTransactionReceipt("0x298bb20795a584c9bd038d47323f592e072b66eace172833d27acaad8278281a")
{
  blockHash: "0x1e3ae615a7d676cc51983ea066e2cda3a5d7702d27255065b330591cc3b9062f",
  blockNumber: 2503,
  contractAddress: null,
  cumulativeGasUsed: 21000,
  effectiveGasPrice: 1000000007,
  from: "0xa9e98368b44b371cec7d205f9fe2b074b6134c95",
  gasUsed: 21000,
  logs: [],
  logsBloom:
      "0x000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
  status: "0x1",
  to: "0xc061b852a26bedec5bd457b88c031c46a622f4ab",
  transactionHash:
      "0x298bb20795a584c9bd038d47323f592e072b66eace172833d27acaad8278281a",
  transactionIndex: 0,
  type: "0x2"
}
```

Ao enviar outra quantidade para a segunda conta é preciso autorizar a transação no console do `clef` e minerar novamente para ela ser efetivada.

```

> eth.sendTransaction({from: "0xa9e98368b44b371cec7d205f9fe2b074b6134c95", to:
    "0xc061b852a26bedec5bd457b88c031c46a622f4ab", value: 1000})
"0x667f9570b629fdd437b112e4b5f388cbbe68efb43e0bdbb75bf9db473cbd00fc"
> eth.getBalance(eth.accounts[1])

100
> miner.start()
null
> eth.getBalance(eth.accounts[1])
1100
> miner.stop()
null
> eth.getBalance(eth.accounts[1])
1100
> eth.getBalance(eth.accounts[0])
5.177999999999997049e+21
>
```

No *terminal* de execução do `geth` aparece que a transação foi submetida e após a mineração os saldos das contas foram atualizados.

```
INFO [04-19|15:10:01.663] Submitted transaction
    hash=0x667f9570b629fdd437b112e4b5f388cbbe68efb43e0bdbb75bf9db473cbd00fc
```

```
    from=0xa9e98368B44b371ceC7d205F9fE2b074b6134C95 nonce=1
    recipient=0xc061b852A26BEdeC5Bd457b88c031c46a622f4ab value=1000
```

A transação `0x667f9570b629fdd437b112e4b5f388cbbe68efb43e0bdbb75bf9db473cbd00fc` foi autorizada e podemos verificar o saldo das carteiras.

```
>
    eth.getTransactionReceipt("0x667f9570b629fdd437b112e4b5f388cbbe68efb43e0bdbb75bf9db473cbd00fc")
{
  blockHash: "0x30643a61e054cc3a2e5ce04c67e4c14ed192a889cede500bcf214d754540fe6b",
  blockNumber: 2544,
  contractAddress: null,
  cumulativeGasUsed: 21000,
  effectiveGasPrice: 1000000007,
  from: "0xa9e98368b44b371cec7d205f9fe2b074b6134c95",
  gasUsed: 21000,
  logs: [],
  logsBloom:
      "0x000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
  status: "0x1",
  to: "0xc061b852a26bedec5bd457b88c031c46a622f4ab",
  transactionHash:
      "0x667f9570b629fdd437b112e4b5f388cbbe68efb43e0bdbb75bf9db473cbd00fc",
  transactionIndex: 0,
  type: "0x2"
}
> eth.accounts
["0xa9e98368b44b371cec7d205f9fe2b074b6134c95",
    "0xc061b852a26bedec5bd457b88c031c46a622f4ab"]
> web3.fromWei(eth.getBalance("0xa9e98368b44b371cec7d205f9fe2b074b6134c95"), "ether")
5177.9999999999997049
> web3.fromWei(eth.getBalance("0xc061b852a26bedec5bd457b88c031c46a622f4ab"), "ether")
1.1e-15
```

1.16 Leitura Recomendada

Leitura Recomendada

Capítulo 11: Ethereum 101
Livro: IMRAN BASHIR. Mastering Blockchain : Distributed Ledger Technology, Decentralization, and Smart Contracts Explained, 2nd Edition.
Capítulo 12: Futher Ethereum
Livro: IMRAN BASHIR. Mastering Blockchain : Distributed Ledger Technology, Decentralization, and Smart Contracts Explained, 2nd Edition.

2 Prática: Criando uma Rede Ethereum Privada

O objetivo dessa prática é criarmos uma Rede *Ethereum* Privada.

2.1 Criando uma Rede Privada Local

Para a criação de uma nova Rede Privada Local é necessário fazermos algumas configuraçãos. Precisamos criar um diretório `mkdir ~/.etherprivate` para ser a base de armazenamento para a nova rede. Temos que fornecer a configurações iniciais para a nova rede, criando um arquivo `privategenesis.json` em `~/.etherprivate`.

Passos

1. Criar um diretório `mkdir ~/.etherprivate`
2. Criar um arquivo `privategenesis.json` em `~/.etherprivate`.

O conteúdo do arquivo `privategenesis.json` deve ser o listado no Código 1.

```
{
  "config": {
    "chainId": 786,
    "homesteadBlock": 0,
    "eip150Block": 0,
    "eip155Block": 0,
    "eip158Block": 0,
    "byzantiumBlock": 0,
    "constantinopleBlock": 0,
    "petersburgBlock": 0,
    "istanbulBlock": 0,
    "berlinBlock": 0,
    "ethash": {}
  },
  "nonce": "0x0000000000000042",
  "timestamp": "0x00",
  "parentHash": "0x0000000000000000000000000000000000000000000000000000000000000000",
  "extraData": "0x00",
  "gasLimit": "0x8000000",
  "difficulty": "0x0400",
  "mixhash": "0x0000000000000000000000000000000000000000000000000000000000000000",
  "coinbase": "0x3333333333333333333333333333333333333333",
  "alloc": {}
}
```

Código 1: Genesis File

Utilizando esse *genesis file* a rede será criada sem nenhum usuário. Será possível criar posteriormente conforme descrito na [Seção@#sec:criando:contas].

Uma outra forma de estabelecer a configuração inicial com uma conta que já seja alocado algum valor é criando o usuário antes de inicializar a rede.

```
[rogerio@ryzen-nitro .etherprivate]$ geth --networkid 786 --datadir ~/.etherprivate/
    account new
INFO [04-19|16:22:04.258] Maximum peer count ETH=50 LES=0 total=50
```

```
INFO [04-19|16:22:04.258] Smartcard socket not found, disabling err="stat
    /run/pcscd/pcscd.comm: no such file or directory"
Your new account is locked with a password. Please give a password. Do not forget this
    password.
Password:
Repeat password:

Your new key was generated

Public address of the key: 0x2db017e44b03b37755a4b15e14cd799f83de4c13
Path of the secret key file:
    /home/rogerio/.etherprivate/keystore/UTC--2023-04-17T12-22-11.261468773Z--2db017e44b03b37755a4b15

- You can share your public address with anyone. Others need it to interact with you.
- You must NEVER share the secret key with anyone! The key controls access to your
    funds!
- You must BACKUP your key file! Without the key, it's impossible to access account
    funds!
- You must REMEMBER your password! Without the password, it's impossible to decrypt
    the key!
```

Com um `id` da conta criada (`0x2db017e44b03b37755a4b15e14cd799f83de4c13`) é possível completarmos algumas informações para inicializar a nova rede. Altere o conteúdo do arquivo `privategenesis.json` tal como listado no Código 2.

```
{
  "config": {
    "chainId": 786,
    "homesteadBlock": 0,
    "eip150Block": 0,
    "eip155Block": 0,
    "eip158Block": 0,
    "byzantiumBlock": 0,
    "constantinopleBlock": 0,
    "petersburgBlock": 0,
    "istanbulBlock": 0,
    "berlinBlock": 0,
    "ethash": {}
  },
  "nonce": "0x0000000000000042",
  "timestamp": "0x00",
  "parentHash": "0x0000000000000000000000000000000000000000000000000000000000000000",
  "extraData": "0x00",
  "gasLimit": "0x8000000",
  "difficulty": "0x0400",
  "mixhash": "0x0000000000000000000000000000000000000000000000000000000000000000",
  "coinbase": "0x2db017e44b03b37755a4b15e14cd799f83de4c13",
  "alloc": {
    "2db017e44b03b37755a4b15e14cd799f83de4c13": { "balance": "300000" }
  }
}
```

Código 2: Genesis File Atualizado

Após a configuração inicial no *genesis file*, o `Geth` é utilizado para a criação e inicialização da

nova Rede. O `geth` deve ser executado com os parâmetros `--datadir`, indicando o diretório onde os dados da nova rede serão armazenados e com o `init` indicando o caminho para o *genesis file*, conforme Código 3.

```
[rogerio@ryzen-nitro .etherprivate]$ geth --identity "RAGPrivateEtherem" --datadir
    ~/.etherprivate init ~/.etherprivate/privategenesis.json
INFO [04-19|16:01:05.715] Maximum peer count ETH=50 LES=0 total=50
INFO [04-19|16:01:05.717] Smartcard socket not found, disabling err="stat
    /run/pcscd/pcscd.comm: no such file or directory"
INFO [04-19|16:01:05.720] Set global gas cap cap=50,000,000
INFO [04-19|16:01:05.721] Using leveldb as the backing database
INFO [04-19|16:01:05.721] Allocated cache and file handles
    database=/home/rogerio/.etherprivate/geth/chaindata cache=16.00MiB handles=16
INFO [04-19|16:01:05.734] Using LevelDB as the backing database
INFO [04-19|16:01:05.741] Opened ancient database
    database=/home/rogerio/.etherprivate/geth/chaindata/ancient/chain readonly=false
INFO [04-19|16:01:05.741] Writing custom genesis block
INFO [04-19|16:01:05.742] Persisted trie from memory database nodes=1 size=142.00B
    time="56.922us" gcnodes=0 gcsize=0.00B gctime=0s livenodes=1 livesize=0.00B
INFO [04-19|16:01:05.743] Successfully wrote genesis state database=chaindata
    hash=c2469e..234a72
INFO [04-19|16:01:05.743] Using leveldb as the backing database
INFO [04-19|16:01:05.744] Allocated cache and file handles
    database=/home/rogerio/.etherprivate/geth/lightchaindata cache=16.00MiB handles=16
INFO [04-19|16:01:05.747] Using LevelDB as the backing database
INFO [04-19|16:01:05.755] Opened ancient database
    database=/home/rogerio/.etherprivate/geth/lightchaindata/ancient/chain
    readonly=false
INFO [04-19|16:01:05.755] Writing custom genesis block
INFO [04-19|16:01:05.756] Persisted trie from memory database nodes=1 size=142.00B
    time="29.054us" gcnodes=0 gcsize=0.00B gctime=0s livenodes=1 livesize=0.00B
INFO [04-19|16:01:05.757] Successfully wrote genesis state database=lightchaindata
    hash=c2469e..234a72
[rogerio@ryzen-nitro .etherprivate]$
```

Código 3: Inicialização da Rede Privada Local

2.2 Executando a nova Rede

O mesmo cliente `Geth` pode ser iniciado, executando com base na nova rede criada.

```
[rogerio@ryzen-nitro .etherprivate]$ geth --networkid 786 --datadir ~/.etherprivate/
    --syncmode full --allow-insecure-unlock --http --http.addr 127.0.0.1 --http.port
    8559 --http.api "eth,net,web3,personal,engine,admin,debug" --keystore
    ~/.etherprivate/keystore --authrpc.addr localhost --authrpc.port 8551
    --authrpc.vhosts localhost --authrpc.jwtsecret ~/.etherprivate/geth/jwtsecret
    --nodiscover --maxpeers 15
INFO [04-17|09:14:28.084] Maximum peer count ETH=15 LES=0 total=15
INFO [04-17|09:14:28.085] Smartcard socket not found, disabling err="stat
    /run/pcscd/pcscd.comm: no such file or directory"
INFO [04-17|09:14:28.087] Set global gas cap cap=50,000,000
INFO [04-17|09:14:28.089] Allocated trie memory caches clean=154.00MiB dirty=256.00MiB
INFO [04-17|09:14:28.089] Using leveldb as the backing database
INFO [04-17|09:14:28.089] Allocated cache and file handles
```

```
    database=/home/rogerio/.etherprivate/geth/chaindata cache=512.00MiB handles=262,144
8 INFO [04-17|09:14:28.096] Using LevelDB as the backing database
9 INFO [04-17|09:14:28.097] Opened ancient database
    database=/home/rogerio/.etherprivate/geth/chaindata/ancient/chain readonly=false
10 INFO [04-17|09:14:28.097] Disk storage enabled for ethash caches
    dir=/home/rogerio/.etherprivate/geth/ethash count=3
11 INFO [04-17|09:14:28.097] Disk storage enabled for ethash DAGs
    dir=/home/rogerio/.ethash count=2
12 INFO [04-17|09:14:28.097] Initialising Ethereum protocol network=786 dbversion=8
13 INFO [04-17|09:14:28.098]
14 INFO [04-17|09:14:28.098]
    -----------------------------------------------------------------------------------------------
15 INFO [04-17|09:14:28.098] Chain ID: 786 (unknown)
16 INFO [04-17|09:14:28.098] Consensus: unknown
17 INFO [04-17|09:14:28.098]
18 INFO [04-17|09:14:28.098] Pre-Merge hard forks (block based):
19 INFO [04-17|09:14:28.098] - Homestead: #0
    (https://github.com/ethereum/execution-specs/blob/master/network-upgrades/mainnet-upgrades/homest
20 INFO [04-17|09:14:28.098] - Tangerine Whistle (EIP 150): #0
    (https://github.com/ethereum/execution-specs/blob/master/network-upgrades/mainnet-upgrades/tanger
21 INFO [04-17|09:14:28.098] - Spurious Dragon/1 (EIP 155): #0
    (https://github.com/ethereum/execution-specs/blob/master/network-upgrades/mainnet-upgrades/spuric
22 INFO [04-17|09:14:28.098] - Spurious Dragon/2 (EIP 158): #0
    (https://github.com/ethereum/execution-specs/blob/master/network-upgrades/mainnet-upgrades/spuric
23 INFO [04-17|09:14:28.098] - Byzantium: #<nil>
    (https://github.com/ethereum/execution-specs/blob/master/network-upgrades/mainnet-upgrades/byzant
24 INFO [04-17|09:14:28.098] - Constantinople: #<nil>
    (https://github.com/ethereum/execution-specs/blob/master/network-upgrades/mainnet-upgrades/consta
25 INFO [04-17|09:14:28.098] - Petersburg: #<nil>
    (https://github.com/ethereum/execution-specs/blob/master/network-upgrades/mainnet-upgrades/peters
26 INFO [04-17|09:14:28.098] - Istanbul: #<nil>
    (https://github.com/ethereum/execution-specs/blob/master/network-upgrades/mainnet-upgrades/istanb
27 INFO [04-17|09:14:28.098] - Berlin: #<nil>
    (https://github.com/ethereum/execution-specs/blob/master/network-upgrades/mainnet-upgrades/berlin
28 INFO [04-17|09:14:28.098] - London: #<nil>
    (https://github.com/ethereum/execution-specs/blob/master/network-upgrades/mainnet-upgrades/londor
29 INFO [04-17|09:14:28.098]
30 INFO [04-17|09:14:28.098] The Merge is not yet available for this network!
31 INFO [04-17|09:14:28.098] - Hard-fork specification:
    https://github.com/ethereum/execution-specs/blob/master/network-upgrades/mainnet-upgrades/paris.m
32 INFO [04-17|09:14:28.098]
33 INFO [04-17|09:14:28.098] Post-Merge hard forks (timestamp based):
34 INFO [04-17|09:14:28.098]
35 INFO [04-17|09:14:28.098]
    -----------------------------------------------------------------------------------------------
36 INFO [04-17|09:14:28.098]
37 INFO [04-17|09:14:28.098] Loaded most recent local block number=0 hash=6650a0..b5c158
    td=1024 age=54y3w3d
38 INFO [04-17|09:14:28.099] Loaded local transaction journal transactions=0 dropped=0
39 INFO [04-17|09:14:28.099] Regenerated local transaction journal transactions=0
    accounts=0
40 INFO [04-17|09:14:28.101] Gasprice oracle is ignoring threshold set threshold=2
41 WARN [04-17|09:14:28.101] Engine API enabled protocol=eth
42 WARN [04-17|09:14:28.101] Engine API started but chain not configured for merge yet
43 INFO [04-17|09:14:28.101] Starting peer-to-peer node
    instance=Geth/v1.11.5-stable-a38f4108/linux-amd64/go1.20.2
```

```
INFO [04-17|09:14:28.111] IPC endpoint opened url=/home/rogerio/.etherprivate/geth.ipc
INFO [04-17|09:14:28.112] Loaded JWT secret file
    path=/home/rogerio/.etherprivate/geth/jwtsecret crc32=0x612a2337
INFO [04-17|09:14:28.112] New local node record seq=1,681,733,283,410
    id=ffcbe33c92bbd417 ip=127.0.0.1 udp=0 tcp=30303
INFO [04-17|09:14:28.112] Started P2P networking
    self="enode://4e6d847eea6db022f2a8a44453f4009a52141bc87eae8204f84d968e91f18a60c9ed3f60890e4e6d189
INFO [04-17|09:14:28.112] HTTP server started endpoint=127.0.0.1:8559 auth=false
    prefix= cors= vhosts=localhost
INFO [04-17|09:14:28.113] WebSocket enabled url=ws://127.0.0.1:8551
INFO [04-17|09:14:28.113] HTTP server started endpoint=127.0.0.1:8551 auth=true
    prefix= cors=localhost vhosts=localhost
```

2.3 *Interagindo com a nova Rede*

O console pode ser utilizado na interação com a instância da nova rede em execução.

```
$ geth attach ~/.etherprivate/geth.ipc
```

2.4 *Criando contas na nova Rede*

Vamos criar duas contas para testes na nova rede. Utilizaremos para fins de teste a senha *"admin12345"*. O Código abaixo mostra a execução quando era possível utilizar o método `personal.newAccount(...)` via console JavaScript.

```
> personal.newAccount("admin12345")
"0xedbc36d74d5a1cd64db36e53798bd1781f0c4955"

> eth.accounts
["0xedbc36d74d5a1cd64db36e53798bd1781f0c4955"]

> personal.newAccount("admin12345")

"0x1478d95f8754b3ba7127100dd0bb46578fe7d22a"

> eth.accounts
["0xedbc36d74d5a1cd64db36e53798bd1781f0c4955",
    "0x1478d95f8754b3ba7127100dd0bb46578fe7d22a"]
```

Com a depreciação do `personal` é recomendado utilizar o `clef` com o parâmetro `newaccount`. É importante indicar o diretório `keystore` do `.etherprivate`.

Se seguimos a opção de criar uma conta antes para configurar o *genesis file* nossa base já terá uma conta. Essa conta pode ser verificada utilizando o `geth --networkid 786 --datadir ~/.etherprivate/ account` ou o comando do `clef`.

```
[rogerio@ryzen-nitro keystore]$ clef list-accounts --keystore ~/.etherprivate/keystore

WARNING!

Clef is an account management tool. It may, like any software, contain bugs.

```

```
Please take care to
- backup your keystore files,
- verify that the keystore(s) can be opened with your password.

Clef is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY;
without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR
PURPOSE. See the GNU General Public License for more details.

Enter 'ok' to proceed:
> ok


0x2db017E44b03B37755A4b15e14Cd799f83DE4c13
    (keystore:///home/rogerio/.etherprivate/keystore/UTC--2023-04-17T12-22-11.261468773Z--2db017e44b0
```

A conta `0x2db017E44b03B37755A4b15e14Cd799f83DE4c13` que foi gerada, utilizei a senha `admin12345`.

```
[rogerio@ryzen-nitro .etherprivate]$ clef newaccount --keystore keystore

WARNING!

Clef is an account management tool. It may, like any software, contain bugs.

Please take care to
- backup your keystore files,
- verify that the keystore(s) can be opened with your password.

Clef is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY;
without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR
PURPOSE. See the GNU General Public License for more details.

Enter 'ok' to proceed:
> ok

## New account password

Please enter a password for the new account to be created (attempt 0 of 3)
>
-----------------------
INFO [04-17|09:28:56.607] Your new key was generated
    address=0x7A7686aD451d2865A2246E239B674aeFd4c6c27c
WARN [04-17|09:28:56.607] Please backup your key file!
    path=/home/rogerio/.etherprivate/keystore/UTC--2023-04-17T12-28-54.934614755Z--7a7686ad451d2865a2
WARN [04-17|09:28:56.607] Please remember your password!
Generated account 0x7A7686aD451d2865A2246E239B674aeFd4c6c27c
[rogerio@ryzen-nitro .etherprivate]$
```

Neste ponto é para termos duas contas criadas, que na minha máquina são: `0x2db017E44b03B37755A4b15e14` e `0x7A7686aD451d2865A2246E239B674aeFd4c6c27c`, ambas com a senha `admin12345`.

Verifiquemos no *console* `JavaScript` se elas são listadas:

```
[rogerio@ryzen-nitro .etherprivate]$ geth attach ~/.etherprivate/geth.ipc
Welcome to the Geth JavaScript console!

instance: Geth/v1.11.5-stable-a38f4108/linux-amd64/go1.20.2
```

```
at block: 0 (Wed Dec 31 1969 21:00:00 GMT-0300 (-03))
 datadir: /home/rogerio/.etherprivate
 modules: admin:1.0 debug:1.0 engine:1.0 eth:1.0 ethash:1.0 miner:1.0 net:1.0 rpc:1.0
     txpool:1.0 web3:1.0

To exit, press ctrl-d or type exit
> eth.accounts
["0x2db017e44b03b37755a4b15e14cd799f83de4c13",
    "0x7a7686ad451d2865a2246e239b674aefd4c6c27c"]
>
```

2.5 *Verificando o saldo das carteiras*

Vamos verificar os valores em cada uma das carteiras:

```
[rogerio@ryzen-nitro .etherprivate]$ geth attach ~/.etherprivate/geth.ipc
Welcome to the Geth JavaScript console!

instance: Geth/v1.11.5-stable-a38f4108/linux-amd64/go1.20.2
at block: 0 (Wed Dec 31 1969 21:00:00 GMT-0300 (-03))
 datadir: /home/rogerio/.etherprivate
 modules: admin:1.0 debug:1.0 engine:1.0 eth:1.0 ethash:1.0 miner:1.0 net:1.0 rpc:1.0
     txpool:1.0 web3:1.0

To exit, press ctrl-d or type exit
> eth.accounts
["0x2db017e44b03b37755a4b15e14cd799f83de4c13",
    "0x7a7686ad451d2865a2246e239b674aefd4c6c27c"]
>
> web3.fromWei(eth.getBalance("0x2db017e44b03b37755a4b15e14cd799f83de4c13"), "ether")
0
> web3.fromWei(eth.getBalance("0x7a7686ad451d2865a2246e239b674aefd4c6c27c"), "ether")
0
>
```

Se criamos a primeira conta antes da configuração do *genesis file,* alocamos para ela 300000 então a consulta de valores iniciais terá resultado diferente.

```
> eth.accounts
["0x2db017e44b03b37755a4b15e14cd799f83de4c13",
    "0x7a7686ad451d2865a2246e239b674aefd4c6c27c"]
> eth.getBalance(eth.accounts[0])
300000
> eth.getBalance(eth.accounts[1])
0
> web3.fromWei(eth.getBalance("0x2db017e44b03b37755a4b15e14cd799f83de4c13"), "ether")
3e-13
> web3.fromWei(eth.getBalance("0x7a7686ad451d2865a2246e239b674aefd4c6c27c"), "ether")
0
>
```

2.6 Gerar algum saldo para as carteiras

Se não alocamos nenhuma valor para alguma conta criada antes da inicialização da rede, para acumular algum valor é necessário minerar.

Vamos iniciar o `geth` indicando a carteira que irá receber as recompensas pela mineração utilizando o parâmetro `--miner.etherbase`, utilizei aqui a primeira conta criada `0x2db017e44b03b37755a4b15e14cd79`
Note que é possível colocar uma identificação para sua rede com `--identity "RAGPrivateEtherem"`.

```
[rogerio@ryzen-nitro .etherprivate]$ geth --datadir ~/.etherprivate/ --syncmode full
    --allow-insecure-unlock --networkid 786 --identity "RAGPrivateEtherem" --http
    --http.addr 127.0.0.1 --http.port 8559 --http.api
    "eth,net,web3,personal,engine,admin,debug" --keystore ~/.etherprivate/keystore
    --authrpc.addr localhost --authrpc.port 8551 --authrpc.vhosts localhost
    --authrpc.jwtsecret ~/.etherprivate/geth/jwtsecret --nodiscover --maxpeers 15
    --miner.etherbase=0x2db017e44b03b37755a4b15e14cd799f83de4c13
```

O cliente `geth` irá iniciar normalmente.
Em um console JavaScript vamos verificar o saldo inicial e iniciar a mineração.

```
[rogerio@ryzen-nitro .etherprivate]$ geth attach ~/.etherprivate/geth.ipc
Welcome to the Geth JavaScript console!

instance: Geth/RAGPrivateEtherem/v1.11.5-stable-a38f4108/linux-amd64/go1.20.2
coinbase: 0x2db017e44b03b37755a4b15e14cd799f83de4c13
at block: 0 (Wed Dec 31 1969 21:00:00 GMT-0300 (-03))
 datadir: /home/rogerio/.etherprivate
 modules: admin:1.0 debug:1.0 engine:1.0 eth:1.0 ethash:1.0 miner:1.0 net:1.0 rpc:1.0
     txpool:1.0 web3:1.0

To exit, press ctrl-d or type exit
> eth.accounts

["0x2db017e44b03b37755a4b15e14cd799f83de4c13",
    "0x7a7686ad451d2865a2246e239b674aefd4c6c27c"]
> web3.fromWei(eth.getBalance("0x2db017e44b03b37755a4b15e14cd799f83de4c13"), "ether")
0
> miner.start()

null
> web3.fromWei(eth.getBalance("0x2db017e44b03b37755a4b15e14cd799f83de4c13"), "ether")
2890
>
```

2.7 Transferências entre as carteiras

Vamos enviar 100 *ethers* da primeira para a segunda carteira.

```
[rogerio@ryzen-nitro .etherprivate]$ geth attach ~/.etherprivate/geth.ipc
Welcome to the Geth JavaScript console!

instance: Geth/RAGPrivateEtherem/v1.11.5-stable-a38f4108/linux-amd64/go1.20.2
coinbase: 0x2db017e44b03b37755a4b15e14cd799f83de4c13
at block: 0 (Wed Dec 31 1969 21:00:00 GMT-0300 (-03))
```

```
 datadir: /home/rogerio/.etherprivate
 modules: admin:1.0 debug:1.0 engine:1.0 eth:1.0 ethash:1.0 miner:1.0 net:1.0 rpc:1.0
     txpool:1.0 web3:1.0

To exit, press ctrl-d or type exit
> eth.accounts

["0x2db017e44b03b37755a4b15e14cd799f83de4c13",
    "0x7a7686ad451d2865a2246e239b674aefd4c6c27c"]
> web3.fromWei(eth.getBalance("0x2db017e44b03b37755a4b15e14cd799f83de4c13"), "ether")
0
> miner.start()

null
> web3.fromWei(eth.getBalance("0x2db017e44b03b37755a4b15e14cd799f83de4c13"), "ether")
2890
> miner.stop()

null
> web3.fromWei(eth.getBalance("0x7a7686ad451d2865a2246e239b674aefd4c6c27c"), "ether")
0
> web3.fromWei(eth.getBalance("0x7a7686ad451d2865a2246e239b674aefd4c6c27c"), "ether")
0
> web3.fromWei(eth.getBalance("0x2db017e44b03b37755a4b15e14cd799f83de4c13"), "ether")
4335
> eth.sendTransaction({from: "0x2db017e44b03b37755a4b15e14cd799f83de4c13", to:
    "0x7a7686ad451d2865a2246e239b674aefd4c6c27c", value: 100})
Error: authentication needed: password or unlock
       at web3.js:6365:9(39)
       at send (web3.js:5099:62(29))
       at <eval>:1:20(9)

> personal.unlockAccount(eth.accounts[0])


ReferenceError: personal is not defined
       at <eval>:1:1(0)

> eth.accounts
["0x2db017e44b03b37755a4b15e14cd799f83de4c13",
    "0x7a7686ad451d2865a2246e239b674aefd4c6c27c"]
> eth.sendTransaction({to: '0x7a7686ad451d2865a2246e239b674aefd4c6c27c', from:
    eth.accounts[0], value: 100});
Error: authentication needed: password or unlock
       at web3.js:6365:9(39)
       at send (web3.js:5099:62(29))
       at <eval>:1:20(12)

> eth.sendTransaction({
  to: '0x7a7686ad451d2865a2246e239b674aefd4c6c27c',
  from: eth.accounts[0],
> eth.accounts

["0x2db017e44b03b37755a4b15e14cd799f83de4c13",
    "0x7a7686ad451d2865a2246e239b674aefd4c6c27c"]
```

O erro **"Error: authentication needed: password or unlock"** ocorre por que precisamos autorizar a transação. Em versões anteriores era possível desbloquear as contas via console JavaScript, conforme tentamos no Código acima `personal.unlockAccount(eth.accounts[0])`. Como o `personal` foi depreciado não pode ser mais utilizado.

Na versão corrente é preciso utilizar o `clef` para fazer a autenticação em um console. Em um outro terminal inicie a instância do `clef` com o comando:

```
[rogerio@ryzen-nitro .etherprivate]$ clef --chainid 786 --keystore
    ~/.etherprivate/keystore --configdir ~/.etherprivate/clef --http

WARNING!

Clef is an account management tool. It may, like any software, contain bugs.

Please take care to
- backup your keystore files,
- verify that the keystore(s) can be opened with your password.

Clef is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY;
without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR
PURPOSE. See the GNU General Public License for more details.

Enter 'ok' to proceed:
> ok

INFO [04-17|10:23:09.630] Using CLI as UI-channel
INFO [04-17|10:23:10.050] Loaded 4byte database embeds=268,621 locals=0
    local=./4byte-custom.json
WARN [04-17|10:23:10.050] Failed to open master, rules disabled err="failed stat on
    /home/rogerio/.etherprivate/clef/masterseed.json: stat
    /home/rogerio/.etherprivate/clef/masterseed.json: no such file or directory"
INFO [04-17|10:23:10.050] Starting signer chainid=786
    keystore=/home/rogerio/.etherprivate/keystore light-kdf=false advanced=false
INFO [04-17|10:23:10.052] Smartcard socket file missing, disabling err="stat
    /run/pcscd/pcscd.comm: no such file or directory"
INFO [04-17|10:23:10.052] Audit logs configured file=audit.log
INFO [04-17|10:23:10.053] HTTP endpoint opened url=http://127.0.0.1:8550/
INFO [04-17|10:23:10.053] IPC endpoint opened
    url=/home/rogerio/.etherprivate/clef/clef.ipc

------- Signer info -------
* extapi_version : 6.1.0
* extapi_http : http://127.0.0.1:8550/
* extapi_ipc : /home/rogerio/.etherprivate/clef/clef.ipc
* intapi_version : 7.0.1

------- Available accounts -------
0. 0x2db017E44b03B37755A4b15e14Cd799f83DE4c13 at
    keystore:///home/rogerio/.etherprivate/keystore/UTC--2023-04-17T12-22-11.261468773Z--2db017e44b03
1. 0x7A7686aD451d2865A2246E239B674aeFd4c6c27c at
    keystore:///home/rogerio/.etherprivate/keystore/UTC--2023-04-17T12-28-54.934614755Z--7a7686ad451c
-------- List Account request--------------
A request has been made to list all accounts.
You can select which accounts the caller can see
  [x] 0x2db017E44b03B37755A4b15e14Cd799f83DE4c13
```

```
    URL:
        keystore:///home/rogerio/.etherprivate/keystore/UTC--2023-04-17T12-22-11.261468773Z--2db017e4
  [x] 0x7A7686aD451d2865A2246E239B674aeFd4c6c27c
    URL:
        keystore:///home/rogerio/.etherprivate/keystore/UTC--2023-04-17T12-28-54.934614755Z--7a7686ad
-----------------------------------------
```

E volte ao terminal onde iniciou o `geth` e indique que as autentificações serão via `clef` com o parâmetro `--signer=/home/rogerio/.etherprivate/clef/clef.ipc` passando o caminho dado pelo `clef`.

```
[rogerio@ryzen-nitro .etherprivate]$ geth --datadir ~/.etherprivate/ --syncmode full
    --allow-insecure-unlock --networkid 786 --identity "RAGPrivateEtherem" --http
    --http.addr 127.0.0.1 --http.port 8559 --http.api
    "eth,net,web3,personal,engine,admin,debug" --keystore ~/.etherprivate/keystore
    --authrpc.addr localhost --authrpc.port 8551 --authrpc.vhosts localhost
    --authrpc.jwtsecret ~/.etherprivate/geth/jwtsecret --nodiscover --maxpeers 15
    --miner.etherbase=0x2db017e44b03b37755a4b15e14cd799f83de4c13
    --signer=/home/rogerio/.etherprivate/clef/clef.ipc
```

Cada transação executada no console JavaScript deverá ser autorizada no console do `clef`.

```

> eth.sendTransaction({
  to: '0x7a7686ad451d2865a2246e239b674aefd4c6c27c',
  from: eth.accounts[0],
> eth.accounts

["0x2db017e44b03b37755a4b15e14cd799f83de4c13",
    "0x7a7686ad451d2865a2246e239b674aefd4c6c27c"]
> eth.sendTransaction({
  to: '0x7a7686ad451d2865a2246e239b674aefd4c6c27c',
  from: eth.accounts[0],
  value: 100
});
"0xb579cc595601e4aca546ce4e46bdcded7841bd7f50a0a78c505e839dd039b8b9"
> eth.sendTransaction({
  to: '0x7a7686ad451d2865a2246e239b674aefd4c6c27c',
  from: eth.accounts[0],
> eth.sendTransaction({
  to: '0x7a7686ad451d2865a2246e239b674aefd4c6c27c',
  from: eth.accounts[0],
> eth.getBalance(eth.accounts[1])

0
```

No console do `clef` é possível autorizar e ver a transação assinada com `hash: 0xb579cc595601e4aca546ce4e4` que é o mesmo `id` devolvido no console JavaScript.

```
-------- List Account request--------------
A request has been made to list all accounts.
You can select which accounts the caller can see
  [x] 0x2db017E44b03B37755A4b15e14Cd799f83DE4c13
    URL:
```

```
        keystore:///home/rogerio/.etherprivate/keystore/UTC--2023-04-17T12-22-11.261468773Z--2db017e4
  [x] 0x7A7686aD451d2865A2246E239B674aeFd4c6c27c
    URL:
        keystore:///home/rogerio/.etherprivate/keystore/UTC--2023-04-17T12-28-54.934614755Z--7a7686ad
-------------------------------------------
Request context:
        NA -> ipc -> NA

Additional HTTP header data, provided by the external caller:
        User-Agent: ""
        Origin: ""
Approve? [y/N]:
> y
-------- List Account request--------------
A request has been made to list all accounts.
You can select which accounts the caller can see
  [x] 0x2db017E44b03B37755A4b15e14Cd799f83DE4c13
    URL:
        keystore:///home/rogerio/.etherprivate/keystore/UTC--2023-04-17T12-22-11.261468773Z--2db017e4
  [x] 0x7A7686aD451d2865A2246E239B674aeFd4c6c27c
    URL:
        keystore:///home/rogerio/.etherprivate/keystore/UTC--2023-04-17T12-28-54.934614755Z--7a7686ad
-------------------------------------------
Request context:
        NA -> ipc -> NA

Additional HTTP header data, provided by the external caller:
        User-Agent: ""
        Origin: ""
Approve? [y/N]:
> y
--------- Transaction request-------------
to: 0x7A7686aD451d2865A2246E239B674aeFd4c6c27c
from: 0x2db017E44b03B37755A4b15e14Cd799f83DE4c13 [chksum ok]
value: 100 wei
gas: 0x5208 (21000)
gasprice: 1000000000 wei
nonce: 0x0 (0)
chainid: 0x312

Request context:
        NA -> ipc -> NA

Additional HTTP header data, provided by the external caller:
        User-Agent: ""
        Origin: ""
-------------------------------------------
Approve? [y/N]:
> y
## Account password

Please enter the password for account 0x2db017E44b03B37755A4b15e14Cd799f83DE4c13
>
-----------------------
Transaction signed:
 {
```

```
  "type": "0x0",
  "nonce": "0x0",
  "gasPrice": "0x3b9aca00",
  "maxPriorityFeePerGas": null,
  "maxFeePerGas": null,
  "gas": "0x5208",
  "value": "0x64",
  "input": "0x",
  "v": "0x648",
  "r": "0xe497ab329bf31af61f371e2eb251ca979ec8ba45e099318d44468e8703358418",
  "s": "0x518b4e7e5906e4abc86e486523a8ad290727529f240a201905acc84c5f95417f",
  "to": "0x7a7686ad451d2865a2246e239b674aefd4c6c27c",
  "hash": "0xb579cc595601e4aca546ce4e46bdcded7841bd7f50a0a78c505e839dd039b8b9"
}
```

No *console* `JavaScript` é possível recuperar o recibo da transação:

```
eth.getTransactionReceipt("0xb579cc595601e4aca546ce4e46bdcded7841bd7f50a0a78c505e839dd039b8b9")
{
  blockHash: "0x6756c8d89be5a01ff3d300eab5c50add20a2bca691221c4690fc663336ef2cf8",
  blockNumber: 868,
  contractAddress: null,
  cumulativeGasUsed: 21000,
  effectiveGasPrice: 1000000000,
  from: "0x2db017e44b03b37755a4b15e14cd799f83de4c13",
  gasUsed: 21000,
  logs: [],
  logsBloom:
      "0x00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
  root: "0x9d1d510b5517e7c772e36c67e99d47de6d10c25ba9d24311d39eb85b7db6c071",
  to: "0x7a7686ad451d2865a2246e239b674aefd4c6c27c",
  transactionHash:
      "0xb579cc595601e4aca546ce4e46bdcded7841bd7f50a0a78c505e839dd039b8b9",
  transactionIndex: 0,
  type: "0x0"
}
```

É preciso minerar para a transação ser efetivada:

```
> miner.start()
null
> eth.getBalance(eth.accounts[1])
100
> miner.stop()
null
```

Ao enviar outra quantida para a segunda conta é preciso autorizar a transação no console do `clef` e minerar novamente para ela ser efetivada.

```

> eth.sendTransaction({from: "0x2db017e44b03b37755a4b15e14cd799f83de4c13", to:
    "0x7a7686ad451d2865a2246e239b674aefd4c6c27c", value: 100})
"0xa4cdfb3d4f5fcd98db211bab41eb15b2eace3cd938250faec9d2c4feac242980"
> eth.getBalance(eth.accounts[1])

```

```
100
> miner.start()
null
> eth.getBalance(eth.accounts[1])
200
> miner.stop()
```

Console do `clef`para a segunda transação:

```
-------- List Account request--------------
A request has been made to list all accounts.
You can select which accounts the caller can see
  [x] 0x2db017E44b03B37755A4b15e14Cd799f83DE4c13
    URL:
        keystore:///home/rogerio/.etherprivate/keystore/UTC--2023-04-17T12-22-11.261468773Z--2db017e4
  [x] 0x7A7686aD451d2865A2246E239B674aeFd4c6c27c
    URL:
        keystore:///home/rogerio/.etherprivate/keystore/UTC--2023-04-17T12-28-54.934614755Z--7a7686ad
-------------------------------------------
Request context:
        NA -> ipc -> NA

Additional HTTP header data, provided by the external caller:
        User-Agent: ""
        Origin: ""
Approve? [y/N]:
> y
-------- List Account request--------------
A request has been made to list all accounts.
You can select which accounts the caller can see
  [x] 0x2db017E44b03B37755A4b15e14Cd799f83DE4c13
    URL:
        keystore:///home/rogerio/.etherprivate/keystore/UTC--2023-04-17T12-22-11.261468773Z--2db017e4
  [x] 0x7A7686aD451d2865A2246E239B674aeFd4c6c27c
    URL:
        keystore:///home/rogerio/.etherprivate/keystore/UTC--2023-04-17T12-28-54.934614755Z--7a7686ad
-------------------------------------------
Request context:
        NA -> ipc -> NA

Additional HTTP header data, provided by the external caller:
        User-Agent: ""
        Origin: ""
Approve? [y/N]:
> y
--------- Transaction request-------------
to: 0x7A7686aD451d2865A2246E239B674aeFd4c6c27c
from: 0x2db017E44b03B37755A4b15e14Cd799f83DE4c13 [chksum ok]
value: 100 wei
gas: 0x5208 (21000)
gasprice: 1000000000 wei
nonce: 0x1 (1)
chainid: 0x312

Request context:
        NA -> ipc -> NA
```

```
44
45 Additional HTTP header data, provided by the external caller:
46        User-Agent: ""
47        Origin: ""
48 ------------------------------------------
49 Approve? [y/N]:
50 > y
51 ## Account password
52
53 Please enter the password for account 0x2db017E44b03B37755A4b15e14Cd799f83DE4c13
54 >
55 ----------------------
56 Transaction signed:
57  {
58     "type": "0x0",
59     "nonce": "0x1",
60     "gasPrice": "0x3b9aca00",
61     "maxPriorityFeePerGas": null,
62     "maxFeePerGas": null,
63     "gas": "0x5208",
64     "value": "0x64",
65     "input": "0x",
66     "v": "0x647",
67     "r": "0x33d621389272cdbee73d2c50d91b846b325bb2d7b94f2c32d726c6fa21151f9a",
68     "s": "0x5510f0a5611c8c7640904b5a489422352fae1517eb7307bb6496468cff545726",
69     "to": "0x7a7686ad451d2865a2246e239b674aefd4c6c27c",
70     "hash": "0xa4cdfb3d4f5fcd98db211bab41eb15b2eace3cd938250faec9d2c4feac242980"
71   }
72 -------- List Account request--------------
73 A request has been made to list all accounts.
74 You can select which accounts the caller can see
75   [x] 0x2db017E44b03B37755A4b15e14Cd799f83DE4c13
76     URL:
           keystore:///home/rogerio/.etherprivate/keystore/UTC--2023-04-17T12-22-11.261468773Z--2db017e4
77   [x] 0x7A7686aD451d2865A2246E239B674aeFd4c6c27c
78     URL:
           keystore:///home/rogerio/.etherprivate/keystore/UTC--2023-04-17T12-28-54.934614755Z--7a7686ad
79 ------------------------------------------
80 Request context:
81        NA -> ipc -> NA
82
83 Additional HTTP header data, provided by the external caller:
84        User-Agent: ""
85        Origin: ""
86 Approve? [y/N]:
87 > y
88 -------- List Account request--------------
89 A request has been made to list all accounts.
90 You can select which accounts the caller can see
91   [x] 0x2db017E44b03B37755A4b15e14Cd799f83DE4c13
92     URL:
           keystore:///home/rogerio/.etherprivate/keystore/UTC--2023-04-17T12-22-11.261468773Z--2db017e4
93   [x] 0x7A7686aD451d2865A2246E239B674aeFd4c6c27c
94     URL:
           keystore:///home/rogerio/.etherprivate/keystore/UTC--2023-04-17T12-28-54.934614755Z--7a7686ad
95 ------------------------------------------
```

```
Request context:
        NA -> ipc -> NA

Additional HTTP header data, provided by the external caller:
        User-Agent: ""
        Origin: ""
Approve? [y/N]:
> y
```

A transação `0xa4cdfb3d4f5fcd98db211bab41eb15b2eace3cd938250faec9d2c4feac242980` foi autorizada e podemos verificar o saldo das carteiras:

```
>
    eth.getTransactionReceipt("0xa4cdfb3d4f5fcd98db211bab41eb15b2eace3cd938250faec9d2c4feac242980")
{
  blockHash: "0x89dd3dd5175ca3d85549aeb8387be343ef1cecfd2a96134148f39618c4b587fd",
  blockNumber: 1021,
  contractAddress: null,
  cumulativeGasUsed: 21000,
  effectiveGasPrice: 1000000000,
  from: "0x2db017e44b03b37755a4b15e14cd799f83de4c13",
  gasUsed: 21000,
  logs: [],
  logsBloom:
      "0x000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
  root: "0xfbaf589acffe7f4f701301e31e7549e99e102bfd36145c7f55a38800e3714194",
  to: "0x7a7686ad451d2865a2246e239b674aefd4c6c27c",
  transactionHash:
      "0xa4cdfb3d4f5fcd98db211bab41eb15b2eace3cd938250faec9d2c4feac242980",
  transactionIndex: 0,
  type: "0x0"
}
> web3.fromWei(eth.getBalance("0x2db017e44b03b37755a4b15e14cd799f83de4c13"), "ether")
5484.9999999999999998
> web3.fromWei(eth.getBalance("0x7a7686ad451d2865a2246e239b674aefd4c6c27c"), "ether")
2e-16
>
```

Mais detalhes de como implantar uma rede privada local do *Ethereum* podem ser visto em Private Networks.

2.8 Leitura Recomendada

Leitura Recomendada

Capítulo 12: *Futher Ethereum* (Imran 2018)
Livro: IMRAN BASHIR. Mastering Blockchain : Distributed Ledger Technology, Decentralization, and Smart Contracts Explained, 2nd Edition.

3 Prática: Instalando o `Solidity`

Para os testes com o desenvolvimento de Contratos Inteligentes iremos utilizar a linguagem `Solidity`. O Compilador para a linguagem `Solidity` é o `solc`. O `solc` converte código de alto nível escrito em `Solidity` para *bytecode* da *Ethereum Virtual Machine (EVM)*.

O comando para instalação em distribuições `Ubuntu` ou derivados do `Debian`:

```
$ sudo apt-get install solc
```

Outras distribuições como o `Manjaro Linux`, o pacote `solidity` deve ser instalado:

```
$ pacaur -S solidity
```

Feita a instalação, para verificar a versão instalada execute o comando:

```
$ solc --version
solc, the solidity compiler commandline interface
Version: 0.8.19+commit.7dd6d404.Linux.g++
```

3.1 Compilando um Exemplo

Para verificar o funcionamento e algumas funcionalidades vamos criar um contrato simples, com o nome `Addition.sol` e com o seguinte conteúdo:

```
pragma solidity ^0.8.19;

contract Addition {
   uint8 x;

   function addx(uint8 y, uint8 z ) public {
      x = y + z;
   }
   function retrievex() view public returns (uint8) {
      return x;
   }
}
```

Se a versão do `solidity` na sua máquina for diferente, basta ajustar no arquivo fonte colocando a versão correta.

Para a compilação simples execute:

```
[rag@ryzen-nitro]$ solc Addition.sol
Compiler run successful. No output generated.
[rag@ryzen-nitro]$
```

3.2 Visualizando o bytecode *gerado*

O `solc` tem alguns parâmetros interessantes que nos permite verificar o formato binário do contrato, que é a sequência dos *bytecodes* gerados para a **EVM**:

```
$ solc --bin Addition.sol
Warning: SPDX license identifier not provided in source file. Before publishing,
    consider adding a comment containing "SPDX-License-Identifier: <SPDX-License>" to
    each source file. Use "SPDX-License-Identifier: UNLICENSED" for non-open-source
    code. Please see https://spdx.org for more information.
--> Addition.sol


======= Addition.sol:Addition =======
Binary:
608060405234801561001057600080fd5b506101f680610020600396000f3fe608060405234801561001057
600080fd5b50600436106100365760003560e01c806336718d801461003b578063ac04e0a014610057575b60
0080fd5b610055600480360381019061005091906100f2565b610075565b005b61005f61009e565b60405161
006c9190610141565b60405180910390f35b808261008191906100f2565b6000806101000a81548160ff0219
16908360ff160217905550505056 5b6000806000905490610100a900460ff16905090565b600080fd5b6000
60ff82169050919050565b6100cf816100b9565b81146100da57600080fd5b50565b6000813590506100ec81
6100c6565b92915050565b600080604083850312156101095761010861006100b4565b5b6000610117858286016100
00dd565b9250506020610128858286016100dd565b9150509250929050565b61013b816100b9565b82525050
565b6000602082019050610156600083018461013256 5b92915050565b7f4e487b7100000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
506101a1836100b9565b9250828201905060ff8111156101ba576101b961015c565b5b9291505056fea26469
70667358221220e0ec16eaf684603f4f7c74f327a27e4a1a981dfac0cb258479ffe452abda2e4964736f6c63
430008110033
```

3.3 Estimando a taxa `gas`

Como uma taxa de `gas` é cobrada para cada operação que a **EVM** executa, é uma boa prática estimar o `gas` antes de implantar um contrato em uma rede ativa. O parâmetro `--gas` pode ser utilizado para fazer essa estimativa.

```
$ solc --gas Addition.sol
======= Addition.sol:Addition =======
Gas estimation:
construction:
   147 + 100400 = 100547
external:
   addx(uint8,uint8): infinite
   retrievex(): 2479
```

3.4 Gerando a ABI

A *Application Binary Interface (ABI)* é uma forma padrão de interagir com os contratos, sabermos como os métodos estão disponíveis e quais parâmetros utilizam. Para a gerar a `ABI` do contrato utilize o `solc` com o parâmetro `--abi`.

```
$ solc --abi Addition.sol
======= Addition.sol:Addition =======
Contract JSON ABI
[{"inputs":[{"internalType":"uint8","name":"y","type":"uint8"},{"internalType":"uint8",
"name":"z","type":"uint8"}],"name":"addx","outputs":[],"stateMutability":"nonpayable",
"type":"function"},{"inputs":[],"name":"retrievex","outputs":[{"internalType":"uint8",
```

```
"name":"","type":"uint8"}],"stateMutability":"view","type":"function"}]
```

3.5 *Processo de Compilação Completo*

O processo de compilação completo do contrato `Addition.sol` pode ser feito com o comando:

```
$ solc --bin --abi -o bin Addition.sol
Compiler run successful. Artifact(s) can be found in directory "bin".
```

Se erros ocorrerem serão mostrados no terminal, caso contrário o compilador irá mostrar uma mensagem de sucesso. Com o parâmetro de diretório de saída `-o bin`, serão gerados os arquivos no diretório `bin`:

- **Addition.abi:** Contém a ABI do contrato no formato JSON.
- **Addition.bin:** Contém a representação binária do código do contrato.

O conteúdo de cada um dos arquivos pode ser visualizado:

```
$ cat bin/Addition.bin
608060405234801561001057600080fd5b506101f6806100206000396000f3fe608060405234801561001057
600080fd5b50600436106100365760003560e01c806336718d801461003b578063ac04e0a014610057575b60
0080fd5b61005560048036038101906100509190610f2565b610075565b005b61005f61009e565b60405161
006c9190610141565b60405180910390f35b808261008191906101 8b565b6000806101000a81548160ff0219
16908360ff160217905550505050565b60008060009054906101000a900460ff16905090565b600080fd5b6000
60ff82169050919050565b6100cf816100b9565b81146100da57600080fd5b50565b600081359050610 0ec81
6100c6565b92915050565b6000806040838503121561010957610108 6100b4565b5b600061011785828601 61
00dd565b9250506020610128858286016100dd565b9150509250929050565b61013b816100b9565b82525050
565b600060208201905061015660008301846101 32565b92915050565b7f4e487b71000000000000000000000
000000000000000000000000000000000060005260116004526024 6000fd5b60006101968261 00b9565b91
506101a1836100b9565b925082820190 5060ff8111156101ba576101b961015c565b5b9291505056fea26469
70667358221220e0ec16eaf684603f4f7c74f327a27e4a1a981dfac0cb258479ffe452abda2e4964736f6c63
430008110033

$ cat bin/Addition.abi
[{"inputs":[{"internalType":"uint8","name":"y","type":"uint8"},{"internalType":"uint8",
"name":"z","type":"uint8"}],"name":"addx","outputs":[],"stateMutability":"nonpayable",
"type":"function"},{"inputs":[],"name":"retrievex","outputs":[{"internalType":"uint8",
"name":"","type":"uint8"}],"stateMutability":"view","type":"function"}]
```

3.6 *Visualizando os* Opcodes

Os *opcodes* da instruções geradas para a **EVM** podem ser visualizados compilando-se com o parâmetro `--opcodes`:

```
[rag@ryzen-nitro ]$ solc --opcodes Addition.sol

======= Addition.sol:Addition =======
Opcodes:
PUSH1 0x80 PUSH1 0x40 MSTORE CALLVALUE DUP1 ISZERO PUSH2 0x10 JUMPI PUSH1 0x0 DUP1
    REVERT JUMPDEST POP PUSH2 0x1F6 DUP1 PUSH2 0x20 PUSH1 0x0 CODECOPY PUSH1 0x0
    RETURN INVALID PUSH1 0x80 PUSH1 0x40 MSTORE CALLVALUE DUP1 ISZERO PUSH2 0x10 JUMPI
```

```
      PUSH1 0x0 DUP1 REVERT JUMPDEST POP PUSH1 0x4 CALLDATASIZE LT PUSH2 0x36 JUMPI
      PUSH1 0x0 CALLDATALOAD PUSH1 0xE0 SHR DUP1 PUSH4 0x36718D80 EQ PUSH2 0x3B JUMPI
      DUP1 PUSH4 0xAC04E0A0 EQ PUSH2 0x57 JUMPI JUMPDEST PUSH1 0x0 DUP1 REVERT JUMPDEST
      PUSH2 0x55 PUSH1 0x4 DUP1 CALLDATASIZE SUB DUP2 ADD SWAP1 PUSH2 0x50 SWAP2 SWAP1
      PUSH2 0xF2 JUMP JUMPDEST PUSH2 0x75 JUMP JUMPDEST STOP JUMPDEST PUSH2 0x5F PUSH2
      0x9E JUMP JUMPDEST PUSH1 0x40 MLOAD PUSH2 0x6C SWAP2 SWAP1 PUSH2 0x141 JUMP
      JUMPDEST PUSH1 0x40 MLOAD DUP1 SWAP2 SUB SWAP1 RETURN JUMPDEST DUP1 DUP3 PUSH2
      0x81 SWAP2 SWAP1 PUSH2 0x18B JUMP JUMPDEST PUSH1 0x0 DUP1 PUSH2 0x100 EXP DUP2
      SLOAD DUP2 PUSH1 0xFF MUL NOT AND SWAP1 DUP4 PUSH1 0xFF AND MUL OR SWAP1 SSTORE
      POP POP POP JUMP JUMPDEST PUSH1 0x0 DUP1 PUSH1 0x0 SWAP1 SLOAD SWAP1 PUSH2 0x100
      EXP SWAP1 DIV PUSH1 0xFF AND SWAP1 POP SWAP1 JUMP JUMPDEST PUSH1 0x0 DUP1 REVERT
      JUMPDEST PUSH1 0x0 PUSH1 0xFF DUP3 AND SWAP1 POP SWAP2 SWAP1 POP JUMP JUMPDEST
      PUSH2 0xCF DUP2 PUSH2 0xB9 JUMP JUMPDEST DUP2 EQ PUSH2 0xDA JUMPI PUSH1 0x0 DUP1
      REVERT JUMPDEST POP JUMP JUMPDEST PUSH1 0x0 DUP2 CALLDATALOAD SWAP1 POP PUSH2 0xEC
      DUP2 PUSH2 0xC6 JUMP JUMPDEST SWAP3 SWAP2 POP POP JUMP JUMPDEST PUSH1 0x0 DUP1
      PUSH1 0x40 DUP4 DUP6 SUB SLT ISZERO PUSH2 0x109 JUMPI PUSH2 0x108 PUSH2 0xB4 JUMP
      JUMPDEST JUMPDEST PUSH1 0x0 PUSH2 0x117 DUP6 DUP3 DUP7 ADD PUSH2 0xDD JUMP
      JUMPDEST SWAP3 POP POP PUSH1 0x20 PUSH2 0x128 DUP6 DUP3 DUP7 ADD PUSH2 0xDD JUMP
      JUMPDEST SWAP2 POP POP SWAP3 POP SWAP3 SWAP1 POP JUMP JUMPDEST PUSH2 0x13B DUP2
      PUSH2 0xB9 JUMP JUMPDEST DUP3 MSTORE POP POP JUMP JUMPDEST PUSH1 0x0 PUSH1 0x20
      DUP3 ADD SWAP1 POP PUSH2 0x156 PUSH1 0x0 DUP4 ADD DUP5 PUSH2 0x132 JUMP JUMPDEST
      SWAP3 SWAP2 POP POP JUMP JUMPDEST PUSH32
      0x4E487B7100000000000000000000000000000000000000000000000000000000 PUSH1 0x0
      MSTORE PUSH1 0x11 PUSH1 0x4 MSTORE PUSH1 0x24 PUSH1 0x0 REVERT JUMPDEST PUSH1 0x0
      PUSH2 0x196 DUP3 PUSH2 0xB9 JUMP JUMPDEST SWAP2 POP PUSH2 0x1A1 DUP4 PUSH2 0xB9
      JUMP JUMPDEST SWAP3 POP DUP3 DUP3 ADD SWAP1 POP PUSH1 0xFF DUP2 GT ISZERO PUSH2
      0x1BA JUMPI PUSH2 0x1B9 PUSH2 0x15C JUMP JUMPDEST JUMPDEST SWAP3 SWAP2 POP POP
      JUMP INVALID LOG2 PUSH5 0x6970667358 0x22 SLT KECCAK256 GT 0x5E RETURNDATACOPY
      SLOAD PUSH24 0xA690F575038FE9F1805C21F5FB9C1486E175C2D740F794C0 SDIV DUP5 MSIZE
      PUSH5 0x736F6C6343 STOP ADDMOD SGT STOP CALLER
6 [rag@ryzen-nitro]$
```

Para uma lista completa de parâmetros aceitos pelo `solc` execute no teminal o comando `solc --help`.

3.7 Leitura Recomendada

Leitura Recomendada

Capítulo 14: *Development Tools and Frameworks* (Imran 2018)
Livro: IMRAN BASHIR. Mastering Blockchain : Distributed Ledger Technology, Decentralization, and Smart Contracts Explained, 2nd Edition.

4 Prática: Introdução ao Web3

Nesta prática são apresentadas algumas ferramentas de Desenvolvimento e Frameworks para o desenvolvimento e implantação de Contratos Inteligentes. Apresenta uma introdução ao `Web3`, métodos de desenvolvimento, teste e verificação de contratos inteligentes com `Ganache`, console do cliente `Geth` e Remix IDE. Introduz o `Truffle` *framework*, que também pode ser usado para testar, migrar contratos inteligentes e o Drizzle, para criar *frontends* de `DApps` de maneira mais fácil, com IPFS, para hospedar as páginas web da aplicação.

A proposta é explorarmos a biblioteca `Web3` com o cliente `Geth`, e os métodos de desenvolvimento, teste e verificação de contratos inteligentes com `Ganache`, console do cliente `Geth`. Fazer o *deploy* de contratos inteligentes utilizando o *console* `Geth` e o `Truffle`. O `Truffle` pode ser usado para testar, migrar contratos inteligentes.

4.1 Instalação das Ferramentas

3. Instale as outras ferramentas: `Node.js`, `Ganache` e `Ganache-CLI`, `Truffle`, `Drizzle`, `Embark` e outras ferramentas indicadas no capítulo. O `truffle` utiliza o `nodejs` nas versões `v14-v18` (https://trufflesuite.com/docs/truffle/how-to/install/)

4. Instale o `nvm` para configurar a versão desejada para o `Node.js` e as bibliotecas necessárias.

```
$ sudo apt-get install nvm
$ nvm install 18
```

2. Instale o `Truffle`.

```
npm install -g truffle
```

4.2 Explorando Web3 com Geth

A `Web3` é uma biblioteca `JavaScript` que pode ser usada na comunicação com um Nó *Ethereum* via comunicação `RPC`. `Web3` expõe métodos que o acesso está disponível sobre `RPC`.

A interação com o cliente `Geth` é possível via *Geth JavaScript Console*, que expõe vários métodos de consulta e gerenciamento do *blockchain*.

Todos os comandos que vimos até o momento continuam válidos.

1. Iniciar `clef` em um terminal.

```
[rogerio@ryzen-nitro .etherprivate]$ clef --chainid 786 --keystore
    ~/.etherprivate/keystore --configdir ~/.etherprivate/clef --http
```

2. Iniciar o cliente de execução `geth` com suporte ao `web3`:

```
[rogerio@ryzen-nitro execution]$ geth --datadir ~/.etherprivate/ --syncmode full
    --allow-insecure-unlock --networkid 786 --identity "RAGPrivateEtherem" --http
    --http.addr 127.0.0.1 --http.port 8559 --http.api
    "eth,net,web3,personal,engine,admin,debug" --keystore ~/.etherprivate/keystore
    --authrpc.addr localhost --authrpc.port 8551 --authrpc.vhosts localhost
    --authrpc.jwtsecret ~/.etherprivate/geth/jwtsecret --nodiscover --maxpeers 15
    --miner.etherbase=0x2db017e44b03b37755a4b15e14cd799f83de4c13
    --signer=/home/rogerio/.etherprivate/clef/clef.ipc
```

3. Iniciar um *console* `JavaScript` para a interação com a execução.

```
[rogerio@ryzen-nitro .etherprivate]$ geth attach ~/.etherprivate/geth.ipc
Welcome to the Geth JavaScript console!

instance: Geth/RAGPrivateEtherem/v1.11.5-stable-a38f4108/linux-amd64/go1.20.2
coinbase: 0x2db017e44b03b37755a4b15e14cd799f83de4c13
at block: 0 (Wed Dec 31 1969 21:00:00 GMT-0300 (-03))
 datadir: /home/rogerio/.etherprivate
 modules: admin:1.0 debug:1.0 engine:1.0 eth:1.0 ethash:1.0 miner:1.0 net:1.0 rpc:1.0
     txpool:1.0 web3:1.0

To exit, press ctrl-d or type exit
>
```

Para verificar se os recursos `web3` estão disponíveis:

```
> web3.version
{
  api: "0.20.1",
  ethereum: undefined,
  network: "786",
  node: "Geth/RAGPrivateEtherem/v1.11.5-stable-a38f4108/linux-amd64/go1.20.2",
  whisper: undefined,
  getEthereum: function(callback),
  getNetwork: function(callback),
  getNode: function(callback),
  getWhisper: function(callback)
}
```

4.3 Web3 deployment

- Faremos o *deploy* de um contrato usando o *console*.
- O passo a passo pode ser visto no livro e iremos reproduzir aqui, seguindo a sequência de passos:
  - Executar o cliente de execução `geth`.
  - Criar um script de *deployment*, usando a `ABI` e o *bytecode*, e algum código `JavaScript`.
  - Faremos o *deploy* do contrato via linha de comando pelo *console* `JavaScript`.
  - Interagir com o contrato via um *frontend* web.

4.4 *Web3 deployment: Executar o* Geth client

- Executar o `geth`. [✓]
- Executar o *console* `JavaScript`. [✓]

4.5 *Web3 deployment: Criar um script de* deployment

O exemplo de contrato que iremos compilar e fazer o *deploy* é o `valueChecker`:

```
// SPDX-License-Identifier: Apache-2.0 OR MIT
pragma solidity ^0.8.19;

contract valueChecker {
    uint price=10;
    event valueEvent(bool returnValue);
    function Matcher (uint8 x) public returns (bool) {
        if (x>=price) {
            emit valueEvent(true);
            return true;
        }
    }
}
```

Compile o contrato com o `solc` ou utilizando o `Remix IDE`, gerando o binário e a `ABI`:
geth –http –http.corsdomain="https://remix.ethereum.org" –http.api web3,eth,debug,personal,net –vmdebug –datadir –dev console

```
$ solc --bin --abi -o bin ValueChecker.sol
$ ls
bin deploy.js ValueChecker.sol
$ cd bin
$ ls
valueChecker.abi valueChecker.bin
$ cat valueChecker.bin
6080604052600a600055348015610015576000 80fd5b5061018b8061002560003960 00f3fe60806040523480
1561001057600080fd5b506004361061002b5760003560e01c8063f9d55e2114610030575b600080fd5b6100
4a600480360381019061004591906100f2565b610060565b604051610057919061013a565b60405180910390
f35b600080548260ff16106100ae577f3eb1a229ff7995457774a4bd31ef7b13b6f4491ad1ebb8961af120b8
b4b6239c600160405161009d919061013a565b60405180910390a1600190506100af565b5b919050565b6000
80fd5b600060ff82169050919050565b6100cf816100b9565b81146100da57600080fd5b50565b6000813590
506100ec816100c6565b92915050565b60006020828403121561010857610076100b4565b5b600061011684
8285016100dd565b91505092915050565b60008115159050919050565b610134816101 1f565b82525050565b
60006020820190506101 4f60008301846101 2b565b9291505056fea2646970667358221220 88a7e63726327b
857c0d0a6d073976f05d5073826c629671c857a375db35d51c64736f6c63430008110033

$ cat valueChecker.abi
[{"anonymous":false,"inputs":[{"indexed":false,"internalType":"bool","name":"returnValue",
"type":"bool"}],"name":"valueEvent","type":"event"},{"inputs":[{"internalType":"uint8",
"name":"x","type":"uint8"}],"name":"Matcher","outputs":[{"internalType":"bool",
"name":"","type":"bool"}],"stateMutability":"nonpayable","type":"function"}]
```

Preparação do código do *script JavaScript*:

```
var valuecheckerContract = web3.eth.contract([{ "anonymous": false, "inputs": [{
    "indexed": false, "internalType": "bool", "name": "returnValue", "type": "bool"
    }], "name": "valueEvent", "type": "event" }, { "inputs": [{ "internalType":
    "uint8", "name": "x", "type": "uint8" }], "name": "Matcher", "outputs": [{
    "internalType": "bool", "name": "", "type": "bool" }], "stateMutability":
    "nonpayable", "type": "function" }]);
var valuechecker = valuecheckerContract.new({
    from: web3.eth.accounts[0],
    data:
        '0x6080604052600a600055348015610015576000 80fd5b5061010d806100256000396000f3006080
         6040526004361060 3f576000357c0100000000000000000000000000000000000000000000000000
         0000900463ffffffff 168063f9d55e21146044575b600080fd5b348015604f57600080fd5b5060
         6f6004803603810190 80803560ff16906020019092919050505060 89565b6040518082151515158
         1526020019150506040 5180910390f35b600080548260ff1610151560db577f3eb1a229ff79954577
         74a4bd31ef7b13b6f4491ad1ebb8961af120b8b4b6239c60016040518082151515158152602001 91
         505060405180910390a16001905060dc565b5b9190505600a165627a7a723058209ff756514f1ef4
         6f5650d800506c4eb6be2d8d71c0e2c8b0ca50660fde82c7680029', gas: '4700000'
},
    function (e, contract) {
        console.log(e, contract);
        if (typeof contract.address !== 'undefined') {
            console.log('Contract mined! address: ' + contract.address +
                'transactionHash: ' + contract.transactionHash);
        }
    })
```

4.6 *Web3 deployment: Fazendo o* deploy *pelo* Geth console

Para fazer o *deploy* cole o código `JavaScript` no *console*:

```
> var valuecheckerContract = web3.eth.contract([{ "anonymous": false, "inputs": [{
    "indexed": false, "internalType": "bool", "name": "returnValue", "type": "bool"
    }], "name": "valueEvent", "type": "event" }, { "inputs": [{ "internalType":
    "uint8", "name": "x", "type": "uint8" }], "name": "Matcher", "outputs": [{
    "internalType": "bool", "name": "", "type": "bool" }], "stateMutability":
    "nonpayable", "type": "function" }]);
undefined
> var valuechecker = valuecheckerContract.new({
...... from: web3.eth.accounts[0],
...... data:
    '0x6080604052600a600055348015610015576000 80fd5b5061010d806100256000396000f3006080604052600436106
    gas: '4700000'
...... },
... function (e, contract) {
...... console.log(e, contract);
...... if (typeof contract.address !== 'undefined') {
......... console.log('Contract mined! address: ' + contract.address +
    'transactionHash: ' + contract.transactionHash);
......... }
...... })
Error: insufficient funds for gas * price + value undefined
undefined
>
```

Na execução do *deploy* deu uma mensagem de erro **Error: insufficient funds for gas * price + value undefined**, pois a carteira da conta selecionada não tem saldo suficiente. É necessário minerar para ganhar algum saldo.

```
> miner.start()
null
> miner.stop()
> null
```

Repetindo o processo de *deploy*:

```
> var valuechecker = valuecheckerContract.new({
    from: web3.eth.accounts[0],
    data:
        '0x6080604052600a600055348015610015576000 80fd5b5061010d806100256000396000f3006080604052600436
        gas: '4700000'
},
    function (e, contract) {
        console.log(e, contract);
        if (typeof contract.address !== 'undefined') {
            console.log('Contract mined! address: ' + contract.address +
                'transactionHash: ' + contract.transactionHash);
        }
    })
null [object Object]
undefined
```

Após as confirmações no *console* do `clef`:

```
--------- Transaction request-------------
to: <contact creation>
from: 0x2db017E44b03B37755A4b15e14Cd799f83DE4c13 [chksum ok]
value: 0 wei
gas: 0x47b760 (4700000)
gasprice: 1000000000 wei
nonce: 0x0 (0)
chainid: 0x312
data:
    0x6080604052600a600055348015610015576000 80fd5b5061010d806100256000396000f3006080604052600436106043610603

Request context:
        NA -> ipc -> NA

Additional HTTP header data, provided by the external caller:
        User-Agent: ""
        Origin: ""
--------------------------------------------
Approve? [y/N]:
> y
## Account password

Please enter the password for account 0x2db017E44b03B37755A4b15e14Cd799f83DE4c13
>
-----------------------
Transaction signed:
```

```
{
   "type": "0x0",
   "nonce": "0x0",
   "gasPrice": "0x3b9aca00",
   "maxPriorityFeePerGas": null,
   "maxFeePerGas": null,
   "gas": "0x47b760",
   "value": "0x0",
   "input":
       "0x6080604052600a600055348015610015576000 80fd5b5061010d806100256000396000f3006080604052600436
   "v": "0x648",
   "r": "0x7085d0359b0cdef9c553d3e6f32a0a0a82dc0a376ed8fa18dae76dc93e274171",
   "s": "0x50f8eb26062a35825f0c960856cec0c3d20fab15ed9ab272f9ab85b59bdfedf8",
   "to": null,
   "hash": "0xd3422f91fc4063682fcaed37cc7eb8b7b438cae9f0c9b56596bc49ededaf2081"
}
```

No *terminal* de execução `geth` irá aparecer a mensagem de que o contrato foi submetido.

```
INFO [04-19|17:42:57.418] Submitted contract creation
    hash=0xd3422f91fc4063682fcaed37cc7eb8b7b438cae9f0c9b56596bc49ededaf2081
    from=0x2db017E44b03B37755A4b15e14Cd799f83DE4c13 nonce=0
    contract=0xe0203C7AEE6512789d63b54773dEDaCd84b1d06B value=0
```

Iniciando a mineração o contrato será minerado:

```
> miner.start()
null
> null [object Object]
Contract mined! address: 0xe0203c7aee6512789d63b54773dedacd84b1d06btransactionHash:
    0xd3422f91fc4063682fcaed37cc7eb8b7b438cae9f0c9b56596bc49ededaf2081

> miner.stop()
```

4.7 Web3 deployment: Interagindo com o contrato

Depois de minerado é possível interagir com o contrato via *console* `JavaScript`, pois após o *deployment* através da sua `ABI` o contrato estará disponível no *console*.

```
> valuechecker
valuechecker valuecheckerContract
> valuechecker.
valuechecker.Matcher valuechecker.allEvents
valuechecker._eth valuechecker.constructor
valuechecker.abi valuechecker.transactionHash
valuechecker.address valuechecker.valueEvent
> valuechecker.address
"0xe0203c7aee6512789d63b54773dedacd84b1d06b"
> valuechecker.transactionHash
"0xd3422f91fc4063682fcaed37cc7eb8b7b438cae9f0c9b56596bc49ededaf2081"
```

Percebam o mesmo *address* e *transactionHash* que foram devolvidos no processo de *deploy*. A `ABI` do `valuechecker` está disponível:

```
> valuechecker.abi
[{
    anonymous: false,
    inputs: [{
        indexed: false,
        internalType: "bool",
        name: "returnValue",
        type: "bool"
    }],
    name: "valueEvent",
    type: "event"
}, {
    inputs: [{
        internalType: "uint8",
        name: "x",
        type: "uint8"
    }],
    name: "Matcher",
    outputs: [{
        internalType: "bool",
        name: "",
        type: "bool"
    }],
    stateMutability: "nonpayable",
    type: "function"
}]
>
```

A função `Matcher` pode ser invocada para a verificação de valores.

```
> eth.getBalance(valuechecker.address)
0
> valuechecker.Matcher.call(12)
true
> valuechecker.Matcher.call(10)
true
> valuechecker.Matcher.call(5)
false
>
```

4.8 *Interagir com o contrato via um* frontend *web.*

Vimos que é possível interagir com o cliente de execução `geth` via *POST requests* utilizando a API `JSON RPC` sobre o `HTTP`. Para esse teste utilizaremos o `curl`. Lembrando que a porta utilizando foi a 8559.

A lista de contas podem ser recuperadas, como vimos, com o comando:

```
[rogerio@ryzen-nitro .etherprivate]$ curl -X POST --insecure --header "Content-Type:
    application/json" --data '{"jsonrpc":"2.0","method":"eth_accounts","params":[],
    "id":64}' --location http://localhost:8559
{"jsonrpc":"2.0","id":64,"result":["0x2db017e44b03b37755a4b15e14cd799f83de4c13","0x7a7686ad451d2865a2
[rogerio@ryzen-nitro .etherprivate]$
```

Um objeto `JSON` é retornado com a lista de contas.

No comando `curl`, o parâmetro `--request` é usado para especificar que o comando é uma requisição do tipo `POST` e `--data` é usado para especificar os parâmetros e valores. Finalmente, o `localhost:8559` é usando para indicar o endereço que o `HTTP endpoint` do `geth` está respondendo.

4.9 Utilizando o REMIX IDE

Acesse o REMIX IDE em https://remix.ethereum.org/.

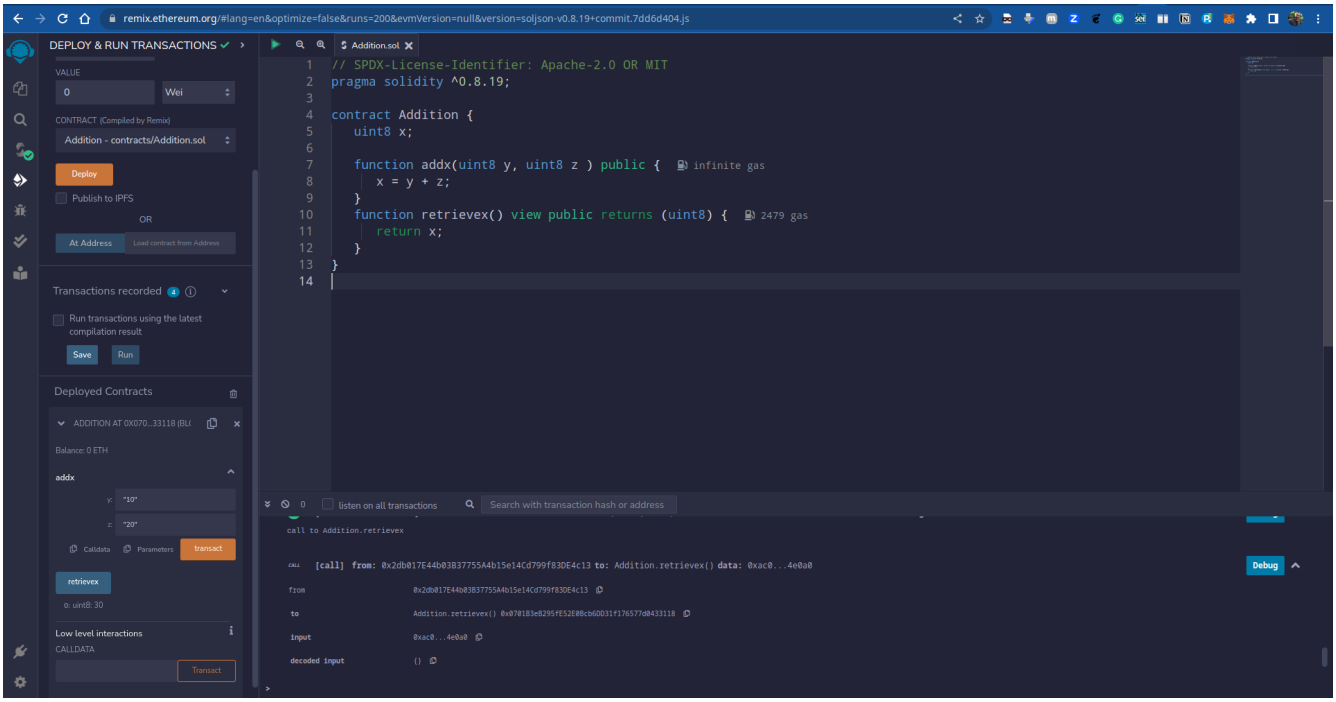


Figura 3: REMIX

4.10 Conectar o REMIX com o MetaMask.

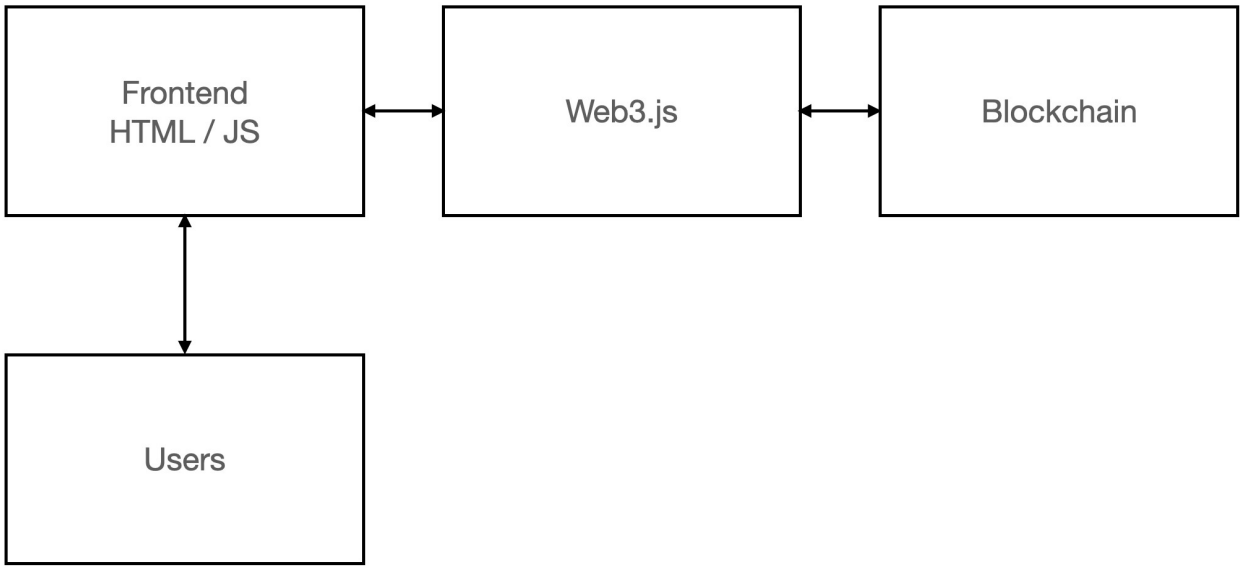
4.11 Executando o `geth` para aceitar conexão com o REMIX

Para aceitar conexões do REMIX IDE inicie o cliente de execução com o comando:

```
[rogerio@ryzen-nitro execution]$ geth --datadir ~/.etherprivate/ --syncmode full
    --allow-insecure-unlock --networkid 786 --identity "RAGPrivateEtherem" --http
    --http.addr 127.0.0.1 --http.port 8559 --http.api
    "eth,net,web3,personal,engine,admin,debug"
    --http.corsdomain="https://remix.ethereum.org" --vmdebug --keystore
    ~/.etherprivate/keystore --authrpc.addr localhost --authrpc.port 8551
    --authrpc.vhosts localhost --authrpc.jwtsecret ~/.etherprivate/geth/jwtsecret
    --nodiscover --maxpeers 15
    --miner.etherbase=0x2db017e44b03b37755a4b15e14cd799f83de4c13
```

4.12 Interagindo com contratos via frontends web

- A interação com *smart contracts* como parte de uma `DApps` é normalmente feito usando uma interface web desenvolvida utilizando `HTML/JS/CSS`. Algumas bibliotecas e *frameworks* como `React`, `Redux`, e `Drizzle`, podem também ser usadas.



4.13 Biblioteca Javascript Web3.js

- Se ainda não instalou a biblioteca `web3.js`, pode instalá-la via `npm` com o comando:

```
$ npm install web3
```

A biblioteca `Web.js` disponibiliza alguns módulos, sendo eles:

- **web3-eth:** Ethereum blockchain e smart contracts.
- **web3-shh:** Protocolo `Whisper` (Comunicação e *broadcast* P2P).
- **web3-bzz:** Protocolo `Swarm`, que fornece armazenamento descentralizado.
- **web3-utils:** Fornece funções úteis para o desenvolvimento de `DApps`.
- Criando um servidor http para testar a app.

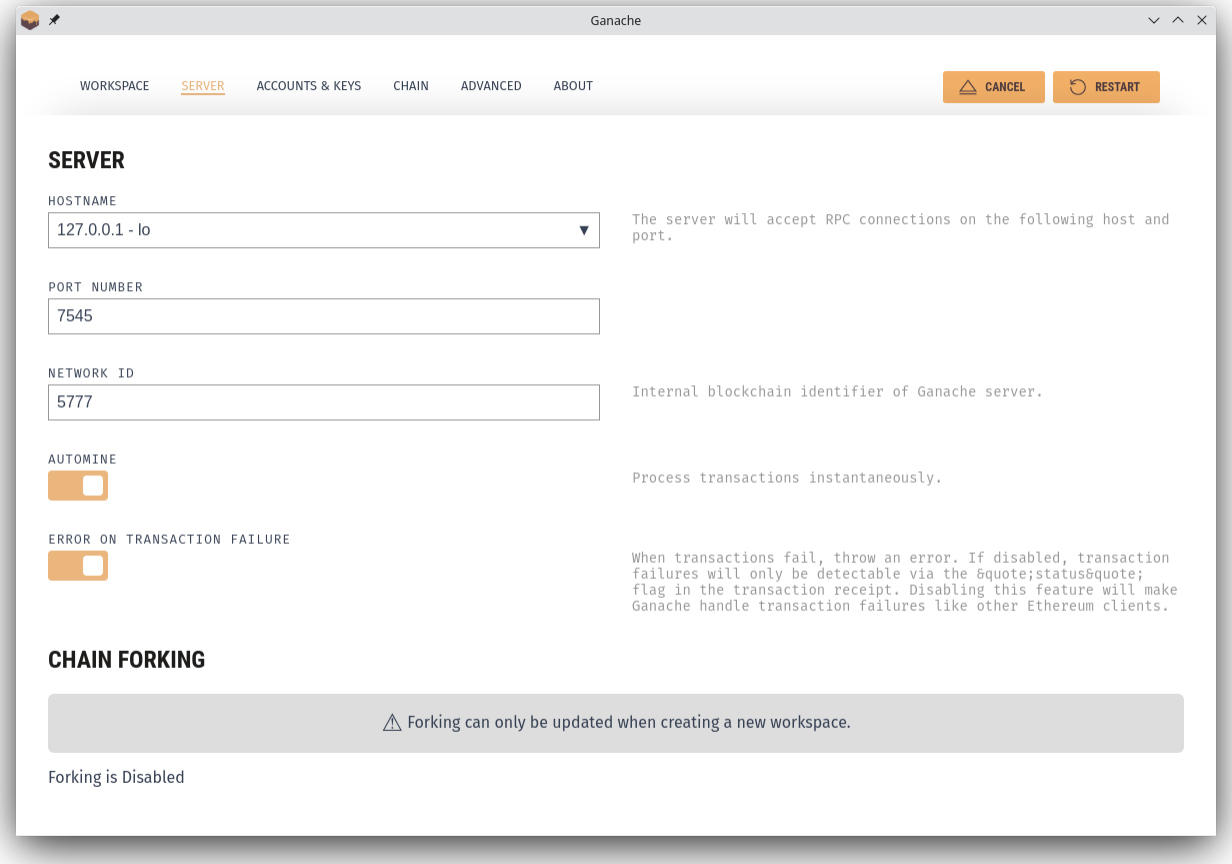
```
# Python 3.x
python3 -m http.server 7777
# If Python version returned above is 2.X
python -m SimpleHTTPServer 7777
```

4.14 Development frameworks

- Installing and initializing Truffle
  - Truffle initialization is perfumed using the Truffle init command, which generates a skeleton structure for a project
- Compiling, testing, and migrating using Truffle
  - Several commands available in Truffle can be used to compile, test and deploy smart contracts

4.15 Configuração do Ganache

- We can use Ganache as a local blockchain to provide the RPC interface.



4.16 Interagindo com um contrato

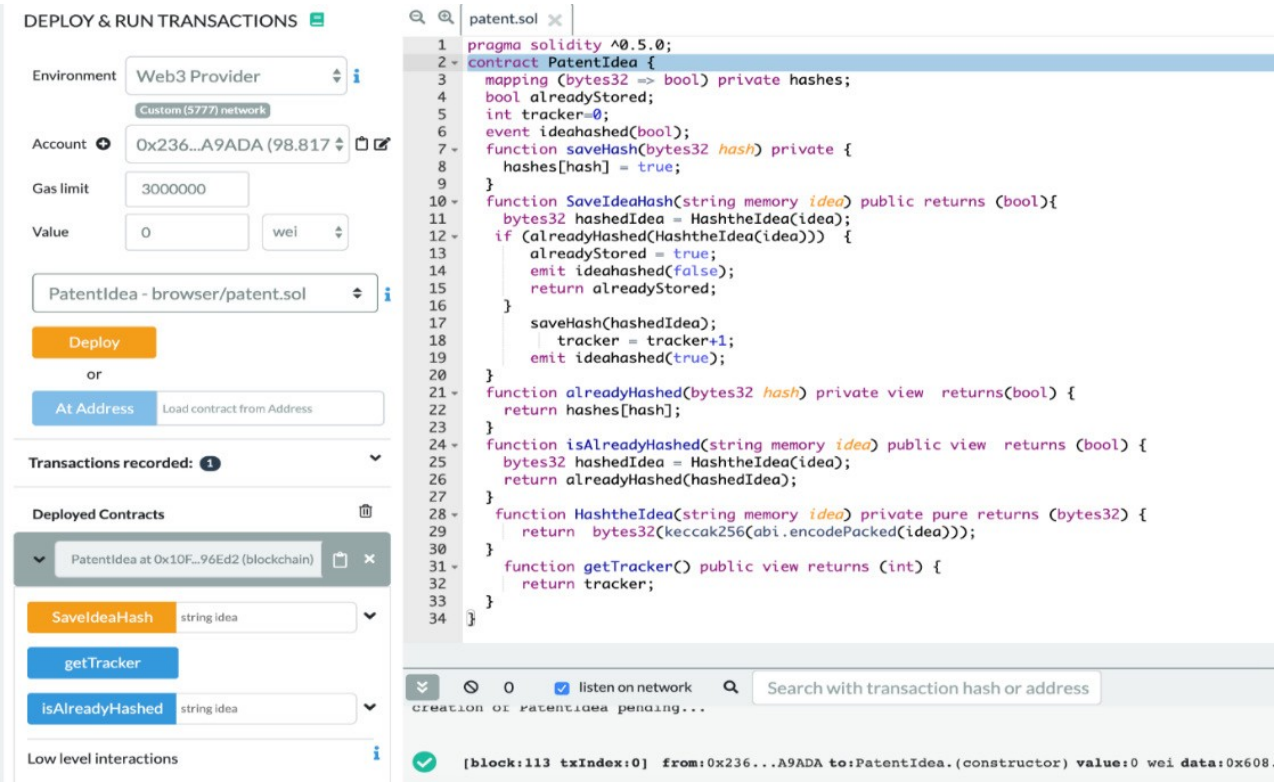
- O console do `Truffle` expõe vários métodos que podem ser usados para interagir com contratos.

```
truffle(development)> MetaCoin.
MetaCoin.__defineGetter__       MetaCoin.__defineSetter__       MetaCoin.__lookupGetter__       MetaCoin.__lookupSetter__
MetaCoin.__proto__              MetaCoin.hasOwnProperty         MetaCoin.isPrototypeOf          MetaCoin.propertyIsEnumerable
MetaCoin.toLocaleString         MetaCoin.valueOf

MetaCoin.apply                  MetaCoin.bind                   MetaCoin.call                   MetaCoin.constructor
MetaCoin.toString

MetaCoin._constructorMethods    MetaCoin._json                  MetaCoin._properties            MetaCoin._property_values
MetaCoin.abi                    MetaCoin.addProp                MetaCoin.address                MetaCoin.arguments
MetaCoin.ast                    MetaCoin.at                     MetaCoin.autoGas                MetaCoin.binary
MetaCoin.bytecode               MetaCoin.caller                 MetaCoin.class_defaults         MetaCoin.clone
MetaCoin.compiler               MetaCoin.configureNetwork       MetaCoin.contractName           MetaCoin.contract_name
MetaCoin.currentProvider        MetaCoin.decodeLogs             MetaCoin.defaults               MetaCoin.deployed
MetaCoin.deployedBinary         MetaCoin.deployedBytecode       MetaCoin.deployedSourceMap      MetaCoin.detectNetwork
MetaCoin.devdoc                 MetaCoin.ens                    MetaCoin.events                 MetaCoin.gasMultiplier
MetaCoin.hasNetwork             MetaCoin.interfaceAdapter       MetaCoin.isDeployed             MetaCoin.legacyAST
MetaCoin.length                 MetaCoin.link                   MetaCoin.links                  MetaCoin.metadata
MetaCoin.name                   MetaCoin.network                MetaCoin.networkType            MetaCoin.network_id
MetaCoin.networks               MetaCoin.new                    MetaCoin.numberFormat           MetaCoin.prototype
MetaCoin.resetAddress           MetaCoin.schemaVersion          MetaCoin.schema_version         MetaCoin.setNetwork
MetaCoin.setNetworkType         MetaCoin.setProvider            MetaCoin.setWallet              MetaCoin.source
MetaCoin.sourceMap              MetaCoin.sourcePath             MetaCoin.timeoutBlocks          MetaCoin.toJSON
MetaCoin.transactionHash        MetaCoin.unlinked_binary        MetaCoin.updatedAt              MetaCoin.updated_at
MetaCoin.userdoc                MetaCoin.web3
```

4.17 Developing a proof of idea project



4.18 Creating the ideap project

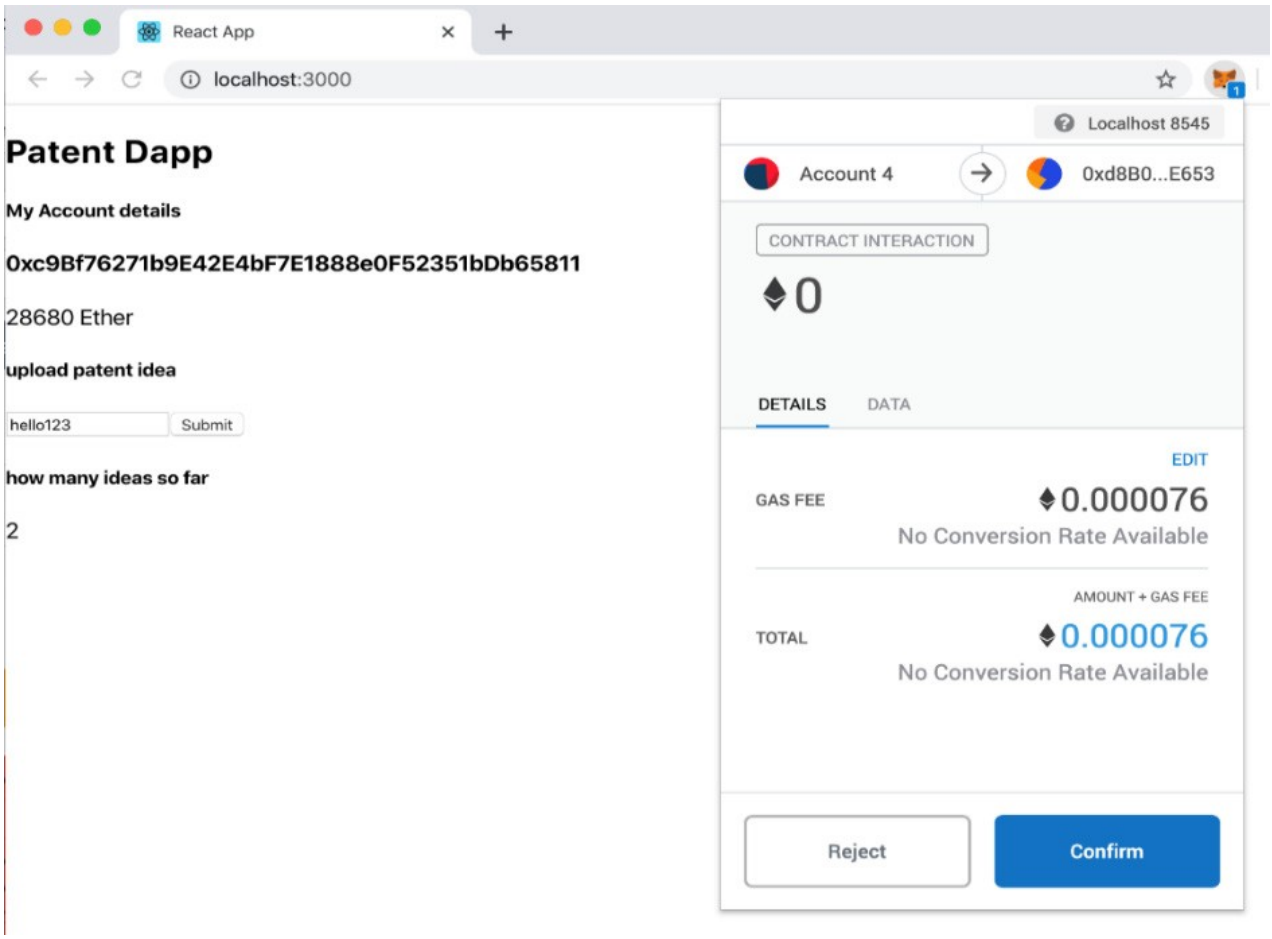
The necessary steps to create a proof of idea project, as detailed in the core Mastering Blockchain book, are as follows:

- Write the ideap smart contract
- Compile and test it in the Remix IDE

- Deploy to Ganache using Truffle
- Deploy to your network of choice (this is optional)
- Build a web frontend using Drizzle
- Run the DApp!

4.19 Patent DApp

This is the resulting interactive frontend of the proof of idea DApp.



4.20 IPFS

- Traditionally, storage is centralized.
- In order to decentralize the entire blockchain ecosystem, storage services should also be decentralized, and serve as decentralized storage layer of the blockchain.
- DApps can benefit from decentralized storage, where backend data can be stored without fear of censorship or centralized control.

4.21 Atividade

- Instalar as ferramentas do Capítulo e implementar os projetinhos de exemplos.
- Utilizando o `Truffle` baixar o exemplo de projeto `MetaCoin` e fazer o `deploy` no `Ganache`.

«»

Utilizando o `Truffle` baixar o exemplo de projeto `MetaCoin` e fazer o `deploy` no `Ganache`.

4.22 Leitura Recomendada

Leitura Recomendada

Capítulo 15: Introducing Web3
Livro: IMRAN BASHIR. Mastering Blockchain : Distributed Ledger Technology, Decentralization, and Smart Contracts Explained, 2nd Edition.

5 Prática: Introdução à Tokenização

5.1 Desenvolvendo um Token

A proposta desta prática é explorarmos o desenvolvimento de *Tokens* e o suporte à Tokenização.

6 *Word Cloud*

Referências

Imran, Bashir. 2018. *Mastering Blockchain : Distributed Ledger Technology, Decentralization, and Smart Contracts Explained, 2nd Edition.* Packt Publishing. https://search.ebscohost.com/login.aspx?direct=true&db=e000xww&AN=1789486&lang=pt-br&site=eds-live&scope=site.