

Tutorial para Instalação e Utilização da Rede Blockchain Ethereum no Ubuntu Linux



*Empresa Brasileira de Pesquisa Agropecuária
Embrapa Informática Agropecuária
Ministério da Agricultura, Pecuária e Abastecimento*

DOCUMENTOS 162

Tutorial para Instalação e Utilização da Rede Blockchain Ethereum no Ubuntu Linux

*Inácio Henrique Yano
Alexandre de Castro
Fabio Cesar da Silva
Geraldo Magela de Almeida Caçado*

Autores

Exemplares desta publicação podem ser adquiridos na:

Embrapa Informática Agropecuária

Av. Dr. André Tosello, 209 - Cidade Universitária
Campinas, SP, Brasil
CEP. 13083-886
Fone: (19) 3211-5700
www.embrapa.br

www.embrapa.br/fale-conosco/sac

Comitê Local de Publicações
da Unidade Responsável

Presidente

Stanley Robson de Medeiros Oliveira

Secretário-Executivo

Carla Cristiane Osawa

Membros

*Adriana Farah Gonzalez; Carla Geovana do
Nascimento Macário; Jayme Garcia Arnal Barbedo;
Kleber Xavier Sampaio de Souza; Luiz Antonio
Falaguasta Barbosa; Magda Cruciol; Paula Regina
Kuser Falcão; Ricardo Augusto Dante; Sônia Ternes*

Suplentes

Goran Nesic

Michel Eduardo Beleza Yamagishi

Supervisão editorial

Kleber Xavier Sampaio de Souza

Revisão de texto

Nadir Pereira Rodrigues

Normalização bibliográfica

Victor Paulo Marques Simão

Projeto gráfico da coleção

Carlos Eduardo Felice Barbeiro

Editoração eletrônica

Felipe Prado Jaconi sob supervisão de Magda Cruciol

Foto da capa

Pixabay

1ª edição

Versão digital (2019)

Todos os direitos reservados.

A reprodução não autorizada desta publicação, no todo ou em parte,
constitui violação dos direitos autorais (Lei nº 9.610).

Dados Internacionais de Catalogação na Publicação (CIP)

Embrapa Informática Agropecuária

Tutorial para Instalação e Utilização da Rede Blockchain Ethereum no Ubuntu Linux
/ Inácio Henrique Yano... [et al.]. - Campinas : Embrapa Informática
Agropecuária, 2019.

PDF (23 p.) : il. - (Documentos / Embrapa Informática Agropecuária, ISSN 1677-
9274; 162).

1. Banco de Dados. 2. Blockchain. 3. Rastreamento. 4. Inteligência Artificial. 5.
Big data. I. Yano, Inácio Henrique. II. Título III. Embrapa Informática Agropecuária.
IV. Série.

CDD (21. ed.) 005.74

Autores

Inácio Henrique Yano

Tecnólogo em Processamento de Dados e Economista, doutor em Engenharia Agrícola, analista da Embrapa Informática Agropecuária, Campinas, SP.

Alexandre de Castro

Físico, doutor em Ciências, pesquisador da Embrapa Informática Agropecuária, Campinas, SP.

Fabio Cesar da Silva

Engenheiro agrônomo e engenheiro florestal, doutor em Solos e Nutrição de Plantas, pesquisador da Embrapa Informática Agropecuária, Campinas, SP

Geraldo Magela de Almeida Cançado

Agrônomo, doutor em Genética e Biologia Molecular, pesquisador da Embrapa Informática Agropecuária, Campinas, SP

Apresentação

Blockchains são bases de registros de transações e de dados distribuídos e compartilhados que têm a função de criar um índice global para todas as transações que ocorrem em um determinado mercado ou cadeia produtiva. Essa tecnologia está por trás de criptomoedas (moedas digitais) como bitcoin, ether, ripple, entre tantas outras, mas seu uso como conduíte transacional vai muito além.

Por conceito, trata-se de uma tecnologia que permite a formação de parcerias colaborativas, e que pode reduzir o custo e a complexidade de transações entre empresas por meio da criação de redes eficientes e altamente seguras. A tecnologia blockchain pode também facilitar o gerenciamento da cadeia de suprimentos.

À medida que blockchain funciona como um banco de dados descentralizado, em que é possível compartilhar informações com diferentes agentes envolvidos na cadeia de produção, a adoção dessa tecnologia confere uma maior segurança às transações permitindo o rastreamento de toda a produção.

Essa tecnologia permite a geração de massas de dados que podem no futuro próximo otimizar ainda mais as cadeias produtivas com base em análise de Big Data por meio da inteligência artificial. A implantação de mecanismos de rastreabilidade via blockchain pode proporcionar inúmeros benefícios e, por isso, merece atenção e incentivo de P&D&I.

Silvia Maria Fonseca Silveira Massruhá

Chefe-geral

Embrapa Informática Agropecuária

Sumário

Introdução	9
Blockchain	9
Simulação de Rastreamento de Caixas de Manga Utilizando Smart Contract	10
Tutorial para Instalação e Utilização da Rede Blockchain Ethereum no Ubuntu Linux	10
Instalação da Máquina Virtual Ethereum no Ubuntu Linux (Ethereum Foundation, 2018)	10
Configurando uma rede blockchain (Mercury Protocol, 2017)	10
Manipulando contas (Externally owned account) na rede blockchain (Chakravarty, 2017)	12
Adicionando novo ponto na rede blockchain (Tam, 2019)	13
Implementando um Smart Contract na rede blockchain	14
Utilização do Smart Contract pelas contas EOAs da rede blockchain	18
Conclusão	23
Referências	23

Introdução

A Embrapa estabelece em seu documento Visão 2014-2034 que:

O avanço nas tecnologias da informação e comunicação (TICs) reduziu as barreiras físicas, políticas e culturais entre as nações. Globalizou o acesso às matérias-primas, aos bens e aos serviços e deu a todas as pessoas o poder para influenciar os rumos do desenvolvimento tecnológico da formatação de bens e serviços. Munida de equipamentos e sensores, sem limites de conexão, a população mundial exerce seu poder de escolha em escala global, consubstanciando a realidade Big Data, em que um grande volume de dados e informações sobre tendências e demandas reflete, entre outros, manifestações de caráter cultural e psicossocial. Investir intensivamente em ferramentas e processos que apoiem previsões sobre as necessidades tecnológicas e sobre a demanda futura por bens e serviços, cada vez mais difusa e dinâmica, é essencial para as organizações de pesquisa e inovação. (Embrapa, 2014, p. 11).

O contexto agropecuário é marcado pela Era do Big Data, com a geração de grandes volumes de dados que necessitam ser organizados, armazenados e processados para a geração de novos conhecimentos (Yano et al., 2018).

O mundo já vive a era Big Data, com a possibilidade de gerar, medir, coletar e armazenar assombrosas quantidades de dados, que são a matéria-prima do conhecimento. Uma vasta gama de tecnologias emergentes ajuda as organizações a extraírem valor desses grandes conjuntos de dados, o que torna possível, por exemplo, inferir padrões de comportamento e de consumo e ajustar o design e a logística de entrega de produtos e de serviços para cada indivíduo, com enormes ganhos de eficiência operacional e econômica. Daqui para o futuro, o setor privado vai usar Big Data para multiplicar o acesso a serviços e bens de consumo. O setor público vai usá-lo para suporte à formulação, melhoria e implementação de políticas públicas em áreas sensíveis tais como medicina, saúde pública, produção de alimentos e meio ambiente. Na agropecuária, a Era Big Data ainda irá impactar o melhoramento genético, a previsão de clima, a agricultura de precisão, o entendimento da dinâmica dos mercados, entre outros aspectos. (Embrapa, 2014, p. 52-53).

No âmbito do agronegócio, a realidade Big Data, associada às novas tecnologias como cadeia de blocos Blockchain, poderá facilitar o registro distribuído de operações de rastreamento de produtos agrícolas e transações de commodities visando à descentralização como medidas de segurança (Yano et al., 2018).

O objetivo deste trabalho é apresentar a implementação de um contrato inteligente em uma rede privada Ethereum Blockchain, de forma a tornar possível o uso desta rede nos mais diversos tipos de aplicação, principalmente nos casos de rastreabilidade; neste caso, foi escolhido como exemplo a possibilidade de rastreamento de caixas de manga.

Blockchain

O Blockchain (ou cadeia de blocos, em português) é um tipo de banco de dados distribuído cujo modelo de armazenamento permite a guarda de registros de modo permanente e inviolável. É mundialmente conhecido por ser a tecnologia sobre a qual se desenvolveu a criptomoeda Bitcoin, sendo sua origem datada de 2008, quando seu autor, sob o pseudônimo de Satoshi Nakamoto, publicou um artigo na Internet (Nakamoto, 2008) sobre a criação de um sistema de pagamento eletrônico descentralizado, seguro e baseado em uma rede do tipo peer-to-peer P2P (Embrapa, 2017).

Simulação de Rastreamento de Caixas de Manga Utilizando Smart Contract

A simulação foi realizada no ambiente de desenvolvimento Remix2, que é uma ferramenta desenvolvida para trabalhar com a linguagem Solidity, sendo de fácil interação com o contrato e com o acompanhamento das variações dos valores das variáveis (Yano et al., 2018).

A plataforma é a Ethereum, cujo ambiente de desenvolvimento Remix permite montar as transações envolvidas via linguagem de programação Solidity. Na plataforma Ethereum (Ethereum Foundation, 2018), os blocos encadeados carregam um determinado conteúdo junto a uma assinatura hash digital, de forma que o bloco seguinte sempre contém a função hash do bloco anterior e seu próprio conteúdo, gerando, assim, sua própria hash. A função hash, ou função de dispersão, é uma fórmula matemática que possibilita o mapeamento de dados de tamanho arbitrário para um conjunto de tamanho fixo. Dessa forma é possível verificar alterações mínimas em um bloco de dados volumoso, examinando a hash gerada. Todas as informações dos blocos serão escritas e gravadas em um livro-razão digital (ledger) e depois de escritas não poderão mais ser apagadas. Eventualmente, se qualquer conteúdo do bloco for alterado, a função hash também será alterada e não mais coincidirá com a assinatura hash. É importante destacar que em uma cadeia padrão de blocos existem nós mineradores constituídos de participantes que verificam se o bloco escrito é válido. Sempre que um participante da cadeia validar um bloco, recebe uma recompensa na forma da moeda digital corrente da plataforma, no caso da Ethereum, o ether. Essa recompensa é o pagamento pelo custo de processamento dos cálculos matemáticos que asseguram que o hash criptográfico do bloco é válido.

Tutorial para Instalação e Utilização da Rede Blockchain Ethereum no Ubuntu Linux

1. Instalação da Máquina Virtual Ethereum no Ubuntu Linux (Ethereum Foundation, 2018)

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

2. Configurando uma rede blockchain (Mercury Protocol, 2017)

2.1. Todo blockchain inicia com o Genesis Block (bloco ZERO), criado a partir de um arquivo Genesis file, conforme exemplo abaixo (arquivo Genesis1.json)

```
{
  "config": {
    "chainId": 987,
    "homesteadBlock": 0,
    "eip155Block": 0,
    "eip158Block": 0
  },
  "difficulty": "0x400",
  "gasLimit": "0x8000000",
  "alloc": {}
}
```

Cujos atributos estão descritos abaixo:

-config

chainId—Este é o identificador da sua cadeia e é usado na proteção contra duplicação.

-difficulty

Isso dita como é difícil minerar um bloco. Definir esse valor como baixo (~ 10–10000) é útil em um blockchain privado, pois permite minerar blocos rapidamente, o que equivale a transações rápidas e muito ETH para testar.

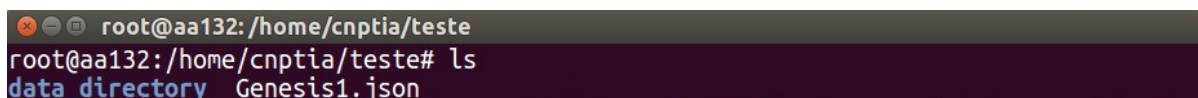
-gasLimit

O número máximo de cálculos que qualquer bloco nessa cadeia pode suportar.

-alloc

Este é o campo que determina quem começa com quantos ether para iniciar o blockchain.

2.2. Crie um subdiretório para os dados da rede blockchain (neste exemplo: `data_directory`, conforme Figura 1):



```
root@aa132: /home/cnptia/teste
root@aa132: /home/cnptia/teste# ls
data_directory Genesis1.json
```

Figura 1. Diretório de rede blockchain.

2.3. Em seguida dê o comando abaixo para criar o primeiro bloco da rede blockchain (Figura 2):

```
geth init /home/cnptia/teste/Genesis1.json --datadir /home/cnptia/teste/data_directory
```

```

root@aa132: /home/cnptia/teste
root@aa132:/home/cnptia/teste# geth init /home/cnptia/teste/Genesis1.json --datadir
/home/cnptia/teste/data_directory
WARN [01-21|16:28:06.519] Sanitizing cache to Go's GC limits      provided=1024 up
dated=1007
INFO [01-21|16:28:06.525] Maximum peer count      ETH=25 LES=0 tot
al=25
INFO [01-21|16:28:06.526] Allocated cache and file handles      database=/home/c
nptia/teste/data_directory/geth/chaindata cache=16 handles=16
INFO [01-21|16:28:06.554] Writing custom genesis block
INFO [01-21|16:28:06.554] Persisted trie from memory database      nodes=0 size=0.0
0B time=3.82µs gcnodes=0 gcsiz=0.00B gctime=0s livenodes=1 livesize=0.00B
INFO [01-21|16:28:06.555] Successfully wrote genesis state      database=chainda
ta hash=d1a12d...4c8725
INFO [01-21|16:28:06.555] Allocated cache and file handles      database=/home/c
nptia/teste/data_directory/geth/lightchaindata cache=16 handles=16
INFO [01-21|16:28:06.564] Writing custom genesis block
INFO [01-21|16:28:06.566] Persisted trie from memory database      nodes=0 size=0.0
0B time=3.362µs gcnodes=0 gcsiz=0.00B gctime=0s livenodes=1 livesize=0.00B
INFO [01-21|16:28:06.566] Successfully wrote genesis state      database=lightch
aindata hash=d1a12d...4c8725
root@aa132:/home/cnptia/teste#

```

Figura 2. Log da criação do primeiro bloco da rede blockchain.

2.4. Inicie a rede blockchain com o comando abaixo (Figura 3):

```
geth --datadir ./data_directory/ --networkid 987 console 2>> eth1.log
```

```

root@aa132: /home/cnptia/teste
root@aa132:/home/cnptia/teste# geth --datadir ./data_directory/ --networkid 987 con
sole 2>> eth1.log
Welcome to the Geth JavaScript console!

instance: Geth/v1.8.21-stable-9dc5d1a9/linux-amd64/go1.10.4
modules: admin:1.0 debug:1.0 eth:1.0 ethash:1.0 miner:1.0 net:1.0 personal:1.0 rpc
:1.0 txpool:1.0 web3:1.0

>

```

Figura 3. Inicialização da rede blockchain.

3. Manipulando contas (Externally owned account na rede blockchain (Chakravarty, 2017))

3.1. O acesso à rede blockchain é feito a partir de uma console Javascript, conforme comando abaixo (Figura 4):

```
geth attach /home/cnptia/teste/data_directory/geth.ipc
```

```

root@aa132: /home/cnptia/teste/data_directory
root@aa132:/home/cnptia/teste/data_directory# geth attach /home/cnptia/teste/dat
a_directory/geth.ipc
WARN [01-22|10:34:24.982] Sanitizing cache to Go's GC limits      provided=1024
updated=1007
Welcome to the Geth JavaScript console!

instance: Geth/v1.8.21-stable-9dc5d1a9/linux-amd64/go1.10.4
modules: admin:1.0 debug:1.0 eth:1.0 ethash:1.0 miner:1.0 net:1.0 personal:1.0
rpc:1.0 txpool:1.0 web3:1.0

>

```

Figura 4. Acesso à console Javascript.

3.2. Por meio da console Javascript, crie uma conta (EOA para interagir com a rede blockchain (Figura 5):

personal.newAccount()

```

root@aa132: /home/cnptia/teste/data_directory
> personal.newAccount()
Passphrase:
Repeat passphrase:
"0x017c8b0da6d8d571467694540c2080b8e3093d4f"
>

```

Figura 5. Criação de nova conta na rede blockchain e retorno do hash EOA.

3.3. Para consultar o saldo da conta EOA (Figura 6):

eth.getBalance("hash EOA")

```

root@aa132: /home/cnptia/teste/data_directory
> eth.getBalance("0x017c8b0da6d8d571467694540c2080b8e3093d4f")
0
>

```

Figura 6. Consultar saldo de conta EOA.

3.4. Por tratar-se de uma rede de teste, pode-se minerar alguns blocos para obter algum saldo em wei que é a menor fração do Ether (criptomoeda do Blockchain Ethereum) (Figura 7):

```

root@aa132: /home/cnptia/teste/data_directory
> miner.start()
null
> eth.getBalance("0x017c8b0da6d8d571467694540c2080b8e3093d4f")
0
> eth.getBalance("0x017c8b0da6d8d571467694540c2080b8e3093d4f")
0
> eth.getBalance("0x017c8b0da6d8d571467694540c2080b8e3093d4f")
5000000000000000000
> ^C
> eth.getBalance("0x017c8b0da6d8d571467694540c2080b8e3093d4f")
10000000000000000000
> miner.stop()
null
>

```

Figura 7. Mineração de blocos para aquisição de wei/Ether.

4. Adicionando novo ponto na rede blockchain (Tam, 2019):

4.1. Instale a Máquina Virtual Ethereum conforme item 1, na máquina que será o segundo nó da rede blockchain (Figura 8).

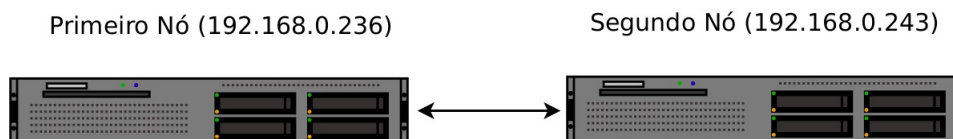


Figura 8. Rede Privada Ethereum formada por dois nós.

4.2. Utilizando o mesmo arquivo de configuração *Genesis1.json* e criando um novo diretório no segundo nó para dados da rede blockchain, dê o comando:

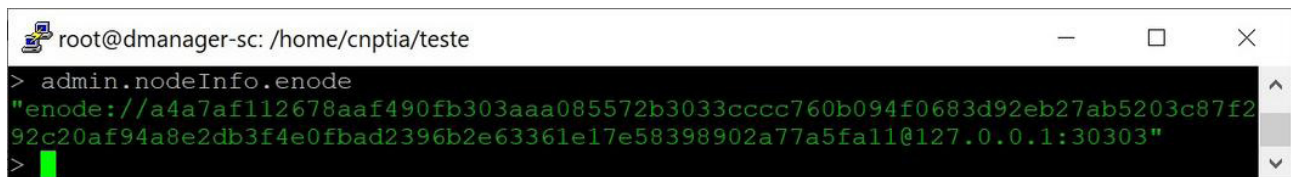
```
geth init /home/cnptia/teste/Genesis1.json --datadir /home/cnptia/teste/data_directory2
```

4.3. Inicie o segundo nó da rede blockchain com o comando:

```
geth --datadir ./data_directory2/ --networkid 987 --port 30304 console 2>> eth2.log
```


4.4. Obtenha os dados do primeiro nó da rede blockchain, utilizando o comando abaixo na console Javascript (Figura 9):

admin.nodeInfo.enode



```

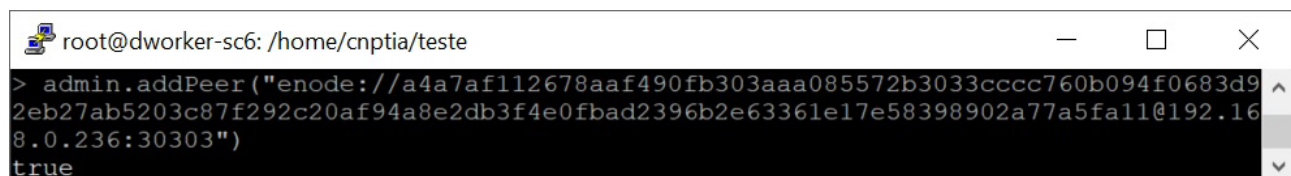
root@dmanager-sc: /home/cnptia/teste
> admin.nodeInfo.enode
"enode://a4a7af112678aaf490fb303aaa085572b3033cccc760b094f0683d92eb27ab5203c87f292c20af94a8e2db3f4e0fbad2396b2e63361e17e58398902a77a5fa11@127.0.0.1:30303"
>

```

Figura 9. Resultado do comando `admin.nodeInfo.enode`.

4.5. Substitua o IP 127.0.0.1 pelo IP do servidor do primeiro nó da rede blockchain (192.168.0.236) no resultado do comando `admin.nodeInfo.enode`, e na console Javascript do segundo nó dê o comando para adicionar o novo nó (Figura 10):

admin.addPeer("enode://a4a7af112678aaf490fb303aaa085572b3033cccc760b094f0683d92eb27ab5203c87f292c20af94a8e2db3f4e0fbad2396b2e63361e17e58398902a77a5fa11@192.168.0.236:30303")



```

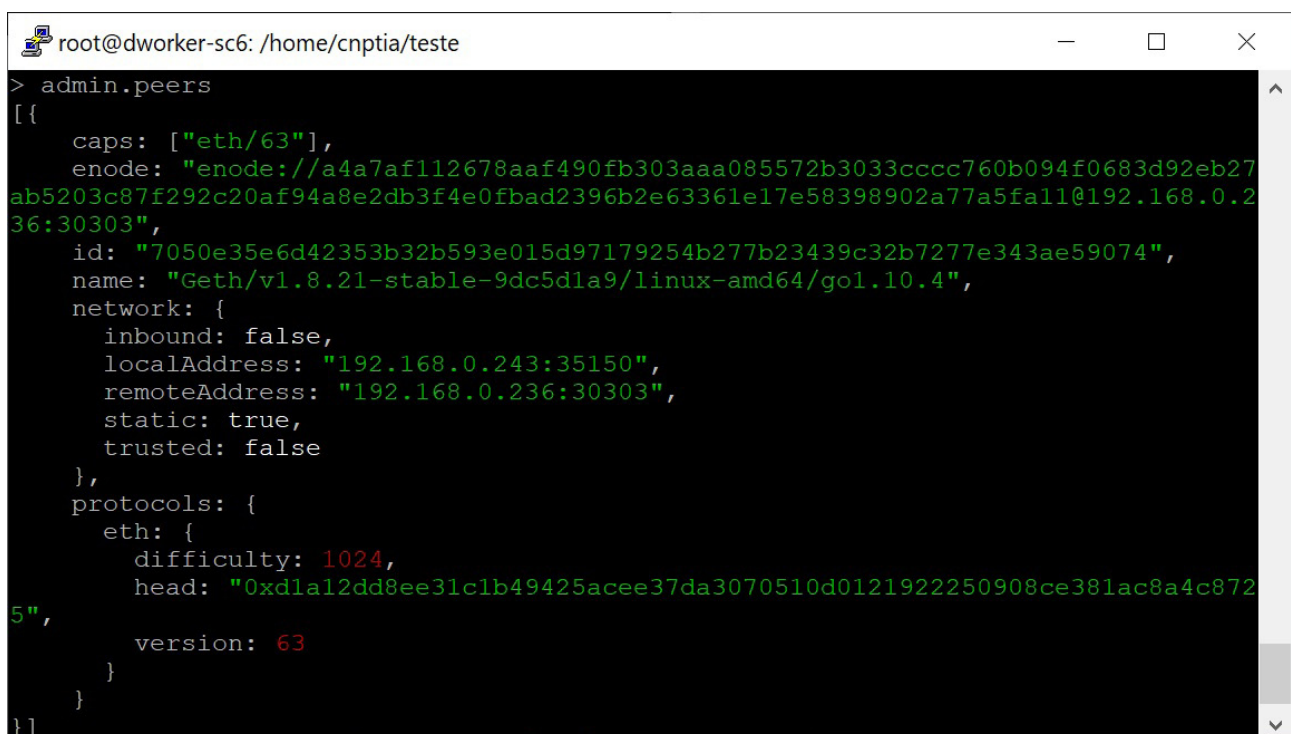
root@dworker-sc6: /home/cnptia/teste
> admin.addPeer("enode://a4a7af112678aaf490fb303aaa085572b3033cccc760b094f0683d92eb27ab5203c87f292c20af94a8e2db3f4e0fbad2396b2e63361e17e58398902a77a5fa11@192.168.0.236:30303")
true

```

Figura 10. Resultado do comando `admin.addPeer`.

4.6. Para confirmar a comunicação entre os nós, dê o comando (Figura 11):

admin.peers



```

root@dworker-sc6: /home/cnptia/teste
> admin.peers
[
  {
    caps: ["eth/63"],
    enode: "enode://a4a7af112678aaf490fb303aaa085572b3033cccc760b094f0683d92eb27ab5203c87f292c20af94a8e2db3f4e0fbad2396b2e63361e17e58398902a77a5fa11@192.168.0.236:30303",
    id: "7050e35e6d42353b32b593e015d97179254b277b23439c32b7277e343ae59074",
    name: "Geth/v1.8.21-stable-9dc5d1a9/linux-amd64/go1.10.4",
    network: {
      inbound: false,
      localAddress: "192.168.0.243:35150",
      remoteAddress: "192.168.0.236:30303",
      static: true,
      trusted: false
    },
    protocols: {
      eth: {
        difficulty: 1024,
        head: "0xd1a12dd8ee31c1b49425acee37da3070510d0121922250908ce381ac8a4c8725",
        version: 63
      }
    }
  }
]

```

Figura 11. Resultado do comando `admin.peers`.

5. Implementando um Smart Contract na rede blockchain

Nesta implementação será utilizado um exemplo simplificado de Smart Contract para rastreamento

de uma caixa de mangas, conforme o código da Lista 1 abaixo:

Lista 1. Linhas de Código do Smart Contract Mango_box123

```
1    pragma solidity ^0.5.0;
2    contract Mango_box123 {
3
4        address payable owner;
5        uint public price = 1 ether;
6        uint constant monetary_unit = 1 ether;
7        mapping (address => uint) public purchasers;
8
9        constructor() public {
10           owner = msg.sender;
11           purchasers[owner] = 1;
12       }
13
14
15       function buyMango(uint amount) public payable returns(bool) {
16           require(msg.value == (amount * price));
17           require(amount == 1);
18           require(purchasers[owner] >= 1);
19
20
21           owner.transfer(msg.value);
22
23           purchasers[owner] -= amount;
24           purchasers[msg.sender] += amount;
25           owner = msg.sender;
26
27           return true;
28       }
29
30
31       function setPrice(uint _price) public returns(uint) {
32           require(owner == msg.sender);
33
34           price = _price * monetary_unit;
35
36           return price;
37       }
38
39       function getAmount(address addr) view public returns (uint) {
40           return purchasers[addr];
41       }
42   }
```

5.1. A Figura 12 apresenta o contrato escrito na linguagem Solidity e compilado no Ambiente de Desenvolvimento Integrado (IDE) Remix¹. Detalhes sobre a programação e utilização do IDE podem ser consultados em Yano et al. (2018).

1 Disponível em: <<https://remix.ethereum.org/>>.

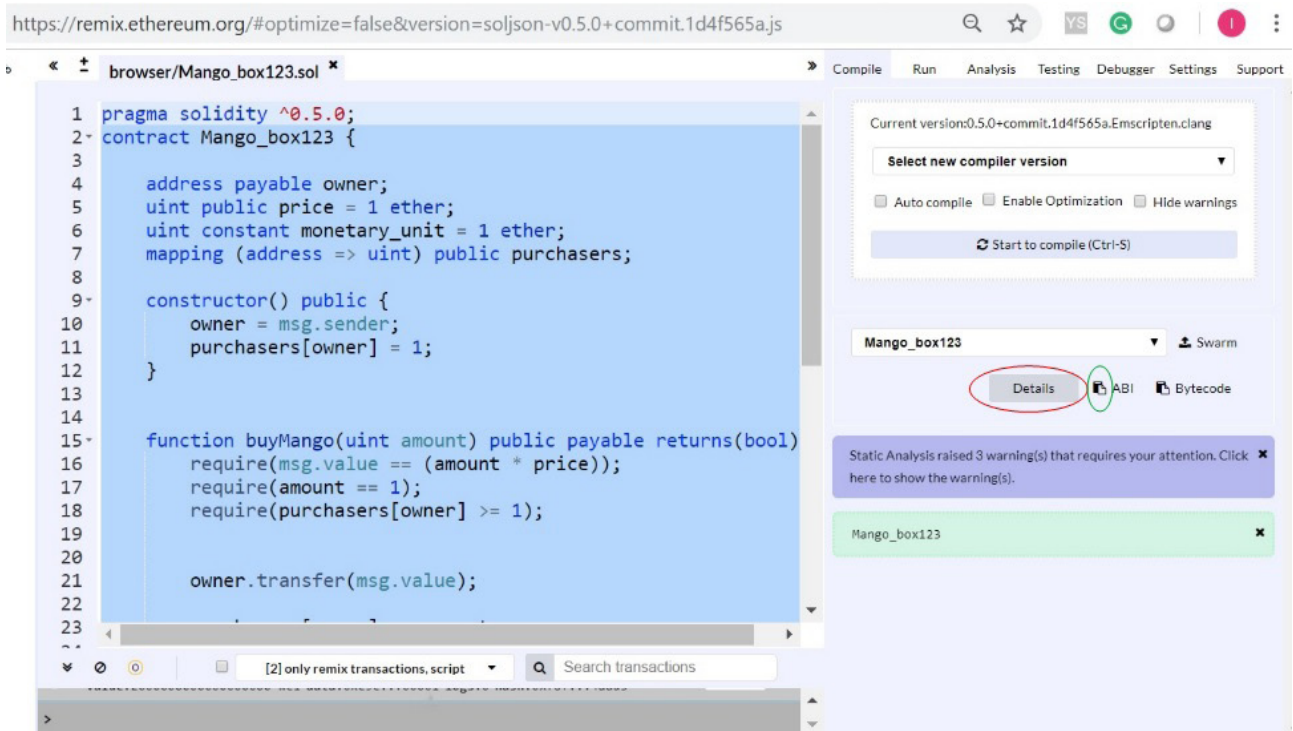


Figura 12. Contrato Mango_box123 no IDE Remix.

5.2. A implementação do contrato na rede blockchain será pelo código WEB3DEPLOY; deve-se clicar em Details destacado em vermelho na Figura 12 e depois no símbolo de cópia destacado em vermelho na Figura 13.

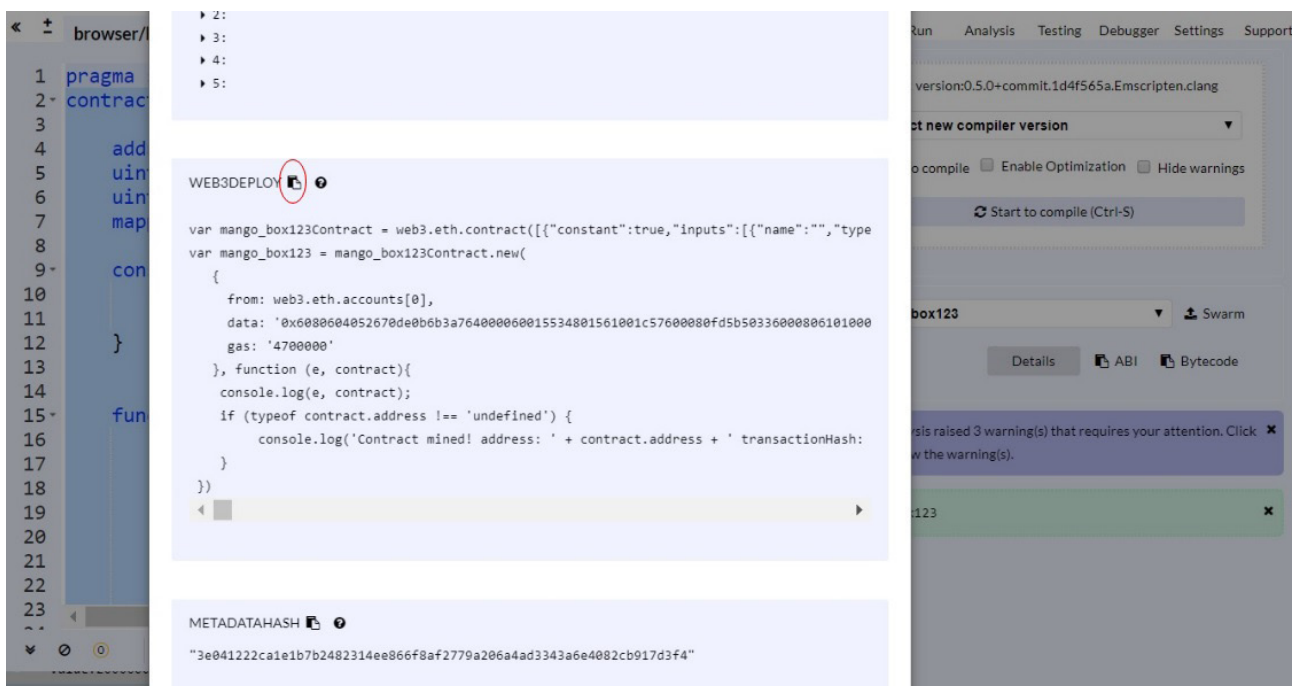


Figura 13. Cópia do código WEB3DEPLOY do contrato Mango_box123.

5.3. Na console Javascript de uma das máquinas da rede blockchain (192.168.0.243), selecione uma conta para fazer a implementação do contrato com o comando (Figura 14).

personal.unlockAccount(hash EOA)

```

root@dworker-sc6: ~/home/cnptia/teste
> personal.unlockAccount(eth.accounts[0])
Unlock account 0x2bdd4836a87a2d39db32aa164586c16c14f46d36
Passphrase:
true
>
    
```

Figura 14. Resultado do comando personal.unlockAccount.

Em seguida cole o contrato no formato WEB3DEPLOY na console Javascript (Figura 15).

```

root@dworker-sc6: ~/home/cnptia/teste
> var mango_box123Contract = web3.eth.contract([{"constant":true,"inputs":[{"name":"","type":"address"}],"name":"purchasera","outputs":[{"name":"","type":"uint256"}],"payable":false,"stateMutability":"view","type":"function"}, {"constant":false,"inputs":[{"name":"price","type":"uint256"}],"name":"setPrice","outputs":[{"name":"","type":"uint256"}],"payable":false,"stateMutability":"nonpayable","type":"function"}, {"constant":true,"inputs":[],"name":"price","outputs":[{"name":"","type":"uint256"}],"payable":false,"stateMutability":"view","type":"function"}, {"constant":false,"inputs":[{"name":"amount","type":"uint256"}],"name":"buyMango","outputs":[{"name":"","type":"bool"}],"payable":true,"stateMutability":"payable","type":"function"}, {"constant":true,"inputs":[{"name":"addr","type":"address"}],"name":"getAmount","outputs":[{"name":"","type":"uint256"}],"payable":false,"stateMutability":"view","type":"function"}, {"inputs":[],"payable":false,"stateMutability":"nonpayable","type":"constructor"}]);
undefined
> var mango_box123 = mango_box123Contract.new(
.....
    from: web3.eth.accounts[0],
    ....
    data: '0x608060405270de0b3a76400006001553480156100c57600080fd5b5033600806101000a90473ffffffffffffffffffffffffffffffffffff021916908373fffffffffffffffffffff
    ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff1602179055050600160026000806000905490610100a900473ffffffffffffffffffffffffffffffffffff1673fffff
    ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff681526020019081526020160020819055061050608e1004260003960003fe50806405260043106100e0d57600037c0100000000000000000000000000000000000000
    009004463fffffffffffff16046339ad02121461007257806391b7f5ed146100d7578063a035bffe14610126578063e9cdc814610151578063f5a7976714610197575b600080fd5b34801561007e57600080fd5b506100c160048
    0360360208101561009557600080fd5b81019080803573fffffffffffffffffffffffffffffffffffff169062001909291905050506101fc565b6040518082815260200191505060405180910390f35b3480156100e3
    57600080fd5b50610110600480360360208110156100fa57600080fd5b8101908080359060200190929190505050610214565b6040518082815260200191505060405180910390f35b34801561013257600080fd5b5061013
    b61026c56b6040518082815260200191505060405180910390f35b61017d6004803603602081101561016757600080fd5b8101908080359060200190929190505050610292565b6040518082815151515152602001915050
    60405180910390f35b3480156101a357600080fd5b6101e6600480360360208110156101ba57600080fd5b81019080803733fffffffffffffffffffffffffffffffffffff16930200190929190505050610491565b
    6040518082815260200191505060405180910390f35b6002602052806000526040600206000915090505481565b60003733fffffffffffffffffffffffffffffffffffff166000809054906101000a900473fffff
    ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff1681526020019081526020016002060008282540192505081905550336000806101000a8154
    5081600260003733fffffffffffffffffffffffffffffffffffff1673fffff1681526020019081526020016002060008282540192505081905550336000806101000a8154
    8173fffff166108fc349081150290604051600060405180830381858888f193505050515801561038c5736000803e3d6000fd5b508160026000806009054906101000a900473ffff
    ffffff1673fffff1681526020019081526020016002054905091905056feaf165627a7a723058203e041222ca1eb7b2482314ee866f8a2779a206a4d334a6e4082cb917d3f4
    0029',
    ....
    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
    
```

Figura 15. Contrato copiado para a console Javascript.

Para finalizar é necessário minerar alguns blocos para gerar o hash do contrato (Figura 16).

```

root@dworker-sc6: ~/home/cnptia/teste
> miner.start()
null
> null [object Object]
Contract mined! address: 0x81f18096662ec26e262f9f679f35708d0910e39f transactionHash: 0x3d9f132efadc3c8646282738a25a86c804e488fe487a3a88264dc1a5cf3f857
> miner.stop()
null
>
    
```

Figura 16. Obtenção da hash do contrato após o processo de mineração de blocos.

5.4. O contrato foi armazenado em bytecode e, assim como as Externally Owned Accounts (EOAs), possui um endereço hash; neste exemplo: “0x81f18096662ec26e262f9f679f35708d0910e39f”. O acesso a esse bycode é feito pela interface Application Binary Interface (ABI), que também deve ser copiada do IDE Remix, destacada em verde na figura 12 e transportada para console Javascript como uma variável (neste exemplo: “abi=”) (Figura 17).

```

root@dworker-sc6: /home/cnptia/teste
> abi=[ { "constant": true, "inputs": [ { "name": "", "type": "address" } ], "name": "purchasers", "outputs": [ { "name": "",
"constant": true, "inputs": [ { "name": "", "type": "uint256" } ], "payable": false, "stateMutability": "view", "type": "function" }, { "constant": false, "inputs": [ { "
name": "_price", "type": "uint256" } ], "name": "setPrice", "outputs": [ { "name": "", "type": "uint256" } ], "payable": false, "stateMutability": "nonpayable", "type": "function" }, { "constant": true, "inputs": [], "name": "price", "outputs": [ { "na
me": "", "type": "uint256" } ], "payable": false, "stateMutability": "view", "type": "function" }, { "constant": false, "inputs": [ { "name": "amount", "type": "uint256" } ], "name": "buyMango", "outputs": [ { "name": "", "type": "bool" } ], "payable":
true, "stateMutability": "payable", "type": "function" }, { "constant": true, "inputs": [ { "name": "addr", "type": "address"
} ], "name": "getAmount", "outputs": [ { "name": "", "type": "uint256" } ], "payable": false, "stateMutability": "view", "typ
e": "function" }, { "inputs": [], "payable": false, "stateMutability": "nonpayable", "type": "constructor" } ]
[[
  constant: true,
  inputs: [{
    name: "",
    type: "address"
  }],
  name: "purchasers",
  outputs: [{
    name: "",
    type: "uint256"
  }],
  payable: false,
  stateMutability: "view",
  type: "function"
}], {
  constant: false,
  inputs: [{
    name: "_price",
    type: "uint256"
  }],
  name: "setPrice",
  outputs: [{
    name: "",
    type: "uint256"
  }],
  payable: false,
  stateMutability: "nonpayable",
  type: "function"
}

```

Figura 17. Implementação da ABI no nó 192.168.0.243.

5.5. A última etapa da implementação do contrato na rede blockchain é a disponibilização do contrato para interação com as contas EOAs, conforme comando abaixo:

```
var contract=web3.eth.contract(ABI code).at("Contract account")
```

A Figura 18 apresenta a disponibilização do contrato no nó 192.168.0.243 para interação com os usuários da rede blockchain. Os hashes dos contratos e das contas EOAs estão disponíveis para toda a rede, mas as variáveis, como eth.account[0], eth.account[1] e abi, são específicas dos nós. Após a disponibilização do contrato já é possível interagir com ele. Na Figura 18 foram consultadas as variáveis de preço e quantidade do primeiro possuidor da caixa de mangas, que foi o usuário que implementou o contrato.

```

root@dworker-sc6: /home/cnptia/teste
>
> var contract=web3.eth.contract(abi).at("0x8f1f8096662ec26e262f9f679f35708d0910e39f")
undefined
>
> contract.price ()
1000000000000000000
>
> contract.getAmount ("0x2bdd4836a87a2d39db32aa164586c16c14f46d36")
1

```

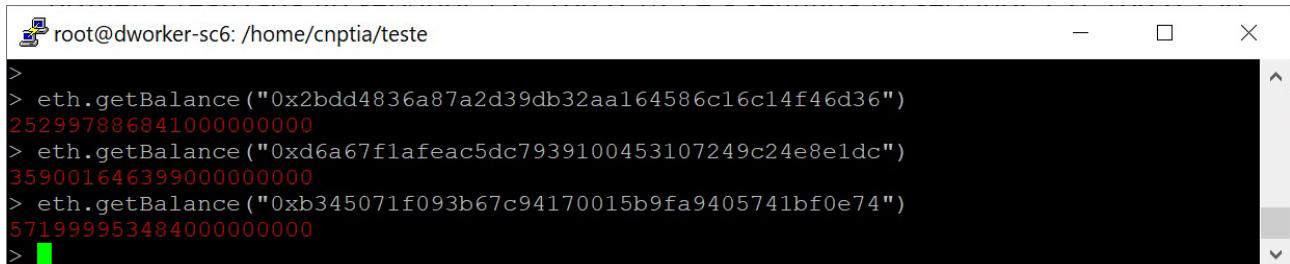
Figura 18. Disponibilização do contrato no nó 192.168.0.243.

6. Utilização do Smart Contract pelas contas EOAs da rede blockchain

Neste exemplo de utilização do Smart Contract serão realizadas duas operações de venda, sendo a primeira realizada no servidor 192.168.0.243 e a segunda no servidor 192.168.0.236.

6.1. Definição do preço de venda

A primeira operação é a definição do preço de venda, a partir da conta detentora da caixa de mangas. A fim de se apurar os saldos ao final de transações, primeiro serão consultados os saldos das contas a serem utilizadas neste exemplo (Figura 19).



```
root@dworker-sc6: /home/cnptia/teste
> eth.getBalance ("0x2bdd4836a87a2d39db32aa164586c16c14f46d36")
25299788684100000000
> eth.getBalance ("0xd6a67f1afeac5dc7939100453107249c24e8e1dc")
35900164639900000000
> eth.getBalance ("0xb345071f093b67c94170015b9fa9405741bf0e74")
57199995348400000000
```

Figura 19. Consulta ao saldo inicial das contas EOAs.

Conforme seção anterior, a conta detentora da caixa de mangas é a `eth.accounts[0]` do servidor 192.168.0.243 ou "0x2bdd4836a87a2d39db32aa164586c16c14f46d36"; a alteração do preço dar-se-á por meio da função `setPrice()`, por se tratar de uma transação que envolve alteração de valores tem uma sintaxe própria, que além dos dados da função, envolve dados da transação (conta de origem/destino, *gas limit*, valores em wei etc.) e tratamento de exceções:

contrato.função(dados da função, dados da transação, tratamento de exceções)

Este tipo de transação gera uma hash e necessita de mineração de blocos para ser concluída; para consultar se a transação está pendente pode-se executar o comando:

eth.pendingTransactions

A Figura 20 mostra todos os passos para a alteração do preço da caixa de mangas de 1 para 2 *Ether*.


```

root@dworker-sc6: /home/cnptia/teste
> contract.price ()
10000000000000000000
>
> contract.setPrice (2, {from:"0x2bdd4836a87a2d39db32aa164586c16c14f46d36",gas:6000000}
, function(error,result){if (!error){console.log(result);} else{ console.log(error);}}
);
0x769b57f88296b3e001653ed67b49bda7881ebe14f937207071d059ac43ae1fe
undefined
>
> eth.pendingTransactions
[[
  blockHash: null,
  blockNumber: null,
  from: "0x2bdd4836a87a2d39db32aa164586c16c14f46d36",
  gas: 6000000,
  gasPrice: 1000000000,
  hash: "0x769b57f88296b3e001653ed67b49bda7881ebe14f937207071d059ac43ae1fe",
  input: "0x91b7f5ed0000000000000000000000000000000000000000000000000000000000000002",
  nonce: 17,
  r: "0xd4135234a48b9e9e80e90acf6ea13e67ec600cd43c429a0c1cf48f14fb99f0c8",
  s: "0x3950b95ccd2d15e50b2108c14be9da754d8cb2ad1720653614769e9c69915e39",
  to: "0x8f1f8096662ec26e262f9f679f35708d0910e39f",
  transactionIndex: 0,
  v: "0x7da",
  value: 0
]]
>
> miner.start ()
null
> eth.pendingTransactions
[]
> miner.stop ()
null
>
> contract.price ()
20000000000000000000
>

```

Figura 20. Alteração de preço da caixa de mangas.

6.2. Realização da compra

Uma vez que o preço de venda esteja definido, a caixa de mangas já pode ser vendida e, neste exemplo, será comprada pela conta `eth.accounts[1]` do servidor `192.168.0.243` ou `"0xd6a67f1afeac5dc7939100453107249c24e8e1dc"`. A compra será realizada pela função `buyMango()`, de forma semelhante à realizada na execução da função `setPrice()`, porém passando o valor em wei da caixa de mangas e utilizando o recurso `sendTransaction()`, que é outra forma de se executar uma função e é usada, principalmente, quando existem muitos parâmetros a serem passados:

contrato.função.sendTransaction(dados da função, dados da transação, tratamento de exceções)

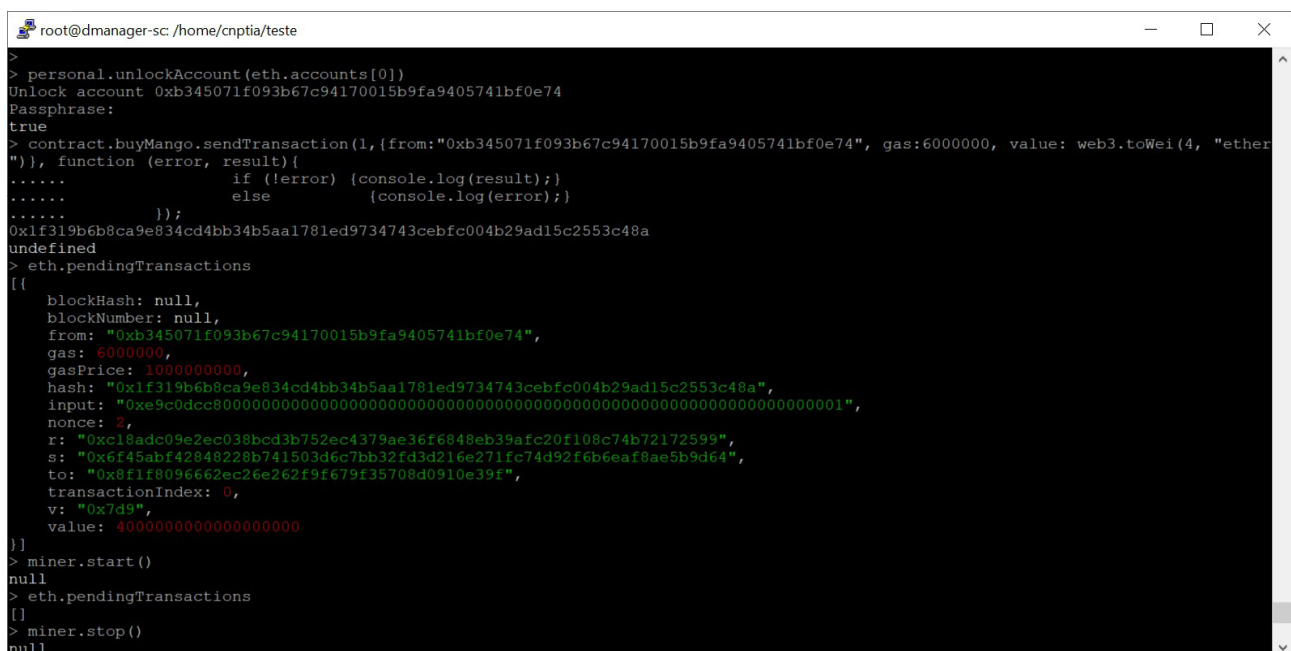
A Figura 21 apresenta os passos para se realizar a compra da caixa de mangas, na qual, primeiro o comprador deve autenticar-se, para depois executar a função de compra e concluir minerando os blocos que a transação exige. Ao final foram feitas consultas para confirmar que o novo detentor da caixa de mangas é a conta `"0xd6a67f1afeac5dc7939100453107249c24e8e1dc"`.


```
abi=[ { "constant": true, "inputs": [ { "name": "", "type": "address" } ], "name": "purchasers", "outputs": [ { "name": "", "type": "uint256" } ], "payable": false, "stateMutability": "view", "type": "function" }, { "constant": false, "inputs": [ { "name": "_price", "type": "uint256" } ], "name": "setPrice", "outputs": [ { "name": "", "type": "uint256" } ], "payable": false, "stateMutability": "nonpayable", "type": "function" }, { "constant": true, "inputs": [], "name": "price", "outputs": [ { "name": "", "type": "uint256" } ], "payable": false, "stateMutability": "view", "type": "function" }, { "constant": false, "inputs": [ { "name": "amount", "type": "uint256" } ], "name": "buyMango", "outputs": [ { "name": "", "type": "bool" } ], "payable": true, "stateMutability": "payable", "type": "function" }, { "constant": true, "inputs": [ { "name": "addr", "type": "address" } ], "name": "getAmount", "outputs": [ { "name": "", "type": "uint256" } ], "payable": false, "stateMutability": "view", "type": "function" }, { "inputs": [], "payable": false, "stateMutability": "nonpayable", "type": "constructor" } ]
```

Assim como é necessário implementar o contrato (procedimento descrito no item 5.5):

```
var contract=web3.eth.contract(abi).at("0x8f1f8096662ec26e262f9f679f35708d0910e39f")
```

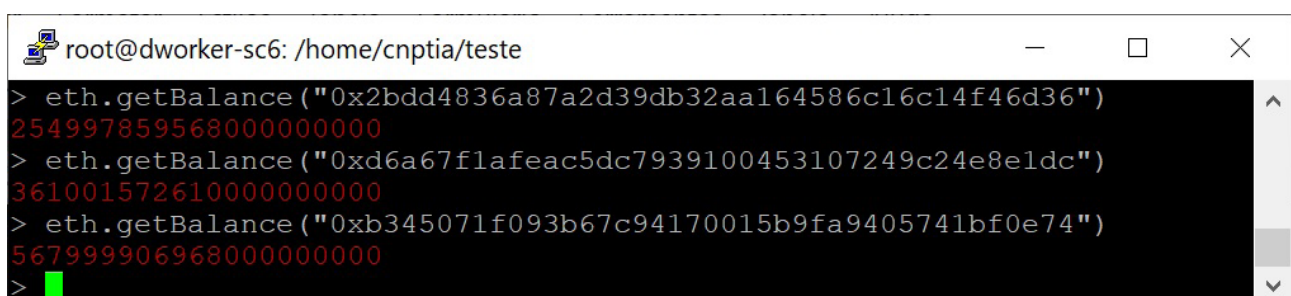
Uma vez implementado o contrato, a operação de compra já pode ser realizada no servidor 192.168.0.236 (Figura 23).



```
root@dmanager-sc: /home/cnptia/teste
> personal.unlockAccount(eth.accounts[0])
Unlock account 0xb345071f093b67c94170015b9fa9405741bf0e74
Passphrase:
true
> contract.buyMango.sendTransaction(1,{from:"0xb345071f093b67c94170015b9fa9405741bf0e74", gas:6000000, value: web3.toWei(4, "ether")}, function (error, result){
.....         if (!error) {console.log(result);}
.....         else {console.log(error);}
.....     });
0xf319b6b8ca9e834cd4bb34b5aa1781ed9734743cebfc004b29ad15c2553c48a
undefined
> eth.pendingTransactions
[[
  {
    blockHash: null,
    blockNumber: null,
    from: "0xb345071f093b67c94170015b9fa9405741bf0e74",
    gas: 6000000,
    gasPrice: 1000000000,
    hash: "0xf319b6b8ca9e834cd4bb34b5aa1781ed9734743cebfc004b29ad15c2553c48a",
    input: "0xe9c0dcc8000000000000000000000000000000000000000000000000000000000000000001",
    nonce: 2,
    r: "0xc18adc09e2ec038bcd3b752ec4379ae36f6848eb39afc20f108c74b72172599",
    s: "0x6f45abf42848228b741503d6c7bb32fd3d216e271fc74d92f6b6eaf8ae5b9d64",
    to: "0x8f1f8096662ec26e262f9f679f35708d0910e39f",
    transactionIndex: 0,
    v: "0x7d9",
    value: 40000000000000000000
  }
]]
> miner.start()
null
> eth.pendingTransactions
[]
> miner.stop()
null
```

Figura 23. Operação de compra realizada a partir do servidor 192.168.0.236.

Ao final foi realizada nova consulta de saldo das contas envolvidas nas transações com o Smart Contract e foi criado o quadro 1, com o resumo das movimentações de wei realizadas (Figura 24).



```
root@dworker-sc6: /home/cnptia/teste
> eth.getBalance("0x2bdd4836a87a2d39db32aa164586c16c14f46d36")
25499785956800000000
> eth.getBalance("0xd6a67f1afeac5dc7939100453107249c24e8e1dc")
36100157261000000000
> eth.getBalance("0xb345071f093b67c94170015b9fa9405741bf0e74")
56799990696800000000
>
```

Figura 24. Saldo final das contas EOAs utilizadas neste exemplo.

Quadro 1. Resumo do saldo das contas EOAs

conta	servidor de origem	saldo inicial	saldo final	diferença
"0x2bdd4836a87a2d39db32aa164586c16c14f46d36"	192.168.0.243	2,52998E+20	2,54998E+20	1,99997E+18
"0xd6a67f1afeac5dc7939100453107249c24e8e1dc"	192.168.0.243	3,59002E+20	3,61002E+20	1,99993E+18
"0xb345071f093b67c94170015b9fa9405741bf0e74"	192.168.0.236	5,72E+20	5,68E+20	-4E+18

Observando o quadro 1 é possível notar que o valor da diferença não corresponde exatamente ao resultado das operações de compra e venda da caixa de mangas. Isto acontece devido ao custo de mineração de blocos para realizar estas operações que, apesar de serem relativamente baixos, interferem no resultado final.

Conclusão

A implantação de mecanismos de rastreabilidade via blockchain pode proporcionar inúmeros benefícios e, por isso, merece atenção e incentivo de P&D&I. À medida que o blockchain funciona como um banco de dados descentralizado, em que é possível compartilhar informações com diferentes agentes envolvidos na cadeia de produção, a adoção dessa tecnologia pode conferir uma maior segurança às transações permitindo o rastreamento de toda a produção do alimento. Por meio das funções criptográficas, a tecnologia blockchain pode assegurar a fidelidade dos dados que circulam nas transações on-line, possibilitando realizar transferência de valores e diversos acordos comerciais. Além disso, a tecnologia blockchain permite a geração de massas de dados que podem no futuro próximo otimizar ainda mais as cadeias produtivas de alimento com base em análise de Big Data por meio da Inteligência Artificial (Yano et al., 2018).

Referências

CHAKRAVARTY, A. **Part-4 of the product manager's guide to the Blockchain**. 2017. Disponível em: <<https://hackernoon.com/heres-how-i-built-a-private-blockchain-network-and-you-can-too-62ca7db556c0>>. Acesso em: 21 jan. 2019.

EMBRAPA. Departamento de Tecnologia da Informação. **Blockchain**. Brasília, DF, 2018. Nota técnica.

EMBRAPA. **Visão 2014-2034: o futuro do desenvolvimento tecnológico da agricultura brasileira**. Brasília, DF, 2014. Disponível em: <<http://ainfo.cnptia.embrapa.br/digital/bitstream/item/108955/1/Documento-Visao-versao-completa.pdf>>. Acesso 20 out. 2018.

ETHEREUM FOUNDATION. **Ethereum command line tools**. 2018. Disponível em: <<https://www.ethereum.org/cli>>. Acesso em: 21 jan. 2019.

MERCURY PROTOCOL. **How To: Create Your Own Private Ethereum Blockchain**. 2017. Disponível em: <<https://medium.com/mercuryprotocol/how-to-create-your-own-private-ethereum-blockchain-dad6af82fc9f>>. Acesso em: 22 jan. 2019.

TAM, K. C. **Two-Node Setup of a Private Ethereum**. 2019. Disponível em: <<https://blockgeeks.com/two-node-setup-of-a-private-ethereum/>>. Acesso em: 22 jan. 2019.

YANO, I. H.; SANTOS, E. H. dos; CASTRO, A. de; BERGIER, I.; SANTOS, P. M.; OLIVEIRA, S. R. de M.; ABREU, U. G. P. de. **Modelo de rastreamento bovino via Smart Contracts com tecnologia Blockchain**. Campinas: Embrapa Informática Agropecuária, 2018. 21 p. (Embrapa Informática Agropecuária. Comunicado técnico, 130).



Informática Agropecuária

