Universidade Federal de Minas Gerais
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Wanderson da Silva Sena

MONOGRAFIA EM SISTEMAS DE INFORMAÇÃO

Utilização de blockchains públicas para fortalecer blockchains privadas

Belo Horizonte

2019/2º semestre
Universidade Federal de Minas Gerais
Instituto de Ciências Exatas
Departamento de Ciência da Computação
Curso de Bacharelado em Sistemas de Informação

Utilização de blockchains públicas para fortalecer blockchains privadas

por

Wanderson da Silva Sena

Monografia em Sistemas de Informação

Apresentado como requisito da disciplina de Sistemas de Informação do Curso de Bacharelado em Sistemas de Informação da UFMG

Prof. Dr. Jeroen van de Graaf
Orientador

Belo Horizonte
2019/2º semestre

Aos meus familiares,
a meus professores,
a meus amigos,
dedico este trabalho

RESUMO

O principal objetivo dessa pesquisa é verificar a possibilidade de realizar interações entre blockchains públicas e blockchains privadas, que estão hospedadas em redes diferentes, com o objetivo de melhorar a robustez e adicionar mais vantagens a blockchain privada. Para alcançar esse objetivo na primeira etapa dessa pesquisa foi criada um protótipo de uma aplicação com a função de ser uma interface para a comunicação entre a rede de testes Ropsten, da Ethereum, e uma blockchain privada hospedada em uma rede local. Esse protótipo provou que é possível realizar essa comunicação, guardar dados simples e executar funções disponíveis em contratos, porém para isso rer possível foi preciso utilizar um serviço gratuito para realizar a conexão com a rede de testes. Dessa forma, a próxima etapa dessa pesquisa irá focar em melhorar o protótipo e desenvolver uma interação mais profunda entre as redes.

Palavras-chave: blockchains, web3, infura, blockchains públicas, blockchains privadas

ABSTRACT

The main objective of this research is to verify the possibility of making interactions between public and private blockchains, which are hosted on different networks, in order to improve the robustness and add more advantages to the private blockchain. To achieve this goal in the first step of this research, a prototype application was created to function as an interface for communication between Ethereum's Ropsten test network and a private blockchain hosted on a local network. This prototype proved that it is possible to perform this communication, save simple data and perform functions available in contracts, but to do this it was necessary to use a free service to make the connection to the test network. Thus, the next step of this research will focus on improving the prototype and developing a deeper interaction between networks.

Keywords: blockchains, web3, infura, public blockchains, private blockchains

LISTA DE FIGURAS

LISTA DE SIGLAS

ABI Application Binary Interface
API Application Programming Interface

SUMÁRIO

1 Introdução

Em outubro de 2009 o whitepaper “Bitcoin: A Peer-to-Peer Electronic Cash System” foi publicado contendo a ideia para uma rede peer-to-peer com a finalidade de revolucionar o modo com que o dinheiro e as transações monetárias são feitas na internet. Para isso, foi necessário a construção de um sistema que pudesse garantir a integridade das transações realizadas pelos usuários, o que implicou na criação da blockchain. A blockchain possui o objetivo de ser um banco de dados distribuído e independente, para isso, cada nó na rede pode conter uma cópia completa da blockchain ou somente a cópia de blocos contendo as suas transações. Isso possibilita que qualquer usuário dentro da rede possa realizar uma consulta aos dados da blockchain, além de aumentar a segurança dela.

A blockchain, juntamente com as medidas de segurança que foram desenvolvidas para garantir a integridade da Bitcoin como moeda e de suas transações, foram responsáveis pelo aumento da popularidade desse projeto. Com o passar do tempo, a popularidade da Bitcoin conquistou um grande número de usuários. Por ser um projeto de código aberto, ela ganhou diversas propostas de melhoria resultando em um código ainda mais robusto e seguro. Devido a natureza do projeto, a blockchain da Bitcoin possui limitações que dificultam o seu uso em situações que não sejam realizar o registro de transações. Por isso, com o passar dos anos, surgiram diversos projetos com o objetivo de ampliar o uso das blockchains, e em especial o projeto Ethereum.

A rede Ethereum foi lançada em julho de 2015 com o objetivo de agregar novas funcionalidades as blockchains sendo o seu principal diferencial é a possibilidade de executar smart contracts dentro da blockchain. Isso significa que é possível escrever aplicações distribuídas que são executadas sem a interferência de terceiros, o que as torna mais seguras. Além disso, um yellow paper, contendo as especificações para a criação de redes nesse padrão, também foi publicado junto com a plataforma. Isso permite que usuários possam criar ledges similares as presentes na rede Ethereum, para uso público ou privado, e manter as principais vantagens das blockchains. Além disso, uma ledge que siga os padrões da Ethereum permite que as aplicações desenvolvidas possam ser publicadas nessa ledge migradas para outras redes.

Com isso em mente, uma empresa pode optar pelo uso de uma blockchain com os padrões da Ethereum para garantir a segurança e consistência da sua aplicação. Porém, o uso de uma blockchain privada também tem desvantagens quando comparado com o uso de uma ledge pública. Isso porque ao criar uma uma blockchain privada, os usuários devem ter em mente que a sua estrutura possivelmente será menos descentralizada quando comparada com a ledge pública. Isso implica em algumas desvantagens para a rede privada, sendo as principais delas a diminuição na robustez da rede e o aumento na vulnerabilidade da blockchain.

Porém, como foi dito anteriormente, aplicações que tenham sido criadas conforme os padrões da Ethereum são compatíveis entre si. Dessa forma, é possível escrever programas que possam ser executados em diferentes redes, independente delas serem públicas ou privadas. Com isso em mente, o objetivo da primeira etapa dessa pesquisa é verificar a possibilidade de realizar interações entre duas blockchains, uma pública e a outra privada, que estejam hospedadas em redes, diferentes através da utilização de um protótipo. Esse protótipo também servirá de base para a segunda etapa dessa pesquisa que terá como foco a utilização dessa interação para agregar mais vantagens para a rede privada.

2 Revisão de literatura

Essa primeira etapa dessa pesquisa buscou desenvolver um protótipo com o objetivo de validar a hipótese de que blockchains públicas podem ser utilizadas por blockchains privadas, em especial, se é possível realizar a interação entre essas blockchains. Devido a especificidade da pesquisa e o fato da blockchain ainda ser uma tecnologia nova, não foram encontradas pesquisas que abordassem um tema similar. Por isso, a literatura foi utilizada principalmente para fornecer apoio teórico sobre o funcionamento das tecnologias e como suporte durante o desenvolvimento do protótipo.

Em relação ao funcionamento das blockchains, os principais matérias utilizados foram livros e artigos que tinham como foco explicar como ela funciona e quais são os primeiros passos que devem ser realizados para que elas possam ser utilizadas. Para o desenvolvimento do protótipo, não foi utilizado nenhum material acadêmico, em compensação foram realizadas consultas a documentação da biblioteca web3.py, utilizada durante o desenvolvimento do protótipo, e fóruns de programação destinados a esclarecer dúvidas sobre a linguagem solidity e o desenvolvimento de contratos inteligentes.

3 Metodologia Utilizada

Ao longo do desenvolvimento dessa pesquisa, o principal objetivo foi adquirir conhecimento teórico sobre o funcionamento das blockchains para que o protótipo pudesse ser desenvolvido. A principal teoria que foi utilizada é que essa interação pode ser feita utilizando um programa externo que serve como interface para a comunicação entre as duas redes. Essa escolha ocorreu devida a dificuldade em utilizar um contrato para realizar essa comunicação.

Dessa forma, os primeiros passos para o desenvolvimento do protótipo envolvem a criação da carteira que será utilizada na rede de testes, a criação da rede local e o desenvolvimento dos contratos que serão utilizados. Para isso, foi utilizado o metamask para criar a carteira, o Ganache para criar a rede e a blockchain local e a Remix para desenvolver os contratos. Com a conta criada, é possível publicar os contratos que foram desenvolvidos em ambas as redes e utilizar a conta associada a carteira para interagir com a rede.

Em sequência, é preciso obter uma URL que será responsável por conectar a aplicação a rede de testes de Ropsten, para isso foi utilizada uma conta gratuita do Infura que permite obter essa URL. Para o desenvolvimento da interface, primeiro é preciso instalar a biblioteca web3.py, ela será responsável por possibilitar a conexão e a interação com as redes e as blockchains. Com ela instalada o próximo passo é iniciar a configuração das redes, juntamente com a conexão é preciso informar as chaves privadas pertencentes as contas que serão utilizadas para enviar as transações e para publicar os contratos.

Em sequência, caso os contratos são publicados na blockchain pública e na privada, caso esses contratos já tenham sido publicados é possível informar ao algoritmo o endereço em que eles se encontram e evitar essa etapa. O próximo passo consiste em recuperar o último bloco presente na blockchain local, com ele o algorítimo verifica qual é a primeira transação desse bloco e a envia para a blockchain pública. Após isso, o programa espera a transação ser minerada para recuperar o recibo da transação, com ele é possível verificar o hash da transação realizada e o número do bloco em que ela foi armazenada. Feito isso, esses dados são enviados para a blockchain local com o objetivo de armazená-los para um uso futuro. As próximas sessões demonstram como esse procedimento foi realizado e como o algoritmo do protótipo foi construído.

3.1 Os Básicos

Inicialmente é preciso realizar alguns pré requisitos antes de começar a elaborar o protótipo, sendo o primeiro criar uma carteira digital ethereum para armazenar o ether que será utilizado na rede de testes. Para o desenvolvimento do protótipo a aplicação responsável pela carteira não é muito importante, o objetivo principal é escolher uma aplicação que permita a exportação da chave privada associada a carteira.

Além disso, também é preciso configurar a blockchain privada, para isso foi utilizada uma blockchain hospedada em uma única máquina local através da utilização do Ganache. Em sequência, é preciso programar os contratos que serão enviados para as blockchains, embora nessa etapa ainda não seja preciso compilar os contratos. As próximas sessões irão demonstrar como esses pré requisitos podem ser alcançados.

3.1.2 MetaMask

Para o desenvolvimento dessa pesquisa, a MetaMask foi escolhida para ser a aplicação responsável pela carteira ethereum. Ela foi escolhida pela praticidade de ser uma extensão para navegadores tendo versões disponíveis para Mozilla Ffirefox, Google Chome, Opera, Android, IOS e outros, além de também permitir a visualização da chave privada e possuir uma interface amigável. Além disso, é possível utilizá-la em conjunto com a Remix, ferramenta utilizada para o desenvolvimento e compilação dos contratos inteligentes, para realizar a publicação dos contratos diretamente pelo navegador de internet.

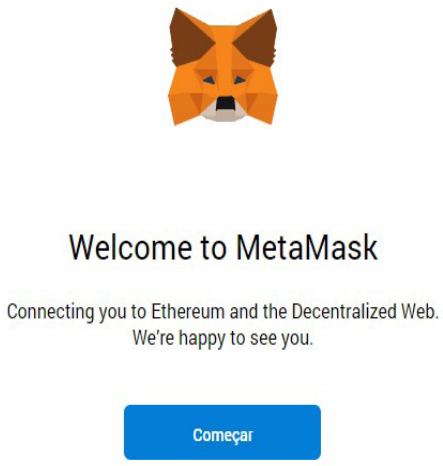

Figura 1 Tela de boas vindas do MetaMask

Embora a extensão do MetaMask possa ser encontrada através do gerenciador de extensões do navegador, é mais confiável obtê-la pelo seu site oficial https://metamask.io/. Após adicionar a extensão será adicionado um ícone de uma raposa ao navegador, dessa forma já é possível criar a carteira que será utilizada para interagir com a rede de teste. Para isso, basta abrir o MetaMask e clicar em começar, em sequencia é possível escolher entre importar uma carteira existente ou criar uma nova.

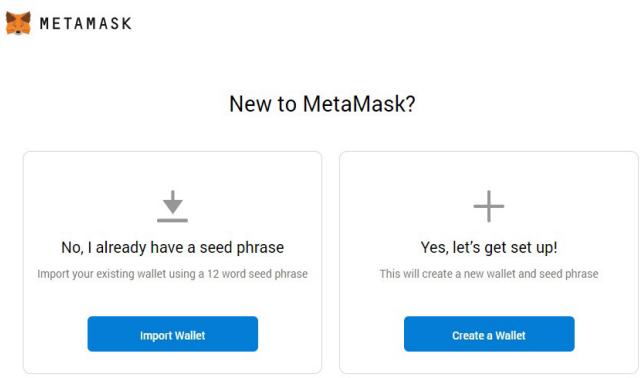

Figura 2 Importação ou criação de uma nova carteira

Ao criar uma nova carteira, o MetaMask irá pedir que o usuário crie uma senha para a conta, esse procedimento permite que outros usuários utilizem o navegador sem conseguir interagir com a carteira. Após definir a senha, o MetaMask irá exibir a frase de backup, essa frase é uma sequência de 12 palavras em inglês pode ser utilizada para importar a carteira em outros clientes ou em outra instância do próprio MetaMask. É importante guardar essa frase em um local seguro pois ela é o principal meio para realizar o backup de uma conta, além disso quem possuir o controle dessa frase também possui o controle sobre a conta e qualquer valor que esteja presente nela.

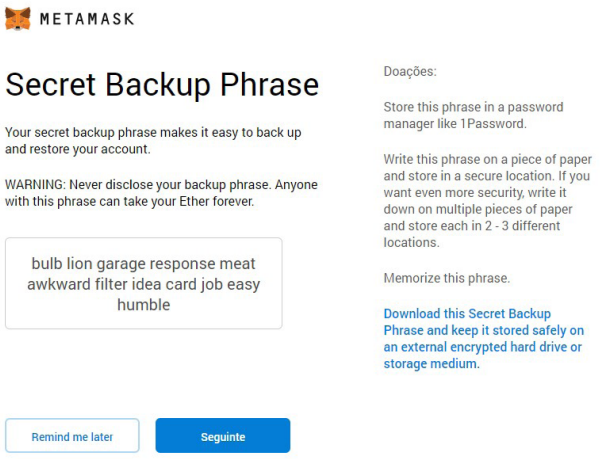

Figura 3 Exemplo de uma frase secreta gerada pelo Metamask

O último passo para a criação da conta é confirmar a frase de backup, para isso é preciso organizar as palavras de forma que elas montem a frase que foi exibida na tela anterior. Feito isso, a carteira é criada e já está pronta para o uso, também já é possível se conectar a rede de testes, realizar transações, publicar e consumir contratos.

3.1.2 Blockchain Local

O próximo pré-requisito é criar e configurar a blockchain local responsável por armazenar os dados e contratos da rede privada. Dessa forma, foi utilizando o Ganache para criação e configuração da blockchain, ele foi escolhido por apresentar uma interface amigável e facilitar o processo de criação e administração da blockchain quando comparado a outros programas como o geth. Além disso, o Ganache cria as contas que servirão como mineradores e usuários juntamente com a blockchain e facilita a interação e obtenção de dados dela. O Ganache pode ser encontrado no site https://www.trufflesuite.com/ganache e está disponível para os sistemas operacionais Windows, Linux e MacOS.

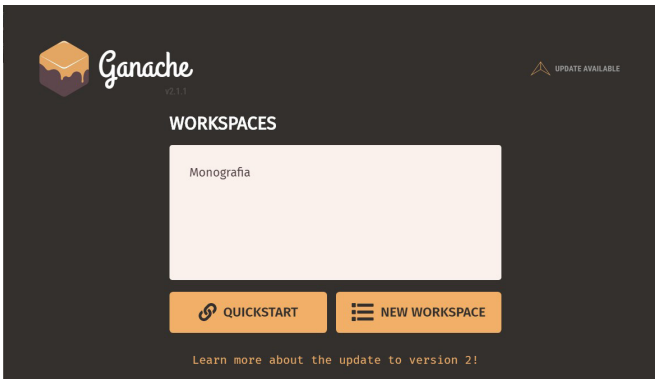

Figura 4 Tela inicial do Ganache

Após realizado o download do arquivo executável basta prosseguir normalmente com a instalação. Com o Ganache instalado, é preciso criar um novo projeto através do botão “New Workspace” presente na tela inicial, esse projeto será responsável por hospedar a blockchain local. Em seguida, é preciso informar um nome para o projeto e escolher a opção “Save Workspace”. Feito isso, o Ganache irá criar a blockchain e os usuários que poderão interagir com ela, em cada usuário é possível ver qual a chave privada associada com aquela conta. Essas chaves serão utilizadas pelo protótipo para realizar a interação com a blockchain local.

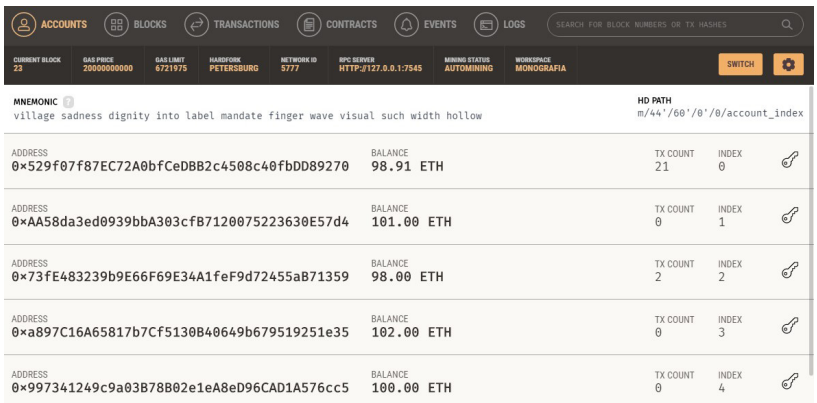

Figura 5 Página do Ganache exibindo as contas locais

3.1.3 Smart Contracts

Uma vez que a blockchain local foi criada, a próxima etapa consiste em preparar os contratos que serão publicados blockchain local e na presente na rede de testes. Esses contratos possuem o objetivo de armazenar dados simples para garantir que o protótipo conseguiu não só se conectar com ambas as redes como também interagir com elas.

Devido a integração com o MetaMask e a possibilidade de programar, compilar, e exportar o código dos contratos, a Remix foi escolhida como ferramenta para criação dos contratos. Ela é uma aplicação web disponível no site https://remix.ethereum.org/ que permite aos usuários não só desenvolver os contratos, em solidity ou vyper, como também compilar e publicá-los em uma blockchain seguindo os padrões da Ethereum, Utilizando o MetaMask também é possível publicar o contrato em uma das redes principais da Ethereum. Para a utilização dos contratos pelo protótipo será preciso exportar o bytecode e a ABI dos contratos, esse processo será detalhado mais a frente junto com o desenvolvimento do protótipo.

O contrato que será publicado na rede de testes foi desenvolvido com o propósito de guardar o código hash de uma única transação. Para isso, é preciso criar uma variável responsável por armazenar esse hash juntamente com dois métodos, um para poder atribuir um valor para a variável e o outro para poder ler o valor presente nela. Como o código hash da transação é recuperado no formato hexadeximal, foi escolhida uma variável do tipo bytes, que consegue armazenar uma sequência de bytes, para receber esse código.

```
1   pragma solidity >=0.5.11;
2
3   contract BackupBlock {
4       bytes transaction;
5
6       function setTransaction(bytes memory y) public {
7           transaction = y;
8       }
9
10      function getTransaction() public view returns (bytes memory) {
11          return transaction;
12      }
13  }
```

Figura 6 Exemplo da implementação do contrato público

O contrato que será utilizado na blockchain privada possui como objetivo guardar um número inteiro, representando o número de um bloco qualquer da blockchain pública, e um código hash de um endereço similar ao contrato anterior. Para isso, ele possui duas váriáveis e três métodos implementados, a primeira variável é do tipo uint o que permite que ela armazena números inteiros, já a segunda é do tipo bytes para armazenar o hash da transação que será enviada para a rede pública. O primeiro método é responsável por enviar ambos os dados que serão armazenados no bloco, o segundo tem o propósito de recuperar o hash do endereço que foi armazenado e o terceiro foi desenvolvido para recuperar o número do bloco que foi armazenado. Uma vez que os contratos foram implementados é possível prossequir para o desenvolvimento do protótipo.

```
pragma solidity >=0.5.11;

contract AddressHistory {
    bytes addressHash;
    uint blockNumber;

    function set(bytes memory x, uint y) public {
        addressHash = x;
        blockNumber = y;
    }

    function getAddress() public view returns (bytes memory) {
        return addressHash;
    }

    function getBlock() public view returns (uint) {
        return blockNumber;
    }

}
```

Figura 7 Exemplo da implementação do contrato privado

3.2 Construção do protótipo

Para a construção do protótipo é preciso utilizar a biblioteca web3 para realizar as conexões com as blockchains. Essa biblioteca possui versões para python, javascript e node através do npm, para esse desenvolvimento foi escolhida a biblioteca implementada em python e, por causa disso, toda a implementação do protótipo também foi realizada em nessa linguagem. Dessa forma, é possível instalar a biblioteca utilizando o pip, um gerenciador de pacotes para python. Dessa forma, basta digitar o comando “pip install web3” no terminal de comando do SO e o pip irá se encarregar de baixar e instalar o pacote e as suas dependências, quando tudo estiver pronto é possível começar desenvolver o protótipo.

De início é preciso realizar a importação das bibliotecas que serão utilizadas, sendo elas a web3 e a json. Com isso a próxima etapa é configurar a conexão com ambas as redes, para isso é preciso obter a chave privada da carteira criada no MetaMask e de uma das contas que foi criada pelo Ganache. Para recuperar a chave do Ganache basta clicar no ícone de chave que fica ao lado das contas que foram criadas. Feito isso, o Ganache irá mostrar o endereço da conta e a chave privada dela.

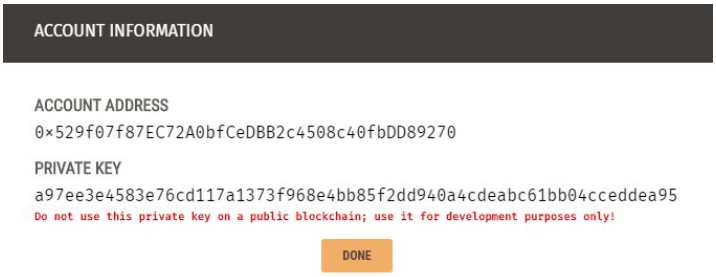

Figura 8 Detalhes de uma conta do Ganache

A próxima etapa é obter a chave do MetaMask e para isso é preciso abri-lo, entrar no menu, clicar em “detalhes” e escolher a opção “exportar chave privada” como pode ser visto nas imagens abaixo. Feito isso, o MetaMask irá pedir a senha da conta antes de exibir a chave, após informar a senha a chave privada será exibida na tela é poderá ser copiada para utilização no protótipo.

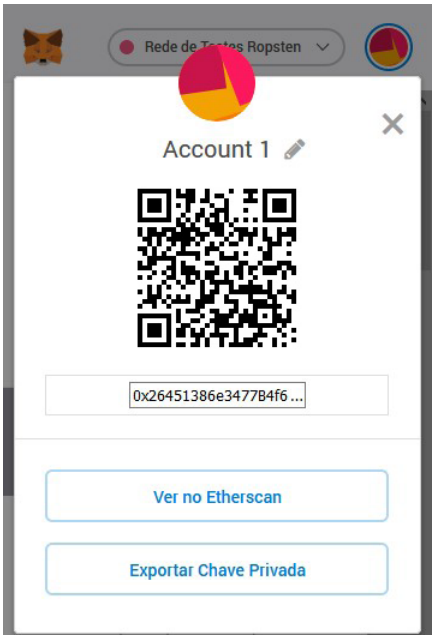

Figura 9 Detalhes da carteira no MetaMask

Figura 10 Chave privada exibida pelo MetaMask

A próxima etapa consistem em obter os pontos de entrada para realizar a conexão com as redes que hospedam as blockchains. Como a blockchain criada pelo Ganache está hospedada na máquina local, é possível acessá-la através da URL “http://127.0.0.1:7545”. Para conectar na rede de testes de Ropsten, foi preciso criar um projeto no Infura com a finalidade de obter a URL de entrada para essa rede. O Infura é um serviço que fornece APIs para a conexão com rede Ethereum, para o desenvolvimento do protótipo a URL que é fornecida de forma gratuita é o suficiente.

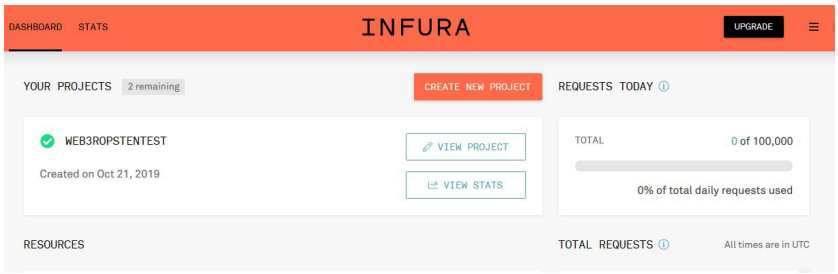

Figura 11 Tela principal da Infura

Para obter essa URL, é preciso criar uma conta gratuita no site do Infura, https://infura.io/register. Com a conta criada será necessário criar um novo projeto para poder utilizar as APIs, para isso basta clicar em “Create new project” e escolher um nome para o projeto. Feito isso, já será possível visualizar o projeto na tela inicial, ao clicar no projeto será possivel ver e editar os detalhes dele além de ter acesso as URL s para acesso as redes da Ethereum. Por padrão o infursa mostra a URL para realizar a conexão na rede principal, como essa pesquisa utiliza a rede de testes, é preciso alterar o endpoint de “Mainnet” para “Ropsten”.

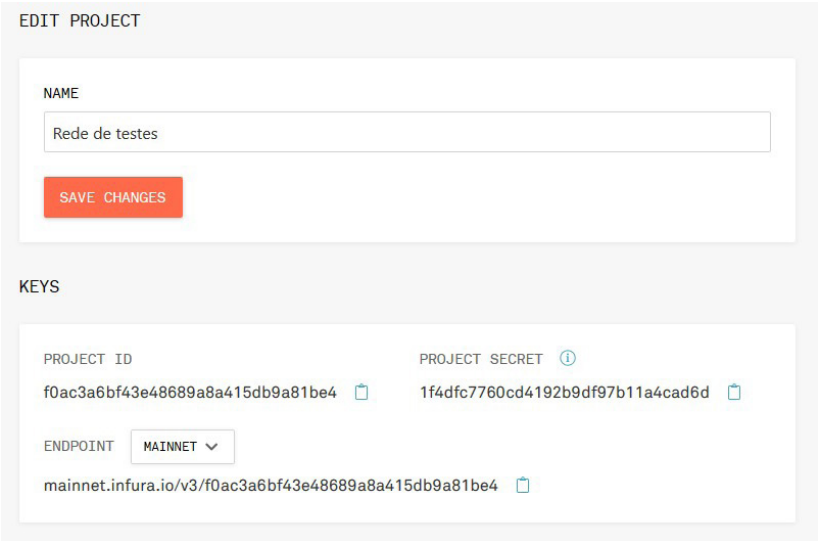

Figura 12 Detalhes de um projeto no Infura

Feito isso, é preciso utilizar o método ‘Web3(Web3.HTTPProvider(url_de_entrada))’, disponível na web3.py, para realizar a conexão com cada rede. Em sequência, é preciso especificar a conta que será utilizada para assinar e realizar as transações nas redes, para isso, pode-se utilizar o método ‘.eth.account.privateKeyToAccount(private_key)’ a partir da conexão que foi realizada. Nessa etapa é importante que as chaves privadas sejam utilizadas em suas respectivas redes, sendo a chave do MetaMask utilizada na rede pública e a chave do Ganeche utilizada na rede privada, caso contrário as contas que serão utilizadas poderão apresentar problemas ao realizar as transações.

```python
import json
from web3 import Web3
from web3.contract import ConciseContract

# Prepara a conexão com o ganache, ele precisa estar aberto
ganache_url = "http://127.0.0.1:7545"
ganache_private_key = "a97ee3e4583e76cd117a1373f968e4bb85f2dd940a4cdeabc61bb04cceddea95"
ganache_web3 = Web3(Web3.HTTPProvider(ganache_url))
ganache_acct = ganache_web3.eth.account.privateKeyToAccount(ganache_private_key)

# Set up web3 connection with Ethereum Ropsten testnet
privateKey = '50b67f37c12c02dd7d9ba921e38853f1f7bdaf80fddbd37da3dad861003e579e'
ropsten_web3 = Web3(Web3.HTTPProvider("https://ropsten.infura.io/v3/4998bb597a5d4e00b7d14ab1e6ca5035"))
ropsten_acct = ropsten_web3.eth.account.privateKeyToAccount(privateKey)
```

Figura 13 Exemplo da conexão com as redes

Para a próxima etapa é preciso compilar os contratos que foram desenvolvidos nas etapas anteriores e para isso será utilizado a Remix. Para realizar a compilação é preciso primeiro abrir o arquivo que contém a implementação do contrato, em sequência é preciso escolher a segunda opção no menu lateral para abrir o compilador, selecionar o arquivo contendo a implementação do contrato e clicar em “Compile nome_do_arquivo”.

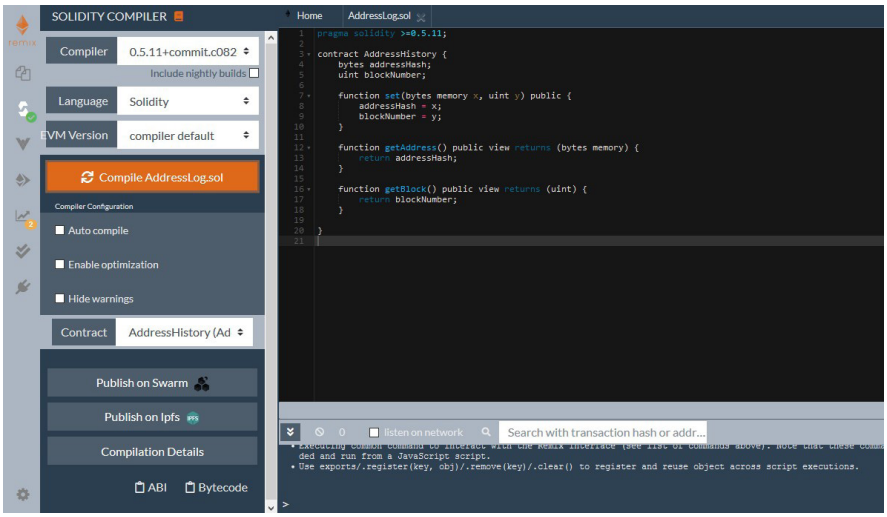

Figura 14 Interface gráfica da Remix

Após o código ser compilado, é preciso copiar a ABI e o Bytecode do contrato, isso pode ser feito ao clicar nos botões ABI e Bytecode presentes no fundo da página. Feito isso, os campos serão copiados para a área de transferência e poderão ser utilizados pelo protótipo. No caso da ABI será utilizado todo o código que foi fornecido pela Remix, já no caso do Bytecode só é preciso utilizar o valor presente no campo “object”. Também foi criada uma variável para guardar o endereço do contrato caso ele seja publicado pelo Remix, nesse exemplo ele será publicado utilizando o próprio protótipo.

```python
# set contract abi and bytecode
backup_block_abi = json.loads('''[ ···
]''')
backup_block_bytecode = "6080604052600a600055348015610015576000080fd5b506103538061002560000396000f3fe6080604
backup_block_contract_addr = None

# Set local contract abi and bytecode
address_history_abi = json.loads('''[ ···
]''')
address_history_bytecode = "6080604052348015610010576000080fd5b506103548061002060000396000f3fe6080604052348
address_history_contract_addr = None
```

Figura 15 Utilização da ABI e do Bytecode

A próxima etapa do algoritmo é realizar a publicação dos contratos em ambas as redes, para isso é preciso construir e assinar uma transação contendo o contrato a ser publicado. Essa transação será enviada para as redes com o objetivo de inserir os contratos nas blockchains e os disponibilizar para uso. Dessa forma, primeiro deve-se construir a estrutura contrato a partir da ABI e do Bytecode. Em sequência, é preciso construir a transação a partir do contrato do passo anterior, esse processo é realizado para ambos os contratos.

```
# If public contract does not exist
if backup_block_contract_addr is None:
    #create contract drom ABI and Bytecode
    contract_to_deploy = ropsten_web3.eth.contract(abi=backup_block_abi, bytecode=backup_block_bytecode)
    # Create contrac transacton
    construct_txn = contract_to_deploy.constructor().buildTransaction({
        'from': ropsten_acct.address,
        'nonce': ropsten_web3.eth.getTransactionCount(ropsten_acct.address),
        'gas': 1728712,
        'gasPrice': ropsten_web3.toWei('21', 'gwei'),
        'chainId': 3}) #Chain id = 3 for deploy in ropsten testnet

    # sign and send transaction
    signed_transaction = ropsten_acct.signTransaction(construct_txn)
    # Submit the transaction that deploys the contract
    transaction_hash = ropsten_web3.eth.sendRawTransaction(signed_transaction.rawTransaction)
    # Wait for the transaction to be mined, and get the transaction receipt
    transactionx_receipt = ropsten_web3.eth.waitForTransactionReceipt(transaction_hash)
    # Set contract address
    backup_block_contract_addr = transactionx_receipt.contractAddress

# Create the contract instance with the newly-deployed address
backup_block_contract = ropsten_web3.eth.contract(
    address=backup_block_contract_addr,
    abi=backup_block_abi,
)
```

Figura 16 Publicação do contrato público

A partir disso, a conta que foi informada na criação da rede é utilizada para assinar a transação que foi construída. Com ela assinada, é possível enviar a transação para rede utilizando o método “eth.sendRawTransaction(transaction)”, esse método retorna um hash que será utilizado para esperar a transação ser minerada e recuperar o recibo resultante da mineração. A partir do recibo, é possível recuperar o endereço no qual o contrato foi publicado e, com ele, criar a instância do contrato que será utilizada para interagir com as blockchains.

Com ambos os contratos publicados, o próximo passo para o algoritmo é recuperar o último bloco minerado na blockchain local, isso pode ser feito utilizando o método “.eth.getBlock(‘latest’)”. Esse método retorna um objeto JSON contendo diversas informações interessantes acerca do bloco que foi recuperado, o objetivo aqui é recuperar o primeiro valor da lista de transações. Esse valor é enviado para a blockchain pública a partir de uma nova transação criada utilizando a instância do contrato público como base, isso é feito a partir do método “.functions.nome_do_metodo(parâmetros)” presente na instância.

```
# Get local ganache block
block = ganache_web3.eth.getBlock('latest')

# Backup last block
transaction = backup_block_contract.functions.setTransaction(Web3.toBytes(block['transactions'][0])).buildTransaction({
    'gas': 70000,
    'gasPrice': ropsten_web3.toWei('1', 'gwei'),
    'from': ropsten_acct.address,
    'nonce': ropsten_web3.eth.getTransactionCount(ropsten_acct.address)
    })

# sign and send transaction
signed_transaction = ropsten_acct.signTransaction(transaction)
# Submit the transaction that deploys the contract
transaction_hash = ropsten_web3.eth.sendRawTransaction(signed_transaction.rawTransaction)
# Wait for the transaction to be mined, and get the transaction receipt
backup_receipt = ropsten_web3.eth.waitForTransactionReceipt(transaction_hash)

# Display the resulting values
print("Public block transaction hash:")
print(backup_block_contract.functions.getTransaction().call())
```

Figura 17 Envido dos dados para o contrato público

Após isso, o processo é feito de forma similar a publicação do contrato, a transação é assinada e enviada para rede pública. O protótipo recebe o hash da transação e ele é utilizado para esperar ela ser minerada. Com isso, o recibo contendo os dados da transação é recuperado. A última etapa do algoritmo consiste em enviar alguns dos dados presentes no recibo da transação anterior para a blockchain privada. Para isso uma nova transação é construída a partir da instância do contrato disponibilizado na rede privada, nessa transação são enviados os valores "transactionHash" e "blockNumber" presentes no recibo do contrato anterior. Esse processo segue as mesmas etapas que foram utilizadas para enviar os dados do bloco para a blockchain pública. Ao final desse processo, o contrato presente na blockchain privada armazena dados que foram enviados pela blockchain pública, isso encerra a execução do protótipo. Para garantir que esses dados foram armazenados, o protótipo exibe no terminal o hash da primeira transação enviada a blockchain pública e os dados enviados para a blockchain privada.

4 Resultados

Após o desenvolvimento do protótipo, foi provado que é possível realizar a comunicação entre as duas ledges utilizando uma interface. Ele também demonstrou que é possível realizar interações com contratos queque estão presentes nas blockchains, além de poder realizar transações e recuperar dados presentes nos blocos e na rede. Porém, para realizar essa comunicação foi preciso utilizar um serviço externo para obter a url de entrada necessária para realizar as interações com a rede de testes da Ethereum, gerando no mínimo uma dependência para o protótipo. Todo o código fonte produzido pode ser encontrado em https://bitbucket.org/Wanderson_Sena/prototipo-msi/src/master/

Embora também tenha sido utilizada uma ferramenta externa para realizar a compilação do contrato, é possível adicionar um compilador de solidity ao protótipo e remover essa dependência. Entretanto, o Remix foi utilizado devido a praticidade de compilar os contratos e obter tando a ABI quanto o Bytecode, isso foi possível pelo fato do protótipo não possuir a necessidade de alterar e republicar os contratos. Como essa primeira implementação do protótipo sempre publica os contratos, os dados são salvos em locais diferentes a cada interação. Contudo, isso pode ser evitado caso o endereço onde os contratos estão publicados sejam inseridos no código.

Também é preciso informar ao protótipo as chaves privadas pertencentes a contas, e garantir que elas possuam ether suficiente para executar as transações, e devido a simplicidade das interações não é necessário mais de 1 ether em cada conta. Caso a rede de testes de Ropsten esteja sendo utilizada, é possível realizar uma chamada em um contrato para obter ether. Esses contratos podem ser encontrados em diversos endereços, o contrato utilizado no desenvolvimento dessa pesquisa pode ser encontrado no endereço https://faucet.metamask.io/.

5 Conclusões

Essa pesquisa permitiu desenvolver um protótipo demonstrando uma interação simples entre duas blockchains hospedadas em redes diferentes, as próximas pesquisas terão como foco criar interações mais profundas e interessantes entre essas redes. Além disso, futuras evoluções na tecnologia das blockchains e dos contratos podem permitir que essa interação seja realizada diretamente pelos contratos publicados nas redes. Embora essa pesquisa tenha tido como foco blockchains públicas e privadas, o protótipo pode ser utilizado para realizar interações entre quaisquer duas blockchains desde que exista uma url de entrada para a rede que hospeda as blockchains.

6 Referências Bibliográficas

Nakamoto S. Bitcoin: A Peer-to-Peer Electronic Cash System. Disponível em: https://bitcoin.org/bitcoin.pdf. Acesso em: 7 set. 2019.

Antonopoulos A. Mastering Bitcoin, 2 ed. O'Reilly Media Inc, 2017. Disponível em:https://www.oreilly.com/library/view/mastering-bitcoin-2nd/9781491954379/#toc-start. Acesso em: 7 set. 2019.

Wood G.; Antonopoulos A. Mastering Ethereum, 1 ed. O'Reilly Media Inc, 2018. Disponível em: https://www.oreilly.com/library/view/mastering-ethereum/9781491
971932/. Acesso em: 7 set. 2019.

Cachin C. Architecture of the Hyperledger Blockchain Fabric. Disponível em: https://pdfs.semanticscholar.org/f852/c5f3fe649f8a17ded391df0796677a59927f.pdf. Acesso em: 7 set. 2019.

Pilkington, M. Blockchain technology: principles and applications. Research Handbook on Digital Transformations, Montréal, Canadá, 1 ed, p.255 - p.254, 2016

Crosby, M. BlockChain Technology: Beyond Bitcoin. Applied Innovation Review., Montréal, Canadá, 1 ed, 2 volume, 2016