MINISTÉRIO DA EDUCAÇÃO
UTFPR – UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
CAMPUS CORNÉLIO PROCÓPIO
GRADUAÇÃO EM ENGENHARIA DE COMPUTAÇÃO

Professor: Lucas Hiera Sampaio

Aluno: Mateus Yi Muraoka

Atividade 2

EC38D – Segurança e Auditoria de Sistema – C81(2022/1)

Blockchain

CORNÉLIO PROCÓPIO

JUNHO– 2022

Sumário

1. INTRODUÇÃO

Blockchain é um livro-razão que é compartilhado e imutável, que permite um processo de armazenamento e consulta a informações de maneira mais acessível. Podendo guardar registros tangíveis (carros, casas e afins) e intangíveis (propriedade intelectual, direitos autorais, patentes). Desta maneira as informações se tornam imprescindíveis, logo é blockchain entra como um serviço que permite armazenar e recuperar as informações de forma imediata, compartilhadas e transparentes, sendo armazenadas no livrão- razão imutável, podendo ser acessada somente por membros de rede autorizada.

As criptomoedas são moedas virtuais, para que essa moeda se torne segura é necessário a blockchain pois é necessária uma estrutura apropriada para a quantidade massiva de transações mantendo a segurança e a transparência. Já o processo de obtenção destas moedas é necessário a mineração das moedas, verificando autenticidade das informações que estão contidas nos blocos, caso as informações forem verdadeiras, o bloco será incorporado na rede e recebera criptografia.

Os processos de validação são incorporados a blockchain através de um algoritmo, mais comum é o Prova de Trabalho (*Prof-of-Work(PoW)*). O PoW é utilizado validar algumas criptomoedas existentes atualmente como Bitcoin. É considerado um algoritmo de consenso onde é necessário que ocorra verificação das informações do bloco por vários mineradores, para decidir se este bloco entrara ou não na corrente “chain”.

Na atividade 2 é proposto a construção de uma blockchain local para o compreendimento do conceito e execução prática permite entender os processos necessários para a criação, processo de adicionar blocos, garantir a validação e imutabilidade da cadeia.

2. MATERIAIS E METODOS

O desenvolvimento da atividade sobre blockchain, onde foi utilizado como base o link disponibilizado no documento da Atividade 2, que encaminha para primeira video aula da playlist “Building a blockchain with Javascript” do canal Simply Explained no YouTube.

Figura 1 –Playlist utilizada para execução da atividade.

A linguagem de programação utilizada para o desenvolvimento da atividade foi o Javascript, assim como no vídeo, o IDE utilizada foi Visual Code Studio e o processador é um Intel Core i7-10750H de notebook, visto que foi gasto processamento durante a mineração do bloco . Além disto, foi necessário a importação de duas bibliotecas sendo, a “crypto-js” na versão 4.1.1 e a “elliptic” na versão 6.5.4, ambas foram importadas através do terminal para dentro do projeto. A “crypto-js” foi importada para que pudesse utilizar o algoritmo de criptografia SHA256, enquanto a “elliptic” foi utilizada para criação de uma chave publica e um privada para verificação uma assinatura.

Seguindo os primeiros 4 vídeos da playlist foram criados 3 arquivos bases, “blockchain.js”, “keygenerator.js” e o “main.js”.

Figura 2 – Arquivos criados

O arquivo “keygenerator.js” serve para gerar as chaves publicas e privadas a serem utilizadas, o algoritmo se encontra na figura 3 e as chaves geradas na figura 4.

```
//Geracao do par de chaves
const key = ec.genKeyPair();
//Extrarir a chave publica no formato hex
const publicKey = key.getPublic('hex');
//Extrarir a chave privada no formato hex
const privateKey = key.getPrivate('hex');

//Imprime no terminal
console.log();
console.log('Private key: ', privateKey);

//Imprime no terminal
console.log();
console.log('Public key:  ', publicKey);
```

Figura 3 – Código para gerar chaves publicas e privadas

```
Private key:  c6c7f38907cd11fd1880d6f0011a422572ffabc643ee0db5f9c2951929263f73

Public key:   04d83bf687197b805180474cfc2d2bfb76391122134193ee6a9eb6011b3cbe7e43fd44909aad818efa1d62298e14fe263b99e3b80659ba8b852c75b6d4f7a57a29
```

Figura 4 – Chaves públicas e privadas geradas

No arquivo “blockchain.js” foi criado 3 classes principais: Transaction, Block e Blockchain.

```
//Criando a classe transaction
class Transaction {
    // onde transacao sempre vem de alguem, vai para alguem e carrega uma certa quantia de moedas
    constructor(fromAddress, toAddress, amount) {
        this.fromAddress = fromAddress;
        this.toAddress = toAddress;
        this.amount = amount;
    }
    // Metodo para retornar a Hash sha256 da transacao
    calculateHash() {
        return SHA256(this.fromAddress + this.toAddress + this.amount).toString();
    }
    // Metodo para assinar as transacao
    signTransaction(signingKey) {
        //Funcao para verificar se a chave eh igual ao  do endereco
        if (signingKey.getPublic('hex') !== this.fromAddress) {
            throw new Error('Você não pode assinar transacoes para outras carteiras!');
        }
        // Eh calculado a hash da transacao
        //Assina com a chave em formato hex que se encontra no arquivo keygenerator.js
        // Guarda o objeto da transacao
        const hashTx = this.calculateHash();
        const sig = signingKey.sign(hashTx, 'base64');
        this.signature = sig.toDER('hex');
    }
    //Metodo para verificar se assinatura esta correta
    isValid() {
        //verifica se a transacao eh nula se for eh um transacao valida
        if (this.fromAddress === null) return true;
        //verifica se nao ha assinatura ou se assinatura estiver vazia
        if (!this.signature || this.signature.length === 0) {
            throw Error('Sem assinatura nesta transacao!');
        }
        const publicKey = ec.keyFromPublic(this.fromAddress, 'hex');
        //verifica se a hash do bloco esta assinada com esta assinatura
        return publicKey.verify(this.calculateHash(), this.signature);
    }
}
```

Figura 5 – Classe Transaction

Este código da classe “Transaction”, demostrado na figura 5, tem como função lidar com as informações de origem e de destino, a quantidade de informações a serem transferidas, as validar e verificar as informações e checar se as assinaturas estão corretas.

```
//criar a classe block
class Block {
    //cria classe construtor que receberar//as propriedades , sendo que a previousHash esta vazia
    //"time" stamp dira quando o block foi criado
    //previousHash eh um string que contem a hash anterior podemndo armzenar o valor de transacao
    constructor(timestamp, transactions, previousHash = '') {
        //ira acompanhar todos os valores
        this.previousHash = previousHash;
        this.timestamp = timestamp;
        this.transactions = transactions;
        //"hash" ira conter a hash do nosso bloco // precisamos de um caminho para calcular isto
        // enquanto mantem isto como vazio por agora
        this.hash = this.calculateHash();
        this.nonce = 0;
    }
    //criar um novo metodo para calcular a funcao hash do bloco//pegando a propriedades do bloco, roda-los dentro da funcao hash
    // e retornar a hash que identificara o bloco na blockchain//sera utilizado sha-256 como funcao hash
    calculateHash() {
        //retorna as propriedades do bloco
        return SHA256(this.previousHash + this.timestamp + JSON.stringify(this.transactions) + this.nonce).toString();
    }
    //Metodo verifica a dificuldade de mineração e a incrementa
    mineBlock(difficulty) {
        // Condição criada para verificar a quandidade de zeros necessarias para minerar
        while (this.hash.substring(0, difficulty) !== Array(difficulty + 1).join("0")) {
            this.nonce++;
            this.hash = this.calculateHash();
        }
        console.log("BLOCK MINED: " + this.hash);
    }
    // Metodo verifica se todas as transacoes do bloco sao validas
    hasValidTransactions() {
        for (const tx of this.transactions) {
            if (!tx.isValid()) {
                return false;
            }
        }
        return true;
    }
```

Figura 6 – Classe Block

O código da figura 6, é a o Block que possui os atributos que armazenam a hash do bloco atual, a hash bloco anterior, o tempo em que a informações foram armazenadas e as informações que serão armazenadas, assim como o método de validação de transação e método para verificar a dificuldade de mineração do bloco.

```
//Criar uma nova classe blockchain
class Blockchain {
    //criar um metodo constructor
    constructor() {
        //criar propriedade chain dentro da classe que sera um array dos blocos
        this.chain = [this.createGenesisBlock()];
        this.difficulty = 1;
        this.pendingTransactions = [];
        this.miningReward = 100;
    }
    //o primeiro bloco na blockchain eh bloco de Origem(GenesisBlock)
    // e ele dever ser adicionado manualmente

    //Metodo cria o Genesis block e
    createGenesisBlock() {
        //retornara um novo bloco e sera iniciado como zero pois nao ha bloco anterior
        return new Block("01/05/2022", "Genesis block", "0");
    }
```

Figura 7 – Classe Blockchain – parte 1

```
//Metodo retorna o ultimo bloco da Blockchain
getLatestBlock() {
    return this.chain[this.chain.length - 1];
}
//Metodo resposavel por minerar, caso a mineracao ocorra com sucesso,
//o bloco enviara o reward para este endereco
minePendingTransactions(miningRewardAddress) {
    const rewardTx = new Transaction(null, miningRewardAddress, this.miningReward);
    this.pendingTransactions.push(rewardTx);

    let block = new Block(Date.now(), this.pendingTransactions, this.getLatestBlock().hash);
    block.mineBlock(this.difficulty);

    console.log('Bloco Minerado com sucesso!');
    this.chain.push(block);

    this.pendingTransactions = [];
}
//Metodo resposavel por adicionar uma Transacao
addTransaction(transaction) {
    //  cria condicao para verificar se a transacao tem endereço de origem e destino
    if (!transaction.fromAddress || !transaction.toAddress) {
        throw new Error('A transacao deve incluir de qual endereco e para qual endereço');
    }
    //  cria condicao para verificanr se a transacao que sera adicionada a cadeia eh valida
    if (!transaction.isValid()) {
        throw new Error('Nao pode adicionar transacao invalida a chain');
    }
    this.pendingTransactions.push(transaction);
}
```

Figura 8 – Classe Blockchain – parte 2

```
//Metodo resposavel para calcular o Saldo
getBalanceOfAddress(address) {
    let balance = 0;
    for (const block of this.chain) {
        for (const trans of block.transactions) {
            if (trans.fromAddress === address) {
                balance -= trans.amount;
            }

            if (trans.toAddress === address) {
                balance += trans.amount;
            }
        }
    }
    return balance;
}
```

Figura 9 – Classe Blockchain – parte 3

```
    //Método para validar o valores da blockchain
    isChainValid() {
        for (let i = 1; i < this.chain.length; i++) {
            const currentBlock = this.chain[i];
            const previousBlock = this.chain[i - 1];

            // cria condicao para verificar se todas transacoes do bloco sao validas
            if (!currentBlock.hasValidTransactions()) {
                return false;
            }
            //cria condicao para verificar se o  hash bloco atual nao eh igual ao do hashcalculado bloco
            if (currentBlock.hash != currentBlock.calculateHash()) {
                return false;
            }
            //cria condicao para verificar se o bloco esta apontando corretamente para o  bloco anterior
            if (currentBlock.previousHash !== previousBlock.hash) {
                return false;
            }
        }
        return true;
    }
}

module.exports.Blockchain = Blockchain;
module.exports.Transaction = Transaction;
```

Figura 10 – Classe Blockchain – parte 4

O código das figuras 7, 8, 9 e 10 são referentes a classe Blockchain, onde na figura 6 foi criado um método que possui os atributos de nível de dificuldade, o valor da recompensa pela mineração, pedido de transação e defini criação do bloco Genesis que é o primeiro bloco. Na figura 8, é demonstrado 3 funções, o método para retornar até o último bloco, método mineração e endereçamento dos ganhos e a última função é para criação de uma transação. Enquanto na figura 9 tem o método para calcular o saldo e na figura 10, método para validar os valores da blockchain.

3. RESULTADOS

O arquivo “main.js”, tem como função executar a mineração dos blocos e a validação através chave privada, das informações que foram mineradas durante o processo e para demonstração foram criados 3 blocos para mineração, como mostrados na figura 12, 15 e 16. Antes da execução da mineração de todos os blocos, foi executado um teste de capacidade da blockchain de invalidar alterações que ocorram no bloco.

O código da figura 13 foi executado, onde foi alterado o valor do atributo “amount” da classe Transaction. Com o valor de “amount” alterado, foi executado o método de verificação, onde o resultado obtido pode ser observado na figura 14, mostrando o resultado falso, logo a “chain” não é mais válida para ser utilizada.

Após, foi comentado a linha de código do teste e o valor de “amount” é retornado a zero. Os 3 blocos para mineração que foram criados testados em 5 dificuldades sendo elas 2, 3, 4 ,5 e 6, logo as respostas obtidas durante a execução do código podem ser vistas nas figuras 17, 18, 19, 20 e 21.

```
//seta  a chave privada para ser utilizada
const myKey = ec.keyFromPrivate('c6c7f38907cd11fd1880d6f0011a422572ffabc643ee0db5f9c2951929263f73');
const myWalletAddress = myKey.getPublic('hex');

let savjeeCoin = new Blockchain();
```

Figura 11 – Utilização da chave privada

```
//Seta a Transacao 1
const tx1 = new Transaction(myWalletAddress, 'public keys goes here',10);
//Assina transacao com key pessoal
tx1.signTransaction(myKey);
// Adiciona o Transacao 1 na blockchain
savjeeCoin.addTransaction(tx1);

console.log('\n Começou a mineração 1...');
savjeeCoin.minePendingTransactions(myWalletAddress);
const time1 =  new Date(savjeeCoin.chain[1].timestamp);
console.log("Tempo: \n", time1);
console.log('\nO saldo da carteira eh: ', savjeeCoin.getBalanceOfAddress(myWalletAddress));
console.log('A Chain eh valida?' , savjeeCoin.isChainValid());
```

Figura 12 – Código para a primeira transação

```
//Codigo para alterar os dados do bloco diretamente
savjeeCoin.chain[1].transactions[0].amount =1;
console.log('\nTentativa de alteracao')
//checa se chain eh valida
console.log('\nA Chain eh valida?' , savjeeCoin.isChainValid());
```

Figura 13 – Código para a primeira transação

```
A Chain eh valida? true

Tentativa de alteracao

A Chain eh valida? false
```

Figura 14 – Código para a primeira transação

```
//Seta a Transacao 2
const tx2 = new Transaction(myWalletAddress, 'public keys goes here',10);
//Assina transacao com key pessoal
tx2.signTransaction(myKey);
// Adiciona o Transacao 2 na blockchain
savjeeCoin.addTransaction(tx2);

console.log('\n Começou a mineração 2...');
savjeeCoin.minePendingTransactions(myWalletAddress);
const time2 =  new Date(savjeeCoin.chain[2].timestamp);
console.log("Tempo: \n", time2);
console.log('\nO saldo da carteira eh: ', savjeeCoin.getBalanceOfAddress(myWalletAddress));
console.log('A Chain eh valida?' , savjeeCoin.isChainValid());
```

Figura 15 – Código para a segunda transação

```
//Seta a Transacao 3
const tx3 = new Transaction(myWalletAddress, 'public keys goes here',10);
//Assina transacao com key pessoal
tx3.signTransaction(myKey);
// Adiciona o Transacao 3 na blockchain
savjeeCoin.addTransaction(tx3);

console.log('\n Começou a mineração 3...');
savjeeCoin.minePendingTransactions(myWalletAddress);
const time3 =  new Date(savjeeCoin.chain[3].timestamp);
console.log("Tempo: \n", time3);
console.log('\nO saldo da carteira eh: ', savjeeCoin.getBalanceOfAddress(myWalletAddress));
console.log('A Chain eh valida?' , savjeeCoin.isChainValid());
```

Figura 16 – Código para a terceira transação

```
 Começou a mineração 1...
BLOCK MINED: 00587396bfa8fd6858a0ca99b292e911ff20e68395976e1d93ac64ba51094daf
Bloco Minerado com sucesso!
Tempo:
 2022-06-09T04:57:09.010Z

O saldo da carteira eh:  90
A Chain eh valida? true

 Começou a mineração 2...
BLOCK MINED: 000344271bde8d737c3f089a8c4e28b589625156972880a35cc38e648bc596ae
Bloco Minerado com sucesso!
Tempo:
 2022-06-09T04:57:09.033Z

O saldo da carteira eh:  180
A Chain eh valida? true

 Começou a mineração 3...
BLOCK MINED: 00eed5d5283bc090a01949aa94d6364ea2f6ab28f3f940ef3568141eea3ff14c
Bloco Minerado com sucesso!
Tempo:
 2022-06-09T04:57:09.057Z

O saldo da carteira eh:  270
A Chain eh valida? true
```

Figura 17 – Resultado da mineração - Dificuldade 2

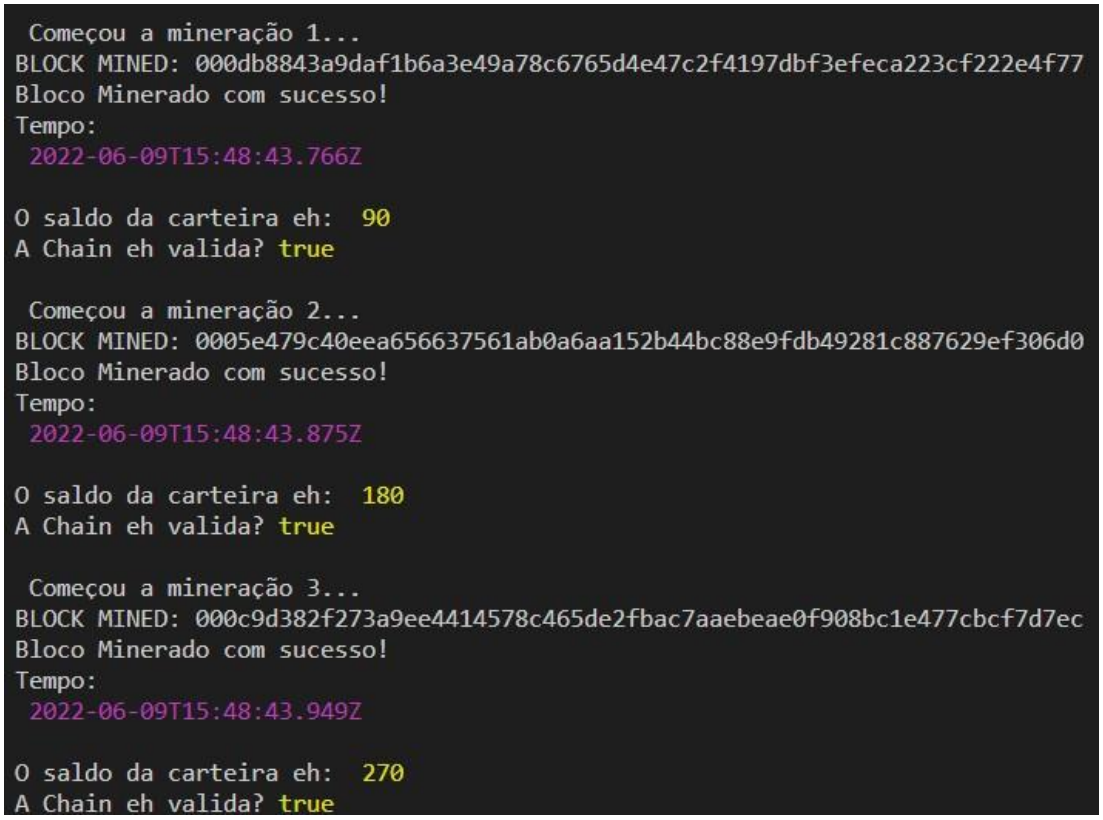

Figura 18 – Resultado da mineração - Dificuldade 3

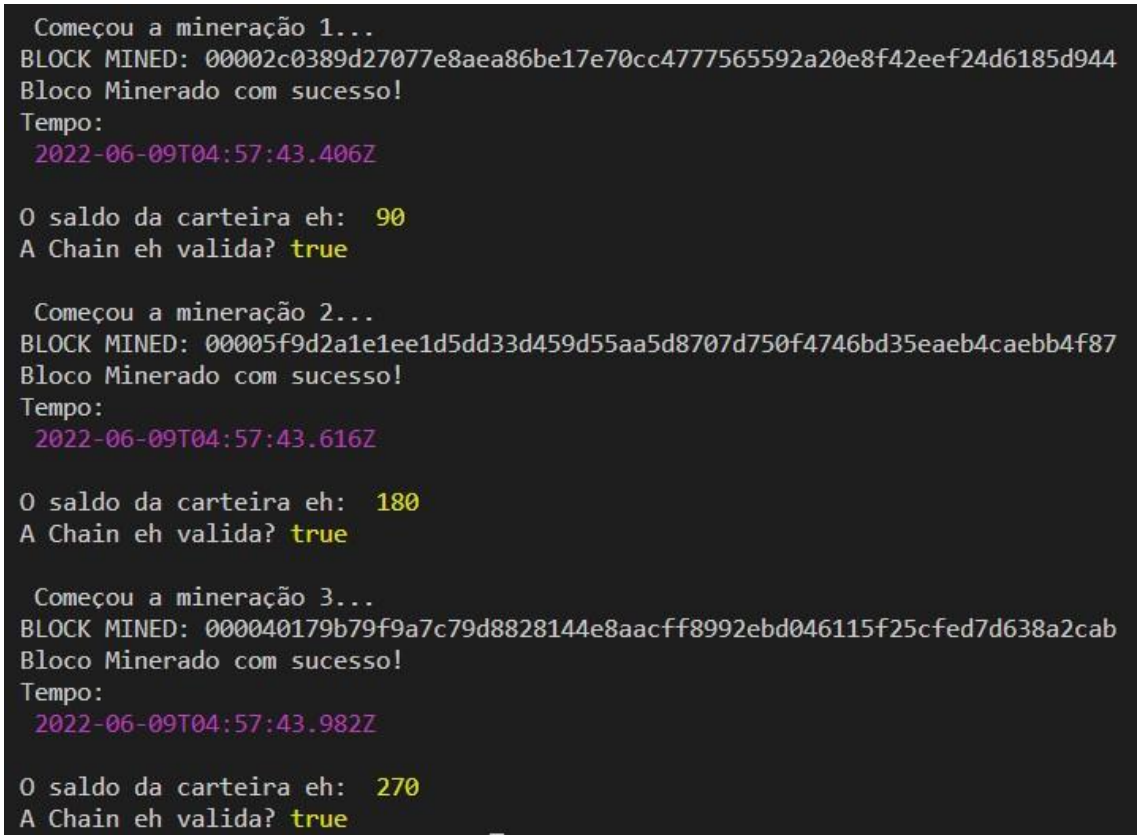

Figura 19 – Resultado da mineração - Dificuldade 4

```
 Começou a mineração 1...
BLOCK MINED: 00000f58f830f182b8aa4d27269a527237603636ef5be5990dd6e1654760b4ee
Bloco Minerado com sucesso!
Tempo:
 2022-06-09T04:58:18.284Z

O saldo da carteira eh:  90
A Chain eh valida? true

 Começou a mineração 2...
BLOCK MINED: 0000047521b0c8fee718e0b557b2e1fdaf31c9e105ee41a6889d99e5d2633ec7
Bloco Minerado com sucesso!
Tempo:
 2022-06-09T04:58:20.757Z

O saldo da carteira eh:  180
A Chain eh valida? true

 Começou a mineração 3...
BLOCK MINED: 000007902e1f1d6af06cdfcc87438536593cd2abffb2c6bf1a50ceaa8dcc6305
Bloco Minerado com sucesso!
Tempo:
 2022-06-09T04:58:24.126Z

O saldo da carteira eh:  270
A Chain eh valida? true
```

Figura 20 – Resultado da mineração - Dificuldade 5

```
 Começou a mineração 1...
BLOCK MINED: 00000082e07354df9b86922aa230c9418929a163f4fdbb36d288f2832e4107fa
Bloco Minerado com sucesso!
Tempo:
 2022-06-09T04:59:15.209Z

O saldo da carteira eh:  90
A Chain eh valida? true

 Começou a mineração 2...
BLOCK MINED: 000000fe91dcb780afc7068e84c7441688be504fdfe21d6c670515a0bf5980e6
Bloco Minerado com sucesso!
Tempo:
 2022-06-09T05:00:25.685Z

O saldo da carteira eh:  180
A Chain eh valida? true

 Começou a mineração 3...
BLOCK MINED: 0000000a00e8b96df5d06f616cac132b156d02c78bea419d58e5a4213fc79889
Bloco Minerado com sucesso!
Tempo:
 2022-06-09T05:01:39.743Z

O saldo da carteira eh:  270
A Chain eh valida? true
```

Figura 21 – Resultado da mineração - Dificuldade 6

Dificuldade/tempo	Execução 1(ms)	Execução 2(ms)	Execução 3(ms)	Media (ms)
Nivel 2	26,000	37,000	32,000	32
Nivel 3	116,000	82,000	109,000	102
Nivel 4	273,000	408,000	381,000	354
Nivel 5	3188,000	8987,000	6361,000	6179
Nivel 6	12528,000	74766,000	39604,000	42299

Figura 22 – Dados obtidos durante 3 testes feitos

Figura 23 – Gráficos obtidos através da média feita entre os dados obtidos.

Foram feitos 3 testes de mineração para o bloco 1 a cada nível de dificuldade testada, onde foi calculado o tempo de cada teste e colocado em na tabela como a figura 22, o tempo desta mineração é calculado através do horário de início do bloco 1 e começo do bloco 2, pois o bloco 2 inicia a ser minerado quando o bloco 1 termina. Assim foi calculado a média entre os 3 valores obtidos para cada nível dificuldade e demonstrado de forma gráfica na figura 23. A partir do gráfico e dos resultados obtidos foi possível notar que o aumento de dificuldade é proporcional ao aumento do tempo gasto para mineração de cada bloco, podendo ver que o crescimento é exponencial. Da dificuldade 4 em diante os valores de tempo estrão sempre crescendo exponencialmente, além da necessidade de processamento do hardware que estiver minerando os blocos. Foi feito um teste para a dificuldade 8 entretanto devido os recursos da máquina, o teste não foi finalizado com sucesso, devido a necessidade de recursos de processamento.

4. CONCLUSÃO

Deste modo os resultados obtidos foram satisfatórios, onde foi possível atingir os objetivos descritos na atividade 2. Permitindo a compreensão do conteúdo sobre “Blockchain”, “Prof-of-Work”, o conceito de recompensa da mineração e transação e assinaturas que devem ocorrer durante todo processo. Além da compreensão, a atividade permitiu a execução pratica de todo o conceito abordado, permitindo entender como funciona o processo criação, mineração e sua segurança das Criptomoedas, que estão presentes atualmente e tem tendencia a continuar crescendo e se estabilizando.

5. REFERÊNCIA

1. IBM. O que é a tecnologia blockchain? Disponível em: “https://www.ibm.com/br-pt/topics/what-is-blockchain “. Acesso em: 18/maio. 2022
2. Equipe Coinext. Proof of Work(PoW): O que é e por que é necessário? . Disponível em: “https://coinext.com.br/blog/proof-of-work-o-que-e#:~:text=Proof%20of%20Work%20%C3%A9%20o,realizado%20pelos%20usu%C3%A1rios%20dessa%20rede. “. Acesso em: 18/maio. 2022
3. MALAR, João Pedro. Entenda como funciona a mineração de criptomoedas e os efeitos no meio ambiente. 2021. Disponível em: “https://www.cnnbrasil.com.br/business/entenda-como-funciona-a-mineracao-decriptomoedas-e-os-efeitos-no-meio-ambiente/.“ Acesso em: 18/maio. 2022
4. Creating a blockchain with JavaScript (Blockchain, part 1), 2017. (14 min). Disponível em: "https://www.youtube.com/watch?v=zVqczFZr124”. Acesso em: 25/maio. 2022
5. Implementing Proof-of-Work in JavaScript (Blockchain, part 2) 2017. (6 min). Disponível em: “https://www.youtube.com/watch?v=HneatE69814”. Acesso em: 25/maio. 2022
6. Mining rewards & transactions - Blockchain in JavaScript (part 3) 2017. (12 min). Disponível em: https://www.youtube.com/watch?v=fRV6cGXVQ4I Acesso em: 25/maio. 2022
7. Signing transactions - Blockchain in JavaScript (part 4), 2017. (18 min). Disponível em: “https://www.youtube.com/watch?v=kWQ84S13-hw&list=PLzvRQMJ9HDiTqZmbtFisdXFxul5k0F-Q4&index=4”. Acesso em: 25/maio. 2022