Segurança e Auditoria de Sistemas AS36B-N16

Atividade blockchain

Ferramentas

- Python 3.8.11;
- Biblioteca de criptografia do Python (time, json, os, hashlib sha256);

Objetivos

- Criar uma aplicação de blockchain local
- Implementar um algoritmo de prova por trabalho local onde a dificuldade é definida pelo usuário da aplicação
- Armazenar os blocos validados
- Implementar uma rotina que verifica a integridade dos blocos.
- O conteúdo permitido nos blocos deverá ser definido por você.

Procedimentos

Foi desenvolvido um código de uma aplicação em Python e os procedimentos de funcionamento do programa são os seguintes:

- É necessário criar no mesmo diretório do programa uma pasta com o nome de 'blockchain' para armazenar os blocos
- Ao iniciar o programa primeiramente é mandatório selecionar a opção de número um para a criação do bloco de origem.
- Em seguida na opção de número dois é possível adicionar a quantidade de blocos que desejar, lembrando sempre de inserir: remetente, destinatário, transação e dificuldade de criação do bloco
- A terceira opção do menu executa uma verificação de integridade da blockchain, averiguando se não teve adulteração de informações gravadas nos blocos.

Metodologia e Resultados

O programa foi desenvolvido com o uso de um menu de seleção com chamada de funções. A seguir o fluxo das funções seguido de uma breve descrição:

- Ao rodar o programa é exibido um menu, criado pela função ‘menu’

- Ao selecionar a opção um chama a função ‘cria_prim_bloco’ que escreve em um arquivo json as informações já pré determinadas do bloco de origem.
- Ao selecionar a opção dois chama a função ‘cria_bloco’ que passa como parâmetro as variáveis: remetente, destinatário, transação e dificuldade. Sendo a dificuldade a quantidade de zero que tem que ter no hash
    - Primeiramente essa função acessa a pasta que guarda os blocos e extrai pela leitura do nome dos arquivos a quantidade de blocos existentes
    - starta a contagem de tempo para a verificação de desempenho
    - chama uma terceira função de nome ‘minera’ que passa como parâmetro o numero do ultimo bloco e a dificuldade
        - A função ‘minera’ abre o último bloco já criado no modo de leitura de binários e extrai para variável ‘conteudo_bloco’ todas as informações gravadas no bloco
        - em seguida ao entrar em um loop while concatena em uma variável temporária o conteúdo do bloco mais o valor do nonce
        - faz o hash256 dessa variável ‘temp’ no modo hexadecimal
        - faz a verificação por condicional se o início do bloco inicia com a quantidade de zeros determinadas pela variável ‘dificuldade’ informada pelo usuário, caso essa condição tenha sido atendida a função retorna o hash mais o valor do nonce, caso contrário sai do condicional soma um na variável ‘nonce’ e permanece no laço while
- sdasd

A figura 1 mostra a criação do primeiro bloco, de modo que precisa-se de um nó origem para que os sucessores possam ter uma ligação pelo hash do anterior:

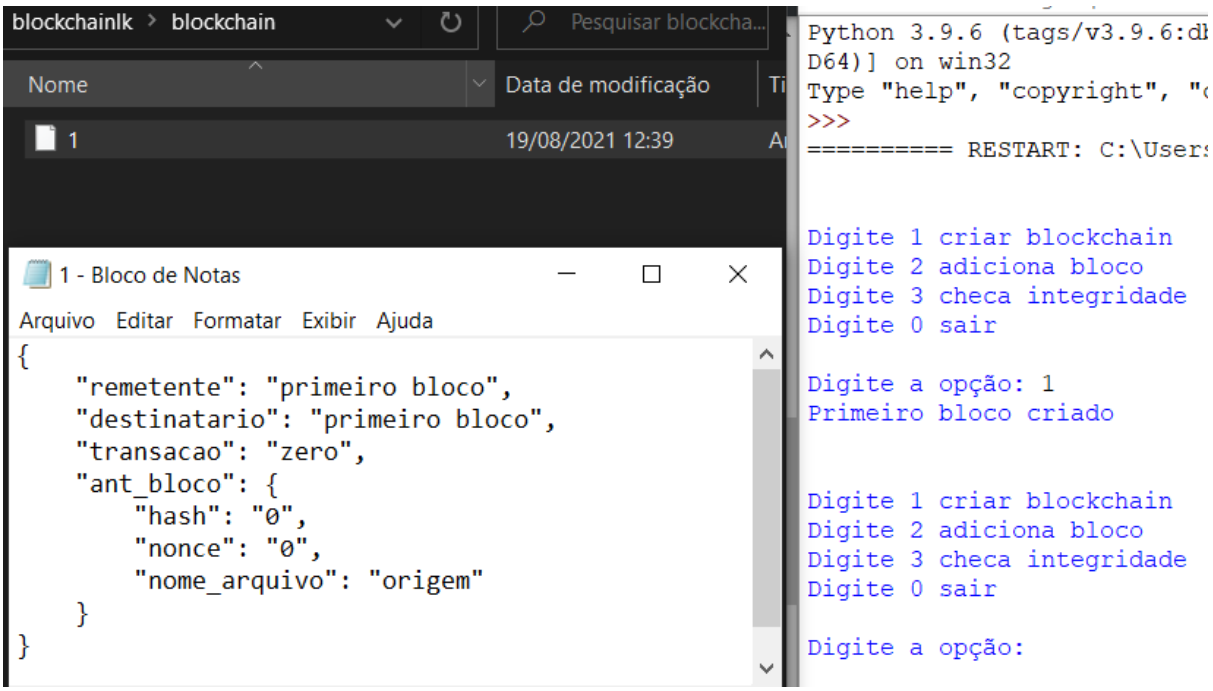

Figura 1

A figura 2 mostra mostra a criação do segundo bloco já com o hash e o nonce do bloco anterior, observa-se que nesse caso ao alterar qualquer informação do bloco origem o hash guardado no bloco dois já não será mais válido e comprovará a alteração a blockchain:

Figura 2

A figura 3 mostra a função de verificação de que os blocos não foram alterados:

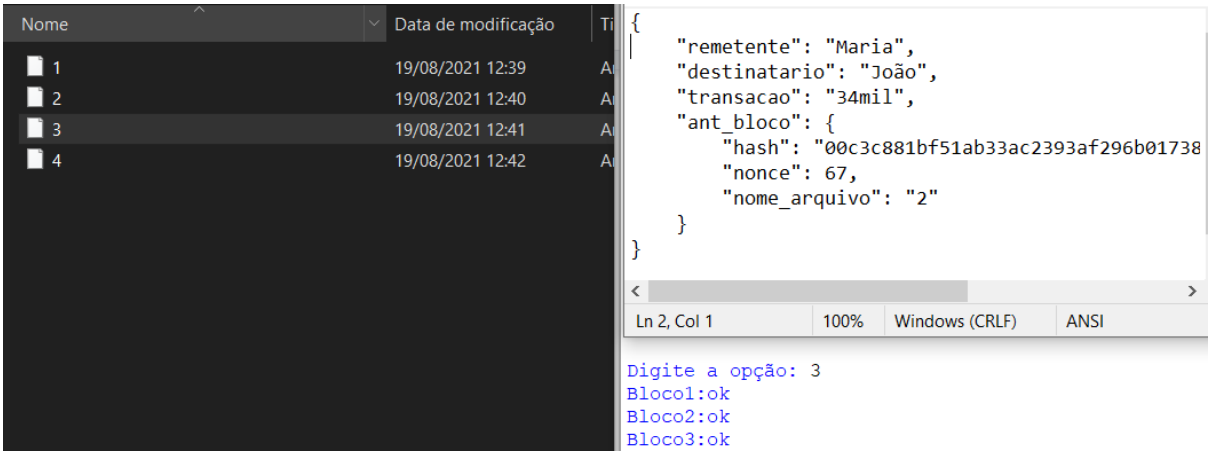

Figura 3

A figura 4 mostra que foi possível identificar uma alteração após alterar o valor da transação no bloco 3:

Figura 4

A figura 5 mostra a criação do hash com dificuldade três, ou seja, três no início:

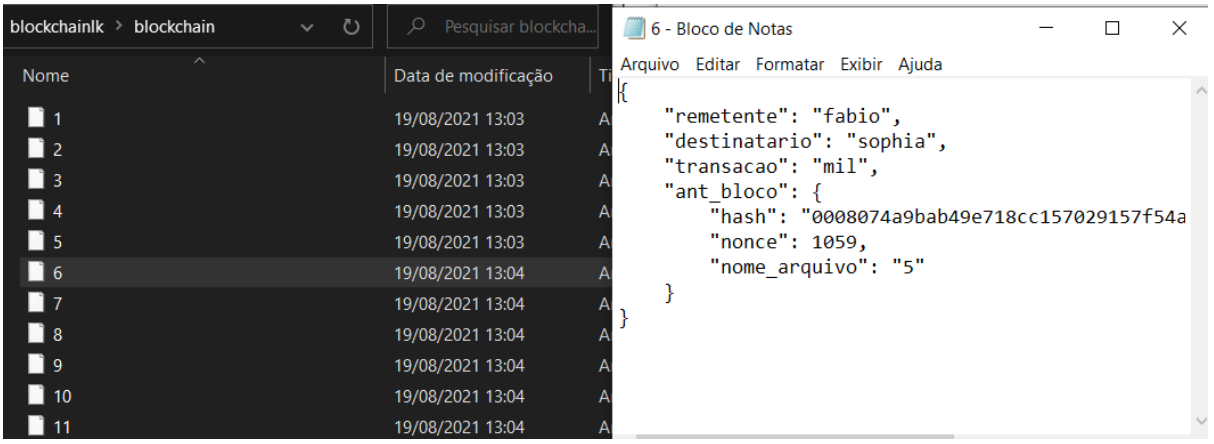

Figura 5

A figura 6 mostra a criação do hash com dificuldade quatro, ou seja, quatro no início:

Figura 6

A figura 7 mostra a criação do hash com dificuldade cinco, ou seja, cinco no início:

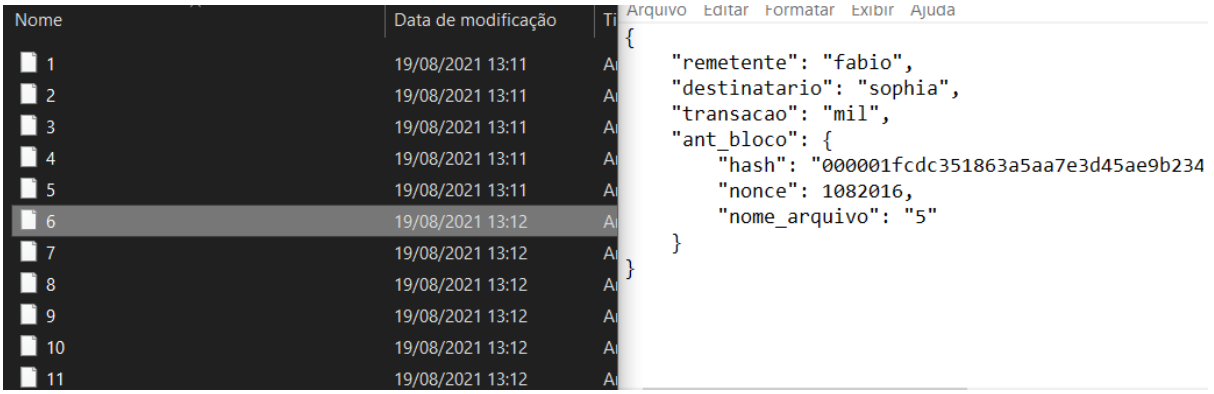

Figura 7

A figura 8 mostra um gráfico comparativo do tempo em segundos necessários para mineração de blocos de acordo com as dificuldades de três, quatro e cinco. No gráfico também está demonstrado o tempo médio de cada dificuldade para uma amostragem de oito blocos:

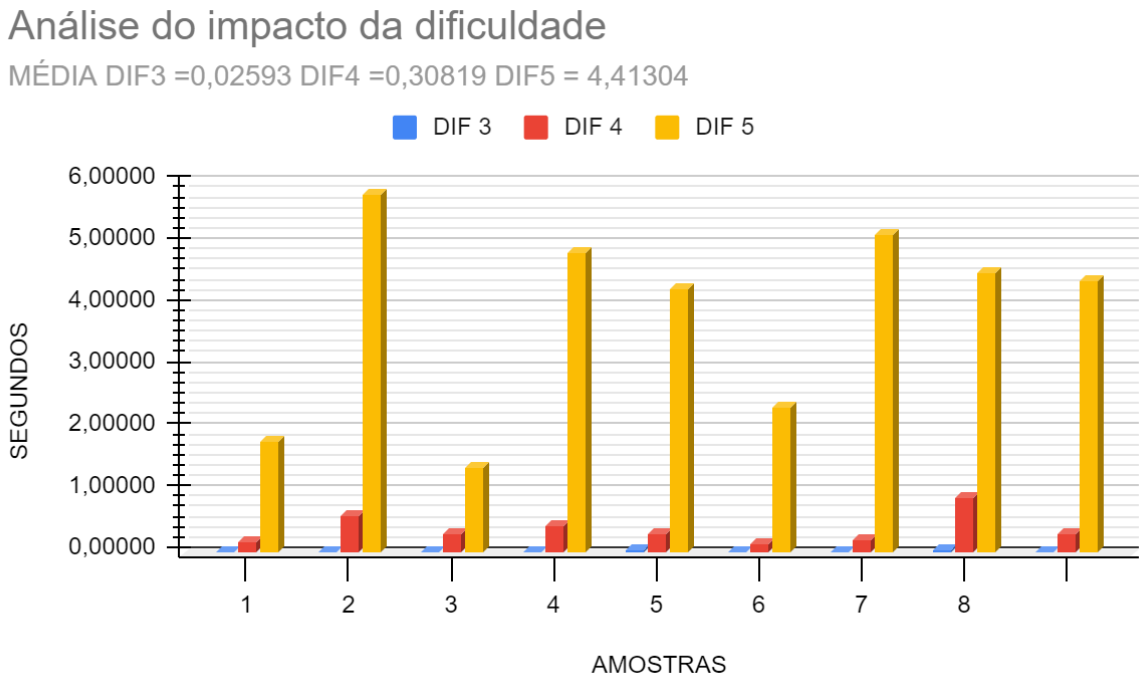

Figura 8

Observa-se um crescimento logaritmos do tempo de acordo com o acréscimo gradual da dificuldade, o que é uma característica do sistema já que tem ligação direta com a dificuldade para alteração dos blocos.

Conclusão

E como melhoria sugere adicionar eventos de verificação try/catch e mensagens de erros como as de inserção de valores e tipos inválidos; Para evitar repetição de código sugere criar uma função que calcula o hash para ser usada na função ‘minera’ e na ‘check_integridade’;

Bibliografia

Making a menu in Python <https://www.youtube.com/watch?v=63nw00JqHo0>

Como Minerar Bitcoin com Python (Código em Python para Minerar Bitcoin) <https://www.youtube.com/watch?v=m3k4kvX6izo>

Blockchain with Python #2: Creating blocks & checking integrity | Python projects <https://www.youtube.com/watch?v=sYwPEenxMN0>