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 1 para a criação do bloco de origem.
- Em seguida na opção de número 2 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 e uma breve descrição:

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

- Ao selecionar a opção 1 é efetuada a chamada da função ‘cria_prim_bloco’ que escreve em um arquivo JSONas informações pré determinadas do bloco de origem.
- Ao selecionar a opção 2 é efetuada a chamada da função ‘cria_bloco’ que passa como parâmetro as variáveis: remetente, destinatário, transação e dificuldade. Sendo a dificuldade a quantidade de zero que deve ter no início do 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 número do último 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 entrar em um loop while que inicialmente 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 e soma um na variável ‘nonce’ e permanece no laço while
    - Retorna para a função ‘cria_bloco’ que finaliza a contagem de tempo de mineração e então escreve em um novo bloco válido as informações recebidas por parâmetro.
- Ao selecionar a opção 3 é efetuada a chamada da função ‘check_integridade’
    - Acessa a pasta com os blocos e os ordena de forma crescente por meio da função ‘sorted’ e do parâmetro lambda
    - Por meio de um laço de repetição guarda na variável ‘blocos’ o conteúdo de um bloco

    - Extrai as informações de hash do bloco anterior e seu nome de arquivo'
    - Chama a função 'minera' que recalcula o hash do bloco
    - Compara o hash recalculado com o gravado no bloco
        - Se for igual retorna que o bloco não foi alterado, caso contrário mostra a informação que o bloco foi alterado
- Ao digitar qualquer valor que não seja no intervalo entre 1 a 3 o programa é encerrado e a mensagem de 'Opção inválida' é exibida

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:

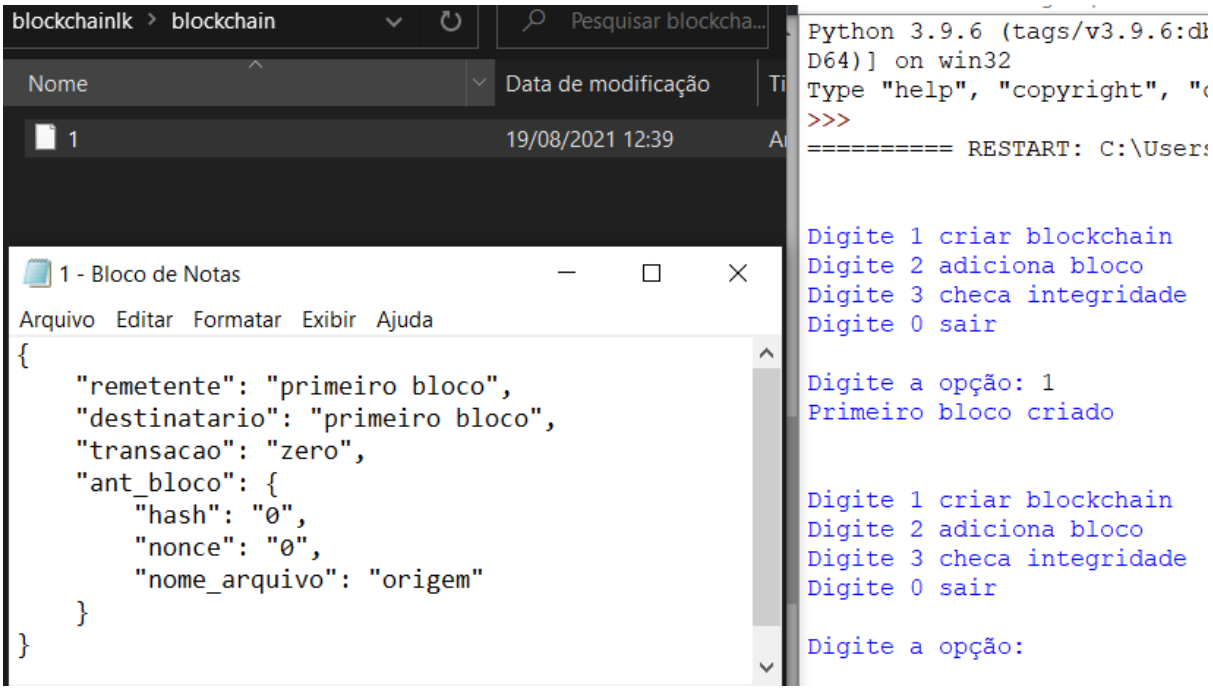

Figura 1

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:

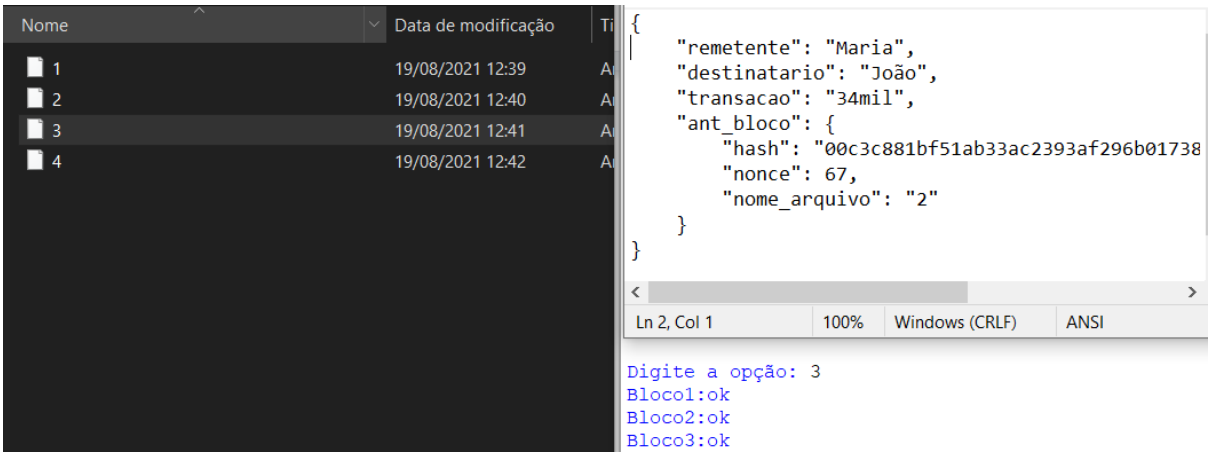

Figura 3

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

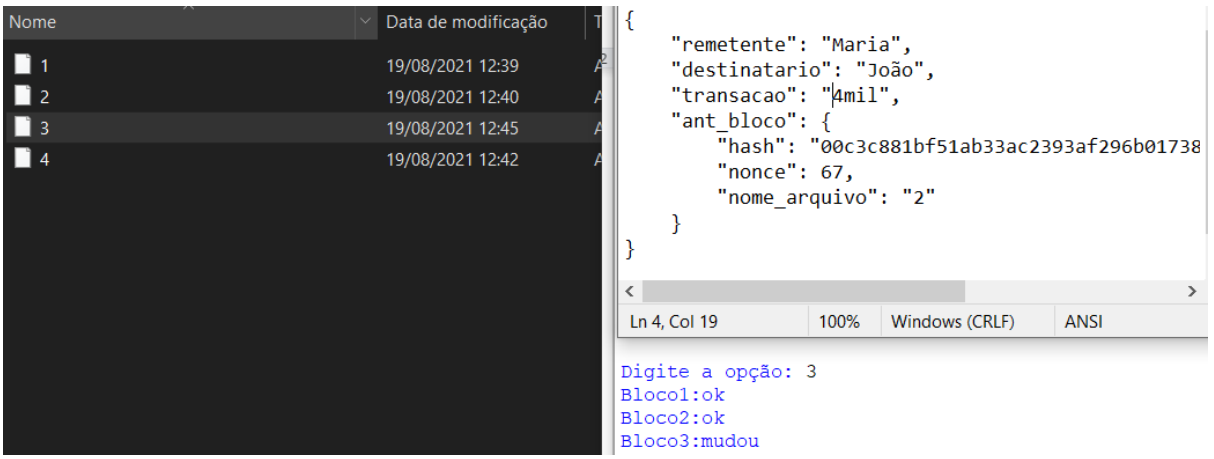

Figura 4

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

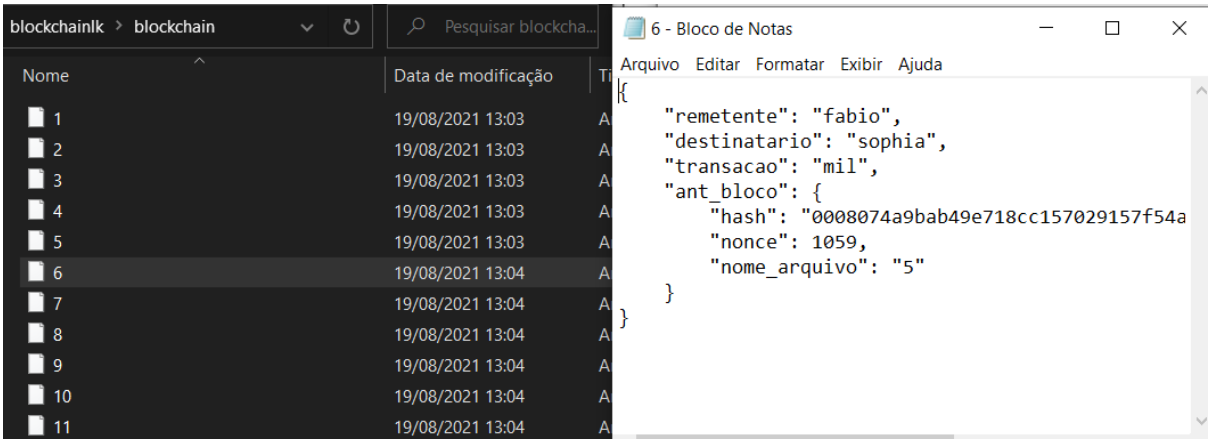

Figura 5

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

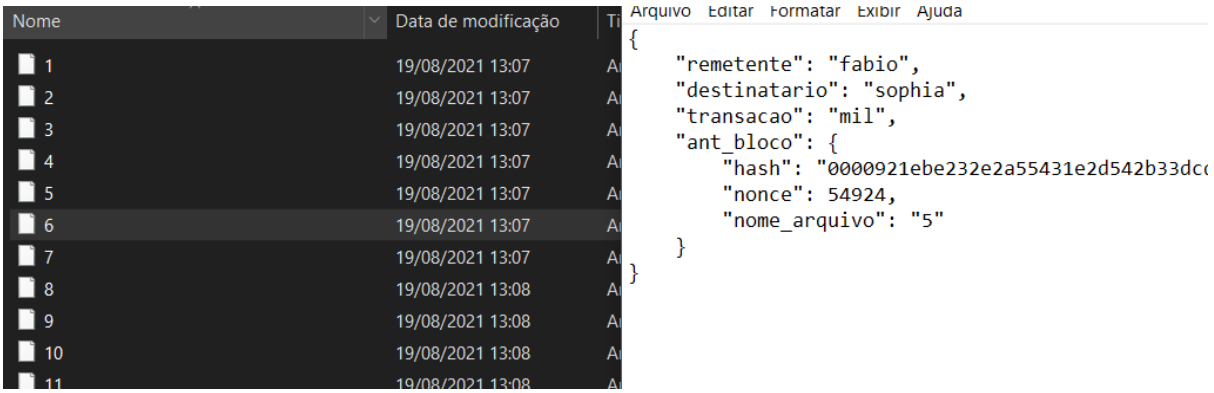

Figura 6

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

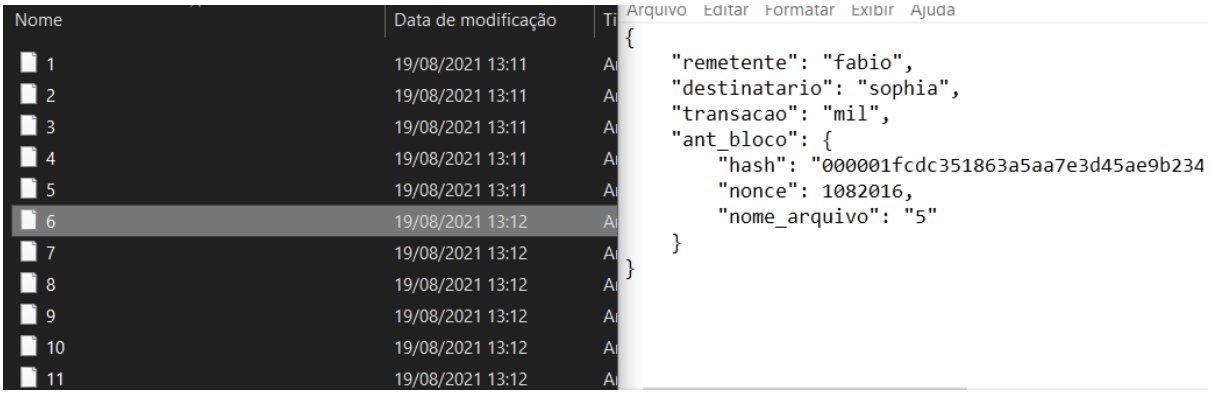

Figura 7

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:

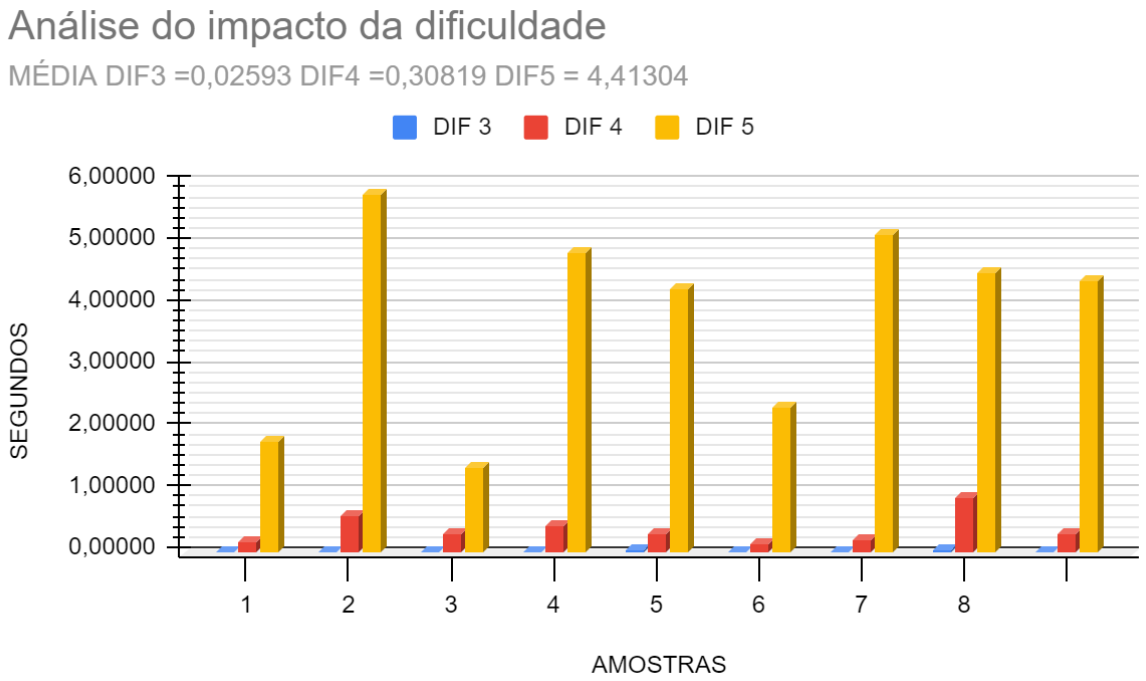

Figura 8

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

Os objetivos desta atividade foram cumpridos como demonstrado ao longo do relatório, o que demonstra que o código desenvolvido exemplifica de uma forma simples e clara o funcionamento de uma blockchain. 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’; No trecho ‘sorted’ da função ‘check_integridade’ sugere testar o uso ‘with open’; Em ‘cria_bloco’ ‘conta_bloco’ pode ser passada direto como parâmetro em ‘ant_bloco’

Bibliografia

Making a menu in Python <https://www.youtube.com/watch?v=63nw00JqHo0> Acesso em: 21 de agosto de 2021

Como Minerar Bitcoin com Python (Código em Python para Minerar Bitcoin) <https://www.youtube.com/watch?v=m3k4kvX6izo> Acesso em: 21 de agosto de 2021

Blockchain with Python #2: Creating blocks & checking integrity | Python projects <https://www.youtube.com/watch?v=sYwPEenxMN0> Acesso em: 21 de agosto de 2021