Atividade 2 – Blockchain e Proof of Work

Instruções de execução e tecnologias usadas

Foram utilizadas as linguagens *HTML*, *CSS* e *Javascript* (Node.js v12.18.3) com o uso do *JQuery-3.5.1*. Para executar o programa, todos os arquivos descompactados devem estar na mesma pasta e então é só iniciar o arquivo "atividade2.html". A função presente no arquivo "*jquery-3.5.1.js*" e a função *sha256* no arquivo "*js.js*" foram pegadas prontas e se encontram nas referências bibliográficas ao final desse texto.

Lógica implementada e como o programa funciona

Assim que o programa é executado, é preciso inserir a dificuldade desejada para a *Blockchain*. É possível inserir apenas números inteiros nesse campo uma vez que é feita uma validação. Depois de definida, não é possível mais alterar a dificuldade. Para isso, é preciso apertar o botão "Recomeçar".

Após essa definição, é possível visualizar um campo de texto no qual é requerido um dado para que o bloco seja criado. Nesse campo, é possível inserir qualquer caractere, só não é possível deixá-lo em branco, pois é feita uma validação de campo. Ao clicar no botão "Criar Bloco", o Bloco criado é exibido na tela. Dessa mesma forma, é possível inserir quantos blocos o usuário desejar. Essa inserção foi implementada via *JQuery*, pelo qual são definidos e escritos no *HTML* os valores de *index*, de *hash* anterior e também são chamadas as funções que calculam o *hash* (SHA-256) do bloco e também seu *nonce*.

O programa nunca gerará um bloco inválido. A única forma de tornar um bloco inválido é modificando seu dado. O dado de cada bloco é exibido em um campo de texto e, para alterá-lo, basta reescrevê-lo. Para que a alteração seja concluída, é necessário apertar o botão "Atualizar, Validar & Minerar Bloco" abaixo do bloco alterado. Se esse botão não for pressionado, a alteração não será feita. Uma vez pressionado, esse botão chama a função "*validaChain*" que é responsável por fazer a atualização do novo dado inserido e por percorrer toda a *Blockchain* a fim de validá-la. É sua função identificar se o *hash* de cada bloco bate com o *hash* calculado para cada um deles e também se o *hash* anterior dele de fato bate com o *hash* do bloco anterior. Uma vez que esses problemas são identificados, a função dispara um *alert* que mostra quais blocos estão inválidos e então atualiza seus *hashs*, seus *hashs* anteriores e também os minera novamente, gerando um novo valor de *nonce* para eles. Esses novos valores são escritos no *HTML* usando o *JQuery*, em substituição aos valores anteriormente registrados. Ao fim de todo esse processo, é exibido um novo *alert* no qual é informado que a *Blockchain* é válida.

Resultados obtidos com a validação e com o tempo de execução para diferentes dificuldades

A Figura 1 mostra uma *Blockchain* composta por quatro blocos sendo que nenhum foi alterado. É possível identificar que todos os *hashs* anteriores coincidem com o *hash* do último bloco. A imagem foi editada para que ficasse menor, entretanto, o programa exibe os blocos um abaixo do outro e não ao lado.

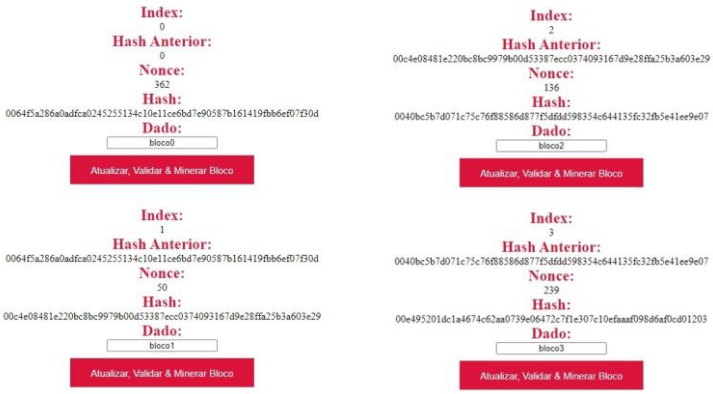

Figura 1 – *Blockchain* composta por quatro blocos não alterados

A Figura 2 mostra os *alerts* que são disparados uma vez que o dado do Bloco 0 é alterado. Eles foram colocados lado a lado na edição.

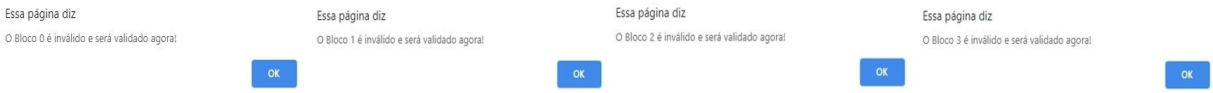

Figura 2 – *Alerts* disparados com a alteração do dado no Bloco 0

A Figura 3 exibe a nova *Blockchain* validada, atualizada e novamente minerada após a alteração realizada. É possível visualizar que os *nonces* e os *hashs* foram atualizados e que os *hashs* anteriores batem com os *hashs* dos últimos blocos.

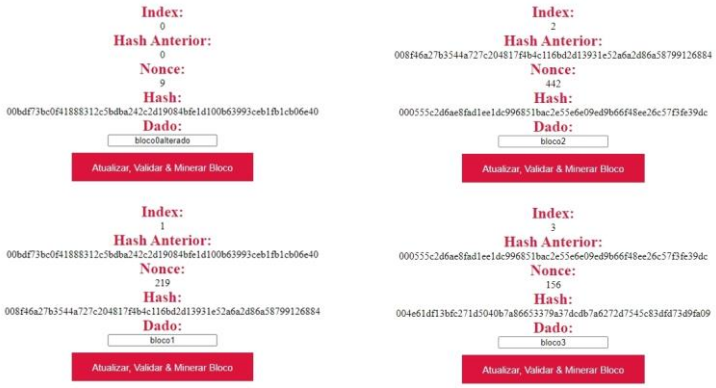

Figura 3 – *Blockchain* atualizada, validada e minerada

Por fim, foram realizados testes escolhendo as dificuldades 2, 3 e 4 a fim de comparar os diferentes tempos de mineração. Foram realizadas 10 minerações por dificuldade. O resultado obtido pode ser visto no Gráfico 1.

Gráfico 1 – Gráfico Dificuldade x Tempo de Execução

Analisando o Gráfico 1, é possível perceber uma grande diferença de tempo de execução da função "*mineraBloco*" de acordo com o aumento da dificuldade estipulada. Isso ocorre porque quanto maior a dificuldade, mais difícil é encontrar um *nonce* que gere um *hash* válido considerando que o número de *hashs* válidos é inversamente proporcional ao aumento da dificuldade. Esses resultados foram obtidos a partir de um computador com um processador Intel® Core™ i7-7500U CPU @ 2.70GHz 2.90GHz com 8GB de memória RAM.

Bibliografia

GERAINTLUFF. **JavaScript SHA-256 demo**. Disponível em: <https://geraintluff.github.io/sha256/>. Acesso em: 28 de agosto de 2020.

JQUERY. **JQuery JavaScript Library**. Disponível em: <https://github.com/jquery/jquery>. Acesso em: 28 de agosto de 2020.