Aula 8 TP – 25/março/2019

Grupo 3

1. *Blockchain*

- ***Pergunta 1.1***

Na experiência 1.1, foi alterado o método que cria o *Genesis Block*, de modo a que o *timestamp* fosse a data do dia em que realizamos este trabalho e o dado incluído nesse Bloco fosse "*Bloco inicial da koreCoin*".

A seguinte imagem ilustra a alteração realizada:

```
class Blockchain{
    constructor(){
        this.chain = [this.createGenesisBlock()];
    }

    createGenesisBlock(){
        return new Block(0, "25/03/2019", "Bloco inicial da koreCoin", "0");
    }

    getlatestBlock(){
        return this.chain[this.chain.length - 1];
    }
```

- ***Pergunta 1.2***

Em resposta a esta questão foram adicionados ao código anterior simulando assim várias transações em cada um deles.

```
koreCoin.addBlock(new Block (1, "01/01/2018", {amount: 20}));
koreCoin.addBlock(new Block (2, "02/01/2018", {amount: 40}));
koreCoin.addBlock(new Block (3, "02/01/2018", {amount: 40}));
koreCoin.addBlock(new Block (4, "02/01/2018", {amount: 40}));
koreCoin.addBlock(new Block (5, "02/01/2018", {amount: 40}));
```

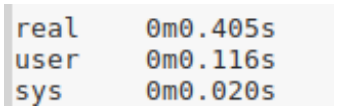
2. *Poof of work Consensus Model*

- ***Pergunta 2.1***

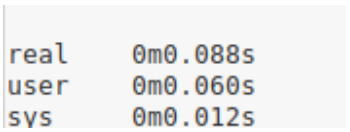
Na experiência 2.1, foi alterada a dificuldade de minerar para 2 e utilizando o comando *time* do *Linux* (*time node main.experiencia2.1.js*). Foram repetidos os exemplos para dificuldade de minerar 3, 4 e 5.

Os resultados de cada uma das alterações realizadas encontram-se nas seguintes imagens:

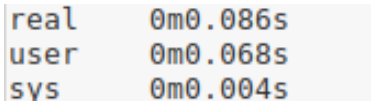
> Dificuldade 2



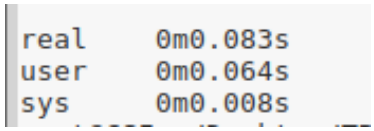
> Dificuldade 3



> Dificuldade 4



> Dificuldade 5



Como podemos observar nas imagens, vemos que à medida que a dificuldade aumenta, o tempo de execução também aumenta.

● ***Pergunta 2.2***

1) Na experiência anterior, o algoritmo de *'proof of work'* é o seguinte:

```
def proof_of_work(last_proof):
  # Create a variable that we will use to find
  # our next proof of work
  incrementor = last_proof + 1
  # Keep incrementing the incrementor until
  # it's equal to a number divisible by 9
  # and the proof of work of the previous
  # block in the chain
  while not (incrementor % 9 == 0 and incrementor % last_proof == 0):
    incrementor += 1
  # Once that number is found,
  # we can return it as a proof
  # of our work
  return incrementor
```

2) O algoritmo *proof of work* não é adequado para minerar, uma vez que o *miner* que publicar o último bloco tem vantagem sobre os outros na publicação dos seguintes, ou seja, consegue pré-computar os próximos valores de prova, porque não existe nenhum factor de aleatoriedade.