RELATÓRIO DE FUNCIONALIDADES

1. Simular uma rede onde diferentes “nós” trocam informações sobre blocos e transações.

2. Aprender a resolver conflitos (forks) e garantir que todos os nós concordem com a mesma cadeia.

3. Implementar um sistema que controla os saldos dos endereços e garante que só transações válidas sejam processadas.

4. Adicionar taxas de transação e recompensas para mineradores

IMPLEMENTAÇÕES

OBSERVAÇÃO INICIAL: A lógica do projeto está baseado em 5 classes do package model:

- Node.java

- Wallet.java

- Blockchain.java

- Block.java

- Transaction.java

1- **Foi criado uma classe Node.java que possui métodos de propagar um novo bloco recém minerado para todos os pares, receber blocos dos pares, e se necessário receber e enviar a blockchain inteira para o nó vizinho que solicitar.**

2- **Sempre que um bloco é recebido em nó, enviado por um vizinho, na classe Node.java, é verificado se o ID do bloco recebido é maior que o ID do bloco mais recente na blockchain do nó.**

- **Caso seja menor ou igual, o nó ignora o bloco recebido e não faz nada, fazendo valer a regra de maior cadeia.**

- **Se o ID for maior em apenas 1, ele primeiro checa o hash do bloco, depois verifica se a dificuldade do bloco está correta e por fim verifica se o hash anterior do bloco é o mesmo que o hash do bloco mais recente de sua blockchain.**
- **Tudo isso sendo confirmado, ele certifica que o bloco é válido e é um bloco novo em relação a sua blockchain, ou seja está um bloco atrasado, então ele o adiciona a mesma e atualiza os saldos das carteiras confirmando as transações, como se faz normalmente quando minera um bloco.**
- **Casa uma das validações não for confirmada, ele recusa o bloco e ignora a transmissão do vizinho.**

- **Por fim, se o ID do bloco recebido for maior em mais que 1 em relação ao seu bloco mais recente, o nó percebe que está 2 blocos ou mais atrasado, logo o nó pedirá ao vizinho a sua blockchain completa.**
- **Recebendo a blockchain o nó executará a validação completa da mesma, método da classe Blockchain.java, casa a validade da blockchain for confirmada, o nó irá substituir a sua blockchain pela a blockchain recebida do nó vizinho.**

3- **Cada blockchain tem uma cópia das carteiras, que um lista de objetos da classe Wallet.java, sempre que um novo bloco é minerado ele é adicionado a uma blockchain, que por sua vez atualiza o saldo das carteiras de acordo com cada transação contida no bloco, garantido que só as transações válidas sejam contabilizadas em cada carteira contida em cada blockchain. Quando uma blockchain é transmitida para outro nó, ela obviamente leva consigo as informações sobre as carteiras contidas nela, incluindo o saldo de cada uma.**

4- **Cada transação é um objeto da classe Transaction.java, que por sua vez, quando instanciada, calcula uma taxa de 5% do valor da transação (menos para transações de mineradores, que é a COINBASE e a FEES), armazena o valor da taxa e subtrai do valor da transação. Quando o nó vai minerar o bloco, ele soma todas as taxas de transações que está minerando e cria uma nova transação: a FEES, e direciona a sua carteira, assim como a coinbase. Caso o nó consiga minerar o bloco, quando adicionar na blockchain o novo bloco, essas transações serão confirmadas e o nó minerador irá receber sua recompensa.**