UNIVERSIDAD NEBRIJA

GRUPO: Grupo 3 David Martínez y Javier Navas
ALUMNO: David Martínez Campos

Asignatura: Programación de Sistemas Distribuidos

Curso: 2021/2022 Fecha: 10-04-2022
Semestre: 2º

PRÁCTICA 4: Blockchain con ruby
Esta práctica es grupal, para la entrega, tenéis que daros de alta como grupo en la asignatura de prácticas y que entregue la práctica el responsable del grupo. Si algún miembro del grupo no participa, el responsable debe comunicarlo en la entrega.

1. **Del siguiente repositorio https://github.com/apradillap/simple-blockchain-in-ruby**
   a) **Ejecuta la blockchain**
   b) **Analiza cómo funciona y añade capturas de pantalla para mostrar evidencias de la compresión de la ejecución.**

```
Enter your name for the new transaction
Javier Navas

What do you want to send ?
NFT Gorilas

How much quantity ?
1

Who do you want to send it to ?
David Martinez

Do you want to make another transaction for this block ? (Y/n)
n
"============================"
#<Block:0x000001fc86020c18
 @hash="00bb651190c02d59ac36c6f088898e877a7972f232a40c11bfdcd404fa35e724",
 @index=3,
 @nonce=331,
 @previous_hash="00bfccc253c7f261b7e76b2c49f067eec0fd68231d15e17cbee619d356fb5bfb",
 @timestamp=2022-05-05 09:54:45.0985726 +0200,
 @transactions=[{:from=>"Javier Navas", :to=>"David Martinez", :what=>"NFT Gorilas", :qty=>"1"}],
 @transactions_count=1>
"============================"
```

Este programa es una simulación de una blockchain. El programa empieza pidiendo el nombre del usuario que tiene que hacer la transición, el elemento a transferir y el numero de copias de ese archivo que deseas guardar, por último, te pide los datos de la blockchain a la que lo quieras mandar. Dichos datos se guardan y simulan la transacción.

2. **¿Qué es el bloque génesis?**

El bloque Genesis es el primer bloque de un blockchain. Sirve para inicializar la criptomoneda.

3. **¿Qué elementos tiene cada bloque?**

   Cada bloque va a tener un índice, un timestramp, una transacción, número de transacciones y el numero hash
4. **¿Qué hace el método def compute_hash_with_proof_of_work? ¿Qué significa Proof of work?**
   Este código va computando los bloques hasta que llegue al bloque inicial. El Proo of work será la blockchain donde tenemos almacenados todos los datos.

5. **Haz un fork de la práctica. ¿Te atreves a realizar algún cambio? Si no se te ocurre ninguno, ¿podrías añadir a cada bloque no solo el hash del bloque anterior, si no del anterior del anterior? Se valorarán más las propuestas ingeniosas.**

```ruby
class Block
  attr_reader :index, :timestamp, :transactions,
                      :transactions_count, :previous_hash, :previous_previous_hash,
                      :nonce, :hash

  def initialize(index, transactions, previous_hash, previous_previous_hash)
    @index                      = index
    @timestamp                = Time.now
    @transactions             = transactions
        @transactions_count  = transactions.size
    @previous_hash            = previous_hash
    @previous_previous_hash  = previous_previous_hash
    @nonce, @hash             = compute_hash_with_proof_of_work
  end

    def compute_hash_with_proof_of_work(difficulty="00")
        nonce = 0
        loop do
            hash = calc_hash_with_nonce(nonce)
            if hash.start_with?(difficulty)
                return [nonce, hash]
            else
                nonce +=1
            end
        end
    end

  def calc_hash_with_nonce(nonce=0)
    sha = Digest::SHA256.new
    sha.update( nonce.to_s +
                          @index.to_s +
                          @timestamp.to_s +
                          @transactions.to_s +
                          @transactions_count.to_s +
                          @previous_hash +
                          @previous_previous_hash    )
    sha.hexdigest
  end

  def self.first( *transactions )    # Create genesis block
    ## Uses index zero (0) and arbitrary previous_hash ("0")
    Block.new( 0, transactions, "0" )
  end

  def self.next( previous, transactions )
    Block.new( previous.index+1, transactions, previous.hash )
  end
end  # class Block
```