Clase Block

Un bloque es donde se guarda la información en blockchain y se cifra.

En nuestro directorio de trabajo vamos a crear un archivo llamado block.py. En él, vamos a crear una nueva clase llamada Block.

In [5]:
```
class Block:
    pass
```

Para crear un bloque siempre vamos a necesitar 3 variables iniciales:

- El hash del bloque anterior (Con él se hace la vinculación entre bloque).
- Una lista que contenga las transacciones pendientes.
- El número del bloque.

In [1]:
```
class Block:
    def __init__(self, preivious_hash: str,
                 list_of_transactions: list,
                 block_number: int):
        pass
```

Nuestra clase Block va a contar con los siguientes atributos:

- Número del bloque: Altura del bloque actual en la cadena.
- Hash del bloque anterior: Firma digital del bloque anterior en la cadena.
- Lista de transacciones: Transacciones contenidas y procesadas.
- Hash del bloque actual: Firma digital del bloque actual.
- Timestamp: Hora en la que el bloque se añadió a la cadena de bloques.
- Nonce: Número mágico que resuelve el "acertijo" en el consenso Proof of Work.

In [9]:
```
class Block:
    def __init__(self, previous_hash: str, list_of_transactions: list, block_number
        self.block_number = block_number
        self.previous_hash = previous_hash
        self.list_of_transactions = list_of_transactions
        self.nonce = 0
        self.hash = 0
        self.time_stamp = datetime.now().strftime("%d/%m/%Y %H:%M:%S")
        # Cuando se instancia un bloque, le asigna el blocknumber a cada transacci
        for tx in self.list_of_transactions:
            tx.block = block_number
        # atributos utilizados con Proof of Stake.
        self.forger = None
```

Los métodos van a darnos utilidad para la blockchain, o para efectos de visualización. La clase Block va a tener los siguientes métodos:

In [2]:
```
def get_block_header(self):
        """Funcion que retorna el encabezado del bloque."""
        return {
            'previous_block_hash':self.previous_hash,
            'nonce': self.nonce,
            'transactions':self.get_tx_in_format(),
        }
```

get_block_header regresa un diccionario que se utiliza para conseguir el hash de un bloque.

In [10]:
```
def get_block_header_pos(self):
        return {
            'previous_block_hash':self.previous_hash,
            'nonce': self.nonce,
            'transactions':self.get_tx_in_format(),
            'forger': self.forger.validator.account.identity
            }
```

get_block_header_pos regresa un diccionario añadiendo al forjador del bloque en el caso de un consenso Proof of Stake. Se utiliza tambien para obtener el hash de un bloque.

In [11]:
```
    def print_block_info(self):
        print("-------------")
        print("Bloque No: ", self.block_number)
        print("Transacciones: ")
        self.print_tx_in_format()
        print("Hash anterior: ", self.previous_hash)
        print("Hash actual: ", self.hash)
        print("Time stamp: ", self.time_stamp)

    def print_tx_in_format(self):
        for tx in self.list_of_transactions:
            print(
                f"- {tx.sender.nickname} send {tx.value} to {tx.recipient.nickname}

    def get_tx_in_format(self):
        tx_list = []
        for tx in self.list_of_transactions:
            tx_in_str = f"{tx.sender.nickname} send {tx.value} to {tx.recipient.nic
            tx_list.append(tx_in_str)
        return str(tx_list)
```

Las funciones anteriores solo imprimen información para visualizarla de mejor manera en la práctica.

El código completo queda de la siguiente manera:

In [4]:
```
from datetime import datetime, date
from bin.account import Account


class Block:
    def __init__(self, previous_hash: str, list_of_transactions: list, block_number
        self.block_number = block_number
        self.previous_hash = previous_hash
```

```
        self.list_of_transactions = list_of_transactions
        self.nonce = 0
        self.hash = 0
        self.time_stamp = datetime.now().strftime("%d/%m/%Y %H:%M:%S")
        # Cuando se instancia un bloque, le asigna el blocknumber a cada transacci
        for tx in self.list_of_transactions:
            tx.block = block_number
        # atributos utilizados con Proof of Stake.
        self.forger = None

    def get_block_header(self):
        """Funcion que retorna el encabezado del bloque."""
        return {
            'previous_block_hash':self.previous_hash,
            'nonce': self.nonce,
            'transactions':self.get_tx_in_format(),
        }

    def get_block_header_pos(self):
        return {
            'previous_block_hash':self.previous_hash,
            'nonce': self.nonce,
            'transactions':self.get_tx_in_format(),
            'forger': self.forger.validator.account.identity
            }

    def print_block_info(self):
        print("-------------")
        print("Bloque No: ", self.block_number)
        print("Transacciones: ")
        self.print_tx_in_format()
        print("Hash anterior: ", self.previous_hash)
        print("Hash actual: ", self.hash)
        print("Time stamp: ", self.time_stamp)

    def print_tx_in_format(self):
        for tx in self.list_of_transactions:
            print(
                f"- {tx.sender.nickname} send {tx.value} to {tx.recipient.nickname}

    def get_tx_in_format(self):
        tx_list = []
        for tx in self.list_of_transactions:
            tx_in_str = f"{tx.sender.nickname} send {tx.value} to {tx.recipient.nic
            tx_list.append(tx_in_str)
        return str(tx_list)
```

Con la clase Block lista, podemos pasar a la clase Account.