Ejemplo práctico

Caso: Proof of Work

Importamos la clase Account y Blockchain

In [2]:
```
from bin.account import Account
from bin.blockchain import Blockchain
```

Inicializar nuestra blockchain conlleva dos cosas:

1. Escoger el consenso con el que vamos a trabajar.
2. Inicializar el bloque génesis.

In [3]:
```
# Inicializamos nuestra cadena de bloques
print("### Inicializamos nuestra blockchain")
blockchain = Blockchain()  # Tambien se crea el bloque genesis.

# Escojemos que protocolo queremos en nuestra blockchain
blockchain.set_consensus('PoW')

# Inicializamos el bloque genes
blockchain.generate_genesis_block()
```

```
### Inicializamos nuestra blockchain
Inicializando bloque genesis...
Dentro de funcion mine...
Nonce Guess:  46378
Resultant Hash: 00001e6acd1d86b401d687712d322ab80a5af8f8f74214d7575cf629c52e135e
Decimal value of hash: 209931745916594597554099974976852832762162364941953807153725092211331934
```

Imprimimos el bloque génesis para revisar su creación.

In [4]:
```
blockchain.chain[0].print_block_info()
```

```
-------------
Bloque No:  0
Transacciones:
- Genesis0 send 0 to Genesis01
Hash anterior:  0
Hash actual:  00001e6acd1d86b401d687712d322ab80a5af8f8f74214d7575cf629c52e135e
Time stamp:  25/09/2022 17:21:29
```

Instanciaremos dos objetos Account que van a interactuar con la blockchain.

In [6]:
```
brian = Account(100, "Brian")
aaron = Account(100, "Aaron")
```

Con dos cuentas, podemos empezar a generar transacciones.

In [7]:
```
# Cada cuenta se instancio con 100 de balance, si nos pasamos, la transaccion no pa
blockchain.new_tx(brian, 120, aaron)
```

```
No tienes suficiente balance en tu cuenta.
```

In [8]:
```
# Cada cuenta se instancio con 100 de balance, si nos pasamos, la transaccion no pa
blockchain.new_tx(brian, 20, aaron)
```

```
Nueva transaccion detectada... Balance suficiente.
Estado: PENDIENTE
Firmando transaccion...
Verificando la firma de la transaccion...
La firma es valida.
Transaccion añadida a la espera.
### Creando nuevo bloque ###
### Bloque No.  1
En PoW
Dentro de funcion mine...
Nonce Guess:  175806
Resultant Hash: 0000972ca1031af0bbaff6eb93dab0aaaece8ee6923984d704ec489ce7566d55
Decimal value of hash: 104336689257510831139951500388979817668974780215209835373731169186704725

### Bloque creado. ###
```

La transaccion pasa por un mar de funciones dentro de la red.

1. La transacción se verifica.
2. Se añade a la espera.
3. Un nuevo bloque se mina y se adjuntan las transacciones en espera.
4. Se anexa a la red.

Observemos los cambios en la red.

In [52]:
```
# ¿Sí se transfierieron los activos?
brian.balance, aaron.balance
```

Out[52]:
```
(80, 120)
```

In [11]:
```
# Status de la transacción
last_transaction = brian.list_of_all_transactions[0]
last_transaction.status
```

Out[11]:
```
<TxStatus.CONFIRMADA: 1>
```

In [53]:
```
# Información de la transacción
last_transaction.to_dict()
```

Out[53]:
```
{'sender': 'Brian',
 'recipient': 'Aaron',
 'value': 20,
 'time': '25/09/2022 17:23:44'}
```

In [55]:
```
# También podemos acceder a la transacción desde los bloques
blockchain.chain[last_transaction.block].list_of_transactions[0].to_dict()
```

Out[55]:
```
{'sender': 'Brian',
 'recipient': 'Aaron',
 'value': 20,
 'time': '25/09/2022 17:23:44'}
```

Veamos la información del bloque a detalle.

In [58]:
```
blockchain.chain[-1].print_block_info()
```

```
-------------
Bloque No:  1
Transacciones:
- Brian send 20 to Aaron
Hash anterior:  00001e6acd1d86b401d687712d322ab80a5af8f8f74214d7575cf629c52e135e
Hash actual:  0000972ca1031af0bbaff6eb93dab0aaaece8ee6923984d704ec489ce7566d55
Time stamp:  25/09/2022 17:23:44
```

También podemos ver las firmas digitales hechas por las cuentas. Asi como, sus llaves, balance, etc.

In [57]:
```
blockchain.chain[-1].list_of_transactions[0].signature
```

Out[57]:
```
b'YqI\x83\xe2\x13\x94Z~\xa9\x0e2\xa14\x7f\xcb\x0f\xb6{\xea%\x07\xbb\xac\xa4\x17\x1
c\xca\xdfQ3y\xca\xd0\xe3g\x03\x0f\xe3\xf6\xa8\x81=\x0bV)\xd4\xa7FIe!\xf4W\x90\xa6
\x9d1\x9b\x90~\xfb\xd3\xd5/\x9b|\x95b\xe4\x86\x95\x8c\x88\x19\xdc\xeb\xa2\x80\xca
\xc4K\x94j\xce#\x7fh\x04\x81EKJ$lXX3g\xc9%\x16\xd5\nk\x99\xc4}\xec\x81\xc6\xf5\t]>
\x15\xa8jSc\xfe\xe0\x0e\x89\xb4\xb9\xaaz'
```

Es posible tambien ajustar el numero de transacciones que mantiene la blockchain en espera antes de ser ejecutadas.

In [59]:
```
blockchain.tx_limit_per_block = 3
```

In [60]:
```
blockchain.new_tx(aaron, 10, brian)
blockchain.new_tx(aaron, 30, brian)
blockchain.new_tx(brian, 10, aaron)
```

```
Nueva transaccion detectada... Balance suficiente.
Estado: PENDIENTE
Firmando transaccion...
Verificando la firma de la transaccion...
La firma es valida.
Transaccion añadida a la espera.
Nueva transaccion detectada... Balance suficiente.
Estado: PENDIENTE
Firmando transaccion...
Verificando la firma de la transaccion...
La firma es valida.
Transaccion añadida a la espera.
Nueva transaccion detectada... Balance suficiente.
Estado: PENDIENTE
Firmando transaccion...
Verificando la firma de la transaccion...
La firma es valida.
Transaccion añadida a la espera.
### Creando nuevo bloque ###
### Bloque No.  2
En PoW
Dentro de funcion mine...
Nonce Guess:  66743
Resultant Hash: 0000deebda7dd1c06da7f1c5390cf966403d77a9c04e78c7fa20b02bea8f45c4
Decimal value of hash: 153854628629294466645807724817771195071070517840276584062997
71981844825540

### Bloque creado. ###
```

In [61]:
```
blockchain.chain[-1].print_block_info()
```

```
-------------
Bloque No:  2
Transacciones:
- Aaron send 10 to Brian
- Aaron send 30 to Brian
- Brian send 10 to Aaron
Hash anterior:  0000972ca1031af0bbaff6eb93dab0aaaece8ee6923984d704ec489ce7566d55
Hash actual:  0000deebda7dd1c06da7f1c5390cf966403d77a9c04e78c7fa20b02bea8f45c4
Time stamp:  25/09/2022 17:38:04
```

Proof of stake

Igual que en Proof of Work, hay que inicializar nuestra blockchain.

El siguiente comando lo podemos utilizar para resetear las variables locales en Jupyter.

In [66]:
```
%reset
```

In [68]:
```
from bin.blockchain import Blockchain
from bin.account import Account, Validator
blockchain = Blockchain()

# Escojemos que protocolo queremos en nuestra blockchain
blockchain.set_consensus('PoS')

# inicializamos el bloque genesis
```

```
blockchain.generate_genesis_block()

blockchain.chain[0].print_block_info()
```

```
Inicializando bloque genesis...
Dentro de funcion mine...
Hash añadido al bloque genesis...
-------------
Bloque No:  0
Transacciones:
- Genesis0 send 0 to Genesis01
Hash anterior:  0
Hash actual:  7b36a0972430bcd8ee49cf3525d4ed88a7ce83e3c20097501fd086c7be62d3a5
Time stamp:  25/09/2022 17:39:44
```

In [69]:
```
# Creamos dos cuenta que interactuaran con la blockchain.
brian = Account(300, "Brian")
aaron = Account(300, "Aaron")
```

In [70]:
```
# Inicializamos 5 cuentas que quieran ser validadores
charles = Account(350, 'charles')
edwin = Account(500, 'edwin')
oliver = Account(200, 'oliver')
erick = Account(90, 'erick')
sonia = Account(275, 'sonia')
```

In [71]:
```
# Los incluimos en nuestra BlockChain
blockchain.set_validators((charles, edwin, oliver, erick, sonia))
```

In [72]:
```
# generamos y subimos las transacciones a la blockchain
blockchain.new_tx(brian, 50, aaron)
```

```
Nueva transaccion detectada... Balance suficiente.
Estado: PENDIENTE
Firmando transaccion...
Verificando la firma de la transaccion...
La firma es valida.
Transaccion añadida a la espera.
### Creando nuevo bloque ###
### Bloque No.  1
En PoS:
Acumulando los tokens de los validadores en el servidor actual...
1325 - tokens acumulados.
Revolviendo la lista...
Lista revuelta!
charles 350
edwin 500
oliver 200
sonia 275
El forjador del nuevo bloque sera... edwin
Testigos:  ['charles', 'oliver', 'sonia']
Forjador:  ['edwin']
El forjador esta verificando las tx.
Verificando la firma de la transaccion...
La firma es valida.
{'sender': 'Brian', 'recipient': 'Aaron', 'value': 50, 'time': '25/09/2022 17:42:5
8'}
### FIRMANDO EL BLOQUE
### BLOQUE FIRMADO Y CON SU HASH ---
### Enviando el bloque a la red...
### Iniciando atestiguamiento del bloque...
charles: Confirmo que la firma del bloque es correcta.
oliver: Confirmo que la firma del bloque es correcta.
sonia: Confirmo que la firma del bloque es correcta.
-----------------
Confirmaciones suficientes para anadir el bloque.
### Bloque creado. ###
```

La transacción pasa por aun más mar de transacciones.

1. Se verifica la firma de la transacción.
2. La transacción se añade a la espera.
3. Si no hay validadores, se crean.
4. Con validadores, se selecciona al forjador y a los testigos.
5. El forjador crea un bloque y adjunta las transacciones verificándolas. Lo firma con su llave privada y lo envía a los testigos.
6. Los testigos lo reciben, y revisan que el trabajo realizado por el forjador este bien.
7. Si la mayoría esta de acuerdo, si llega a una acuerdo y el bloque pasa a ser anexado a la red.
8. Se confirman las transacciones en la red.

Vamos a explorar un poco lo que obtuvimos.

In [73]:
```
# Si se transfieren los activos?
brian.balance, aaron.balance
```

Out[73]:
```
(250, 350)
```

In [74]:
```
# Status de la transaccion
last_transaction = brian.list_of_all_transactions[0]
last_transaction.status
```

Out[74]:
```
<TxStatus.CONFIRMADA: 1>
```

In [75]:
```
# Informacion de la transaccion
last_transaction.to_dict()
```

Out[75]:
```
{'sender': 'Brian',
 'recipient': 'Aaron',
 'value': 50,
 'time': '25/09/2022 17:42:58'}
```

Veamos la informacion del bloque a detalle.

In [77]:
```
blockchain.chain[-1].print_block_info()
```

```
-------------
Bloque No:  1
Transacciones:
- Brian send 50 to Aaron
Hash anterior:  7b36a0972430bcd8ee49cf3525d4ed88a7ce83e3c20097501fd086c7be62d3a5
Hash actual:  ecf72c08ef1ca765315abc4100cba8fa13875bf67713d5a02612489a9f7edbda
Time stamp:  25/09/2022 17:42:58
```

Tambien podemos ver las firmas digitales hechas por las cuentas. Asi como, sus llaves, balance, etc.

In [78]:
```
blockchain.chain[-1].list_of_transactions[0].signature
```

Out[78]:
```
b'\xc1i\n\xfd%y\xccojZ\xb2\x983`;\xad\xf6Z\xa0\x9aZX\x86[\x85\xea\xde\x075\xb3"\xd
3\x0b\x82\xfcr6\xd9c+\r\x13KF\x9c\x9aO\x9b\xc4\xc3,\xe5>\x1a)\xdb#\x15\xa4\x8c\x01
\r5\xaf\xd9\xb5\xbe\x0e\x92\x81\xc5\xec\x941\xe1\x9eD\x0e"\xec%>\xd0\xc6]\'\xa6\xf
1o\x03\xec\rl\xdb\x9fxU\xe2\x04\x10\xcab\xd5z\x12\x00\xb0\x08m\x0c\xa0\xbb\xb3>[z3
\xd6\xb3\\\x86iN\xb7\xdf7C\x84'
```

In [79]:
```
blockchain.chain[-1].print_block_info()
```

```
-------------
Bloque No:  1
Transacciones:
- Brian send 50 to Aaron
Hash anterior:  7b36a0972430bcd8ee49cf3525d4ed88a7ce83e3c20097501fd086c7be62d3a5
Hash actual:  ecf72c08ef1ca765315abc4100cba8fa13875bf67713d5a02612489a9f7edbda
Time stamp:  25/09/2022 17:42:58
```

En este caso, podemos observar quien fue el forjador del bloque.

In [80]:
```
blockchain.chain[-1].forger.validator.account.nickname
```

Out[80]:
```
'edwin'
```

In [88]:
```
blockchain.chain[-1].forger.block_signature
```

Out[88]:
```
b'\xad\x18\xd1\xab"8\x9cn\x13\xcdK\x06?kW=P>\xc3\xb7\xd0yG\xbf$\xabep\xad6\x88\xcf
rLf\xe9t\x171\xbc\xea?\x9b\xae\xb8E\xcb\xc5\x14\x0b\xa3\x90Xak\xf9PU\x8d\x98\x9b\x
d7\xd8\xed\x9c#&\xaa\xb3|\xfb\x87\x11\xed\x89\xaeV5\r\x1d\xb2Dy1\x8e\xd0Q\\\x7f\xc
0\xb6\x8a{\xfe\xea\x8e\x83G4:\x01\x11\x92-&o\xfbC{t\xfbp\xa1\xcb\xdd\x0e.\xd8\xfc%
\xd8\xb9\x8ac\xa0V1u'
```

Es posible tambien ajustar el numero de transacciones que mantiene la blockchain en espera antes de ser ejecutadas.

In [89]:
```
blockchain.tx_limit_per_block = 3
```

In [90]:
```
blockchain.new_tx(aaron, 10, brian)
blockchain.new_tx(aaron, 30, brian)
blockchain.new_tx(brian, 10, aaron)
```

```
Nueva transaccion detectada... Balance suficiente.
Estado: PENDIENTE
Firmando transaccion...
Verificando la firma de la transaccion...
La firma es valida.
Transaccion añadida a la espera.
Nueva transaccion detectada... Balance suficiente.
Estado: PENDIENTE
Firmando transaccion...
Verificando la firma de la transaccion...
La firma es valida.
Transaccion añadida a la espera.
Nueva transaccion detectada... Balance suficiente.
Estado: PENDIENTE
Firmando transaccion...
Verificando la firma de la transaccion...
La firma es valida.
Transaccion añadida a la espera.
### Creando nuevo bloque ###
### Bloque No.  2
En PoS:
Acumulando los tokens de los validadores en el servidor actual...
1325 - tokens acumulados.
Revolviendo la lista...
Lista revuelta!
charles 350
edwin 500
oliver 200
sonia 275
El forjador del nuevo bloque sera... sonia
Testigos:  ['charles', 'edwin', 'oliver']
Forjador:  ['sonia']
El forjador esta verificando las tx.
Verificando la firma de la transaccion...
La firma es valida.
{'sender': 'Aaron', 'recipient': 'Brian', 'value': 10, 'time': '25/09/2022 17:45:2
3'}
Verificando la firma de la transaccion...
La firma es valida.
{'sender': 'Aaron', 'recipient': 'Brian', 'value': 30, 'time': '25/09/2022 17:45:2
3'}
Verificando la firma de la transaccion...
La firma es valida.
{'sender': 'Brian', 'recipient': 'Aaron', 'value': 10, 'time': '25/09/2022 17:45:2
3'}
### FIRMANDO EL BLOQUE
### BLOQUE FIRMADO Y CON SU HASH ---
### Enviando el bloque a la red...
### Iniciando atestiguamiento del bloque...
charles: Confirmo que la firma del bloque es correcta.
edwin: Confirmo que la firma del bloque es correcta.
oliver: Confirmo que la firma del bloque es correcta.
-----------------
Confirmaciones suficientes para anadir el bloque.
### Bloque creado. ###
```

In [91]:
```
blockchain.print_full_chain()
```

```
-------------
Bloque No:  0
Transacciones:
- Genesis0 send 0 to Genesis01
Hash anterior:  0
Hash actual:  7b36a0972430bcd8ee49cf3525d4ed88a7ce83e3c20097501fd086c7be62d3a5
Time stamp:  25/09/2022 17:39:44
-------------
Bloque No:  1
Transacciones:
- Brian send 50 to Aaron
Hash anterior:  7b36a0972430bcd8ee49cf3525d4ed88a7ce83e3c20097501fd086c7be62d3a5
Hash actual:  ecf72c08ef1ca765315abc4100cba8fa13875bf67713d5a02612489a9f7edbda
Time stamp:  25/09/2022 17:42:58
-------------
Bloque No:  2
Transacciones:
- Aaron send 10 to Brian
- Aaron send 30 to Brian
- Brian send 10 to Aaron
Hash anterior:  ecf72c08ef1ca765315abc4100cba8fa13875bf67713d5a02612489a9f7edbda
Hash actual:  4e10a081c6594eed51c82bd49d71a47fb4343d180d6854d27d9943adbf7cb3e6
Time stamp:  25/09/2022 17:45:23
```

Llegamos al final de los ejercicios!