Ejemplo de blockchain sencillo

0. Inicialización del notebook

Importación de librerias

In [12]:
```
import hashlib
import json
```

Variables

In [13]:
```
# Datos del bloque génesis
dato1_bloque_genesis = 'Welcome to Blockchain Demo 2.0! - Clave publica: 0000 - Fecha:
2025-10-07'
```

In [14]:
```
class Block:
    """
    Clase Block para crear bloques correspondientes a la cadena de bloques. Se
inicializa con:
    - index: indice del bloque (secuencial)
    - data: datos a grabar dentro del bloque
    - prior_hash: hash correspondiente al bloque anterior
    - hash: hash de este bloque
    """
    def __init__(self, index, data, prior_hash=''):
        self.index = index
        self.data = data
        self.prior_hash = prior_hash
        self.nonce = 0
        self.hash = self.create_hash()

    def create_hash(self):
        # Devuelve el hash correspondiente al concatenado de los datos
        return hashlib.sha256(f"{self.index}{self.prior_hash}{self.data}
{self.nonce}".encode()).hexdigest()

    def mine_block(self, difficulty):
        # Bucle hasta que el hash comience por un numero determinado de ceros
        while self.hash[:difficulty] != '0' * difficulty:
            self.nonce += 1
            self.hash = self.create_hash()
            # print('Block Hash: ' + self.hash)

class BlockChain:
    """
    Clase BlockChain para crear la cadena de bloques. Se inicializa con:
    - chain: lista de bloques que componen la cadena de bloques
    - difficulty: dificultad para la mineria de bloques
    """
    def __init__(self):
```

```
        self.chain = [self.create_genesis_block()]
        self.difficulty = 4

    def create_genesis_block(self):
        # Crea el bloque génesis con datos predefinidos
        return Block(0, dato1_bloque_genesis, '0')

    def get_last_block(self):
        return self.chain[-1]

    def add_block(self, new_block):
        # Asigna el hash del bloque anterior al nuevo bloque y mina el bloque
        new_block.prior_hash = self.get_last_block().hash
        new_block.mine_block(self.difficulty)
        self.chain.append(new_block)

    def is_bc_valid(self):

        # Verifica la validez de la cadena de bloques
        for i in range(1, len(self.chain)):
            current_block = self.chain[i]
            previous_block = self.chain[i - 1]

            # Verifica el hash del bloque actual
            if current_block.hash != current_block.create_hash():
                return False

            # Verifica el hash del bloque anterior
            if current_block.prior_hash != previous_block.hash:
                return False

        return True
```

Bloque genesis (creamos la blockchain)

In [15]:
```
# Instanciamos la clase BlockChain creando nuestra cadena de bloques
block_chain_1 = BlockChain()
print(json.dumps(block_chain_1.chain, default=lambda o: o.__dict__, indent=4))
```


```
[
    {
        "index": 0,
        "data": "Welcome to Blockchain Demo 2.0! - Clave publica: 0000 - Fecha: 2025-10-0
7",
        "prior_hash": "0",
        "nonce": 0,
        "hash": "5651365fc2ca2a023de2a0897050ac8550e2f9958457e6331fd48f6349878432"
    }
]
```


Agregamos bloque 1

In [16]:
```
# Agregamos bloque a la cadena
block_chain_1.add_block(Block(1, 'Certificado emitido: Barcelona - Fecha: 2025-10-07'))
# Visualizamos el json de nuestra blockchain
```

```
print(json.dumps(block_chain_1.chain, default=lambda o: o.__dict__, indent=4))
```


```
[
    {
        "index": 0,
        "data": "Welcome to Blockchain Demo 2.0! - Clave publica: 0000 - Fecha: 2025-10-0
7",
        "prior_hash": "0",
        "nonce": 0,
        "hash": "5651365fc2ca2a023de2a0897050ac8550e2f9958457e6331fd48f6349878432"
    },
    {
        "index": 1,
        "data": "Certificado emitido: Barcelona - Fecha: 2025-10-07",
        "prior_hash": "5651365fc2ca2a023de2a0897050ac8550e2f9958457e6331fd48f6349878432",
        "nonce": 60734,
        "hash": "0000df2031101761505affd6254f138af7c1c508a11154bd8af03b2f7e583f4b"
    }
]
```


Agregamos bloque 2

In [17]:
```
# Agregamos bloque a la cadena
block_chain_1.add_block(Block(2, 'Certificado emitido: Valencia - Fecha: 2025-10-07'))
# Visualizamos el json de nuestra blockchain
print(json.dumps(block_chain_1.chain, default=lambda o: o.__dict__, indent=4))
```


```
[
    {
        "index": 0,
        "data": "Welcome to Blockchain Demo 2.0! - Clave publica: 0000 - Fecha: 2025-10-0
7",
        "prior_hash": "0",
        "nonce": 0,
        "hash": "5651365fc2ca2a023de2a0897050ac8550e2f9958457e6331fd48f6349878432"
    },
    {
        "index": 1,
        "data": "Certificado emitido: Barcelona - Fecha: 2025-10-07",
        "prior_hash": "5651365fc2ca2a023de2a0897050ac8550e2f9958457e6331fd48f6349878432",
        "nonce": 60734,
        "hash": "0000df2031101761505affd6254f138af7c1c508a11154bd8af03b2f7e583f4b"
    },
    {
        "index": 2,
        "data": "Certificado emitido: Valencia - Fecha: 2025-10-07",
        "prior_hash": "0000df2031101761505affd6254f138af7c1c508a11154bd8af03b2f7e583f4b",
        "nonce": 108991,
        "hash": "00006911a76f258ef07d01f6cccef4f43541ac37fa07820e8d7067f83de1d026"
    }
]
```


Agregamos bloque 3

In [18]:
```
# Agregamos bloque a la cadena
block_chain_1.add_block(Block(3, 'Certificado emitido: Murcia - Fecha: 2025-10-07'))
```

```
# Visualizamos el json de nuestra blockchain
print(json.dumps(block_chain_1.chain, default=lambda o: o.__dict__, indent=4))
```


```
[
    {
        "index": 0,
        "data": "Welcome to Blockchain Demo 2.0! - Clave publica: 0000 - Fecha: 2025-10-0
7",
        "prior_hash": "0",
        "nonce": 0,
        "hash": "5651365fc2ca2a023de2a0897050ac8550e2f9958457e6331fd48f6349878432"
    },
    {
        "index": 1,
        "data": "Certificado emitido: Barcelona - Fecha: 2025-10-07",
        "prior_hash": "5651365fc2ca2a023de2a0897050ac8550e2f9958457e6331fd48f6349878432",
        "nonce": 60734,
        "hash": "0000df2031101761505affd6254f138af7c1c508a11154bd8af03b2f7e583f4b"
    },
    {
        "index": 2,
        "data": "Certificado emitido: Valencia - Fecha: 2025-10-07",
        "prior_hash": "0000df2031101761505affd6254f138af7c1c508a11154bd8af03b2f7e583f4b",
        "nonce": 108991,
        "hash": "00006911a76f258ef07d01f6cccef4f43541ac37fa07820e8d7067f83de1d026"
    },
    {
        "index": 3,
        "data": "Certificado emitido: Murcia - Fecha: 2025-10-07",
        "prior_hash": "00006911a76f258ef07d01f6cccef4f43541ac37fa07820e8d7067f83de1d026",
        "nonce": 42584,
        "hash": "00004afce6fbeeb1daacef25681be9001e2472dc88512a473c9c72f1c94819fa"
    }
]
```


Verificacion de la cadena

Verificación inicial

In [19]:
```
print('Blockchain valida? ' + str(block_chain_1.is_bc_valid()))
```

```
Blockchain valida? True
```

Modificamos algún dato y volvemos a verificar la cadena

In [20]:
```
block_chain_1.chain[1].data
```

Out[20]:
```
'Certificado emitido: Barcelona - Fecha: 2025-10-07'
```

In [21]:
```
# Modificamos los datos de una transaccion
block_chain_1.chain[1].data = 'Certificado emitido: Barcelona - Fecha: 2025-10-06'
# Realizamos validacion de la cadena de bloques
print('Blockchain valida? ' + str(block_chain_1.is_bc_valid()))
```

```
Blockchain valida? False
```

In [23]:
```
print(json.dumps(block_chain_1.chain, default=lambda o: o.__dict__, indent=4))
```


```
[
    {
        "index": 0,
        "data": "Welcome to Blockchain Demo 2.0! - Clave publica: 0000 - Fecha: 2025-10-0
7",
        "prior_hash": "0",
        "nonce": 0,
        "hash": "5651365fc2ca2a023de2a0897050ac8550e2f9958457e6331fd48f6349878432"
    },
    {
        "index": 1,
        "data": "Certificado emitido: Barcelona - Fecha: 2025-10-06",
        "prior_hash": "5651365fc2ca2a023de2a0897050ac8550e2f9958457e6331fd48f6349878432",
        "nonce": 60734,
        "hash": "87ceb40f6b22bd5f6ffa3aad65b0e266e2bdf10606a1b7de1cda3c5b4b32d458"
    },
    {
        "index": 2,
        "data": "Certificado emitido: Valencia - Fecha: 2025-10-07",
        "prior_hash": "0000df2031101761505affd6254f138af7c1c508a11154bd8af03b2f7e583f4b",
        "nonce": 108991,
        "hash": "00006911a76f258ef07d01f6cccef4f43541ac37fa07820e8d7067f83de1d026"
    },
    {
        "index": 3,
        "data": "Certificado emitido: Murcia - Fecha: 2025-10-07",
        "prior_hash": "00006911a76f258ef07d01f6cccef4f43541ac37fa07820e8d7067f83de1d026",
        "nonce": 42584,
        "hash": "00004afce6fbeeb1daacef25681be9001e2472dc88512a473c9c72f1c94819fa"
    }
]
```


Recalculamos el hash del bloque modificado y repetimos la verificación

In [24]:
```
# Modificamos el hash de la cadena que previamente hemos modificado
block_chain_1.chain[1].hash = block_chain_1.chain[1].create_hash()
# Realizamos validacion de la cadena de bloques
print('Blockchain valida? ' + str(block_chain_1.is_bc_valid()))
```

```
Blockchain valida? False
```

In [25]:
```
print(json.dumps(block_chain_1.chain, default=lambda o: o.__dict__, indent=4))
```

```
[
    {
        "index": 0,
        "data": "Welcome to Blockchain Demo 2.0! - Clave publica: 0000 - Fecha: 2025-10-0
7",
        "prior_hash": "0",
        "nonce": 0,
        "hash": "5651365fc2ca2a023de2a0897050ac8550e2f9958457e6331fd48f6349878432"
    },
    {
        "index": 1,
        "data": "Certificado emitido: Barcelona - Fecha: 2025-10-06",
        "prior_hash": "5651365fc2ca2a023de2a0897050ac8550e2f9958457e6331fd48f6349878432",
        "nonce": 60734,
        "hash": "87ceb40f6b22bd5f6ffa3aad65b0e266e2bdf10606a1b7de1cda3c5b4b32d458"
    },
    {
        "index": 2,
        "data": "Certificado emitido: Valencia - Fecha: 2025-10-07",
        "prior_hash": "0000df2031101761505affd6254f138af7c1c508a11154bd8af03b2f7e583f4b",
        "nonce": 108991,
        "hash": "00006911a76f258ef07d01f6cccef4f43541ac37fa07820e8d7067f83de1d026"
    },
    {
        "index": 3,
        "data": "Certificado emitido: Murcia - Fecha: 2025-10-07",
        "prior_hash": "00006911a76f258ef07d01f6cccef4f43541ac37fa07820e8d7067f83de1d026",
        "nonce": 42584,
        "hash": "00004afce6fbeeb1daacef25681be9001e2472dc88512a473c9c72f1c94819fa"
    }
]
```