



5 DE DICIEMBRE DE 2022

# BLOCKCHAIN PYTHON

PRÁCTICA FINAL FF.SS.OO

SERGIO RODRÍGUEZ VIDAL Y JAIME PAZ RODRÍGUEZ

ICAI COMILLAS



## Descripción de proyecto:

El objetivo principal del proyecto es crear una aplicación **Blockchain** con una red “peer-to-peer” donde **cada peer o nodo podrá realizar transferencias y minar bloques**, afectando estos cambios a la Blockchain global **que irá siendo sincronizada**. Este proyecto será enteramente realizado en **Python 3**, aunque se hará uso de otras aplicaciones como **Postman** para la realización de pruebas.

## Estructura del proyecto:

El proyecto está compuesto por una serie de ficheros “.py”. En específico son **Blockchain.py**, **Blockchain\_app.py** y **Blockchain\_requests.py**.

**Blockchain\_app.py** es la **aplicación “web”** del proyecto mediante la cual los usuarios podrán interactuar con las diversas funcionalidades de la Blockchain. Esta aplicación será lanzada **desde una terminal** mediante el comando:

```
$ python Blockchain_app.py -p <especificar un puerto>
```

**Blockchain.py** es el **módulo principal** de la aplicación **Blockchain\_app.py**. Este módulo contiene las clases y objetos fundamentales que definen la **Blockchain**. En específico, se implementan los objetos: **Transacción, Bloque y Blockchain**. La aplicación hará uso **único** de la clase Blockchain, pero esta requiere por su parte la integración de otras dos clases (Bloque y Transacción) para su funcionamiento pleno.

**Blockchain\_requests.py** es un **fichero de pruebas** en el que se hace uso de las principales funciones del programa a modo de ejemplo. Para su uso será necesario iniciar 3 terminales en los puertos 5000, 5001 y 5002 del host local.

## Librerías empleadas:

Empezamos destacando algunas de las **librerías más importantes** en el proyecto, que son requisitos obligatorios de la aplicación.

- flask:

Probablemente esta sea la **librería más importante de toda la aplicación**. Flask es un módulo liviano de Python que facilita el desarrollo de aplicaciones web. Es extensible y **no requiere de una estructura de directorios específica** ni otras complejidades. Por esta simplicidad **se ha escogido frente a otros marcos como Django**.

Las funciones principales empleadas de esta librería son: **Flask, jsonify y request**. **Flask** la instanciamos al principio del programa para **crear la aplicación WSGI**. **Jsonify** se encarga de cargar toda respuesta (diccionario, lista, ...) como una **respuesta de tipo JSON**. **Request**, por último, se encargará de **recibir los datos enviados por los clientes** al servidor.

- requests:

Esta librería permite lanzar **peticiones POST y GET a otros nodos**, para por ejemplo obtener la cadena de algún otro nodo para compararla con la actual. También nos permitirá **hacer pruebas de la app**.

- `argparse`:

Esta librería permite al usuario especificar un puerto al iniciar la app.

- `threading`:

De esta librería se ha hecho uso de dos clases únicamente, pero que sin ellas el programa no funcionaría como es debido. Estas clases son: Semaphore y Timer. La clase **Semaphore** nos sirve para crear un semáforo mutex que **impida los interbloqueos** entre los nodos/peers. A su vez, **Timer** es un temporizador que se inicia cada “x” segundos. En el caso de esta práctica, el temporizador se inicia **cada 60 segundos**. Entonces, crea una copia de seguridad. Ambas clases hacen que el programa sea **más seguro y manejable**.

- `hashlib`:

En esta práctica hay **muchos intercambios y punteros que apuntan a objetos**. La librería `hashlib`, es la que nos permite **hashear** estos objetos para poder **realizar estas funciones más óptimamente**.

- `json`:

JSON es el formato de entrada y salida más común entre las funciones, es por esto que para **crear JSON en Python a partir de diccionarios u objetos** utilizamos la librería con el mismo nombre.

**Otras librerías relevantes** empleadas son: **typing** (para anotar las funciones), **time** (para crear un timestamp para los bloques), **platform y socket** (para coger detalles del host y clientes) y finalmente **datetime** (en específico la función datetime, para especificar la fecha de las copias de seguridad).

## Blockchain.py:

El módulo Blockchain como se explicó previamente, es donde se definen las principales clases de la app, incluyendo la Blockchain misma.

```
class Transaccion(object):
    def __init__(self, origen: str, destino: str, cantidad: int):
        """
        Constructor de la clase 'Transaccion'.
        :param origen: Originario de la transacción.
        :param destino: Destinatario de la transacción.
        :param cantidad: Cantidad de dinero enviada.
        """
        self.origen = origen
        self.destino = destino
        self.cantidad = cantidad
        self.timestamp = time()
```

Figura 1: Transacción

La primera clase es un objeto **simple** que define una **transacción**. Se optó por crear dicha clase para que sea más sencillo crear transacciones en un futuro.

```

class Bloque(object):
    def __init__(self, indice: int, transacciones: TransactionLikeList, hash_previo: str, timestamp: float,
                 prueba: int = 0, calcular_hash: bool = False):
        """
        Constructor de la clase 'Bloque'.
        :param indice: ID unico del bloque.
        :param transacciones: Lista de transacciones.
        :param hash_previo: hash previo.
        :param prueba: prueba de trabajo.
        :param calcular_hash: Calcula el hash del bloque (default: False)
        """
        self.hash_bloque = None
        self.hash_previo = hash_previo
        self.indice = indice
        self.timestamp = timestamp
        self.prueba = prueba
        self.transacciones = transacciones
        if calcular_hash:
            self.hash_bloque = self.calcular_hash()

    def calcular_hash(self):
        """
        Método que devuelve el hash de un bloque.
        :return: hash del bloque
        """
        block_string = json.dumps(self.__dict__, sort_keys=True)
        return str(sha256(block_string.encode()).hexdigest())

```

Figura 2: Bloque

La clase **Bloque** es ligeramente más compleja que una transacción. Esta tiene como **parámetros** los visibles en la Figura 2: Bloque, siendo transacciones **no una lista de objetos Transacción, sino una lista de diccionarios sacados de las transacciones**. El parámetro **calcular hash**, si su valor es **True**, crea automáticamente el hash del bloque en cuestión (útil si sabes que tu bloque se integrará sin problemas y no es modificable).

La función `calcular_hash` **crea un hash para los contenidos actuales del bloque**. Esta función **no cambia el hash del bloque directamente, a no ser que al iniciar este objeto se especifique que calcule el hash**. Entonces sólo se calculará y atribuirá la primera vez, pero no mediante esta función directamente.

Ahora vamos con la clase más importante de toda la práctica.

```

class Blockchain(object):
    dificultad = 4

    def __init__(self):
        """
        Constructor de la clase
        """
        self.cadena = []
        self.transacciones_sin_confirmar = []
        self.primer_bloque()

    def primer_bloque(self) -> Bloque:
        """
        Crea un primer bloque vacío que sirva de raíz para el resto de bloques
        :return: primer bloque
        """
        self.cadena.append(primer_bloque := Bloque(1, [], "1", calcular_hash=True, timestamp=time()))
        return primer_bloque

```

Figura 3: Blockchain (Parte 1)

Esta clase no recibe parámetros, sino que **inicia una cadena vacía, una lista de transacciones igualmente vacía, y crea un primer bloque que sirva de base para el siguiente**. Todo esto es visible en la Figura 3: Blockchain (Parte 1). Además la clase Blockchain tiene una variable local que permanecerá **constante**. Esta es la **dificultad de la Blockchain** que indica el número de 0's por el que empieza el hash de cada bloque.

```
def nueva_transaccion(self, origen: str, destino: str, cantidad: int) → int:
    """
    Crea una nueva transaccion a partir de un origen, un destino y una cantidad y la incluye en las listas de transacciones.
    :param origen: Originario de la transacción.
    :param destino: Destinataria de la transacción.
    :param cantidad: la cantidad.
    :return: el hash del nuevo bloque (dejará el campo de hash del bloque sin modificar)
    """
    nueva_transaccion = Transaccion(origen, destino, cantidad)
    self.transacciones_sin_confirmar.append(nueva_transaccion.__dict__)
    return len(self.cadena) + 1
```

Figura 4: Blockchain (Parte 2)

En la Figura 4: Blockchain (Parte 2) se muestra otro método de la clase, mediante el cuál se podrán añadir nuevas transacciones **sin confirmar** a la Blockchain. Ergo, no serán incluidas en ningún bloque de momento.

```
def nuevo_bloque(self, hash_previo: str) → Bloque:
    """
    Crea un nuevo bloque a partir de las transacciones que no están confirmadas.
    :param hash_previo: el hash del bloque anterior de la cadena
    :return: nuevo bloque
    """
    return Bloque(self.ultimo_bloque.indice + 1, self.transacciones_sin_confirmar, hash_previo, timestamp=time())

def integra_bloque(self, bloque_nuevo: Bloque, hash_prueba: str) → bool:
    """
    Método para integrar correctamente un bloque a la cadena de bloques. Debe comprobar que la prueba de hash es válida y que el hash del bloque último de la cadena coincida con el hash_previo del bloque que se va a integrar. Si pasa las comprobaciones, actualiza el hash del bloque a integrar, lo inserta en la cadena y hace un reset de las transacciones no confirmadas (vuelve a dejar la lista de transacciones no confirmadas a una lista vacía).
    :param bloque_nuevo: el nuevo bloque que se va a integrar.
    :param hash_prueba: prueba del hash del bloque.
    :return: bool. True si se consiguió integrar, False en caso contrario.
    """
    hash_previo = self.ultimo_bloque.hash_bloque
    if hash_previo != bloque_nuevo.hash_previo:
        return False

    if not self.prueba_valida(bloque_nuevo, hash_prueba):
        return False

    bloque_nuevo.hash_bloque = hash_prueba
    self.cadena.append(bloque_nuevo)
    self.transacciones_sin_confirmar = []
    return True
```

Figura 5: Blockchain (Parte 3)

Ahora vamos con la función más importante de esta clase, que es integrar la Blockchain. Esta recibe un bloque con las transacciones sin confirmar de la Blockchain, **creado mediante la función nuevo\_bloque**. Comprueba que el hash previo del bloque sea igual al hash del anterior (para comprobar la noción de cadena). Realiza una prueba que ve el nuevo hash del bloque (hash prueba) esté actualizado correctamente y que cumpla con la dificultad de la Blockchain. Entonces la función **actualiza el hash de dicho bloque y lo añade a la cadena, vaciando la lista de transacciones** sin confirmar.

## Blockchain\_app.py:

Blockchain\_app.py es una aplicación “web” del proyecto mediante la cual los usuarios podrán interactuar con las diversas funcionalidades de la Blockchain.

```
# Instancia del nodo
app = Flask(__name__)

# Instancia de la aplicación
blockchain = Blockchain.Blockchain()

# Nodos registrados en la red
nodos_red = set()

# Para saber mi ip
mi_ip = socket.gethostname(socket.gethostname())

# Semáforo mutex
mutex = Semaphore(1)
```

Figura 6: Inicio de Blockchain\_app

Al inicio del programa, se crea una app con el nombre del fichero, se **inicia una Blockchain y una lista de nodos/peers vacía**, se extrae la IP del host y se crea el semáforo que impida los interbloqueos.

```
if __name__ == '__main__':
    parser = ArgumentParser()
    parser.add_argument('-p', '--puerto', default=5000, type=int, help='puerto para escuchar')
    args = parser.parse_args()
    puerto = args.puerto
    copia_seguridad()
    app.run(host='0.0.0.0', port=puerto)
```

Figura 7: Ejecución de Blockchain\_app.py

Al ejecutarse, **se introduce el puerto deseado como puerto de la app**, y se inicia el temporizador de la copia de seguridad.

```
def copia_seguridad():
    """
    Realiza una copia de seguridad de la blockchain cada 60 segundos.
    """
    # Iniciamos un temporizador cada 60 segundos
    timer = Timer(60., copia_seguridad)
    timer.daemon = True
    timer.start()
    # Creamos un JSON "copia de seguridad" con la blockchain de dicho nodo.
    blockchain.to_json(f'respaldo-nodo{mi_ip}-{puerto}.json', indent=2)
```

Figura 8: Copia de seguridad

En la Figura 8: Copia de seguridad se ve cómo es dicha función. Al ser instanciada, crea un temporizador de 60 segundos. Cada **60 segundos** esta crea un respaldo de dicho nodo, como un fichero json de la Blockchain del nodo.

A partir de ahora sólo haremos referencias a las funciones menos complejas de la app por brevedad. Pese a esto, todas las funciones están perfectamente documentadas en el código.

```
@app.route('/nodos/registrar', methods=['POST'])
def registrar_nodos_completo():
    global blockchain, nodos_red
    # Extrae las direcciones de sus nuevos peers
    direccion_nodos = request.get_json().get('direccion_nodos')

    if direccion_nodos is None:
        return "Error: No se ha proporcionado una lista de nodos", 400
    # Actualiza su set de peers
    nodos_red.update(direccion_nodos)

    # Por cada nodo/peer, manda un diccionario copia de su blockchain, y los nuevos peers de dicho peer incluyéndose a
    # si
    headers = {'Content-Type': "application/json"}
    for nodo in direccion_nodos:
        data = {
            'nodos_direcciones': [request.host_url, * [n for n in direccion_nodos if n != nodo]],
            'blockchain': blockchain.to_dict()
        }
        requests.post(nodo + "/nodos/registro_simple", data=json.dumps(data), headers=headers)

    response = {
        'mensaje': 'Se han incluido nuevos nodos en la red',
        'nodos_totales': List(nodos_red)
    }
    return jsonify(response), 201
```

Figura 9: Registrar nodos

Esta función registra peers/nodos a partir de una lista de direcciones dada a un determinado nodo, que es el encargado de la gestión de estos. Este nodo acogerá las direcciones mandadas como integrantes de su conjunto de peers. Luego, a cada uno de estos nuevos peers les manda una copia de su Blockchain (como un diccionario), y les indica quiénes serán sus peers (o sea una lista de nodos incluyendo al nodo “gestor” mismo, y excluyendo al peer en cuestión). Estos nodos acogerán estos datos mediante la función registrar nodo actualiza blockchain ó registro simple.



```

@app.route('/nodos/registro_simple', methods=['POST'])
def registrar_nodo_actualiza_blockchain():
    """
    Esta función actualiza los nodos_red y blockchain a partir de unos datos. Puede haber algún error
    blockchain si esta no es correcta.
    :return: Respuesta en formato JSON
    """
    # Obtenemos la variable global de blockchain
    global blockchain, nodos_red

    # Extrae los input
    response = request.get_json()
    direccion_nodos = response.get("nodos_direcciones")
    # Actualiza sus peers
    nodos_red.update(direccion_nodos)

    # Crea una blockchain a partir de la leida
    blockchain_leida = response.get("blockchain")
    if blockchain_leida is None:
        return "El blockchain de la red está corrupto", 400
    else:
        mutex.acquire()
        blockchain = crear_blockchain_dump(blockchain_leida["cadena"])
        mutex.release()

    return "La blockchain del nodo" + str(mi_ip) + ":" + str(puerto) + " ha sido correctamente actuali

```

Figura 10: Registro simple de nodos

Esta función **coge las direcciones de nodos enviados y las convierte en sus nuevos peers**. Además, **a través de la cadena de la Blockchain enviada (en formato diccionario), crea un objeto Blockchain que será su nueva Blockchain**.

```

blockchain.nueva_transaccion(origen="0", destino=mi_ip, cantidad=1)
ultimo_bloque = blockchain.ultimo_bloque
nuevo_bloque = blockchain.nuevo_bloque(hash_previo=ultimo_bloque.hash_bloque)
prueba = blockchain.prueba_trabajo(nuevo_bloque)
mutex.release()
# Se comprueba si existen conflictos
resuelve_conflicto = resuelve_conflictos()

# Si ha habido conflictos, se resuelven y se remueven todas las transacciones
if resuelve_conflicto:
    response = {
        'mensaje': "Ha habido un conflicto. Esta cadena se ha actualizado con una version mas larga."
    }
else:
    mutex.acquire()
    resultado = blockchain.integra_bloque(nuevo_bloque, prueba)
    mutex.release()
    # Si no se pudo integrar correctamente, se remueve el pago al minero
    if not resultado:
        mutex.acquire()
        blockchain.transacciones_sin_confirmar.pop()
        mutex.release()
        response = {
            'mensaje': "No es posible integrar el nuevo bloque."
        }
    # Si si se integra correctamente, se manda un mensaje de minado satisfactorio.
    else:
        response = {
            'mensaje': f"El bloque {nuevo_bloque.indice} se ha minado satisfactoriamente."
        }
    return jsonify(response), 200

```

Figura 11: Minar bloque

La función quizás más importante de la aplicación es **minar**. Esta función **no va dentro de la clase Blockchain por motivos de diseño del programa**, ya que al **deber interactuar con la**



**función resuelve conflictos** de Blockchain\_app.py, ha querido meterse en el mismo fichero. Esta función tampoco levanta errores (de integración de bloque por ejemplo) por motivos de diseño.

Esta función, **primero crea una transacción siendo esta el pago del minero, y se acaba efectuando o no dependiendo de qué tal vaya el minado. Si existe algún conflicto entre dicha Blockchain y la de otros nodos (existe una cadena más larga), entonces este pago no se ejecutará (también se borrarán las transferencias sin confirmar, al cambiar la blockchain). Si no existe ningún conflicto entonces se integrará dicho bloque. Si no es posible integrarlo entonces el minado no podrá realizarse por completo y el minero tampoco recibirá el pago.**

```
def resuelve_conflictos():
    """
    Mecanismo para establecer el consenso y resolver los conflictos. Para llegar a un consenso se escoge la cadena
    más larga.
    """
    global blockchain

    # Coge la longitud de la blockchain del nodo actual
    longitud_actual = len(blockchain)

    # Iniciamos una variable cadena_mas_larga, que podrá o no contener una cadena más larga que la actual (en caso de
    # existir)
    cadena_mas_larga = None

    # Por cada nodo, comprueba si su blockchain es mayor que la actual, y si es así, modifica la longitud actual por
    # esta y la cadena más larga también por esta.
    for nodo in nodos_red:
        response = requests.get(f'{nodo}/chain')
        longitud = response.json()['longitud']
        cadena = response.json()['chain']

        if longitud > longitud_actual:
            # Actualiza las variables
            longitud_actual = longitud
            cadena_mas_larga = cadena

    # Si hay una cadena más larga, haz que esta sea tu blockchain
    if cadena_mas_larga:
        blockchain = crear_blockchain_dump(cadena_mas_larga)
        return True
    else:
        return False
```

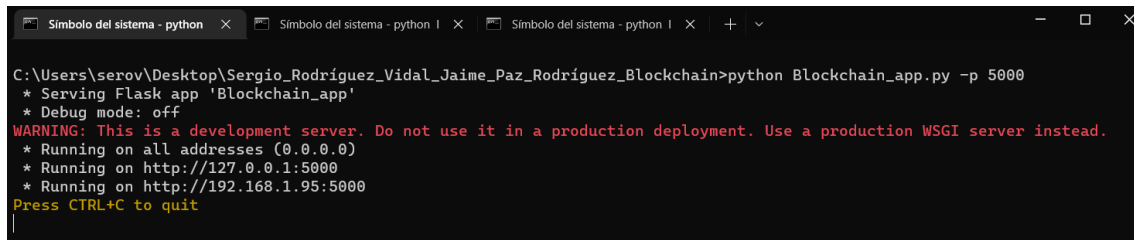
Figura 12: Resuelve conflictos

Anteriormente se especificó que hay un conflicto cuando **existe otro nodo con una Blockchain de mayor longitud**. En la Figura 12: Resuelve conflictos se muestra cómo se observa esto (iterando por los nodos/peers que forman parte de la red). Si sucede un conflicto, entonces la blockchain del nodo cambia por la más larga. (Ojo, esta blockchain no tendrá transferencias sin confirmar).

## Blockchain\_requests.py:

En este fichero **se realizan tests que ponen a prueba a la aplicación en todos los escenarios posibles**. Todas estas pruebas se pueden realizar una a una en Postman para comprobar el funcionamiento de la blockchain.

Antes de ejecutar este fichero es necesario **iniciar la aplicación desde los puertos 5000, 5001 y 5002 del localhost.**



```
C:\Users\serov\Desktop\Sergio_Rodríguez_Vidal_Jaime_Paz_Rodríguez_Blockchain>python Blockchain_app.py -p 5000
* Serving Flask app 'Blockchain_app'
* Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on all addresses (0.0.0.0)
* Running on http://127.0.0.1:5000
* Running on http://192.168.1.95:5000
Press CTRL+C to quit
```

Figura 13: Iniciar Blockchain\_app.py en 3 puertos y 3 terminales

Una vez hecho esto se ejecutará el fichero, que devolverá:

```
{
  "maquina": "AMD64",
  "nombre_sistema": "Windows",
  "version": "10.0.19044"
}

{
  "mensaje": "La transaccion se incluirea en el bloque con indice 2"
}

{
  "mensaje": "El bloque 2 se ha minado satisfactoriamente."
}

{
  "chain": [
    {
      "hash_bloque": "79d3768d1fa2a5740cb02e385986d1a914f87aca76ca8d3a796821707800559e",
      "hash_previo": "1",
      "indice": 1
    }
  ]
}

{
  "mensaje": "Se han incluido nuevos nodos en la red",
  "nodos_totales": [
    "http://localhost:5001",
    "http://localhost:5002"
  ]
}

{
  "mensaje": "La transaccion se incluirea en el bloque con indice 3"
}

{
  "mensaje": "El bloque 3 se ha minado satisfactoriamente."
}

{
  "chain": [
    {
      "hash_bloque": "79d3768d1fa2a5740cb02e385986d1a914f87aca76ca8d3a796821707800559e",
      "hash_previo": "1",
      "indice": 1
    }
  ]
}

{
  "mensaje": "La transaccion se incluirea en el bloque con indice 3"
}

{
  "mensaje": "Ha habido un conflicto. Esta cadena se ha actualizado con una version mas larga."
}

{
  "chain": [
    {
      "hash_bloque": "79d3768d1fa2a5740cb02e385986d1a914f87aca76ca8d3a796821707800559e",
      "hash_previo": "1",
      "indice": 1
    }
  ]
}

{
  "mensaje": "No es posible crear un nuevo bloque. No hay transacciones"
}

{
  "chain": [
    {
      "hash_bloque": "79d3768d1fa2a5740cb02e385986d1a914f87aca76ca8d3a796821707800559e",
      "hash_previo": "1",
      "indice": 1
    }
  ]
}

{
  "mensaje": "La transaccion se incluirea en el bloque con indice 3"
}

{
  "mensaje": "Ha habido un conflicto. Esta cadena se ha actualizado con una version mas larga."
}

{
  "chain": [
    {
      "hash_bloque": "79d3768d1fa2a5740cb02e385986d1a914f87aca76ca8d3a796821707800559e",
      "hash_previo": "1",
      "indice": 1
    }
  ]
}
```

Figura 14: Output Blockchain\_requests.py

```
C:\Users\serov\Desktop\Sergio_Rodríguez_Vidal_Jaime_Paz_Rodríguez_Blockchain>python Blockchain_app.py -p 5000
* Serving Flask app 'Blockchain_app'
* Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on all addresses (0.0.0.0)
* Running on http://127.0.0.1:5000
* Running on http://192.168.1.95:5000
Press CTRL+C to quit
127.0.0.1 - - [04/Dec/2022 13:10:58] "GET /system HTTP/1.1" 200 -
127.0.0.1 - - [04/Dec/2022 13:10:58] "POST /transacciones/nueva HTTP/1.1" 201 -
<Blockchain.Blockchain object at 0x000002086ADBDE40>
127.0.0.1 - - [04/Dec/2022 13:10:59] "GET /minar HTTP/1.1" 200 -
127.0.0.1 - - [04/Dec/2022 13:10:59] "GET /chain HTTP/1.1" 200 -
127.0.0.1 - - [04/Dec/2022 13:10:59] "POST /nodos/registrar HTTP/1.1" 201 -
127.0.0.1 - - [04/Dec/2022 13:10:59] "POST /transacciones/nueva HTTP/1.1" 201 -
<Blockchain.Blockchain object at 0x000002086ADBDE40>
127.0.0.1 - - [04/Dec/2022 13:11:01] "GET /minar HTTP/1.1" 200 -
127.0.0.1 - - [04/Dec/2022 13:11:01] "GET /chain HTTP/1.1" 200 -
127.0.0.1 - - [04/Dec/2022 13:11:03] "GET /chain HTTP/1.1" 200 -
127.0.0.1 - - [04/Dec/2022 13:11:05] "GET /chain HTTP/1.1" 200 -
```

Figura 15: Output localhost:5000

```
C:\Users\serov\Desktop\Sergio_Rodríguez_Vidal_Jaime_Paz_Rodríguez_Blockchain>python Blockchain_app.py -p 5001
* Serving Flask app 'Blockchain_app'
* Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on all addresses (0.0.0.0)
* Running on http://127.0.0.1:5001
* Running on http://192.168.1.95:5001
Press CTRL+C to quit
127.0.0.1 - - [04/Dec/2022 13:10:59] "POST /nodos/registro_simple HTTP/1.1" 200 -
127.0.0.1 - - [04/Dec/2022 13:11:01] "GET /chain HTTP/1.1" 200 -
127.0.0.1 - - [04/Dec/2022 13:11:01] "POST /transacciones/nueva HTTP/1.1" 201 -
127.0.0.1 - - [04/Dec/2022 13:11:03] "GET /minar HTTP/1.1" 200 -
127.0.0.1 - - [04/Dec/2022 13:11:03] "GET /chain HTTP/1.1" 200 -
127.0.0.1 - - [04/Dec/2022 13:11:05] "GET /chain HTTP/1.1" 200 -
```

Figura 16: Output localhost:5001

```
C:\Users\serov\Desktop\Sergio_Rodríguez_Vidal_Jaime_Paz_Rodríguez_Blockchain>python Blockchain_app.py -p 5002
* Serving Flask app 'Blockchain_app'
* Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on all addresses (0.0.0.0)
* Running on http://127.0.0.1:5002
* Running on http://192.168.1.95:5002
Press CTRL+C to quit
127.0.0.1 - - [04/Dec/2022 13:10:59] "POST /nodos/registro_simple HTTP/1.1" 200 -
127.0.0.1 - - [04/Dec/2022 13:11:01] "GET /chain HTTP/1.1" 200 -
127.0.0.1 - - [04/Dec/2022 13:11:03] "GET /chain HTTP/1.1" 200 -
127.0.0.1 - - [04/Dec/2022 13:11:03] "GET /minar HTTP/1.1" 200 -
127.0.0.1 - - [04/Dec/2022 13:11:03] "GET /chain HTTP/1.1" 200 -
127.0.0.1 - - [04/Dec/2022 13:11:03] "POST /transacciones/nueva HTTP/1.1" 201 -
127.0.0.1 - - [04/Dec/2022 13:11:05] "GET /minar HTTP/1.1" 200 -
127.0.0.1 - - [04/Dec/2022 13:11:05] "GET /chain HTTP/1.1" 200 -
```

Figura 17: Output localhost:5002

Ahora, ¿por qué sabemos que la aplicación funciona? Se ha hecho **uso de todas las funciones de la aplicación y no se devuelve ningún error.**

```
transaccion_nueva = {'origen': 'nodoC', 'destino': 'nodoB', 'cantidad': 10}
r = requests.post('http://localhost:5000/transacciones/nueva', data=json.dumps(transaccion_nueva))
print(r.text)

r = requests.get('http://localhost:5000/minar')
print(r.text)

r = requests.get('http://localhost:5000/chain')
print(r.text)

transaccion_nueva = {'origen': 'nodoD', 'destino': 'nodoB', 'cantidad': 10}
r = requests.post('http://localhost:5001/transacciones/nueva', data=json.dumps(transaccion_nueva))
print(r.text)

r = requests.get('http://localhost:5001/minar')
print(r.text)

r = requests.get('http://localhost:5001/chain')
print(r.text)
```

Figura 18: Forzar conflicto


Se **fuera un conflicto** creando una transacción en el puerto 5000 y minándolo, que el 5001 no tiene, por tanto **la cadena del 5001 es menor**, ya que sus blockchain eran idénticas.

También se prueba a minar el 5002, que **no tenía ninguna transacción sin confirmar**, y correctamente, devuelve un mensaje diciendo que no existe dicha transacción.

Por último, **se intenta minar el bloque 2 que como el 3 debería haber un conflicto**.

También se puede comprobar que las copias de seguridad se han realizado correctamente.

Estas mismas requests funcionan en una **virtual box de Ubuntu 24LTS**:



```
{
  "maquina": "x86_64",
  "nombre_sistema": "Linux",
  "version": "#62-Ubuntu SMP Tue Nov 22 19:54:14 UTC 2022"
}

{
  "mensaje": "La transaccion se incluiara en el bloque con indice 4"
}

{
  "hash": "00009d725195cd2244ba0ce6b7d87436f99a0f9d787b9440a44e14312cfefbda",
  "hash_previo": "00007356b44f5f8b9aed01d490d8915c3672e38f311efa0f5407e32faacdee8b",
  "indice": 4,
  "mensaje": "Nuevo bloque minado",
  "prueba": 67396,
  "timestamp": 1670223527.773243,
  "transacciones": [
    {
      "cantidad": 10,
      "destino": "nodoB",
      "origen": "nodoA",
      "tiempo": 1670223527.7107973
    },
    {
      "cantidad": 1,
      "destino": "127.0.0.1",
      "origen": "0",
      "tiempo": 1670223527.7732406
    }
  ]
}

{
  "chain": [
    {
      "hash": "1d581b6b49b933b344512dd031491c4df68d6d19de7abf79cdbac6d3cb0954a",
      "hash_previo": "0",
      "indice": 1,
      "prueba": 0,
      "timestamp": 0,
      "transacciones": []
    },
    {
      "hash": "0000ea2e5af38327b3482c7f78c063f1c1578f489600d7bb21f83a75863a649a",
      "hash_previo": "1d581b6b49b933b344512dd031491c4df68d6d19de7abf79cdbac6d3cb0954a",
      "indice": 2,
      "prueba": 74459,
      "timestamp": 1670223304.5292623,
      "transacciones": [
        {
          "cantidad": 10,
          "destino": "nodoB",
          "origen": "nodoA",
          "tiempo": 1670223304.5201018
        },
        {
          "cantidad": 1,
          "destino": "127.0.0.1",
          "origen": "0",
          "tiempo": 1670223304.5292585
        }
      ],
      "hash": "00007356b44f5f8b9aed01d490d8915c3672e38f311efa0f5407e32faacdee8b",
      "hash_previo": "0000ea2e5af38327b3482c7f78c063f1c1578f489600d7bb21f83a75863a649a",
      "indice": 3,
      "prueba": 37832,
      "timestamp": 1670223305.5570247,
      "transacciones": [
        {
          "cantidad": 10,
          "destino": "nodoB",
          "origen": "nodoA",
          "tiempo": 1670223305.5462477
        },
        {
          "cantidad": 1,
          "destino": "127.0.0.1",
          "origen": "0",
          "tiempo": 1670223305.5570226
        }
      ],
      "hash": "00009d725195cd2244ba0ce6b7d87436f99a0f9d787b9440a44e14312cfefbda",
      "hash_previo": "00007356b44f5f8b9aed01d490d8915c3672e38f311efa0f5407e32faacdee8b",
      "indice": 4,
      "prueba": 67396,
      "timestamp": 1670223527.773243,
      "transacciones": [
        {
          "cantidad": 10,
          "destino": "nodoB",
          "origen": "nodoA",
          "tiempo": 1670223527.7107973
        },
        {
          "cantidad": 1,
          "destino": "127.0.0.1",
          "origen": "0",
          "tiempo": 1670223527.7732406
        }
      ],
      "longitud": 4
    }
  ],
  "mensaje": "Se han incluido nuevos nodos en la red",
  "nodos_totales": [
    "http://127.0.0.1:5001",
    "http://127.0.0.1:5002",
    "http://localhost:5001",
    "http://localhost:5002"
  ]
}

{
  "mensaje": "La transaccion se incluiara en el bloque con indice 5"
}

{
  "hash": "000030ba32001e809dd11353c64ebd21def6797c629303aca82550f44d9cc5dc",
  "hash_previo": "00009d725195cd2244ba0ce6b7d87436f99a0f9d787b9440a44e14312cfefbda",
  "indice": 5,
  "mensaje": "Nuevo bloque minado",
  "prueba": 238499,
  "timestamp": 1670223528.5446908,
  "transacciones": [
    {
      "cantidad": 10,
      "destino": "nodoB",
      "origen": "nodoC",
      "tiempo": 1670223528.5249813
    },
    {
      "cantidad": 1,
      "destino": "127.0.0.1",
      "origen": "0",
      "tiempo": 1670223528.5446892
    }
  ]
}

{
  "chain": [
    {
      "hash": "1d581b6b49b933b344512dd031491c4df68d6d19de7abf79cdbac6d3cb0954a",
      "hash_previo": "0",
      "indice": 1,
      "prueba": 0,
      "timestamp": 0,
      "transacciones": []
    },
    {
      "hash": "0000ea2e5af38327b3482c7f78c063f1c1578f489600d7bb21f83a75863a649a",
      "hash_previo": "1d581b6b49b933b344512dd031491c4df68d6d19de7abf79cdbac6d3cb0954a",
      "indice": 2,
      "prueba": 74459,
      "timestamp": 1670223304.5292623,
      "transacciones": [
        {
          "cantidad": 10,
          "destino": "nodoB",
          "origen": "nodoA",
          "tiempo": 1670223304.5201018
        },
        {
          "cantidad": 1,
          "destino": "127.0.0.1",
          "origen": "0",
          "tiempo": 1670223304.5292585
        }
      ],
      "hash": "00007356b44f5f8b9aed01d490d8915c3672e38f311efa0f5407e32faacdee8b",
      "hash_previo": "0000ea2e5af38327b3482c7f78c063f1c1578f489600d7bb21f83a75863a649a",
      "indice": 3,
      "prueba": 37832,
      "timestamp": 1670223305.5570247,
      "transacciones": [
        {
          "cantidad": 10,
          "destino": "nodoB",
          "origen": "nodoA",
          "tiempo": 1670223305.5462477
        },
        {
          "cantidad": 1,
          "destino": "127.0.0.1",
          "origen": "0",
          "tiempo": 1670223305.5570226
        }
      ],
      "hash": "00009d725195cd2244ba0ce6b7d87436f99a0f9d787b9440a44e14312cfefbda",
      "hash_previo": "00007356b44f5f8b9aed01d490d8915c3672e38f311efa0f5407e32faacdee8b",
      "indice": 4,
      "prueba": 67396,
      "timestamp": 1670223527.773243,
      "transacciones": [
        {
          "cantidad": 10,
          "destino": "nodoB",
          "origen": "nodoA",
          "tiempo": 1670223527.7107973
        },
        {
          "cantidad": 1,
          "destino": "127.0.0.1",
          "origen": "0",
          "tiempo": 1670223527.7732406
        }
      ],
      "longitud": 4
    },
    {
      "hash": "000030ba32001e809dd11353c64ebd21def6797c629303aca82550f44d9cc5dc",
      "hash_previo": "00009d725195cd2244ba0ce6b7d87436f99a0f9d787b9440a44e14312cfefbda",
      "indice": 5,
      "mensaje": "Nuevo bloque minado",
      "prueba": 238499,
      "timestamp": 1670223528.5446908,
      "transacciones": [
        {
          "cantidad": 10,
          "destino": "nodoB",
          "origen": "nodoC",
          "tiempo": 1670223528.5249813
        },
        {
          "cantidad": 1,
          "destino": "127.0.0.1",
          "origen": "0",
          "tiempo": 1670223528.5446892
        }
      ]
    }
  ]
}
```



```
["chain":[{"hash":"1d581b6b49b933b344512dd031491c4df68d6d19de7abf79cdbacf6d3cb0954a","hash_previo":"0","indice":1,"prueba":0,"timestamp":0,"transacciones":[]}, {"hash":"0000ea2e5af38327b3482c7f78c063f1c1578f489600d7bb21f83a75863a649a","hash_previo":"1d581b6b49b933b344512dd031491c4df68d6d19de7abf79cdbacf6d3cb0954a","indice":2,"prueba":74459,"timestamp":1670223304.5292623,"transacciones":[{"cantidad":10,"destino":"nodoB","origen":"nodoA","tiempo":1670223304.5201018}, {"cantidad":1,"destino":"127.0.0.1","origen":"0","tiempo":1670223304.5292585}], {"hash":"00007356b44f5f8b9aed01d490d8915c3672e38f311efa0f5407e32faacdee8b","hash_previo":"0000ea2e5af38327b3482c7f78c063f1c1578f489600d7bb21f83a75863a649a","indice":3,"prueba":37832,"timestamp":1670223305.5570247,"transacciones":[{"cantidad":10,"destino":"nodoB","origen":"nodoA","tiempo":1670223305.5462477}, {"cantidad":1,"destino":"127.0.0.1","origen":"0","tiempo":1670223305.5570226}], {"hash":"00009d725195cd2244ba0ce6b7d87436f99a0f9d787b9440a44e14312cfefbda","hash_previo":"00007356b44f5f8b9aed01d490d8915c3672e38f311efa0f5407e32faacdee8b","indice":4,"prueba":67396,"timestamp":1670223527.773243,"transacciones":[{"cantidad":10,"destino":"nodoB","origen":"nodoA","tiempo":1670223527.7107973}, {"cantidad":1,"destino":"127.0.0.1","origen":"0","tiempo":1670223527.7732406}], {"hash":"000030ba32001e809dd11353c64ebd21def6797c629303aca82550f44d9cc5dc","hash_previo":"00009d725195cd2244ba0ce6b7d87436f99a0f9d787b9440a44e14312cfefbda","indice":5,"prueba":238499,"timestamp":1670223528.5446908,"transacciones":[{"cantidad":10,"destino":"nodoB","origen":"nodoC","tiempo":1670223528.5249813}, {"cantidad":1,"destino":"127.0.0.1","origen":"0","tiempo":1670223528.5446892}]],"longitud":5}

{"mensaje":"La transaccion se incluirea en el bloque con indice 5"}

{"mensaje":"Ha habido un conflicto. Esta cadena se ha actualizado con una version mas larga"}
```

```
["mensaje":"Ha habido un conflicto. Esta cadena se ha actualizado con una version mas larga"]
```

```
["chain":[{"hash":"1d581b6b49b933b344512dd031491c4df68d6d19de7abf79cdbacf6d3cb0954a","hash_previo":"0","indice":1,"prueba":0,"timestamp":0,"transacciones":[]}, {"hash":"0000ea2e5af38327b3482c7f78c063f1c1578f489600d7bb21f83a75863a649a","hash_previo":"1d581b6b49b933b344512dd031491c4df68d6d19de7abf79cdbacf6d3cb0954a","indice":2,"prueba":74459,"timestamp":1670223304.5292623,"transacciones":[{"cantidad":10,"destino":"nodoB","origen":"nodoA","tiempo":1670223304.5201018}, {"cantidad":1,"destino":"127.0.0.1","origen":"0","tiempo":1670223304.5292585}], {"hash":"00007356b44f5f8b9aed01d490d8915c3672e38f311efa0f5407e32faacdee8b","hash_previo":"0000ea2e5af38327b3482c7f78c063f1c1578f489600d7bb21f83a75863a649a","indice":3,"prueba":37832,"timestamp":1670223305.5570247,"transacciones":[{"cantidad":10,"destino":"nodoB","origen":"nodoA","tiempo":1670223305.5462477}, {"cantidad":1,"destino":"127.0.0.1","origen":"0","tiempo":1670223305.5570226}], {"hash":"00009d725195cd2244ba0ce6b7d87436f99a0f9d787b9440a44e14312cfefbda","hash_previo":"00007356b44f5f8b9aed01d490d8915c3672e38f311efa0f5407e32faacdee8b","indice":4,"prueba":67396,"timestamp":1670223527.773243,"transacciones":[{"cantidad":10,"destino":"nodoB","origen":"nodoA","tiempo":1670223527.7107973}, {"cantidad":1,"destino":"127.0.0.1","origen":"0","tiempo":1670223527.7732406}], {"hash":"000030ba32001e809dd11353c64ebd21def6797c629303aca82550f44d9cc5dc","hash_previo":"00009d725195cd2244ba0ce6b7d87436f99a0f9d787b9440a44e14312cfefbda","indice":5,"prueba":238499,"timestamp":1670223528.5446908,"transacciones":[{"cantidad":10,"destino":"nodoB","origen":"nodoC","tiempo":1670223528.5249813}, {"cantidad":1,"destino":"127.0.0.1","origen":"0","tiempo":1670223528.5446892}]],"longitud":5}

{"mensaje":"No es posible crear un nuevo bloque. No hay transacciones"}
```

```

{"mensaje":"No es posible crear un nuevo bloque. No hay transacciones"}

{"chain":[{"hash":"1d581b6b49b933b344512dd031491c4df68d6d19de7abf79cdbacf6d3cb0954a","hash_previo":"0","indice":1,"prueba":0,"timestamp":0,"transacciones":[]},{hash":"0000ea2e5af38327b3482c7f78c063f1c1578f489600d7bb21f83a75863a649a","hash_previo":"1d581b6b49b933b344512dd031491c4df68d6d19de7abf79cdbacf6d3cb0954a","indice":2,"prueba":74459,"timestamp":1670223304.5292623,"transacciones":[{"cantidad":10,"destino":"nodoB","origen":"nodoA","tiempo":1670223304.5201018},{cantidad":1,"destino":"127.0.0.1","origen":"0","tiempo":1670223304.5292585}]}],{"hash":"00007356b44f5f8b9aed01d490d8915c3672e38f311efa0f5407e32faacdee8b","hash_previo":"0000ea2e5af38327b3482c7f78c063f1c1578f489600d7bb21f83a75863a649a","indice":3,"prueba":37832,"timestamp":1670223305.5570247,"transacciones":[{"cantidad":10,"destino":"nodoB","origen":"nodoA","tiempo":1670223305.5462477},{cantidad":1,"destino":"127.0.0.1","origen":"0","tiempo":1670223305.5570226}]}],{"hash":"00009d725195cd2244ba0ce6b7d87436f99a0f9d787b9440a44e14312cfefbda","hash_previo":"00007356b44f5f8b9aed01d490d8915c3672e38f311efa0f5407e32faacdee8b","indice":4,"prueba":67396,"timestamp":1670223527.773243,"transacciones":[{"cantidad":10,"destino":"nodoB","origen":"nodoA","tiempo":1670223527.7107973},{cantidad":1,"destino":"127.0.0.1","origen":"0","tiempo":1670223527.7732406}]}],{"longitud":4}

{"mensaje":"La transaccion se incluira en el bloque con indice 5"}

{"mensaje":"Ha habido un conflicto. Esta cadena se ha actualizado con una version mas larga"}

{"chain":[{"hash":"1d581b6b49b933b344512dd031491c4df68d6d19de7abf79cdbacf6d3cb0954a","hash_previo":"0","indice":1,"prueba":0,"timestamp":0,"transacciones":[]},{hash":"0000ea2e5af38327b3482c7f78c063f1c1578f489600d7bb21f83a75863a649a","hash_previo":"1d581b6b49b933b344512dd031491c4df68d6d19de7abf79cdbacf6d3cb0954a","indice":2,"prueba":74459,"timestamp":1670223304.5292623,"transacciones":[{"cantidad":10,"destino":"nodoB","origen":"nodoA","tiempo":1670223304.5201018},{cantidad":1,"destino":"127.0.0.1","origen":"0","tiempo":1670223304.5292585}]}],{"hash":"00007356b44f5f8b9aed01d490d8915c3672e38f311efa0f5407e32faacdee8b","hash_previo":"0000ea2e5af38327b3482c7f78c063f1c1578f489600d7bb21f83a75863a649a","indice":3,"prueba":37832,"timestamp":1670223305.5570247,"transacciones":[{"cantidad":10,"destino":"nodoB","origen":"nodoA","tiempo":1670223305.5462477},{cantidad":1,"destino":"127.0.0.1","origen":"0","tiempo":1670223305.5570226}]}],{"hash":"00009d725195cd2244ba0ce6b7d87436f99a0f9d787b9440a44e14312cfefbda","hash_previo":"00007356b44f5f8b9aed01d490d8915c3672e38f311efa0f5407e32faacdee8b","indice":4,"prueba":67396,"timestamp":1670223527.773243,"transacciones":[{"cantidad":10,"destino":"nodoB","origen":"nodoA","tiempo":1670223527.7107973},{cantidad":1,"destino":"127.0.0.1","origen":"0","tiempo":1670223527.7732406}]}],{"hash":"000030ba32001e809dd11353c64ebd21def6797c629303aca82550f44d9cc5dc","hash_previo":"00009d725195cd2244ba0ce6b7d87436f99a0f9d787b9440a44e14312cfefbda","indice":5,"prueba":238499,"timestamp":1670223528.5446908,"transacciones":[{"cantidad":10,"destino":"nodoB","origen":"nodoC","tiempo":1670223528.5249813},{cantidad":1,"destino":"127.0.0.1","origen":"0","tiempo":1670223528.5446892}]}],{"longitud":5}

```

```

{"mensaje":"La transaccion se incluira en el bloque con indice 5"}

{"mensaje":"Ha habido un conflicto. Esta cadena se ha actualizado con una version mas larga"}

{"chain":[{"hash":"1d581b6b49b933b344512dd031491c4df68d6d19de7abf79cdbacf6d3cb0954a","hash_previo":"0","indice":1,"prueba":0,"timestamp":0,"transacciones":[]},{hash":"0000ea2e5af38327b3482c7f78c063f1c1578f489600d7bb21f83a75863a649a","hash_previo":"1d581b6b49b933b344512dd031491c4df68d6d19de7abf79cdbacf6d3cb0954a","indice":2,"prueba":74459,"timestamp":1670223304.5292623,"transacciones":[{"cantidad":10,"destino":"nodoB","origen":"nodoA","tiempo":1670223304.5201018},{cantidad":1,"destino":"127.0.0.1","origen":"0","tiempo":1670223304.5292585}]}],{"hash":"00007356b44f5f8b9aed01d490d8915c3672e38f311efa0f5407e32faacdee8b","hash_previo":"0000ea2e5af38327b3482c7f78c063f1c1578f489600d7bb21f83a75863a649a","indice":3,"prueba":37832,"timestamp":1670223305.5570247,"transacciones":[{"cantidad":10,"destino":"nodoB","origen":"nodoA","tiempo":1670223305.5462477},{cantidad":1,"destino":"127.0.0.1","origen":"0","tiempo":1670223305.5570226}]}],{"hash":"00009d725195cd2244ba0ce6b7d87436f99a0f9d787b9440a44e14312cfefbda","hash_previo":"00007356b44f5f8b9aed01d490d8915c3672e38f311efa0f5407e32faacdee8b","indice":4,"prueba":67396,"timestamp":1670223527.773243,"transacciones":[{"cantidad":10,"destino":"nodoB","origen":"nodoA","tiempo":1670223527.7107973},{cantidad":1,"destino":"127.0.0.1","origen":"0","tiempo":1670223527.7732406}]}],{"hash":"000030ba32001e809dd11353c64ebd21def6797c629303aca82550f44d9cc5dc","hash_previo":"00009d725195cd2244ba0ce6b7d87436f99a0f9d787b9440a44e14312cfefbda","indice":5,"prueba":238499,"timestamp":1670223528.5446908,"transacciones":[{"cantidad":10,"destino":"nodoB","origen":"nodoC","tiempo":1670223528.5249813},{cantidad":1,"destino":"127.0.0.1","origen":"0","tiempo":1670223528.5446892}]}],{"longitud":5}

```