Universidad de Huelva
Escuela Técnica Superior de Ingeniería

Implementación básica de una blockchain

José María Baeza-Herrazti Vázquez

Escuela Técnica Superior de Ingeniería de la universidad de Huelva.

18 de Diciembre del 2017.

Índice

Introducción a la tecnología blockchain

En la actualidad, la blockchain (“Cadena de bloques”) se trata de una nueva tecnología, que está en pleno auge, cuya característica principal es la descentralización.

¿Qué es la descentralización?

La descentralización consiste, en el ámbito de la blockchain, a no tener todo el contenido centralizado, es decir, en un solo computador, sino tener ese contenido en varios computadores, teniendo el contenido actualizado continuamente en cada computador.

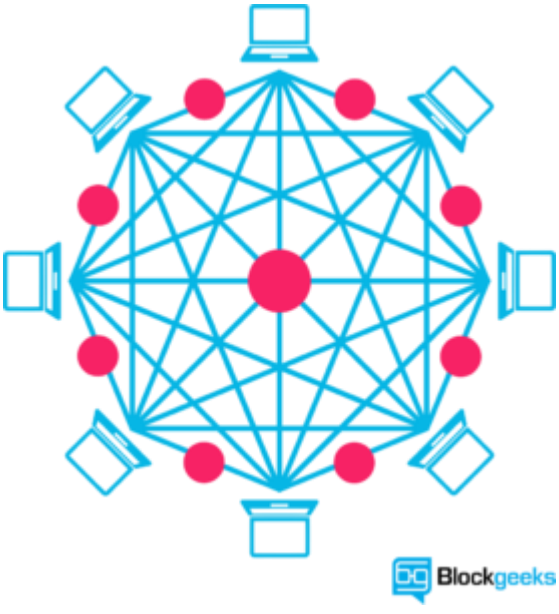
Imagina un Excel que se duplica miles de veces en una red de computadores. Esta red está diseñada para actualizar continuamente este Excel.

La información contenida en una blockchain existe como una base de datos compartida y continuamente actualizada. Esta es una forma de usar la red que tiene unos grandes beneficios. La base de datos de la blockchain no se almacena en una sola ubicación. No existe una versión centralizada de esta información para que un hacker la corrompa. Alojado por millones de ordenadores simultáneamente, sus datos son accesibles a cualquier persona a través de Internet.

La tecnología Blockchain es como Internet en el sentido de que tiene una robustez incorporada. Al almacenar bloques de información que son idénticos en toda su red, la blockchain no puede:

- Estar controlado por una sola entidad.
- No tiene un solo punto de falla.
- Bitcoin se inventó en 2008. Desde entonces, la blockchain de Bitcoin ha funcionado sin interrupción significativa. (Hasta la fecha, cualquiera de los problemas asociados con Bitcoin se han debido a la piratería informática o mala gestión. En otras palabras, estos problemas provienen de la mala intención y del error humano.)

Internet en sí mismo ha demostrado ser duradero durante casi 30 años. Es un récord que es un buen presagio para la tecnología de la blockchain a medida que continúa desarrollándose.



A cada computador que forma parte de la blockchain se le llama ‘nodo’. Con respecto a la blockchain de Bitcoin, cada nodo es un "administrador" de la blockchain y se une a la red voluntariamente (en este sentido, la red está

descentralizada). Sin embargo, cada uno tiene un incentivo para participar en la red: la posibilidad de ganar Bitcoins.

Ahora se reconoce que es sólo la primera de muchas aplicaciones potenciales de la tecnología.

Se estima que ya hay disponibles unas 700 criptomonedas similares a Bitcoin (valor intercambiable de las fichas) disponibles, que utilizan la tecnología blockchain para ofrecer diferentes servicios, como almacenamiento en la nube descentralizado, contratos inteligentes, almacenamiento de tus datos de identificación como dni, etc.

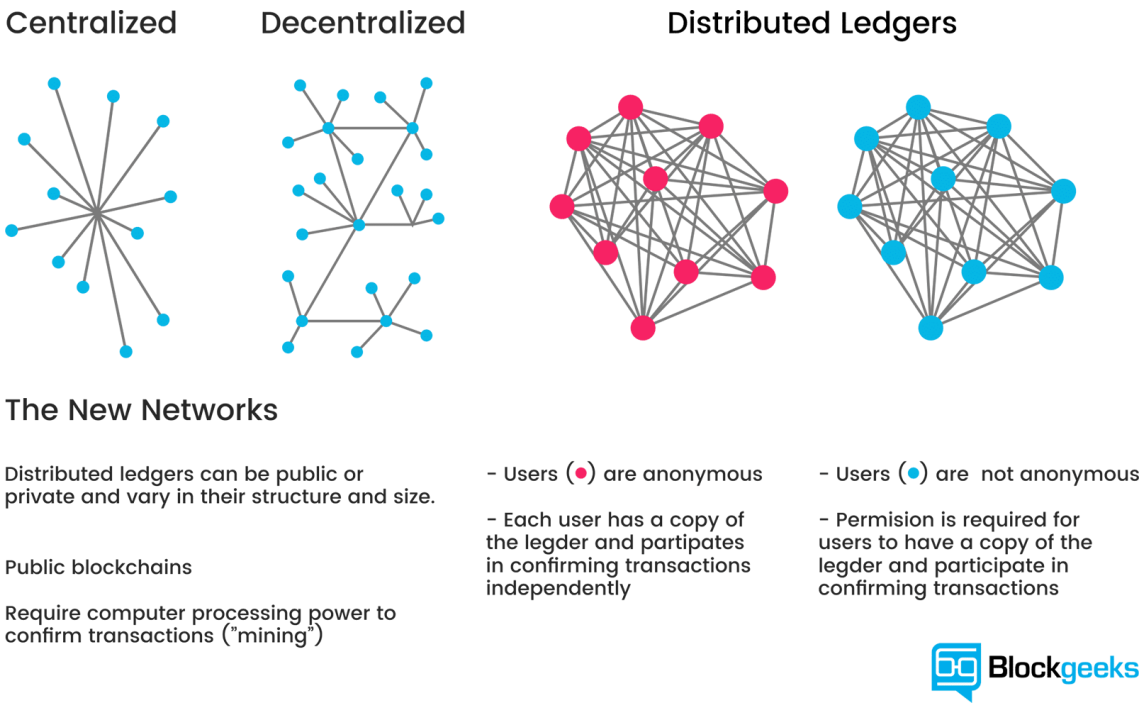
¿Y en cuanto a la seguridad?

Al almacenar datos a través de su red, la blockchain elimina los riesgos que conlleva el almacenamiento centralizado de datos.

Su red carece de puntos de vulnerabilidad centralizados que los piratas informáticos puedan explotar. El Internet de hoy en día tiene problemas de seguridad que son familiares para todos. Todos dependemos del sistema "nombre de usuario/contraseña" para proteger nuestra identidad y activos en línea. Los métodos de seguridad de Blockchain utilizan tecnología de cifrado.

La base para ello son las denominadas "llaves" públicas y privadas. Una "clave pública" (una larga cadena de números generada aleatoriamente) es la dirección de un usuario en la blockchain. Los Bitcoins enviados a través de la red se registra como perteneciente a esa dirección. La "clave privada" es como una contraseña que da a su

propietario acceso a sus Bitcoins u otros activos digitales. Si guardas los activos en la blockchain, es incorruptible. Aunque la protección de sus activos digitales también requerirá proteger su clave privada imprimiéndola.



¿Cómo podremos hacer uso de una blockchain?

Como infraestructura web, usted no necesita saber sobre la blockchain para que sea útil en su vida.

Actualmente, el financiamiento ofrece los casos de mayor uso para la tecnología. Las remesas internacionales, por ejemplo. El Banco Mundial calcula que en 2015 se enviaron más de 430.000 millones de dólares estadounidenses en transferencias de dinero. Y por el momento hay una gran demanda de desarrolladores de blockchains.

La blockchain reduce potencialmente al intermediario para este tipo de transacciones. Esto permite una velocidad muy superior en las transacciones con blockchain, comparado a como se hacen las transacciones en los bancos, de forma que cada transacción tiene que pasar por un intermediario para que este la valide, perdiendo así mucho tiempo. La blockchain en sí misma tiene intermediarios, que son cada uno de los nodos que la forman, ya que son los que validan las transacciones de forma segura, ya que cada transacción nueva debe cumplir una prueba criptográfica computacionalmente alta.

Aplicabilidad

- Contratos inteligentes

  Los leyes de formas distribuidas permiten la codificación de contratos simples que se ejecutarán cuando se cumplan las condiciones especificadas. Ethereum es un proyecto de código abierto de blockchain que fue construido específicamente para realizar esta posibilidad. Sin embargo, en sus primeras etapas, Ethereum tiene el potencial para aprovechar la utilidad de las blockchains en una escala que verdaderamente puede cambiar el mundo.

  En el nivel actual de desarrollo de la tecnología, los contratos inteligentes pueden programarse para realizar funciones sencillas. Por ejemplo, una apuesta de un partido de fútbol podría pagarse cuando tras terminar el partido, el contrato mirase el resultado en internet y le diese el correspondiente dinero al que haya ganado la apuesta, con el uso de la tecnología de blockchain y Bitcoin que permite automatizar el pago.

- La economía compartida

  Con empresas como Uber y AirBnB floreciendo, la economía compartida ya es un éxito probado. En la actualidad, sin embargo, los usuarios que desean disfrutar de un servicio de uso compartido tienen que recurrir a un intermediario como Uber. Al permitir los pagos entre pares, la blockchain abre la puerta a la interacción directa entre las partes, una verdadera

descentralización de los resultados económicos compartidos.

- Gobernanza

  Al hacer que los resultados sean completamente transparentes y accesibles al público, la tecnología de blockchain podría aportar total transparencia a las elecciones o a cualquier otro tipo de votación. Los contratos inteligentes basados en Ethereum ayudan a automatizar el proceso.

  La aplicación, Boardroom, permite la toma de decisiones organizacionales en la blockchain. En la práctica, esto significa que el gobierno de la empresa se vuelve completamente transparente y verificable cuando se gestionan activos digitales, acciones o información.

- Auditoría de la cadena de suministro

  Los consumidores desean cada vez más saber que las afirmaciones éticas que las empresas hacen sobre sus productos son reales. Las leyes distribuidas proveen una manera fácil de certificar que las historias de las cosas que compramos son ciertas. La transparencia viene con la marca de tiempo basada en la blockchain de una fecha y el lugar.

  Provenance, con sede en el Reino Unido, ofrece auditoría de la cadena de suministro para una amplia gama de bienes de consumo. Haciendo uso de la blockchain de Ethereum, un proyecto piloto de

Provenance asegura que el pescado vendido en restaurantes de Sushi en Japón ha sido cosechado de manera sostenible por sus proveedores en Indonesia.

- Almacenamiento de archivos

  Descentralizar el almacenamiento de archivos en Internet ofrece claras ventajas. La distribución de datos a través de la red protege los archivos de ser hackeados o perdidos.

  Inter Planetary File System (IPFS) facilita la conceptualización de cómo podría funcionar una web distribuida. Similar a la forma en que un bittorrent mueve datos por Internet, IPFS elimina la necesidad de relaciones cliente-servidor centralizadas (es decir, la web actual). Un Internet formado por sitios web completamente descentralizados tiene el potencial de acelerar la transferencia de archivos y los tiempos de streaming. Esta mejora no sólo es conveniente. Es una actualización necesaria para los sistemas de entrega de contenido sobrecargados de la web.

- Mercados de predicción

  Se ha demostrado que el crowdsourcing de predicciones sobre la probabilidad de eventos tiene un alto grado de precisión. El promediar opiniones anula los sesgos no examinados que distorsionan el juicio. Los mercados de predicción que pagan según los resultados del evento ya están activos. Blockchains son una tecnología de "sabiduría de la multitud" que

sin duda encontrará otras aplicaciones en los próximos años.

Aún así, en Beta, la aplicación de predicción de mercado Augur hace ofertas de acciones sobre el resultado de eventos del mundo real. Los participantes pueden ganar dinero comprando la predicción correcta. Cuantas más acciones se compren en el resultado correcto, mayor será el pago. Con un pequeño compromiso de fondos (menos de un dólar), cualquiera puede hacer una pregunta, crear un mercado basado en un resultado previsto y cobrar la mitad de todas las comisiones de transacción que el mercado genera.

- Protección de la propiedad intelectual

  Como es bien sabido, la información digital puede ser reproducida infinitamente (y distribuida ampliamente gracias a Internet). Esto le ha dado a los usuarios de la web en todo el mundo una mina de oro de contenido gratuito. Sin embargo, los titulares de derechos de autor no han tenido tanta suerte, perdiendo el control sobre su propiedad intelectual y sufriendo financieramente como consecuencia de ello. Los contratos inteligentes pueden proteger los derechos de autor y automatizar la venta de obras creativas en línea, eliminando el riesgo de copiar y redistribuir archivos.

  Mycelia utiliza la blockchain para crear un sistema de distribución de música punto a punto. Fundada por el

cantautor británico Imogen Heap, Mycelia permite a los músicos vender canciones directamente al público, así como licenciar muestras a los productores y distribuir regalías a compositores y músicos (todas estas funciones se automatizan mediante contratos inteligentes). La capacidad de las blockchains para emitir pagos en cantidades fraccionarias de cripto-divisas (micropagos) sugiere que este caso de uso para la blockchain tiene una gran probabilidad de éxito.

- Internet de las cosas (IoT)

  ¿Qué es la IO? La gestión controlada en red de determinados tipos de dispositivos electrónicos, por ejemplo, el control de la temperatura del aire en una instalación de almacenamiento. Los contratos inteligentes hacen posible la automatización de la gestión de sistemas remotos. Una combinación de software, sensores y red facilita el intercambio de datos entre objetos y mecanismos. El resultado aumenta la eficiencia del sistema y mejora el control de costes.

  Los principales actores de la industria manufacturera compiten por el dominio de IoT. Piensa en Samsung, IBM y AT&T.

Además de estos ejemplos de aplicabilidad, existen muchísimos otros, tan solo hemos visto los que hasta ahora han sido considerados los casos de uso más interesantes para la tecnología blockchain.

Programación funcional Vs Otra programación.

¿Por qué debemos de crear una blockchain con un lenguaje funcional?

La tecnología de Blockchain se basa en el concepto de inmutabilidad, el hecho de que no se puede cambiar una transacción, pero la mayoría de las implementaciones de blockchain se crean utilizando lenguaje de programación estructurados, como C++ o Go. Esto puede conducir a una multitud de posibles vulnerabilidades y efectos secundarios no deseados. Afortunadamente los lenguajes funcionales no tienen variables mutables y una vez que estableces un valor no se puede cambiar.

La programación funcional le permite a uno crear sistemas altamente paralelizados y distribuidos, en parte porque el aspecto de la inmutabilidad conduce a una serie de optimizaciones del compilador. Junto con las funciones integradas para la iteración, como ‘map’ y ‘reduce’, parece que hay muy pocas razones para usar C ++ o cualquier otro lenguaje de programación estructurada para el desarrollo de una blockchain.

Por esto esencialmente, pienso que los lenguajes de programación más adaptados a este tipo de tecnología son los lenguajes funcionales, como Haskell. Se pueden ver varias criptomonedas, que para implementar sus blockchains están utilizando este tipo de lenguaje de programación. Una de estas monedas es ‘Cardano’, que está teniendo un aumento en su valor descomunal (https://iohk.io/projects/cardano/).

Explicación sencilla de cómo funciona una blockchain e implementación en leguaje sencillo.

Vamos a explicar cómo funciona una blockchain por dentro y veréis que de verdad, está tecnología es sumamente segura y viene para cambiar nuestra forma de hacer las cosas.

Una blockchain es una cadena de registros inmutables y secuenciales, llamados bloques. En realidad, puede contener transacciones, archivos o cualquier dato que te guste, lo único que habría que hacer es adaptar los bloques para que puedan contener el tipo de archivo que queramos. Pero lo importante es que están encadenados usando hashes (funciones hash).

Vamos a explicar el funcionamiento de una blockchain, realizando una paso a paso, en un lenguaje sencillo de entender y de implementar como es python. Como hemos visto, el mejor lenguaje para implementar esta nueva tecnología son los lenguajes funcionales, pero como nuestro fin es enseñar cómo funciona la blockchain, vamos a explicarla mediante una implementación en un lenguaje sencillo.

Nos vamos a comunicar con la blockchain con peticiones HTTP.

Nuestra blockchain va a tener:

- Cadena de bloques.
- Transacciones.

Bloque: Cada bloque tendrá:

- Índice: Indica el número de bloque.
- Marca de tiempo (Unix): Tiempo en el que se hizo la transacción.
- Lista de transacciones.
- Una Prueba: Es un número que luego veremos cómo se calcula con un algoritmo llamado Prueba de Trabajo. Para el primer bloque le pondremos Prueba=1.
- Hash del bloque anterior: Para el primer bloque le pondremos un Anterior_Hash = 1. Para los demás, lo que haremos será crear una función hash para el bloque anterior completo, de manera que cada bloque tendrá el hash del bloque anterior.

La idea de una cadena es: cada bloque nuevo contiene en sí mismo, el hash del bloque anterior. Esto es importante porque es lo que hace inmutable a la blockchain: Si un atacante corrompiera un bloque anterior en la cadena, todos los bloques subsiguientes contendrán hashes incorrectos.

Todas las transacciones son registradas y transmitidas a todos los nodos de la red. Así, todos los integrantes tendrán la información constantemente actualizada con todas las transacciones.

Los nodos son los que se conocen como mineros:

- Aseguran que las transacciones son reales y legítimas.

Contra más nodos haya, más segura será nuestra blockchain, debido a que esta estará más descentralizada.

Vamos a comenzar a implementar la blockchain en python:

Empezaremos con lo sencillo… Voy a poner directamente el código, comentado por si no se entendiera algo.

```
import hashlib
import json
from textwrap import dedent
from time import time
from uuid import uuid4

from flask import Flask, jsonify, request

import requests
from urllib.parse import urlparse


class  Blockchain():
    def  __init__(self):
        self.cadena_bloques = []
        self.transacciones = []
        self.nodos = set()
        """Esta es una forma facil de garantizar que la adición de nuevos
        nodos sea idempotente, lo que significa que no importa
        cuántas veces agreguemos un nodo específico, aparece exactamente una vez.
        """

        # Creamos el bloque genesis (Primer bloque sin predecesores)
        self.nuevo_bloque(1, anterior_hash=1)



    def nuevo_bloque(self,prueba, anterior_hash=None):
        #Crea un nuevo bloque y lo agrega a la cadena
        """prueba -> Prueba dada por el algoritmo Proof of Work
        anterior_hash -> hash del bloque anterior
        return el nuevo bloque (DICCIONARIO)"""
        bloque = {
            'indice': len(self.cadena_bloques) + 1,
            'timestamp': time(),
            'transacciones': self.transacciones,
            'prueba': prueba,
            'anterior_hash': anterior_hash or self.hash(self.cadena_bloques[-1]),
        }

        # Reseteamos la lista de transacciones
        self.transacciones = []

        self.cadena_bloques.append(bloque)
        return bloque
```

```
def nueva_transaccion(self, remitente, destinatario, cantidad):
    # Agrega una nueva transaccion a la lista de transacciones
    self.transacciones.append ({
        'remitente' : remitente,
        'destinatario' : destinatario,
        'cantidad' : cantidad,
    })

    return  self.ultimo_bloque['indice'] + 1

@staticmethod
def hash(bloque):
    """
    Crea un SHA-256 hash de un bloque
    SHA-256 es un algoritmo
    """
    #Debemos asegurarnos de que el Diccionario esté Ordenado, o tendremos problemas de inconsistencia.
    bloque_string = json.dumps(bloque, sort_keys=True).encode()
    return hashlib.sha256(bloque_string).hexdigest()

@property
def ultimo_bloque(self):
    # Devuelve el último bloque en la cadena
    return self.cadena_bloques[-1]
```

Tras crear los métodos más básicos, vamos a entender para qué y cómo funciona el algoritmo Proof of Work (Prueba de trabajo), que necesitaremos implementar para la función de añadir nuevos bloques.

Comprendiendo “Proof of Work”

Un algoritmo de prueba de trabajo (PoW) es cómo se crean o extraen nuevos bloques en la cadena de bloques. El objetivo del PoW es descubrir un número que resuelva una operación matemática. El número debe ser difícil de encontrar, pero fácil de verificar por cualquiera de la red.

En Bitcoin, el PoW se llama HashCoin y es un estilo a el siguiente:

```
X=5

Y=0 #En principio no sabemos la Y… vamos a empezar probando con Y=0

While sha256(f’{x*y}’.encode()).hexdigest()[-1] != 0:

        Y += 1

Print(“La ‘y’ que hace que ese algoritmo nos de una clave terminada en 0 es ” + y)
```

Para este ejemplo de X=5, sale una Y=21.

Un algoritmo parecido a ese es el que los mineros compiten por resolver para crear un nuevo bloque. En general, la dificultad viene dada por el número de caracteres buscados en una cadena. Los mineros son recompensados por su solución, recibiendo un % en la transacción.

Vamos a implementar ahora un PoW nosotros:

Nuestra regla se parecerá a la anterior vista: “Encontrar un número ‘p’ que cuando se haya generado un hash con la solución del bloque anterior, se genere un hash con 4 últimas cifras a 0”.

```
class  Blockchain():

    def proof_of_work(self, anterior_prueba):
        """
        Algoritmo de prueba de trabajo simple:
         - Encontrar un numero p tal que hash(p,p') contiene 4 ceros a la izquierda,
         donde p es el anterior de p'
         - p es el anterior prueba, y p' es el nuevo prueba
        """

        prueba_nueva = 0
        while self.valid_proof(anterior_prueba, prueba_nueva) == False:
            prueba_nueva += 1

        return prueba_nueva

    @staticmethod
    def valid_proof(anterior_prueba, prueba_nuevo):
        """
        Valida la prueba: ¿El hash (anterior_prueba, prueba_nuevo) contiene 4 ceros a la izquierda?
        : param anterior_prueba: <int> Prueba anterior
        : prueba de param: <int> Prueba actual
        : return: <bool> Verdadero si es correcto, falso si no.
        """

        guess = f'{anterior_prueba}{prueba_nuevo}'.encode()
        guess_hash = hashlib.sha256(guess).hexdigest()
        return guess_hash[:4] == "0000"

        """Para ajustar la dificultad del algoritmo, podríamos modificar el número
        de ceros a la izquierda. Pero 4 es suficiente. Descubrirá que la adición
        de un cero inicial único marca una gran diferencia con el tiempo
        requerido para encontrar una solución."""
```

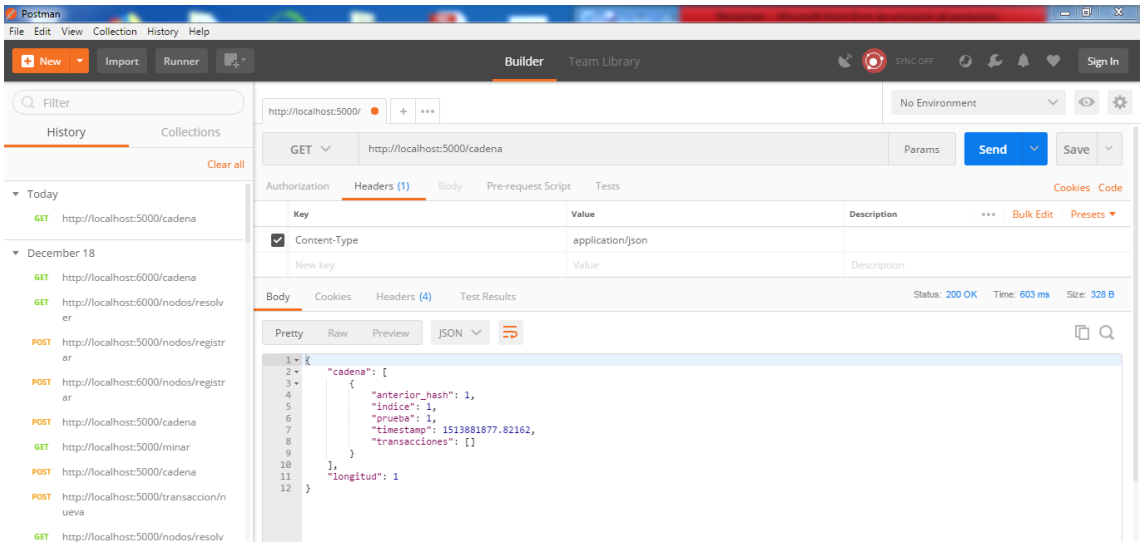
Ahora ya tan solo nos queda crear una API para interactuar con nuestra blockchain mediante llamadas HTTP:

```
class  Blockchain():
...
# Instanciamos nuestro nodo
app = Flask(__name__)

# Genera una dirección global única para este nodo
node_identifier = str(uuid4()).replace('-', '')

# Instanciamos la blockchain
blockchain = Blockchain()


@app.route('/minar', methods=['GET'])
def minar():
    """
    Para que una transaccion se incluya en el libro mayor, es decir,
    sea validada, un minero
    debe aceptarlo, minando
    """
    # Ejecutamos el algoritmo de prueba de trabajo para obtener la siguiente prueba ...
    last_block = blockchain.ultimo_bloque
    last_proof = last_block['prueba']
    proof = blockchain.proof_of_work(last_proof)

    # Debemos recibir una recompensa por encontrar la prueba.
    # El remitente es "0" para indicar que este nodo ha extraído una nueva moneda.
    blockchain.nueva_transaccion(
        remitente="0",
        destinatario="josejosejosejosejosejose", #Ese va a ser mi nodo
        cantidad=1,
    )

    # Forge el nuevo bloque agregándolo a la cadena
    previous_hash = blockchain.hash(last_block)
    block = blockchain.nuevo_bloque(proof, previous_hash)

    response = {
        'message': "Nuevo Bloque Unido",
        'indice': block['indice'],
        'transacciones': block['transacciones'],
        'prueba': block['prueba'],
        'anterior_hash': block['anterior_hash'],
    }
    return jsonify(response), 200
```

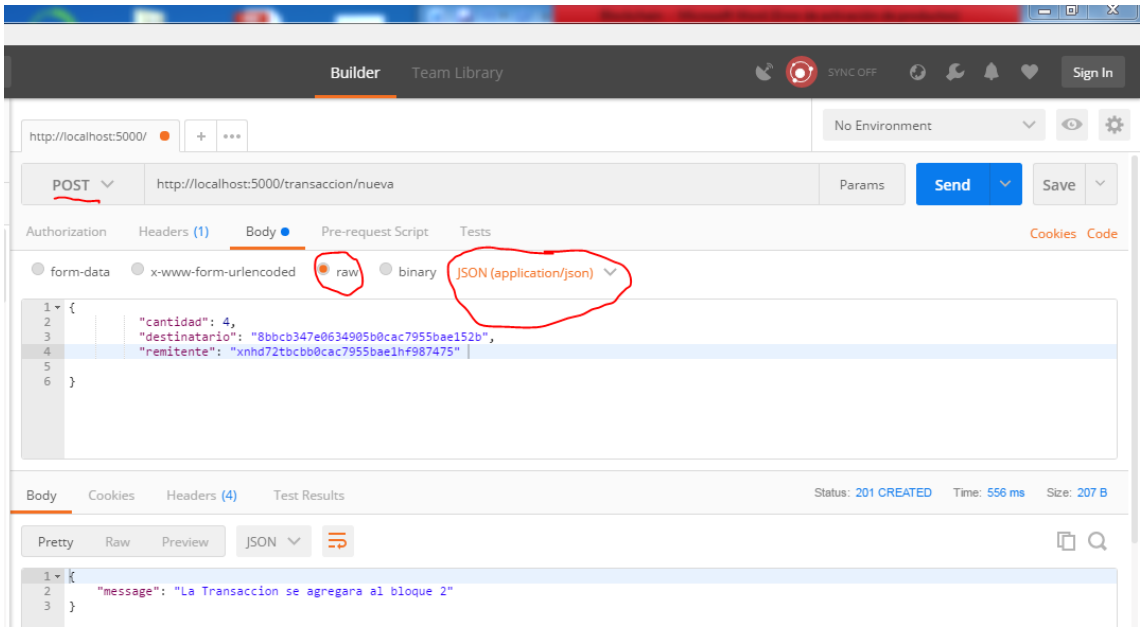
```python
@app.route('/transaccion/nueva', methods=['POST'])
def nueva_transaccion():
    """
    {
    "remitente": "1010",
    "destinatario": "joselito",
    "cantidad": 3
    }
    """
    values = request.get_json()
    print('Valores: '+str(values))

    # Chquea que los campos de las variables POST son correctos
    required = ['remitente', 'destinatario', 'cantidad']
    if not all(k in values for k in required):
        return 'Faltan Valores!', 400

    # Crea una nueva transaccion
    index = blockchain.nueva_transaccion(values['remitente'], values['destinatario'], values['cantidad'])

    response = {'message': f'La Transaccion se agregara al bloque {index}'}
    return jsonify(response), 201


@app.route('/cadena', methods=['GET'])
def Cadena_Total():
    response = {
        'cadena': blockchain.cadena_bloques,
        'longitud': len(blockchain.cadena_bloques),
    }
    return jsonify(response), 200
```

Ahora podemos crear transacciones y minarlas, para que se unan a la cadena de bloques. También podríamos ver la cadena de bloques completa:

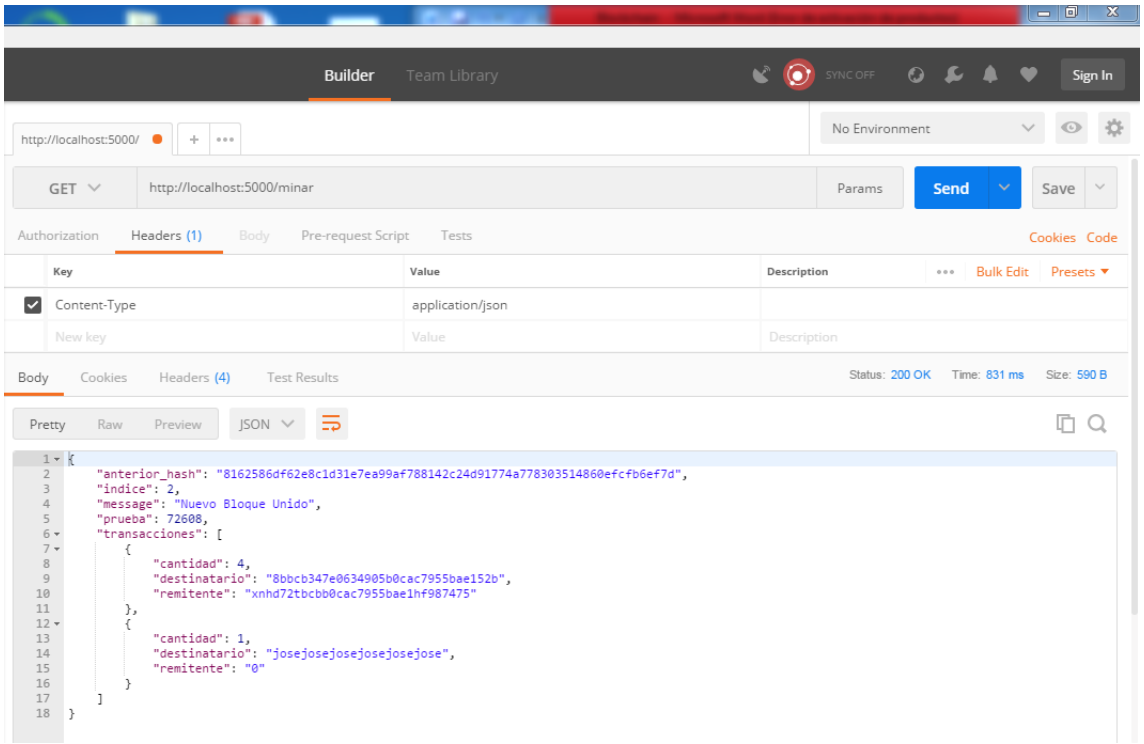
Utilizando el programa Postman por ejemplo, que permite realizar llamadas http con variables post de manera más sencilla, podemos ver la cadena actual:



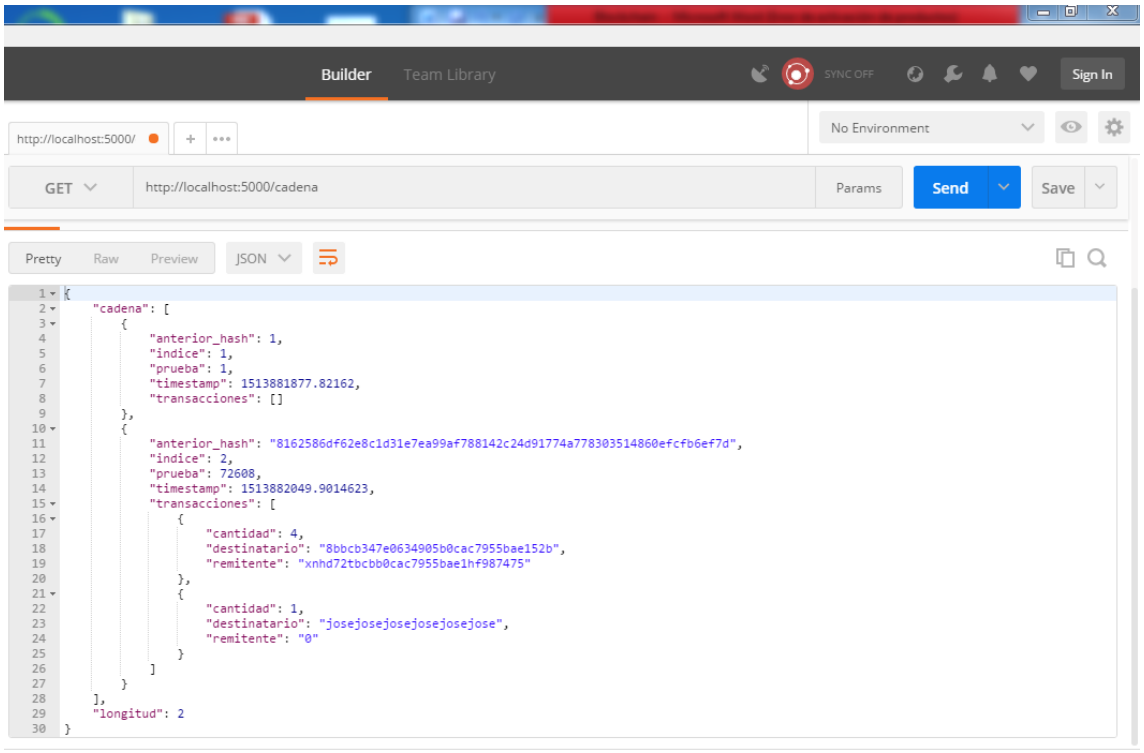
Tras esto, lo que podemos hacer es una nueva transacción, de la siguiente manera:



Ahora lo que haremos será minarla, para unirla a la cadena de bloques:



Y como vemos a continuación, la transacción se unió:



Esto es muy genial. Tenemos un Blockchain básico que acepta transacciones y nos permite extraer nuevos Bloques. Pero el objetivo de Blockchain es que deberían descentralizarse . Y si están descentralizados, ¿cómo diablos nos aseguramos de que todos reflejen la misma cadena? Esto se denomina problema de consenso , y tendremos que implementar un algoritmo de consenso si queremos más de un nodo en nuestra red.

- Registro de nuevos nodos

  Antes de que podamos implementar un Algoritmo de consenso, necesitamos una forma de que un nodo sepa sobre los nodos vecinos en la red. Cada nodo en nuestra red debe mantener un registro de otros nodos en la red. Por lo tanto, necesitaremos más puntos finales:

/nodes/register para aceptar una lista de nuevos nodos en forma de URL.

/nodes/resolve para implementar nuestro Algoritmo de consenso, que resuelve cualquier conflicto, para garantizar que un nodo tenga la cadena correcta.

```
@app.route('/nodos/registrar', methods=['POST'])
def registrar_nodos():
    # {"nodos": "http://192.168.0.119:5000"}
    values = request.get_json()

    nodes = values.get('nodos')
    if nodes is None:
        return "Error: Por favor, ponga una lista correcta de nodos", 400

    blockchain.registrar_nodo(nodes)

    response = {
        'message': 'El nuevo nodo ha sido añadido!',
        'Nodos_Totales': list(blockchain.nodos),
    }
    return jsonify(response), 201


@app.route('/nodos/resolver', methods=['GET'])
def consenso():
    replaced = blockchain.resolver_conflictos()

    if replaced:
        response = {
            'message': 'Nuestra cadena fue reemplazada',
            'new_chain': blockchain.cadena_bloques
        }
    else:
        response = {
            'message': 'Nuestra cadena tiene autoridad',
            'chain': blockchain.cadena_bloques
        }

    return jsonify(response), 200


if __name__ == '__main__':
    app.run(host='0.0.0.0', port=5000)
```

- Implementando el Algoritmo de Consenso

  Como se mencionó, un conflicto ocurre cuando un nodo tiene una cadena diferente a otro nodo. Para resolver esto, estableceremos la regla de que la cadena válida más larga es autoritativa. En otras palabras, la cadena más larga de la red es la verídica. Usando este algoritmo, alcanzamos el consenso entre los nodos en nuestra red.

```
class  Blockchain():

def registrar_nodo(self, direccion):
        """
        Agregue un nuevo nodo a la lista de nodos
        : direccion =  P.ej. 'http://192.168.0.5:5000'
        """

        parsed_url = urlparse(direccion)
        self.nodos.add(parsed_url.netloc) #path


    def validar_cadena(self, cadena):
        """
        El primer método valid_chain()es responsable de verificar si una cadena
        es válida al recorrer cada bloque y verificar tanto el hash como la prueba.
        """
        """
        Determinar si una cadena de bloques dada es válida
        : cadena param: <list> A blockchain
        : return: <bool> True si es válido, False si no
        """

        last_bloque = cadena[0]
        current_index = 1
        while current_index < len(cadena):
            bloque = cadena[current_index]
            print(f'{last_bloque}')
            print(f'{bloque}')
            print("\n-----------\n")
            # Verifica que el hash del bloque sea correcto
            if bloque['anterior_hash'] != self.hash(last_bloque):
                return False

            # Verifica que la Prueba de trabajo sea correcta
            if not self.valid_proof(last_bloque['prueba'], bloque['prueba']):
                return False

            last_bloque = bloque
            current_index += 1

        return True
```

```
def resolver_conflictos(self):
    """
    resolve_conflicts()es un método que recorre todos los nodos
    vecinos, descarga sus cadenas y las verifica utilizando el método
    anterior. Si se encuentra una cadena
    válida, cuya longitud es mayor que la nuestra, reemplazamos la nuestra."""

    neighbours = self.nodos
    nueva_cadena = None

    # Solo estamos buscando cadenas más largas que las nuestras
    max_length = len(self.cadena_bloques)

    new_chain=None
    # Coge y verifica las cadenas de todos los nodos de nuestra red
    for node in neighbours:
        response = requests.get(f'http://{node}/cadena')

        if response.status_code == 200:
            print('JSON: '+str(response.json()))
            length = response.json()['longitud']
            chain = response.json()['cadena']

            # Check if the length is longer and the chain is valid
            if length > max_length and self.validar_cadena(chain):
                max_length = length
                new_chain = chain

    # Replace our chain if we discovered a new, valid chain longer than ours
    if new_chain:
        self.cadena_bloques = new_chain
        return True

    return False
```

El primer método validar_cadena() es responsable de verificar si una cadena es válida al recorrer cada bloque y verificar tanto el hash como la prueba.

resolver_conflictos() es un método que recorre todos los nodos vecinos, descarga sus cadenas y las verifica utilizando el método anterior. Si se encuentra una cadena válida, cuya longitud es mayor que la nuestra, reemplazamos la nuestra.

En este punto, puedes coger una máquina diferente si lo desea y girar diferentes nodos en su red. O desarrolle procesos utilizando diferentes puertos en la misma máquina. Hice girar otro nodo en mi máquina, en un puerto diferente, y lo registré con mi nodo

actual. Por lo tanto, tengo dos nodos: http://localhost:5000 y http://localhost:6000.

Ahora para cada Pc ó puerto tengo que registrar todos los otros nodos de la red.

Cada vez que vayamos a minar un nuevo bloque, ese Pc deberá actualizar su cadena de bloques, por si anteriormente otro Pc había forjado (unido) un bloque a la blockchain. Para actualizar la cadena haremos una llamada a “/nodos/resolver”.

Eso es todo, os voy a dejar el código completo de la blockchain de ejemplo en mi repositorio git para que podáis descargarlo y hacer prueba, incluso poder mejorarla. Os recomiendo ver el código y entender que es lo que hace, porque es muy sencillo aunque al principio parezca muy complicado de entender.

Aquí os dejo el link del repositorio, ¡hasta pronto!

Git: https://github.com/josebaeza14/BlockchainPython