ISIS-1221
INTRODUCCIÓN A LA PROGRAMACIÓN

Proyecto de Nivel 3 - Blockchain de Cupicoin

Objetivo general

El objetivo general de este proyecto es que usted practique todos los conceptos estudiados en el nivel 3 del curso. Recuerde que este proyecto debe realizarse de forma **completamente individual**.

Objetivos específicos

1. Practicar la lectura y escritura de archivos con formato CSV.
2. Ejercitar la implementación de algoritmos de recorrido y modificación de listas y diccionarios.
3. Familiarizarse con la estructura compleja de datos de diccionarios dentro de listas.
4. Fomentar la habilidad de descomponer un problema en subproblemas y de implementar funciones que los resuelven, lo que se conoce comúnmente como la técnica de "Dividir y Conquistar".

Contexto

Blockchain

Blockchain es una estructura para el almacenamiento de datos, los cuales están asegurados criptográficamente para que si alguien hace una modificación, sea fácil descubrir dónde ocurrió el cambio. La estructura *Blockchain* distribuye la información a lo largo de varios nodos conocidos como bloques, que almacenan información, su propio código hash (un código único que sirve como la huella digital del contenido del bloque) y la vinculación con el bloque anterior a través de su código hash. Por lo anterior, cada bloque tiene un lugar específico e inamovible dentro de la cadena. La Figura 1 muestra un diagrama de alto nivel de un Blockchain.

Figura 1. Diagrama de alto nivel de un Blockchain

El código hash de un bloque se genera a partir de la información que contiene incluido el hash del bloque anterior. Por este motivo, si cambia el contenido de un bloque también debería cambiar el hash del bloque, y esto debería verse reflejado en el siguiente bloque y en los posteriores para que la cadena siguiera siendo válida. Para agregar nueva información a un blockchain, la información debe verificarse y luego debe ser añadida a un nuevo bloque que, cuando tenga suficiente información, será encadenado al final de la cadena.
Por lo anterior, un blockchain ofrece una alta seguridad de los datos, casi hasta el punto de ser "inhackeable" ofreciendo confianza y seguridad a quienes deciden utilizarla. Blockchain puede ser utilizado para cualquier

tipo de información que necesite ser preservada de forma intacta y que deba permanecer disponible de manera segura. Por esto, se encuentran aplicaciones de blockchain en la salud con los registros médicos de los pacientes, gestión de documentos digitales, análisis del mercado de Internet de las Cosas (IoT) y, una de sus aplicaciones más conocidas verificar, validar, rastrear y almacenar dinero y transacciones financieras.

Cupicoin

Las criptomonedas son monedas digitales que utilizan un registro electrónico para garantizar la integridad de las transacciones, por ejemplo, utilizando una estructura tipo blockchain. Aparte del famoso caso de Bitcoin, existen otras criptomonedas como Ethereum, Polygon, Dogecoin, Solana y Harmony. Almacenar las transacciones en un blockchain hace extremadamente difícil que alguien anule una transacción con estas monedas o que haga modificaciones (por ejemplo, cambiando el destinatario o el valor de una transferencia): casi cualquier cambio haría que la cadena se volviera inválida.

Una emergente criptomoneda, llamada Cupicoin, basa su esquema de registro de transacciones en la estructura de tipo blockchain. Actualmente, 1 Cupicoin equivale a 12.000 COP. El sistema de registro de Cupicoin tiene 3 componentes principales: cuentas, bloques y transacciones.

- Las cuentas son objetos que almacenan Cupicoins y que pueden interactuar entre sí. Por ejemplo, desde una cuenta “A” es posible enviar Cupicoins a una cuenta “B”. Cada cuenta tiene una dirección única que actúa como su identificador.
- Los bloques en el blockchain de Cupicoin son conjuntos de transacciones. Cada bloque tiene un número único de identificación, un sello temporal (timestamp[1]) que indica su fecha de creación dentro del blockchain representado con un entero y la información de las transacciones almacenadas en el bloque. Cada bloque también contiene su propio código hash que depende del número del bloque, de las transacciones y del código hash del bloque anterior.
- Las transacciones almacenadas en un bloque tienen un código de transacción, un remitente, un destinatario, un valor y un tipo de operación. Las transacciones de tipo “transferencia” son movimientos de Cupicoins entre una cuenta remitente y una cuenta destinataria. Por otra parte, las transacciones de tipo “contrato” son movimientos que tienen únicamente definida la cuenta remitente y no tienen una cuenta destino. Los contratos pueden verse entonces como transacciones que incrementan o disminuyen la cantidad de Cupicoins en una cuenta como respuesta a alguna transacción en el mundo real.

Códigos hash de los bloques

Para que el esquema de seguridad de Cupicoin funcione, es necesario que el código hash de cada bloque sea un valor fácil de calcular con una función que dependa tanto del contenido del bloque como del hash del bloque anterior. De esta manera, si alguien llega a modificar alguna transacción, será muy fácil detectar si el código hash almacenado en el bloque coincide con el contenido o no. Como el código hash de cada bloque estará almacenado también en el siguiente bloque, se tendrá una segunda copia de este valor para comprobar que no haya habido cambios.

Finalmente, la función de hash debe ser una función que requiera cálculos sencillos, pero en grandes cantidades y no debe ser invertible. Lo primero busca que no se pueda romper la seguridad del blockchain a fuerza bruta, utilizando muchos computadores para hacer cálculos rápidamente. Lo segundo busca que sea difícil partir de un código hash para llegar a un conjunto de transacciones falsas que cumplan con las condiciones de consistencia de la cadena.

---

[1] En este proyecto se usará repetidamente el término ‘timestamp’ que hace referencia a un instante preciso de tiempo en que ocurrió algo. Para facilitar los cálculos con fechas y horas, es común que se utilice un valor conocido como Unix Epoch que corresponde a la cantidad de segundos transcurridos desde la media noche del 1 de enero de 1970. Por ejemplo, el número 747270000 corresponde a las 6:00:00 del domingo 5 de septiembre de 1993. Puede usar el sitio web https://www.epochconverter.com/ para saber a qué número corresponde una fecha particular y viceversa.

En el caso de Cupicoin, la función de hash que se utiliza para calcular el hash de cada bloque es una función que realice las siguientes operaciones:

1. Concatenar la información de cada transacción del bloque (código de transacción, remitente, destinatario, valor y tipo de operación).
2. Concatenar las cadenas de las transacciones entre ellas, con el número del bloque y con el hash del bloque anterior (**en ese orden**) para obtener una cadena que representa al bloque entero.
3. Sumar los códigos ASCII de cada letra en la cadena del bloque.
4. Aplicarle al valor anterior la operación módulo (%) usando el *timestamp* del bloque.

Si llamamos *SumaASCII* a la suma de los códigos ASCII de los caracteres de la cadena formada por el contenido del bloque, y *timestamp* al número que representa el momento en que se calculó el hash del bloque, matemáticamente tendríamos la siguiente función:

$$hash(SumaASCII, timestamp) = SumaASCII \ mod \ timestamp$$

Esta misma función en Python se vería de la siguiente manera:

```
def hash (SumaASCII: int, timestamp: int) -> int:
    return SumaASCII % timestamp
```

Finalmente, recuerde que en Python la función **ord** le permite conocer el código ASCII de un carácter. Por ejemplo, el resultado de invocar **`ord('a')`** es el valor 97 porque ese es el número de ese carácter dentro de la tabla de caracteres ASCII.

Descripción de la aplicación

En este proyecto se va a implementar una estructura para analizar las transacciones realizadas en el sistema de Cupicoin. Su programa debe ser capaz de recibir las transacciones a partir de un archivo en formato CSV (Comma-Separated Values) que tiene 6 columnas:

- **codigo:** Código de la transacción.
- **block_number:** Número de identificación único de cada bloque dentro del *Blockchain.*
- **from_address:** Dirección de la cuenta que envía la transacción.
- **to_address:** Dirección de la cuenta que recibe la transacción.
- **value:** Valor transferido en Cupicoins.
- **block_timestamp:** Sello temporal que indica la fecha de cierre del bloque en la blockchain.

Por ejemplo, el siguiente archivo sería válido para ser cargado dentro del sistema:

```
codigo;block_number;from_address;to_address;value;block_timestamp
0x78448c;450400;0x74c4…;0x32be343b…;21000;1634530280
0x4ce389;450410;0x63a9…;0x54cec426…;90.43;1634531382
```

Nota: Las transacciones están ordenadas de acuerdo con el número de bloque. Es decir que primero aparecerán todas las transacciones del bloque 0, luego todas las del bloque 1 y así sucesivamente. Además, las transacciones de un mismo bloque poseen el mismo timestamp, que corresponde al tiempo de cierre del bloque en la cadena.

Nota 2: Las transacciones que tengan la dirección de la cuenta de destino como vacía (“”) son contratos.

La estructura para almacenar la información de todas las transacciones realizadas con Cupicoins debe estar basada en una lista de bloques, donde cada bloque estará representado por un diccionario con las siguientes llaves:

- **numero_bloque:** indica la posición del bloque dentro del blockchain, empezando con 0 para el primer bloque de la cadena.
- **cantidad_transacciones:** indica cuántas transacciones hacen parte del bloque.
- **timestamp:** es un número que indica el momento en que se ingresó la última transacción del bloque y se calculó el hash del bloque.
- **abierto:** es un valor booleano que indica si el bloque está abierto o está cerrado. Un blockchain debe tener siempre un solo bloque abierto al final de la cadena y debe ser el único bloque al cual se le pueden agregar transacciones. Un bloque abierto no tiene ni un timestamp ni un hash. Por el contrario, en un bloque cerrado no es posible agregar transacciones porque ya se calculó su hash y se fijó el timestamp usando la fecha de la última transacción.
- **hash:** es un valor numérico que es el resultado de aplicar la función de hash al contenido del bloque. Para bloques abiertos, el hash será None.
- **hash_anterior:** es el código hash del bloque anterior en la cadena. Para el primer bloque de la cadena, el hash_anterior será None.

Adicionalmente, el diccionario que representa un bloque tendrá llaves numéricas entre 0 y la cantidad_transacciones. Cada una de esas llaves tendrá asociado un diccionario que almacena la información de una transacción, utilizando las siguientes llaves:

- **codigo_transaccion:** es una cadena con un identificador único para cada transacción.
- **remitente:** es una cadena con la dirección de la cuenta de origen para la transacción.
- **destinatario:** es una cadena con la dirección de la cuenta destino para la transacción. Si la transacción no es una transferencia sino un contrato, este valor será una cadena vacía (“”).
- **valor:** es el valor en Cupicoins de la transacción.
- **operacion:** es una cadena que indica el tipo de transacción realizada. Los valores posibles son “transferencia” o “contrato”.

A continuación, se muestra la estructura para manejar los datos de Cupicoin:

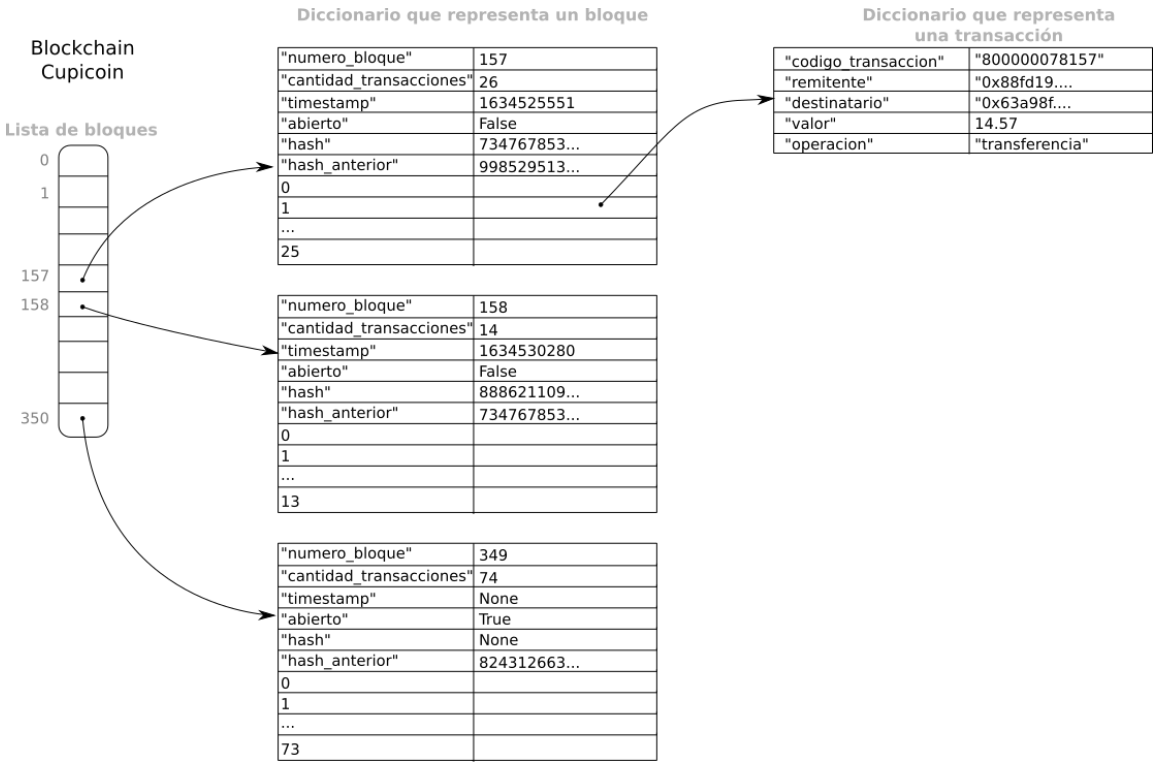

Figura 2. Estructura deseada para el manejo del blockchain de Cupicoin

La aplicación debe permitir al usuario ejecutar las siguientes acciones:

1. Cargar un archivo con la información de las transacciones de Cupicoin.
2. Añadir una nueva transacción al blockchain de Cupicoin.
3. Añadir un nuevo bloque al final del blockchain de Cupicoin.
4. Consultar el número de transacciones en los que una cuenta ingresada por parámetro ha sido registrada como remitente o como destinataria en una transacción.
5. Consultar la información de una transacción dado su código.
6. Encontrar las transacciones que tengan un remitente y un destinatario específicos.
7. Consultar la transacción con el valor máximo.
8. Calcular el saldo de una cuenta.
9. Validar la integridad del blockchain.

Actividad 0: Preparación del ambiente de trabajo

1. Cree una carpeta para trabajar, poniéndole su nombre o login.
2. Descargue de Bloque Neón el archivo con el "esqueleto" del proyecto (n3-cupicoin-esqueleto.zip) y descomprímalo en su carpeta de trabajo. El esqueleto consiste en un conjunto de archivos que usted va a usar o a modificar.
3. Abra Spyder y cambie la carpeta de trabajo para que sea la carpeta con el esqueleto.

Actividad 1: Construir el módulo de funciones

Usando Spyder, cree en su carpeta de trabajo un nuevo archivo con el nombre "cupicoin.py". En este archivo usted va a construir el módulo en el que va a implementar las funciones que responden a los requerimientos de la aplicación. **Defina**, **documente** e **implemente** las funciones descritas a continuación en su nuevo archivo. **Lea cuidadosamente las descripciones de las funciones para determinar los parámetros de entrada junto con sus tipos y el valor de retorno con su respectivo tipo.**

Función 1:
Implemente una función que reciba como parámetro el nombre de un archivo que contiene la información de las transacciones de Cupicoin y la cargue en el programa usando la estructura descrita anteriormente.
La función debe retornar una lista de diccionarios siguiendo la estructura que se muestra en la Figura 2.
Nota: Recuerde que las transacciones del archivo están ordenadas por número de bloque y por timestamp.
Nota 2: Las funciones 2 y 3 que se describen a continuación le podrían ser de utilidad para que la función 1 sea un poco más sencilla.

Función 2:
Implemente una función que reciba por parámetro la lista completa de bloques y un diccionario con la información de una transacción. La función debe añadir la transacción al último bloque de la cadena.

Función 3:
Implemente una función que reciba por parámetro la lista completa de bloques y un timestamp con la hora en que se está cerrando el bloque, y se encargue de cerrar el último bloque de la cadena y agregar un nuevo bloque abierto al final.
Nota: Tenga en cuenta que para cerrar un bloque debe calcular el código hash del bloque a partir de su contenido, y que el nuevo bloque debe tener el código hash del bloque que se acaba de cerrar. Se recomienda crear una función para el cálculo del hash de un bloque, que reciba por parámetro el diccionario que representa el bloque y retorne el hash calculado según las reglas de Cupicoin.
Nota 2: Para poder saber en qué momento se está cerrando un bloque puede utilizar la función **`time()`** que hace parte del módulo **`time`**, la cual retorna un número basado en la hora actual en su computador.

Función 4:
Implemente una función que reciba por parámetro la lista completa de bloques y una dirección de cuenta y retorne el número de transacciones en las que dicha cuenta participó como remitente o como destinatario de una transacción. La función debe retornar un diccionario que tenga las llaves “remitente” y “destinatario” y como valor, el número de transacciones en las que estuvo involucrada como remitente o destinatario respectivamente.

Función 5:
Implemente una función que reciba por parámetro la lista completa de bloques y el código de una transacción y retorne el diccionario con la información de la transacción. Si no existe ninguna transacción con el código ingresado por parámetro, la función debe retornar None.

Función 6:
Implemente una función que reciba por parámetro la lista completa de bloques, una dirección de un remitente y una dirección de un destinatario, y retorne una lista con los diccionarios de las transacciones en las que coincidan las direcciones de remitente y destinatario. En caso de no haber transacciones con ese remitente y destinatario se debe retornar una lista vacía.

Función 7:
Implemente una función que reciba por parámetro la lista completa de bloques y retorne un diccionario con la información completa de la transacción que tiene el valor máximo transferido en Cupicoins.

Función 8:
Implemente una función que reciba por parámetro la lista completa de bloques y la dirección de una cuenta y retorne el saldo final en Cupicoins de la cuenta. Tenga en cuenta que un contrato con un valor positivo significa que se hizo un depósito a la cuenta.

Función 9:
Implemente una función que reciba por parámetro la lista completa de bloques y valide la integridad del blockchain. Para validar la integridad del blockchain se deben verificar 3 cosas: Primero, solo puede existir un único bloque abierto que acepte transacciones. Este bloque abierto debe estar al final de la lista de bloques. Segundo, el código hash registrado en la llave "hash_anterior" de cada bloque, debe coincidir con el valor de la llave "hash" del bloque en la posición anterior en la lista de bloques. Es decir, cada bloque debe tener la referencia correcta al bloque que está inmediatamente antes. Tercero, el código hash de cada bloque debe ser coherente con las transacciones almacenadas. Para esto, debe recalcular el hash del bloque y compararlo con el que está almacenado en la llave "hash" del diccionario del bloque. La función debe retornar True si el blockchain no presenta ninguna irregularidad o False, en caso de que se detecte algún problema en las verificaciones mencionadas. **Nota:** Tenga en cuenta que el primer bloque de la lista no tiene ningún bloque anterior, por lo que el valor de la llave "hash_anterior" debe ser None.

Actividad 2: Completar la interfaz de usuario basada en consola

1. En esta actividad usted tiene que construir la interfaz basada en consola para que el usuario interactúe con la aplicación. Para construir esta interfaz usted debe completar el archivo consola_cupicoin.py, la cual ya tiene una parte implementada que le facilitará su trabajo. Usted debe modificar los elementos marcados con la etiqueta TODO.
2. Pruebe la interfaz por consola ejecutando el archivo "consola_cupicoin.py". Verifique que las funcionalidades de su aplicación se comporten de acuerdo con lo esperado.
3. Pruebe cargando el archivo cupicoin.csv o cree su propio archivo de prueba respetando el mismo formato.

Entrega

1. Comprima la carpeta llamada "esqueleto" con su proyecto resuelto. El archivo debe llamarse "**N3-PROY-login.zip**", donde login es su nombre de usuario de Uniandes.
2. Entregue el archivo comprimido a través de Bloque Neón en la actividad designada como "**Proyecto del Nivel 3**".