“Blockereum”

Sebastián González, David Pazán, Cristian Villavicencio

I. Introduction

A Continuación, en el presente documento, se ha de presentar el desarrollo de la red ”Blockereum”. En este documento se han de abordar diversos tópicos tales como: la motivación del proyecto, por qué se busca trabajar el problema y el enfoque escogido. Una descripción para la Arquitectura implementada, hablando de las capas que componen la red. Detalles más profundos, relacionados a la implementación. Para finalizar, se han de presentar pruebas del funcionamiento de la presente red.

II. Motivación Del Proyecto

La motivación principal detrás del desarrollo de este trabajo radica en poner a prueba los conocimientos adquiridos a lo largo del semestre, especialmente en el área de *Blockchain*, en conjunción con los aprendizajes previos obtenidos en la asignatura de estructuras de datos.

Con estos antecedentes en mente, el objetivo primordial es crear una red de *Blockchain* que aborde el problema de la centralización en las transacciones monetarias, además, como un componente crucial de lo abordado en el curso, se busca resolver el problema del doble gasto el cual puede surgir durante las transacciones entre los usuarios del servicio. Con esto en consideración, la principal orientación de la red Blockereum se enfocará en aspectos monetarios.

III. Descripción de la Arquitectura

En primer lugar hay que aclarar el funcionamiento de la Arquitectura esta efectuada mediante el modelo de capas, de entre las cuales se presentan:

- Capa de Datos: Correspondiente al manejo y almacenamiento de los registros de transacciones e información de los bloques. Se realiza mediante la programación POO y base de datos *LevevlDB*
- Capa de Infraestructura: La generación y comunicación de los nodos, permitiendo entregar un sistema descentralizado. Se configura mediante la librería *Libp2p*
- Capa de Aplicación: La aplicación de REST API que han de permitir a los usuarios realizar transacciones, consultar transacciones y consultar por bloques de la cadena.
- Capa de integración: Se presenta la lógica que utiliza el sistema para asegurar que todas las transacciones sean procesadas de manera segura y transparente.

IV. Detalles implementación

Hablando respecto a los componentes claves, para la implementación del sistema se tiene en consideración:

- Nodos: Tiene una copia completa de la cadena de bloques, es decir, almacena todos los bloques desde el bloque génesis hasta el bloque más reciente. Participa en la validación y propagación de transacciones y bloques en la red. Permite verificar y validar todas las transacciones de acuerdo con las reglas del protocolo.
- Bloques: Los bloques están caracterizados por poseer un Hash propio, una ID, una llave privada.
- Transacciones: Las transacción típicamente incluye información sobre el remitente, el destinatario y la cantidad de valor que se transfiere. Cada transacción está firmada digitalmente con la clave privada del remitente para garantizar la autenticidad.

Para hablar del funcionamiento hay que tratar dos puntos importantes. El primero es el cómo trabajan los bloques, principalmente en manera para efectuar las transacciones. Este proceso se da mediante la verificación del Hash Previo (Bloque previo establecido como nuestro parámetro) del que están consultando, con la finalidad de ir comprobando si el bloque anterior al que hace referencia el bloque existe y es válido (Figura [1]), una vez que se confirma la transacción, este nuevo bloque se ha de unir a la cadena del nodo, como se puede ver una representación en la (Figura [2])

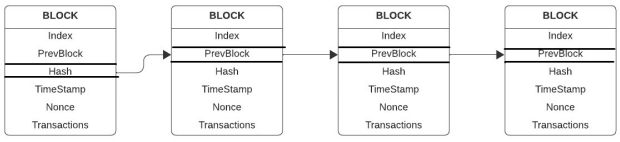

Fig. 1. Representación del funcionamiento del sistema, interacción de bloques de información

El proceso de verificación del Hash Previo en cada bloque juega un papel crucial en la seguridad y la integridad de la cadena de bloques. Al verificar el Hash Previo, cada bloque confirma la existencia y validez del bloque anterior en la cadena. Este proceso crea una conexión segura y secuencial entre los bloques, lo que garantiza que cualquier cambio en un bloque afectaría su hash, invalidando así no solo ese bloque sino también todos los bloques subsiguientes. Esta característica proporciona una inmutabilidad esencial para la integridad global de la cadena de bloques.

La unión de bloques para formar la cadena y la interacción de nodos son fundamentales para la descentralización y redundancia del sistema. Cada nodo mantiene su propia copia de la cadena, y la cohesión entre nodos se logra a través de un proceso descentralizado y consensuado, evitando así la dependencia de un solo punto de control, como se puede apreciar en (Figura [2])

- Validación
- Tiempo de vida del bloque

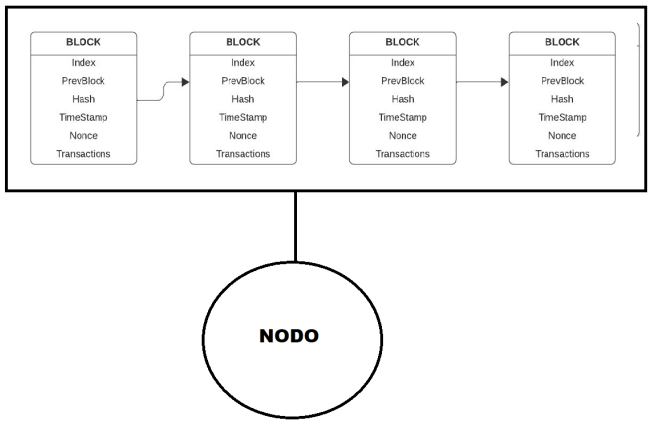

Fig. 2. Representación de la creación y almacenaje del Nodo.

Respecto a la red y la seguridad de esta, tenemos la implementación de los nodos bajo el protocolo Kademlia DHT, el cual entrega la capacidad de realizar búsqueda eficiente y la recuperación de los datos para los sistemas y redes distribuidas basados en P2P. Sobre nuestra red, este protocolo tiene la finalidad de gestionar información clave, como direcciones de nodos, información de transacciones y otros datos necesarios para el funcionamiento de la red.

Teniendo en cuenta el apartado de seguridad, los nodos que se inicializan en el sistema han de ir asignándose a los puertos disponibles de estas, y junto a esto se agregan a una variable de entorno que funciona como una lista, para tener constancia de la existencia. Cabe resaltar que al momento en que un nodo se separa de la red, o es eliminado, este ha de ser borrado de dicha lista, además de perder los datos correspondientes que existen dentro de este.

A. Descripción Modelo de Datos Utilizados

- Bloque:
  - Index: Representa la posicion del bloque dentro de la cadena de bloques.
  - Prev. block: Almacena el hash del bloque anterior en la cadena de bloques.
  - Hash: Almacena el hash del bloque actual.
  - Time Stamp: Representa la marca de tiempo en la que se creó el bloque.
  - Nonce: Es un número entero, se utiliza en algoritmos de consenso.
- Usuario:
  - Private Key: Llave para acceder y controlar los fondos asociados con la dirección de la cuenta en la blockchain.
  - Public Key: Se utiliza como la dirección de la billetera, la cual se deriva de la privada.
  - Address: Representación más amigable de la llave pública.
  - Balance: Saldo del usuario.
- Transacción:
  - Index: Número que identifica la posición de la transacción en el bloque.
  - Sender: Dirección de la cuenta que inicia la transacción.
  - Recipient: Representa la dirección del destinatario de la transacción.
  - Ammount: Cantidad de activos que se están transfiriendo.
  - Signature: Este campo almacena la firma digital asociada a la transacción.
  - TimeStamp: Momento de la creación de esta.
- Escritura
- Lectura

B. Protocolo de Comunicación

La implementación de LibP2P contribuye a entregar la comunicación de la red, junto a esto, se implementa el protocolo DHT (Tabla de Hash Distribuida) los cuales permiten la creación de las identidades descentralizadas (nodos) donde cada uno de los usuarios puede tener el control total sobre sus claves privadas y públicas. Teniendo como resultado que cada usuario puede registrarse en la red blockchain y utilizar la DHT para almacenar y recuperar su información de identidad de manera eficiente.

C. Solución del problema doble gasto

La red asegura que cada transacción sea única y se procese una sola vez. Esto se logra mediante el bloqueo de cuentas, con un mecanismo de “pausa”, el cual permite solo se pueda escribir en un único punto de entrada, teniendo como resultado que si llegase una segunda transacción ha de tener que esperar que este mecanismo pase a un estado de reanudarse, permitiendo que de esta forma la segunda transacción

D. Bloque de origen

El bloque de origen, o génesis, se incluye en el sistema para dar el comienzo a la cadena. Este pose las características previas comentadas, y a la vez posee el saldo que ha de operar en toda la red de blockchain (500 mil pesos). Este bloque no ha de permitir más escritura, estando dese un principio relleno en su totalidad. Y cada bloque se ha definir con un máximo de 5 transacciones para que pase a escribirse en un nuevo bloque.

E. Nodo

Cada nodo es una instancia independiente, con su propia base de datos, que ha de servir para guardar la información de las cuentas creadas, y las transacciones que se efectúan. Además, estas se rigen bajo una base de datos maestra, que ha de guardar toda la información, que posee cada nodo independiente, para el momento que se vuelve a iniciar la red.

V. Pruebas de funcionamiento

A continuación se presentan las distintas pruebas solicitadas en la rubrica, para esto se parte con la muestra de los datos que se tienen previos a crear una cuenta:

Fig. 3. Datos antes de crear una cuenta.

Con esto en mente, se pasa a la creación de la nueva cuenta:

A. Creación de una nueva cuenta

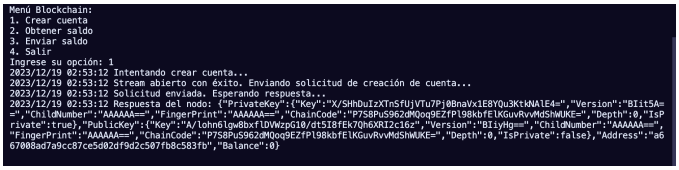

Fig. 4. Creación de cuenta creada, terminal cliente.

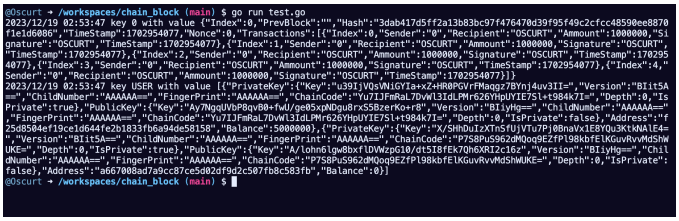

Fig. 5. Base de datos de cuenta creada.

Ahora, ¿Qué pasa si se quiere enviar saldo?, para esto se refleja la siguiente prueba:

B. Envío de saldo de una cuenta a otra

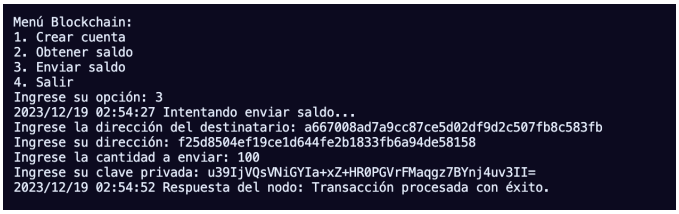

Fig. 6. Envió de saldo desde cliente, interfaz.

Se puede observar los logs que indican que el nodo a recibido la transacción

Fig. 7. Logs del nodo al recibir transacción.

Y por último, se reflejan los cambios en la base de datos con esta transacción

Fig. 8. Base de datos con cambios reflejados.

C. Envío de saldo a una cuenta, sin tener fondos suficientes

¿Qué pasa si se envia saldo a una cuenta, pero no tenemos fondos? Se puede apreciar el mensaje de saldo insuficiente:

Fig. 9. Envio de saldo sin fondos.

Posterior, se aprecian los Logs que envia el Nodo.

Fig. 10. Logs entregado por el nodo.

Y los de la base de datos, demostrando que no ha efectuado esta transacción.

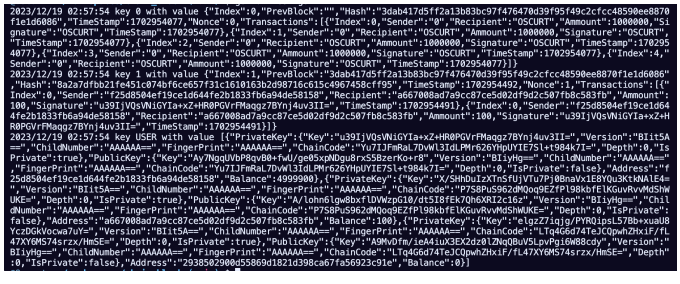

Fig. 11. Vista de la base de datos no afectada.

D. Enviar saldo a una cuenta que no existe

Surge la duda, sobre que pasa si se quiere enviar saldo a una cuenta que no existe, para esto se tiene lo siguiente:

Fig. 12. Envío de saldo a una cuenta no existente, teniendo una alerta sobre que no se realizo la transferencia.

El nodo ha de informar de dicho procedimiento:

Fig. 13. Respuesta por parte del nodo para avisar que no se realizo la transacción y pasa dicho aviso al cliente.

Y por último, la base de datos no presentar un cambio, ya que no ocurre la operación:

Fig. 14. Base de datos no actualizó el saldo de la persona que ha enviado saldo a una cuenta inexistente.

E. Consultar saldos por cuenta

Se puede consultar el saldo de una cuenta, al hacer uso de la opción indicada: Con esto presente, se aprecian los logs que

Fig. 15. Aplicación de la opción consultar saldo.

indican el procesamiento de dicha opción por parte del nodo.

Fig. 16. Logs que entrega el nodo tras la operación.

Y por último, la vista de la información que se encuentra en la base de datos.

Fig. 17. Información que posee la base de datos al momento de efectuar la consulta.

F. Revisar transacciones específicas