INSTITUTO TECNOLÓGICO Y DE ESTUDIOS SUPERIORES DE MONTERREY

CAMPUS ESTADO DE MÉXICO

Desarrollo de proyectos de ingeniería matemática (MA3001B)

Proyecto integrador Blockchain

Realizado por:

Citlali Daniela Martínez Maya	A01747804
Ana Vivar Rojas	A01798128
Eduardo Vargas Soria	A01754700

Grupo: 501

Profesor: Eliseo Sarmiento Rosales

9 de junio de 2025

Documentación de la aplicación

La aplicación del simulador de blockchain esta compuesta por dos partes principales, el archivo *Network_simulation.py* y por los archivos contenidos en el directorio *src*: *Usuario.py, UTXO.py, Transaccion.py, Bloque.py* y *Sistema.py.*

A continuación se describirá la estructura de cada clase y el funcionamiento que realiza.

1. Usuario.py

Clase que representa a un usuario en la red.

Parametros:

- *idx:* Índice del usuario, utilizado para identificarlo de manera única.
- *sign_key:* Llave privada del usuario, utilizada para firmar transacciones.
- *key:* Llave pública del usuario, utilizada para verificar firmas.
- *llave_privada:* Representación hexadecimal de la llave privada.
- *llave_publica:* Representación hexadecimal de la llave pública.
- *direccion:* Dirección del usuario, generada a partir de la llave pública.

Métodos:

- *crear_llaves:* Genera las llaves privada y pública del usuario.
- *generar_dir:* Genera la dirección del usuario a partir de su llave pública.
- *firmar:* Firma un mensaje con la llave privada del usuario.
- *checar_cartera:* Checa la cantidad de monedas en el conjunto de UTXOs del usuario.
- *verificar_firma:* Verifica la firma de un mensaje con la llave pública del usuario.

2. UTXO.py

Clase que representa un UTXO (Unspent Transaction Output) en la red.

Parámetros:

- *propietario:* Dirección del propietario del UTXO.
- *cantidad:* Cantidad de monedas asociadas al UTXO.
- *txid:* Identificador de la transacción a la que pertenece el UTXO.

Métodos:

- *generar_dict:* Genera un diccionario con los atributos del UTXO.

3. Transaccion.py

Clase que representa una transacción en la red de blockchain.

Parámetros:

- *idx:* Índice de la transacción, utilizado para identificarla de manera única.

- *sistema:* Sistema de blockchain al que pertenece la transacción, contiene el conjunto de UTXOs.
- *emisor:* Usuario que envía la transacción, None si es una transacción coinbase.
- *receptor:* Usuario que recibe la transacción, se le asignará un nuevo UTXO.
- *dir_emisor:* Dirección del emisor, se obtiene del usuario.
- *dir_receptor:* Dirección del receptor, se obtiene del usuario.
- *mining_fee:* Tarifa de minería, se suma al total de la transacción.
- *cantidad:* Cantidad de monedas.
- *UTXOs_set:* Conjunto de UTXOs del sistema, utilizado para verificar y actualizar los UTXOs.

Métodos:

- *lista_UTXO_emisor:* Crea una lista de UTXOs del emisor.
- *verificar_tx:* Verifica si la transacción es válida.
- *seleccionar_utxos:* Selecciona los UTXOs necesarios para cubrir la cantidad de la transacción.
- *crear_txid:* Crea un identificador único para la transacción.
- *firmar_tx:* Firma la transacción con la llave privada del emisor.
- *verificar_firma:* Verifica la firma de la transacción con la llave pública del emisor.
- *validar_y_preparar_tx:* Valida la transacción y la prepara para enviar a la mempool (sin tocar UTXOs_set).
- *aplicar_tx:* Aplica los cambios en el UTXOs_set (se llama solo cuando la tx se mina).
- *hacer_tx:* Realiza la transacción, actualizando el sistema y los UTXOs (solo se utiliza para crear el bloque genesis).
- *crear_dict:* Crea un diccionario con los atributos de la transacción.

4. Bloque.py

Clase que representa un bloque en la cadena de bloques.

Parámetros:

- *idx:* Índice del bloque, utilizado para identificarlo de manera única.
- *transacciones:* Lista de transacciones incluidas en el bloque.
- *previous_hash:* Hash del bloque anterior, utilizado para enlazar los bloques.

Métodos:

- *crear_dict:* Crea un diccionario con los atributos del bloque.
- *calcular_hash:* Calcula el hash del bloque utilizando sus atributos.

5. Sistema.py

Clase que representa el sistema de blockchain. Contiene la lógica para manejar a los usuario, transacciones, bloques, el minado de bloques y la creación del bloque génesis (donde se crea el primer usuario y se le asigna la primera transacción).

Parámetros:

- *dificultad:* Dificultad de minería.
- *mining_reward:* Recompensa por minar un bloque.
- *mining_fee:* Tarifa de minería por transacción.
- *usuarios:* Lista de usuarios registrados en el sistema.
- *blockchain:* Lista que representa la cadena de bloques.
- *UTXOs_set:* Conjunto de UTXOs disponibles en el sistema.
- *transacciones:* Lista de transacciones realizadas en el sistema.
- *mempool:* Lista de transacciones pendientes de ser minadas.
- *recompensas:* Lista de recompensas obtenidas por minar bloques.
- *fees:* Lista de tarifas de minería acumuladas.
- *idx_usuario:* Índice para identificar usuarios de manera única.
- *idx_tx:* Índice para identificar transacciones de manera única.
- *idx_bloque:* Índice para identificar bloques de manera única.
- *idx_utxo:* Índice para identificar UTXOs de manera única.

Métodos:

- *agregar_usuario:* Agrega un nuevo usuario al sistema.
- *crear_usuario:* Crea un nuevo usuario y lo agrega al sistema.
- *agregar_bloque:* Agrega un bloque a la cadena de bloques.
- *get_utxo_idx:* Genera un identificador único para un UTXO.
- *agregar_tx:* Agrega una transacción al sistema.
- *procesar_tx:* Procesa una transacción entre un emisor y un receptor.
- *get_cartera:* Devuelve un diccionario con las direcciones de los usuarios y sus saldos.
- *crear_coinbase_tx:* Crea una transacción de coinbase para el minero.
- *get_mining_fees:* Calcula las tarifas de minería acumuladas en la mempool.
- *minar_bloque:* Minera un bloque y lo agrega a la cadena de bloques.
- *crear_bloque_genesis:* Crea el bloque génesis y el usuario génesis.
- *crear_json:* Crea un diccionario con los atributos del sistema.

Network_simulation.py

En este archivo se crea la aplicación de streamlit que cuenta con 7 páginas: *Inicio, Resumen, Usuarios, Transacciones, Minería, Blockchain* y *Balances.*

- Inicio

  En esta página se crea el sistema de blockchain con dificultad de minería personalizada (4 ceros por default), inicializando un nuevo sistema mediante la creación de un objeto *sistema* de la clase Sistema al presionar el botón de "*Crear sistema*". También se puede descargar el sistema actual en un archivo *.pkl*. Finalmente, la página te permite cargar un sistema subiendo un archivo *.pkl.*

- Resumen

  Permite la visualización del estado actual del sistema, en esta página se muestran los atributos del sistema:

  - ➔ *Número de usuarios:* Es la longitud de la lista de usuarios.
  - ➔ *Número de bloques:* Es la longitud de la lista de bloques (blockchain).
  - ➔ *Número de transacciones:* Se obtiene de la longitud de la lista de transacciones.
  - ➔ *Recompensas acumuladas:* Es la suma de los valores de la lista de recompensas.
  - ➔ *Transacciones pendientes:* Es la longitud de la lista mempool (transacciones no procesadas).
  - ➔ *Dificultad de minería:* El número de ceros que el hash del bloque debe tener al inicio.

- Usuarios

  Página donde se crean nuevos usuarios y se muestran los usuarios actuales en una tabla donde se puede observar el número de usuario, su dirección y el saldo que tienen en su cuenta.

- Transacción

  Sección que te permite enviar transacciones de un usuario a otro. Al seleccionar al usuario remitente, se muestra su saldo de lado derecho. Después, te deja seleccionar al destinatario y la cantidad a enviar. Finalmente, se tiene que presionar el botón de "*Enviar transacción*" para que la transacción sea enviada al mempool (transacciones pendientes) que es parte del bloque.

- Minería

  Página donde se realiza la minería del bloque actual. En ella se muestra las transacciones pendientes en el bloque y la recompensa que obtiene el minero al minar el bloque. Después se pide seleccionar a un usuario que va a actuar como el minero

del bloque. Finalmente, se presiona el botón de "*Minar bloque*" para realizar el minado.

Una vez minado el bloque, se muestra un mensaje con el número del bloque minado, su hash, el tiempo de minado y la recompensa del minero.

- Blockchain

  Sección que permite visualizar los bloques minados, ya sea en formato de texto o como grafo.

  En el formato de texto, para cada bloque en el blockchain (lista de bloques) del sistema, se muestra:

  - ➔ *Número de bloque:* El índice del bloque.
  - ➔ *Hash:* Hash del contenido del bloque.
  - ➔ *Hash anterior:* Hash del bloque anterior.
  - ➔ *Nonce:* Número con el que al realizar del contenido del bloque se obtiene el número de ceros al inicio del hash especificados en la dificultad del sistema.
  - ➔ *Timestamp:* Fecha y hora en la que es minado el bloque.
  - ➔ *Recompensa:* Cantidad que recibe el minero al realizar el minado de ese bloque.
  - ➔ *Tiempo de minado:* Tiempo que tomo minar el bloque.
  - ➔ *Transacciones:* Lista con los diccionarios (con los identificadores) de las transacciones en el bloque minado.

  En el grafo se muestra únicamente el número de bloque, nonce, hash y el número de transacciones que contiene el bloque.

- Balance

  Muestra el saldo de todos los usuarios en la red en una tabla con el nombre del usuario, su dirección (hash de la llave pública) y su saldo.