Proyecto Integrador

Eliseo Sarmiento

Módulos del Proyecto

Módulo	Descripción
1. Wallets y Claves	Creación de usuarios con par de llaves (privada y pública). La dirección de usuario se define como el hash SHA-256 de su clave pública.
2. Transacciones y UTXO	Implementación de transacciones simples con entradas, salidas, firma digital y verificación. Uso del modelo UTXO y manejo de mining fee.
3. Bloques y Blockchain	Estructura de bloques con hash, cabecera, transacciones y referencia al bloque anterior. Construcción de una cadena de bloques inmutable.
4. Génesis y Minería	Creación del bloque génesis con 1000 monedas iniciales. Minería simplificada con recompensa de 3 monedas por bloque más comisiones por transacción.
5. Simulación con Streamlit	Desarrollo de una interfaz en Streamlit para crear usuarios, enviar transacciones, minar bloques y visualizar balances y la cadena de bloques.

Wallets y Claves

- Cada usuario debe tener una clave privada y una clave pública.
- La dirección del usuario será el **hash SHA-256 de la clave pública**.
- El algoritmo debe ser ECDSA sobre la curva secp256k1.

Puedes usar:

- `ecdsa` — para generación y manejo de claves (ECDSA).
- `hashlib` — para aplicar SHA-256 a la clave pública.

¿Qué se espera lograr?

- Se deben generar al menos **dos usuarios distintos**.
- Para cada usuario se debe mostrar:
  - Su **clave privada** (en formato hexadecimal).
  - Su **clave pública** (punto de la curva, en hexadecimal o como bytes).
  - Su **dirección**: SHA-256 de la clave pública.
- Estas claves deben guardarse para usarse luego al firmar transacciones.
- Las direcciones se usarán como identificadores únicos de usuario.

Transacciones y UTXO

- Define una estructura simple para transacciones: **entradas**, **salidas** y **firma digital**.
- Cada entrada debe referirse a un **UTXO existente**, asegurando que los fondos existen.
- Las salidas indican la cantidad enviada y la dirección del receptor.
- Implementa el modelo **UTXO**: los usuarios solo pueden gastar salidas no utilizadas.
- Las transacciones pueden incluir una **comisión opcional (mining fee)** que se suma a la recompensa del minero.

¿Qué puedes usar?

- `ecdsa` — para firmar las transacciones con la clave privada del remitente.
- `hashlib` — para obtener identificadores únicos de transacciones (hashes).
- `json` — para representar transacciones de forma estructurada.
- Diccionarios o listas — para mantener el estado del conjunto de UTXOs actuales.

Consejo: puedes representar cada UTXO como una entrada en un diccionario con llave = `id de transacción + índice de salida`.

¿Cómo se debe ver una transacción y el UTXO?

Ejemplo de Transacción:

```
inputs:    [ { txid:  "abc123...", index:  0, firma:  "..." } ]
outputs:   [ { cantidad:  5, direccion:  "ef67..." },
           { cantidad:  2, direccion:  "d91a..." } ]
fee:       1.0
txid:      "f3b9..." (hash de todo lo anterior)
```

Ejemplo de UTXO:

```
utxos = { "abc123...:0":  { direccion:  "a1b2...", cantidad:  8
} }
```

Después de la transacción: el UTXO anterior se elimina y se agregan dos nuevos con las salidas.

Bloques y Blockchain

- Cada bloque representa un conjunto de transacciones agrupadas cronológicamente.
- Cada bloque debe incluir:
  - **Hash del bloque anterior**, para mantener la integridad de la cadena.
  - **Lista de transacciones**, firmadas y válidas.
  - **Hash del propio bloque**, calculado a partir del contenido.
  - **Nonce**.
- La cadena debe ser **inmutable**: cualquier cambio en un bloque invalida los siguientes.

¿Qué puedes usar?

- `hashlib` — para generar el hash SHA-256 del contenido del bloque.
- `json` — para serializar el contenido de los bloques.
- `time` — para agregar marca de tiempo opcional a cada bloque.
- `list` o una clase personalizada — para almacenar y recorrer la cadena de bloques.

Consejo: verifica siempre que el hash del bloque anterior coincida con el campo correspondiente del bloque siguiente.

¿Cómo se debe ver un bloque?

Ejemplo de Bloque:

```
bloque: {
  index:          3,
  timestamp:      "2025-05-20 12:45:00",
  transacciones:  [ {...  }, {...  } ],
  prev_hash:      "0000a3f...",
  nonce:          1024,
  hash:           "000048c..."
}
```

Cadena de bloques:

```
blockchain = [bloque_genesis, bloque_1, bloque_2, bloque_3, ...]
```

Cada bloque se construye con base en la información del anterior.

Génesis y Minería

- Crear un bloque génesis con una transacción especial (coinbase) que entrega 1000 monedas.
- Por cada bloque nuevo, el minero recibe 3 monedas + comisiones por transacción.
- Implementar Prueba de Trabajo (PoW): encontrar un nonce tal que el hash del bloque comience con cierta cantidad de ceros.
- Solo el primer bloque tiene monedas "creadas de la nada" (premine).

¿Qué puedes usar?

- `hashlib` — para calcular el hash del bloque y verificar si cumple con PoW.
- `random` o `secrets` — para generar el nonce.
- `time` — para medir cuánto tarda en encontrarse un nonce válido.
- `datetime` — para registrar la hora en que se mina cada bloque.

Ejemplo: Bloque Génesis y PoW

Bloque Génesis:

- `prev_hash`: "0" (sin bloque anterior).
- `transacciones`: una coinbase entregando 1000 monedas a una dirección inicial.
- `nonce`: número aleatorio que se ajusta hasta que el hash cumpla la condición.
- `hash`: comienza con "0000..." si se exige PoW con 4 ceros iniciales.

Bloques siguientes:

- Incluyen lista de transacciones válidas.
- Recompensa fija de 3 monedas + suma de comisiones.
- Se deben encadenar correctamente al bloque anterior.

Simulación con Streamlit

- Usa **Streamlit** para construir una interfaz simple e interactiva.
- Debe incluir las siguientes funcionalidades:
  - Crear nuevos usuarios (wallets).
  - Enviar transacciones entre usuarios.
  - Ver saldos actuales.
  - Minar bloques (usando PoW).
  - Visualizar la cadena de bloques.

¿Qué puedes usar?

- `streamlit` — para crear la interfaz de usuario.
- `pandas` — para mostrar UTXO, historial de transacciones y saldos como tablas.
- `graphviz` — para mostrar la cadena de bloques como un grafo conectado.
- `matplotlib` — si deseas mostrar gráficas de recompensas o actividad.
- `pickle` o `json` — para guardar y cargar el estado del sistema.

Sugerencia de Menú en la Interfaz

Secciones sugeridas en la app:

- **Inicio:** Resumen general del sistema (bloques, transacciones, recompensas).
- **Usuarios:** Crear nueva wallet y mostrar dirección.
- **Transacciones:** Seleccionar remitente, destinatario, monto y comisión.
- **Minería:** Ejecutar minería, mostrar tiempo de minado y recompensa.
- **Blockchain:** Mostrar los bloques como tabla o grafo (hash, transacciones, nonce, etc.).
- **Balances:** Mostrar en una tabla todos los saldos actuales.

Recomendación: Usa `st.sidebar` para navegar entre estas secciones.