Universidad Anáhuac México Norte

Facultad de Ingeniería

Ingeniería en sistemas y tecnologías de la información

Proyecto final - Blockchain

CADENA DE SUMINISTROS

Profesor: José de Jesús Ángel Ángel

Integrantes:

Gerardo Melo Aguirre

Paloma Gutiérrez Ricuad

Mauricio Reyes Bocanegra

Fecha: 3 de diciembre de 2025

Índice

1. Introducción

Las cadenas de suministro actuales son cada vez más complejas. Un solo producto puede pasar por barcos, trenes, fábricas, centros de distribución y tiendas antes de llegar al consumidor final. Lo complicado es que, mientras más actores participan, más difícil se vuelve mantener un registro confiable de qué ocurrió en cada etapa. Esto abre la puerta a errores, pérdidas de información o incluso manipulación de datos cuando todos dependen de una base centralizada o de reportes manuales.

La idea de este proyecto surge justo de esa necesidad de tener un registro que sea claro, seguro y verificable por todos. La propuesta es construir un sistema basado en blockchain que registre cada evento importante dentro de una cadena de suministro: extracción de materias primas, manufactura, envíos, recepciones, ventas, destrucción y, además, las votaciones que determinan quién valida los bloques en la red. Todo queda almacenado en una estructura que no puede modificarse ni borrarse, lo que ayuda a que los participantes confíen en la información sin depender de un intermediario.

El objetivo principal del proyecto es diseñar y programar una blockchain privada y descentralizada donde cada empresa del proceso opera como un nodo independiente. Cada uno tiene su propia copia del ledger y su propio servidor local, y todos se conectan entre sí a través de una red peer-to-peer. A partir de ahí, cualquier transacción que se realice, ya sea un envío, una manufactura o un voto para elegir delegados, se firma digitalmente y se propaga a la red para que los nodos validadores la incluyan en un nuevo bloque.

A lo largo del documento se explica cómo se construyó el sistema, las decisiones técnicas que se tomaron y por qué el enfoque elegido logra resolver el problema de trazabilidad y confianza dentro de una cadena de suministro.

2. Descripción general del sistema

El proyecto consiste en una blockchain privada y completamente descentralizada que simula el funcionamiento de una cadena de suministro amplia. Cada empresa involucrada funciona como un nodo independiente dentro de la red. Todos los nodos se ejecutan de manera local,

pero cada uno opera su propio servidor, base de datos y lógica interna, lo que permite reproducir el comportamiento de una red distribuida real.

A diferencia de un sistema tradicional, aquí no existe una autoridad central que controle la información. Cada nodo mantiene su propia copia del ledger y participa en la propagación de transacciones y bloques a través de un protocolo peer-to-peer. Esto significa que la red sigue funcionando incluso si un nodo se cae o se desconecta, ya que el resto mantiene el historial completo y puede seguir validando nuevos eventos.

Aunque la red es descentralizada, no es pública. Solo quienes tengan una llave privada válida pueden emitir transacciones. La llave pública correspondiente queda registrada en la base de datos de cada nodo, de modo que todos puedan verificar la autenticidad de las operaciones sin revelar la identidad completa del participante. Este enfoque permite tener un sistema cerrado pero confiable, ideal para el tipo de coordinación que requieren las cadenas de suministro.

Cada nodo puede recibir, almacenar y validar transacciones relacionadas con el movimiento de bienes. Estas transacciones representan acciones como extraer materiales, manufacturar productos, enviarlos, recibirlos, venderlos o incluso destruirlos. Toda esta actividad se registra en el ledger usando bloques que contienen las transacciones aceptadas por la red, junto con un hash del bloque anterior y un Merkle root que permite verificar el contenido sin necesidad de procesar cada transacción una por una.

Además, el sistema incorpora gobernanza interna. Los nodos pueden votar por delegados y esos votos se almacenan como transacciones dentro de la misma cadena. Los tres participantes con más votos se convierten en los delegados elegibles para validar nuevos bloques mediante un proceso determinista. Esta parte del diseño permite que la red funcione sin depender completamente del consenso al azar, pero manteniendo descentralización y eficiencia.

En conjunto, el sistema representa una cadena de suministro digital en la que cada evento deja un rastro verificable, inmutable y accesible para todos los participantes con permisos. Toda la comunicación, validación y sincronización ocurre dentro de la red, sin servidores centrales y sin puntos únicos de falla. De esta manera, la blockchain asegura que la información sobre los bienes fluya con transparencia y que cualquier decisión dentro del proceso pueda auditarse fácilmente.

3. Elementos criptográficos

La seguridad del sistema depende completamente de la criptografía. Aunque cada nodo opera de manera independiente, todos deben estar de acuerdo en qué transacciones son válidas y en cuál es la versión correcta de la cadena. Para lograrlo, el proyecto utiliza tres piezas clave: funciones hash, árboles de Merkle y firmas digitales. Cada una cumple un papel distinto dentro de la arquitectura.

3.1 Hash (SHA-256)

En el proyecto se usa SHA-256 para transformar información en un identificador único y de longitud fija. El hash de un bloque depende de su contenido, del hash del bloque anterior y del Merkle root de sus transacciones. Esto crea un efecto de encadenamiento: si alguien intentara modificar un bloque pasado, cambiaría su hash y, por lo tanto, rompería todos los bloques siguientes. Esta estructura es lo que hace que la blockchain sea prácticamente inmutable.

Más allá de eso, también se calcula un hash por cada transacción antes de firmarla y transmitirla. Con esto se garantiza integridad: si alguien tratara de alterar el contenido, aunque fuera un solo carácter, el hash cambiaría por completo.

Se eligió SHA-256 debido a que es una función hash segura, rápida y ampliamente probada en sistemas reales como Bitcoin. Se considera resistente a colisiones, ya que lleva años siendo estudiada sin que existan ataques prácticos que comprometan su integridad, lo cual la hace confiable para un proyecto donde cada bloque depende del anterior. Además, es eficiente incluso cuando se procesa un gran número de transacciones y viene incluida en la biblioteca estándar de Python, así que su uso es sencillo y no requiere librerías externas.

3.2 Árbol de Merkle

Se incluye el árbol de Merkle, ya que este permite verificar la integridad de todas las transacciones de un bloque de manera rápida y eficiente. Si una sola transacción se altera, el cambio se refleja inmediatamente en la raíz, lo que facilita detectar manipulaciones sin revisar el bloque completo. También, es una estructura estándar en blockchains reales y ayuda a que el sistema escale mejor cuando hay muchas operaciones.

En este caso, cada bloque agrupa varias transacciones y a partir de ellas se genera un Merkle root. El proceso consiste en tomar el hash de cada transacción en la lista, agruparlos de dos en dos y volver a aplicar SHA-256 para crear los niveles superiores del árbol, hasta llegar a una sola raíz.

Esto tiene dos ventajas muy importantes para el sistema:

1. **Verificación eficiente:** Se puede comprobar que una transacción pertenece a un bloque sin necesidad de revisar cada una de las demás. El Merkle root funciona como un "resumen" criptográfico del contenido del bloque.
2. **Ahorro de espacio y velocidad:** Permite validar bloques completos de forma más ligera, especialmente cuando la red procesa muchas operaciones seguidas.

3.3 Firmas digitales (ECDSA)

Cada transacción que se envía a la red se firma usando ECDSA sobre la curva SECP256k1, la misma usada en Bitcoin. El nodo que envía una operación utiliza su llave privada para firmar el hash de la transacción, y los demás nodos verifican la firma usando la llave pública que tienen registrada en su base de datos.

La firma digital es lo que garantiza tres cosas esenciales:

1. **Autenticidad:** Solo el dueño de la llave privada pudo haber generado esa transacción.
2. **Integridad:** Si el contenido de la transacción se altera, la firma ya no sería válida.
3. **No repudio:** Ningún participante puede negar haber enviado algo que se firmó con su llave.

Las llaves privadas nunca se comparten. Cada nodo las almacena en su propia carpeta y solo expone la llave pública para verificación. Esto permite que la red sea privada, pero al mismo tiempo completamente verificable.

3.4 Manejo de llaves públicas y privadas

Al crear la red, el programa genera automáticamente un par de llaves para cada participante. La llave privada se guarda en un archivo que solo el nodo puede leer, mientras que la llave pública se inserta en la tabla de participantes dentro de la base de datos local.

Este diseño permite dos cosas: control de acceso, porque solo quienes tienen una llave privada válida pueden enviar transacciones, y verificación distribuida, ya que todos los nodos tienen acceso a las llaves públicas y no necesitan depender de un servidor externo para validar firmas.

Este sistema de llaves es la base de la seguridad del proyecto y la razón por la que se puede confiar en la autenticidad de todos los eventos registrados.

4. Modelo de datos y reglas de validación

El sistema funciona gracias a un conjunto de estructuras de datos que representan los elementos principales de la cadena: transacciones, bloques y el estado compartido de los bienes. Cada nodo mantiene estos datos en su propia base local y aplica una serie de reglas que actúan como un "smart contract", es decir, un conjunto de condiciones que determinan si una transacción es válida o no. Aunque no se usa una plataforma como Ethereum, la lógica cumple el mismo objetivo: garantizar que todo lo que se registra siga reglas claras y coherentes para todos.

4.1 Estructura de la transacción

Cada transacción describe un evento dentro de la cadena de suministro:

- El emisor y el receptor (llaves públicas)
- El tipo de acción (EXTRACTED, MANUFACTURED, SHIPPED, RECEIVED, SOLD, CONSUMED, DESTROYED o VOTE)
- El identificador del embarque
- La cantidad y el tipo de bien
- La ubicación
- Un timestamp
- La firma digital del emisor

Antes de ser enviada a la red, la transacción se firma con la llave privada del participante, y los demás nodos la verifican usando su llave pública. Con esto se evita que alguien pueda falsificar información o manipular transacciones una vez generadas.

4.2 Modelo del bloque

Las transacciones que pasan las validaciones se agrupan en un bloque. Cada bloque contiene su índice dentro de la cadena, la lista de transacciones válidas, el hash del bloque anterior, el nombre del nodo validador, la raíz del árbol de Merkle y su propio hash calculado con SHA-256

Esta estructura asegura que los bloques estén encadenados entre sí y que cualquier cambio en una transacción modifique el Merkle root y, por ende, el hash final del bloque. Esto es lo que garantiza la inmutabilidad del sistema.

4.3 Base de datos local en cada nodo

Cada nodo mantiene una base de datos SQLite con varias tablas:

blocks	Almacena los bloques aceptados por ese nodo
participants	Contiene las llaves públicas, votos y datos básicos de cada empresa
goods	Catálogo de bienes con su unidad de medida
shipments	Estado actual de cada embarque (dueño, ubicación, cantidad, acción más reciente)
mempool	Transacciones pendientes de validar

Aunque cada nodo tiene su propia copia, todas siguen el mismo estado lógico porque los bloques validados se propagan a toda la red.

4.4. Reglas del “smart contract” y actualización del estado

Las reglas de validación funcionan como un contrato inteligente que define cómo debe comportarse la cadena de suministro. Antes de aceptar una transacción, el nodo validador revisa tres cosas: que la firma digital sea válida, que la acción siga un orden lógico y que no genere inconsistencias en el estado actual del sistema.

Por ejemplo, un shipment solo puede crearse con acciones como *EXTRACTED* o *MANUFACTURED*, y solo el dueño actual puede enviarlo, recibirlo o venderlo. Tampoco es

posible operar sobre un embarque que ya fue destruido o consumido. Lo mismo ocurre con las transacciones de voto: solo se aceptan si el destinatario es un participante registrado.

Cuando un bloque se valida, cada transacción se aplica al estado local del nodo. Esto actualiza la propiedad de los bienes, las ubicaciones, las cantidades y el estatus de cada shipment. En el caso de los votos, también se ajustan los conteos y se refleja en la selección de delegados para el consenso. Como todos los nodos ejecutan exactamente las mismas reglas, el estado final es consistente en toda la red y cualquier participante puede reconstruir el historial de un embarque sin depender de un servidor central.

5. Consenso: Delegated Proof of Stake (DPoS)

Para mantener la red sincronizada y decidir qué nodo puede validar el siguiente bloque, el proyecto utiliza un mecanismo de consenso basado en Delegated Proof of Stake (DPoS). La idea es que los participantes voten por delegados y que solo los nodos con más votos puedan validar bloques. Esto hace el proceso más eficiente que otros métodos como Proof of Work, y al mismo tiempo mantiene el sistema descentralizado.

5.1 Votación y elección de delegados

Cada nodo tiene la capacidad de votar por cualquier otro participante enviando una transacción del tipo *VOTE*. Estos votos se almacenan en la cadena y se reflejan en la base de datos local de cada nodo. Los tres participantes con más votos se consideran los delegados principales.

Este método hace que la gobernanza sea transparente: en lugar de depender de un líder fijo o un nodo privilegiado, la comunidad decide quién tiene la responsabilidad de validar los bloques.

```
************************************ DELEGATE STANDINGS ************************************
Candidate Name                | Votes      | Status
-------------------------------------------------------------
 --(Active)-- FreightTrain_Express        | 6          | ACTIVE WITNESS
 --(Active)-- Drone_Delivery_X            | 3          | ACTIVE WITNESS
 --(Active)-- OPEC_Supplier               | 2          | ACTIVE WITNESS
  Truck_Fleet_Alpha           | 1          | Standby
  CleanWater_Services         | 1          | Standby
  Mega_Consumer_Goods         | 1          | Standby
  Corner_Store                | 1          | Standby
  GlobalMining_Corp           | 0          | Standby
  Pacific_Logistics           | 0          | Standby
  TechFoundry_Inc             | 0          | Standby
  CargoShip_EverGiven         | 0          | Standby
```

Figura 1. Tabla de votos y delegados activos en el mecanismo DPoS

La Figura 1 muestra la tabla generada por el sistema que muestra el estado actual de la votación DPoS. En ella se observa cuántos votos ha recibido cada nodo de la red y cuáles han sido seleccionados como delegados activos (“Active Witness”), así como aquellos que permanecen en modo de espera (“Standby”). Esto ilustra el proceso de gobernanza descentralizada del prototipo.

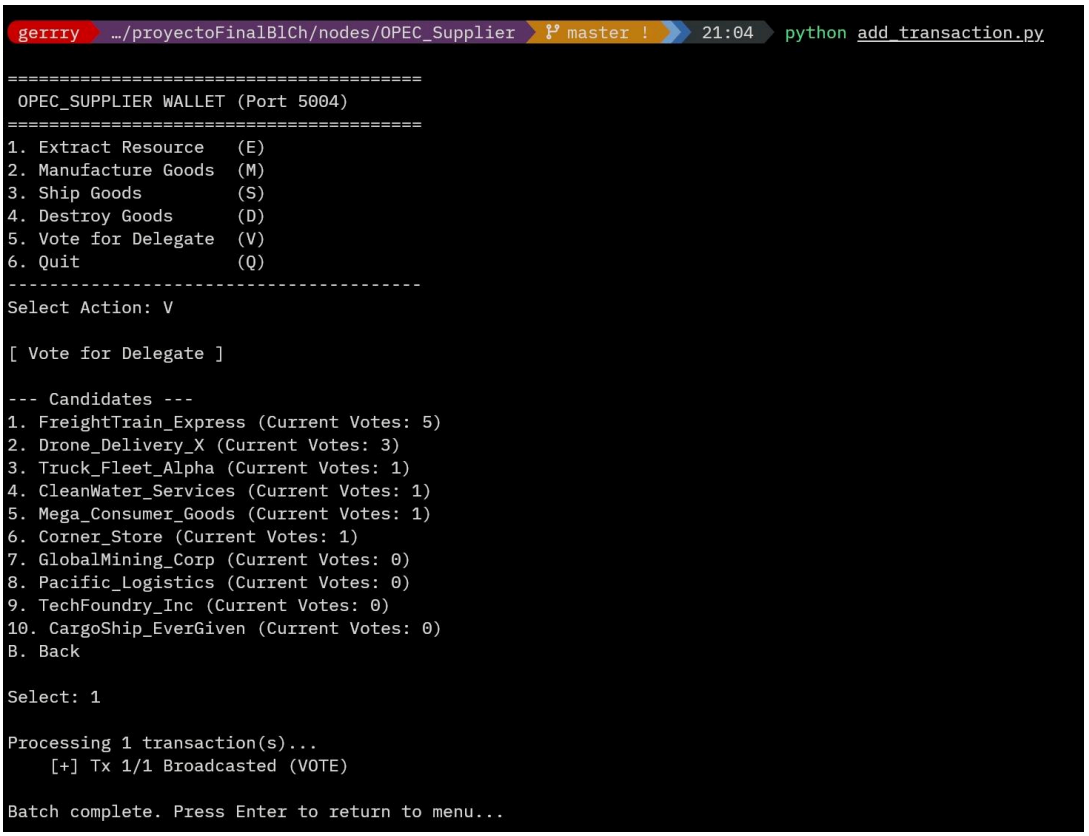


Figura 2. Interfaz de votación de delegados desde un nodo DPoS

La Figura 2 muestra la interfaz del nodo *OPEC_Supplier* donde se despliega el menú de acciones disponibles, incluyendo la opción para votar por un delegado. La imagen muestra la lista de candidatos, sus votos actuales y la confirmación de que la transacción de voto fue firmada, procesada y difundida a la red.

5.2 Selección determinista del validador

Aunque existan tres delegados, solo uno valida cada bloque. Para elegirlo se usa un proceso determinista que toma como entrada el hash del bloque anterior y un valor incremental. Esa

combinación se transforma a un entero y se aplica un módulo sobre el número de delegados. Con esto se obtiene, de forma predecible, cuál de ellos debe validar el siguiente bloque.

Este mecanismo evita la aleatoriedad completa, pero conserva la rotación entre los nodos más votados. Eso reduce la probabilidad de que un solo participante concentre demasiado poder.

5.3. Minado automático y ventajas en un entorno de supply chain

Cada nodo ejecuta un proceso en segundo plano que revisa periódicamente si le toca validar el siguiente bloque. Cuando un nodo es el validador en turno, toma las transacciones del mempool, aplica las reglas del "smart contract" y construye un bloque con las operaciones válidas. Después calcula el hash final y el Merkle root, agrega el bloque a su propia cadena y lo transmite al resto de la red.

Este proceso es ligero y eficiente, lo cual es especialmente útil en una cadena de suministro, donde los eventos ocurren con frecuencia y se necesita que la confirmación sea rápida. A diferencia de mecanismos más pesados como Proof of Work, aquí no se desperdician recursos ni energía, y los bloques pueden validarse prácticamente en tiempo real. También, al existir un conjunto definido de participantes, la rotación entre delegados ofrece un equilibrio adecuado entre rendimiento y descentralización, manteniendo la gobernanza visible y auditable para todos.

6. Comunicación P2P y distribución

La red funciona mediante un protocolo peer-to-peer donde cada nodo actúa como un servidor independiente. No hay un servidor central que coordine la comunicación ni un punto único de falla, si no que todos los nodos se conectan entre sí y comparten información directamente. Este enfoque garantiza que la red sea más resistente y que todos mantengan una copia actualizada del ledger.

6.1 Propagación de transacciones y bloques

La comunicación entre nodos sigue un modelo completamente peer-to-peer. Cuando un nodo genera una transacción, primero la firma y luego la envía a su propio servidor. Si es válida, se agrega al mempool y se propaga al resto de los nodos mediante un sistema tipo "gossip".

Cada nodo que la recibe repite el proceso: valida la firma, revisa que la acción tenga sentido según su estado actual y, si todo está en orden, la almacena y la vuelve a transmitir. Esto asegura que las transacciones nuevas lleguen a toda la red incluso si algunos nodos están momentáneamente desconectados.

Lo mismo ocurre con los bloques. Cuando un delegado valida un bloque, lo envía por HTTP a los demás nodos. Cada uno reconstruye el bloque recibido, comprueba que encaje con su cadena , especialmente el índice y el hash del bloque anterior, y, si es consistente, lo agrega a su propia base de datos y actualiza su estado. Este mecanismo de distribución mantiene a todos los nodos sincronizados y garantiza que el ledger avance de forma uniforme.

6.2 Sincronización entre nodos

La sincronización se activa cuando un nodo descubre que su copia de la cadena no coincide con la de sus pares. En ese caso, consulta a todos los nodos y elige la cadena válida más larga según su altura. Después solicita esa cadena completa y la reconstruye bloque por bloque, aplicando nuevamente todas las validaciones.

Este mecanismo evita que la red se fragmente y asegura que todos los nodos terminen con la misma versión del ledger, incluso si sufrieron desconexiones temporales.

Validación del despliegue de nodos

```
(venv) paloma@palomagtzr proyectoFinalBlCh-master % curl http://localhost:5001/info
{"height":0,"last_hash":"0000000000000000000000000000000000000000000000000000000000000000","node_name":"Truck_Fleet_Alpha"}
(venv) paloma@palomagtzr proyectoFinalBlCh-master % curl http://localhost:5002/info
{"height":0,"last_hash":"0000000000000000000000000000000000000000000000000000000000000000","node_name":"TechFoundry_Inc"}
(venv) paloma@palomagtzr proyectoFinalBlCh-master % curl http://localhost:5003/info
{"height":0,"last_hash":"0000000000000000000000000000000000000000000000000000000000000000","node_name":"Pacific_Logistics"}
(venv) paloma@palomagtzr proyectoFinalBlCh-master % curl http://localhost:5004/info
{"height":0,"last_hash":"0000000000000000000000000000000000000000000000000000000000000000","node_name":"OPEC_Supplier"}
(venv) paloma@palomagtzr proyectoFinalBlCh-master % curl http://localhost:5005/info
{"height":0,"last_hash":"0000000000000000000000000000000000000000000000000000000000000000","node_name":"Mega_Consumer_Goods"}
(venv) paloma@palomagtzr proyectoFinalBlCh-master % curl http://localhost:5006/info
{"height":0,"last_hash":"0000000000000000000000000000000000000000000000000000000000000000","node_name":"GlobalMining_Corp"}
(venv) paloma@palomagtzr proyectoFinalBlCh-master % curl http://localhost:5007/info
{"height":0,"last_hash":"0000000000000000000000000000000000000000000000000000000000000000","node_name":"FreightTrain_Express"}
(venv) paloma@palomagtzr proyectoFinalBlCh-master % curl http://localhost:5008/info
{"height":0,"last_hash":"0000000000000000000000000000000000000000000000000000000000000000","node_name":"Drone_Delivery_X"}
(venv) paloma@palomagtzr proyectoFinalBlCh-master % curl http://localhost:5009/info
{"height":0,"last_hash":"0000000000000000000000000000000000000000000000000000000000000000","node_name":"Corner_Store"}
(venv) paloma@palomagtzr proyectoFinalBlCh-master % curl http://localhost:5010/info
{"height":0,"last_hash":"0000000000000000000000000000000000000000000000000000000000000000","node_name":"CleanWater_Services"}
(venv) paloma@palomagtzr proyectoFinalBlCh-master % curl http://localhost:5011/info
{"height":0,"last_hash":"0000000000000000000000000000000000000000000000000000000000000000","node_name":"CargoShip_EverGiven"}
```

Figura 3. Respuesta de los nodos al endpoint /info

Se muestra, en la Figura 3, la verificación manual del estado de cada nodo mediante solicitudes HTTP al endpoint /info. Cada uno devuelve su nombre, el hash del bloque más reciente y la altura actual de la cadena. Esto confirma que todos los nodos quedaron

correctamente desplegados en sus puertos, se identifican por nombre y corren su propia instancia del servidor Flask.
Esta prueba garantiza que la red P2P funciona y que todos los participantes pueden intercambiar bloques y transacciones.

6.3 Manejo de conflictos y ausencia de autoridad central

La red no depende de un servidor central que decida qué información es válida. Cada nodo mantiene su propia copia del ledger, valida transacciones de manera independiente y participa en la propagación de bloques. Esto hace que el sistema sea más resistente y elimina la necesidad de confiar en un intermediario.

Cuando ocurre un conflicto, por ejemplo, si un nodo recibe un bloque cuyo índice no coincide con su cadena, el nodo no espera instrucciones de un "coordinador". En lugar de eso, inicia un proceso de sincronización en el que consulta a todos los demás nodos para identificar la cadena más larga y coherente. Una vez encontrada, solicita esa versión completa y la reconstruye bloque por bloque, aplicando todas las validaciones necesarias.

Este enfoque permite que la red se mantenga consistente incluso si algunos nodos se desincronizan temporalmente. Al no existir una autoridad central y al basarse en validaciones locales y sincronización distribuida, el sistema evita puntos únicos de falla y asegura que todos los participantes terminen con la misma versión del ledger.

7. Ejecución y demostración del prototipo

El prototipo está desarrollado completamente en Python y organizado en una estructura modular que separa la lógica de la blockchain, la comunicación entre nodos, la validación, la creación de transacciones y las herramientas de visualización. Esto permite que cada nodo funcione como un servidor independiente, con su propia base de datos y su propio estado local.

La demostración del sistema se basa en tres etapas: la creación automática de la red, la ejecución de transacciones y la revisión del estado final de la cadena.

7.1. Creación de la red y generación de nodos

Para iniciar el sistema, primero se ejecuta el script que construye todos los nodos desde cero. Proceso:

1. Crea las carpetas individuales para cada empresa de la cadena de suministro
2. Genera las llaves privadas y públicas de cada participante
3. Inicializa una base de datos SQLite en cada nodo
4. Registra a todos los participantes con su información básica y sus llaves públicas
5. Carga el catálogo de bienes
6. Inserta los shipments iniciales
7. Construye el bloque génesis con los datos necesarios

Durante este proceso se realiza una votación inicial entre los participantes para definir los primeros delegados del consenso. Esta votación queda registrada en las bases de datos locales y determina quiénes comenzarán a validar los bloques en la red.

Una vez creado todo, el script para levantar la red inicia cada nodo en un puerto distinto, lo que permite simular una red distribuida completa usando solo el entorno local.

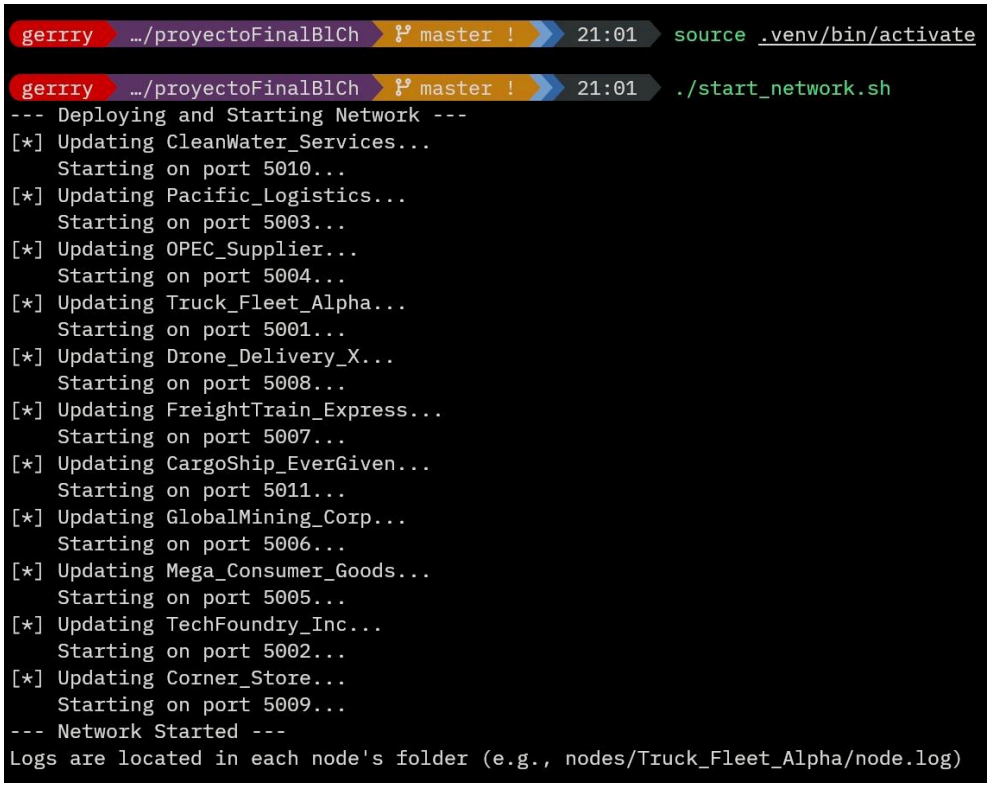


Figura 4. Inicialización de la red y arranque de los nodos del sistema

La Figura 4 muestra la ejecución del script que levanta todos los nodos de la red, iniciando cada servidor Flask en su puerto correspondiente y confirmando que la infraestructura distribuida quedó activa. Esta salida evidencia el arranque coordinado de todos los participantes del sistema y la creación de la red de comunicación P2P.

7.2 Ejecución de transacciones

Cada nodo puede enviar transacciones mediante un cliente interactivo que permite realizar acciones como extraer bienes, manufacturarlos, enviarlos, recibirlos, consumirlos, destruirlos o emitir votos. El cliente lee la llave privada del nodo, consulta su estado actual en la base de datos, construye la transacción según las acciones disponibles, la firma con la llave privada, y finalmente la envía al servidor local del nodo.

Si la transacción pasa las validaciones iniciales, se agrega al mempool y se propaga automáticamente a toda la red. Los delegados irán tomando estas transacciones y las incluirán en los bloques conforme se conviertan en validadores en turno.

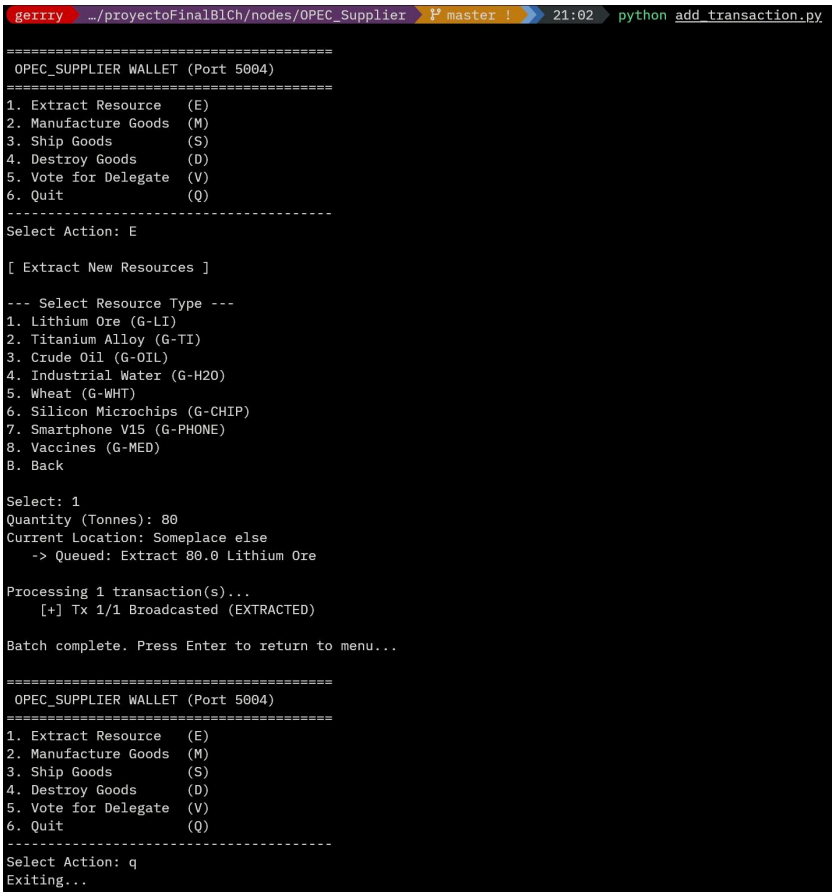


Figura 5. Ejecución de una transacción de extracción desde el nodo OPEC_Supplier

La Figura 5 corresponde al menú interactivo del nodo *OPEC_Supplier*, donde el usuario ejecuta una transacción de tipo EXTRACTED. Se muestra la selección del tipo de recurso, la cantidad, la ubicación y la confirmación de que la transacción fue procesada y difundida a toda la red. Refleja cómo los nodos realizan operaciones sobre los bienes utilizando sus llaves privadas.

7.3. Visualización del ledger y seguimiento de los bienes

Para revisar la información registrada, cada nodo puede ejecutar el script *view_blockchain.py* que muestra los bloques registrados en ese nodo, el hash de cada bloque y el del anterior, las transacciones incluidas y el estado actual de los shipments.

El estado consolidado se calcula aplicando todas las transacciones válidas en orden, por lo que cada nodo puede reconstruir el historial completo de cualquier embarque. Esto permite verificar quién lo creó, por dónde se movió, quién lo tuvo en cada etapa y cuál fue su acción final.

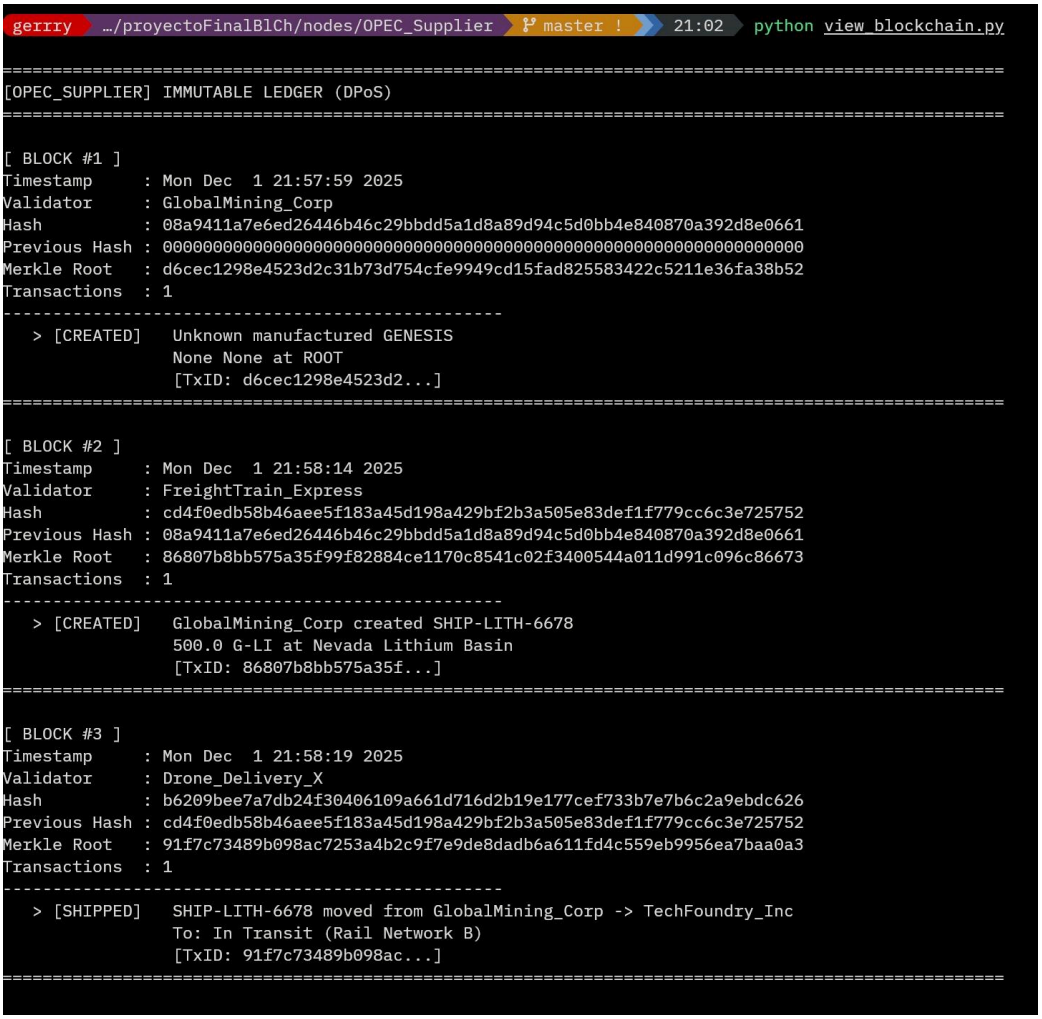


Figura 6. Primeros bloques del ledger: génesis, creación y envío de bienes

La Figura 6 muestra los primeros bloques del ledger, iniciando con el bloque génesis, seguido por la creación del shipment *SHIP-LITH-6678* por GlobalMining_Corp y posteriormente su envío hacia TechFoundry_Inc. Esta salida evidencia cómo el sistema construye la cadena desde su origen y registra paso a paso la creación y movimiento de bienes dentro de la red.

```
######################################## WORLD STATE (Active Inventory) ########################################
Shipment ID        | Good             | Quantity     | Owner                | Location        | Last Action
-------------------------------------------------------------------------------------------------------------
SHIP-96775         | Lithium Ore      | 80.0 Tonnes  | OPEC_Supplier        | Someplace else  | EXTRACTED
SHIP-82125         | Vaccines         | 100.0 Vials  | Drone_Delivery_X     | UIO             | MANUFACTURED
SHIP-35735         | Crude Oil        | 80.0 Barrels | Drone_Delivery_X     | UIO             | RECEIVED
SHIP-63509         | Wheat            | 70.0 Tonnes  | OPEC_Supplier        | McDonalds       | EXTRACTED
SHIP-47655         | Lithium Ore      | 12.0 Tonnes  | OPEC_Supplier        | There           | EXTRACTED
SHIP-1002          | Crude Oil        | 1000.0 Barrels | Pacific_Logistics    | In tranist      | SHIPPED
SHIP-BATT-6021     | Silicon Microch  | 2000.0 Units | Mega_Consumer_Goods  | Retail Loading  | SHIPPED
SHIP-BATT-4363     | Silicon Microch  | 2000.0 Units | Mega_Consumer_Goods  | Retail Loading  | SHIPPED
SHIP-1003          | Industrial Wate  | 50000.0 Liters | CleanWater_Services  | Reservoir A     | EXTRACTED
SHIP-1001          | Lithium Ore      | 500.0 Tonnes | GlobalMining_Corp    | Nevada Mine     | EXTRACTED
```

Figura 7. Estado global de bienes activos en la red (World State)

La figura 7 muestra el estado global del inventario en la red blockchain, mostrando todos los *shipments* activos. Incluye información detallada de cada recurso: ID del envío, tipo de bien, cantidad, propietario actual, ubicación y la última acción registrada sobre ese envío (EXTRACTED, SHIPPED, MANUFACTURED, etc.). Esta vista demuestra cómo el sistema mantiene una trazabilidad completa de cada activo en la cadena de suministro.

```
************************************* DELEGATE STANDINGS *************************************
Candidate Name                | Votes      | Status
--------------------------------------------------------------
 --(Active)-- FreightTrain_Express         | 5          | ACTIVE WITNESS
 --(Active)-- Drone_Delivery_X             | 3          | ACTIVE WITNESS
 --(Active)-- OPEC_Supplier                | 2          | ACTIVE WITNESS
  Truck_Fleet_Alpha            | 1          | Standby
  CleanWater_Services          | 1          | Standby
  Mega_Consumer_Goods          | 1          | Standby
  Corner_Store                 | 1          | Standby
  GlobalMining_Corp            | 0          | Standby
  Pacific_Logistics            | 0          | Standby
  TechFoundry_Inc              | 0          | Standby
  CargoShip_EverGiven          | 0          | Standby

######################################## WORLD STATE (Active Inventory) ########################################
Shipment ID        | Good             | Quantity     | Owner                | Location        | Last Action
-------------------------------------------------------------------------------------------------------------
SHIP-82125         | Vaccines         | 100.0 Vials  | Drone_Delivery_X     | UIO             | MANUFACTURED
SHIP-35735         | Crude Oil        | 80.0 Barrels | Drone_Delivery_X     | UIO             | RECEIVED
SHIP-63509         | Wheat            | 70.0 Tonnes  | OPEC_Supplier        | McDonalds       | EXTRACTED
SHIP-47655         | Lithium Ore      | 12.0 Tonnes  | OPEC_Supplier        | There           | EXTRACTED
SHIP-1002          | Crude Oil        | 1000.0 Barrels | Pacific_Logistics    | In tranist      | SHIPPED
SHIP-BATT-6021     | Silicon Microch  | 2000.0 Units | Mega_Consumer_Goods  | Retail Loading  | SHIPPED
SHIP-BATT-4363     | Silicon Microch  | 2000.0 Units | Mega_Consumer_Goods  | Retail Loading  | SHIPPED
SHIP-1003          | Industrial Wate  | 50000.0 Liters | CleanWater_Services  | Reservoir A     | EXTRACTED
SHIP-1001          | Lithium Ore      | 500.0 Tonnes | GlobalMining_Corp    | Nevada Mine     | EXTRACTED
```

Figura 8. Estado global del consenso DPoS y del inventario activo en la red

La Figura 8 muestra dos vistas críticas del sistema:

- **Delegate Standings:** Lista de candidatos del consenso DPoS, indicando cuántos votos tiene cada uno y quiénes son los “Active Witnesses” responsables de validar los bloques en ese momento.
- **World State (Active Inventory):** Estado global de todos los bienes rastreados por el sistema. Incluye su ID, tipo de mercancía, cantidad, dueño actual, ubicación y última acción registrada.

Esta salida demuestra cómo el blockchain mantiene un ledger distribuido y auditable tanto para la gobernanza de la red como para el seguimiento de los bienes en la cadena de suministro.

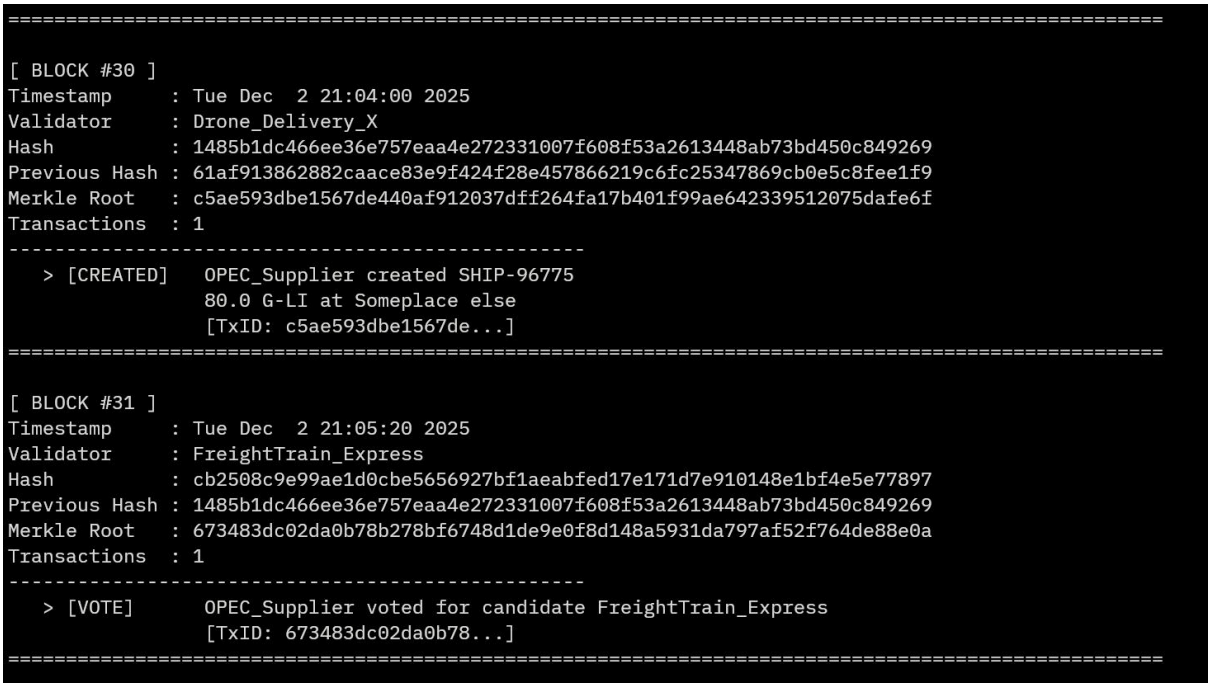
```
[ BLOCK #27 ]
Timestamp     : Tue Dec  2 20:13:41 2025
Validator     : Drone_Delivery_X
Hash          : 44ea90c5bddb8bc5573b422f678515e6155ac3b20075f67c42b035070d82fc9f
Previous Hash : 90f167f0542deb7b066e74d6311cec1b73b1eecbc7640e7541a5a099a0d909fa
Merkle Root   : 62cad24da8d9297bb5036a2d86ec003737757da16739126807660becc7cfdf2c
Transactions  : 1
------------------------------------------------
   > [VOTE]      Drone_Delivery_X voted for candidate OPEC_Supplier
                 [TxID: 62cad24da8d9297b...]
================================================================================================

[ BLOCK #28 ]
Timestamp     : Tue Dec  2 20:14:02 2025
Validator     : FreightTrain_Express
Hash          : c88a8ac03b3210f54d0972c20b80c444a0c7a5d2758b3f4c1e7861cc709d46fe
Previous Hash : 44ea90c5bddb8bc5573b422f678515e6155ac3b20075f67c42b035070d82fc9f
Merkle Root   : 2d0e7443b976faa50e54cd2038d2a68b127826f5dbc2e84728e66d17b02db833
Transactions  : 1
------------------------------------------------
   > [CREATED]   Drone_Delivery_X created SHIP-35735
                 89.0 G-OIL at LKJ
                 [TxID: 2d0e7443b976faa5...]
================================================================================================

[ BLOCK #29 ]
Timestamp     : Tue Dec  2 20:14:42 2025
Validator     : Drone_Delivery_X
Hash          : 61af913862882caace83e9f424f28e457866219c6fc25347869cb0e5c8fee1f9
Previous Hash : c88a8ac03b3210f54d0972c20b80c444a0c7a5d2758b3f4c1e7861cc709d46fe
Merkle Root   : 111196196a3cb068b204f211062f61bc0b5f6abfd01c69855ea2897dc4d2e17e
Transactions  : 2
------------------------------------------------
   > [RECEIVED]  Drone_Delivery_X confirmed/updated SHIP-35735
                 At: UIO
                 Updated Qty: 80.0
                 [TxID: 1c3ee280a04a4ce3...]
   > [CREATED]   Drone_Delivery_X manufactured SHIP-82125
                 100.0 G-MED at UIO
                 Sources: ['SHIP-35735']
                 [TxID: 38a6bf4942f352e6...]
================================================================================================
```

Figura 9. Registro de bloques con operaciones de voto, creación, recepción y manufactura

La Figura 9 muestra tres bloques consecutivos del ledger donde se registran distintos tipos de transacciones: un voto emitido por *Drone_Delivery_X*, la creación del shipment *SHIP-35735* por *FreightTrain_Express*, y un bloque con dos operaciones donde se confirma la recepción de ese shipment y posteriormente se manufactura un nuevo bien (*SHIP-82125*). Estos registros evidencian cómo el sistema documenta de forma secuencial y verificable las



acciones realizadas por los nodos dentro de la red.

Figura 10. Ejemplo de bloques consecutivos en el ledger del sistema

La Figura 10 muestra dos bloques consecutivos: el Bloque #30, validado por *Drone_Delivery_X* con una transacción de creación de bienes; y el Bloque #31, validado por *FreightTrain_Express*, con una transacción de voto del nodo *OPEC_Supplier*. Se observan los hashes, raíz de Merkle y detalles internos de las transacciones, como ejemplo del funcionamiento del blockchain del prototipo.

7.4 Pruebas de carga y simulación con *test.py*

Para comprobar que el sistema funciona bajo múltiples eventos consecutivos, se incluye un script de prueba que genera una secuencia de transacciones simulando el flujo real de una cadena de suministro. El script envía las operaciones a los nodos correspondientes y deja que la red procese los bloques y mantenga la sincronización. Este tipo de prueba demuestra que la

red propaga correctamente las transacciones, selecciona validadores de forma determinista, mantiene la consistencia del estado global, y reconstruye la cadena completa sin depender de un servidor central.

8. Limitaciones del prototipo

Aunque el sistema cumple con su objetivo principal de demostrar cómo una blockchain puede registrar y validar eventos dentro de una cadena de suministro, el prototipo tiene varias limitaciones derivadas de su naturaleza académica y del alcance establecido para el proyecto.

En primer lugar, el modelo de bienes y procesos está simplificado. Para facilitar la implementación, se definieron catálogos básicos de productos y acciones, pero no se incorporaron reglas detalladas sobre cómo se transforman los materiales en productos finales. Esto provoca situaciones poco realistas, como permitir que un minorista “extraiga” productos complejos directamente de la tierra o que se manufacturen bienes sin materias primas compatibles. En un sistema real, sería necesario establecer relaciones claras entre insumos, recetas de producción y restricciones físicas del proceso.

Relacionado con esto, el flujo de manufactura también se usa como una operación genérica para dividir o reorganizar shipments. Por ejemplo, si un embarque llega con cierta cantidad de material y se quiere separar en dos, el prototipo utiliza la acción “MANUFACTURED” para registrar el cambio, cuando en realidad debería existir una operación específica para separación o reasignación de inventario. Esta decisión simplifica el código, pero no representa fielmente la lógica de una cadena de suministro real.

Otra limitación importante es que los shipments solo pueden contener un tipo de bien. Aunque esto funciona para demostrar el sistema, no refleja la mayoría de los casos reales donde un embarque puede incluir múltiples productos, cantidades variadas o incluso lotes mezclados. La estructura del modelo permite agregarlo en un futuro, pero no fue implementado en esta versión.

Finalmente, estas limitaciones no son fallas del enfoque blockchain en sí, sino recortes deliberados del alcance para que la implementación fuera manejable dentro del tiempo del proyecto. El sistema funciona como un prototipo conceptual y operativo, y sienta las bases

para una versión más completa que podría incorporar modelos más robustos de producción, reglas más estrictas de consistencia y un diseño más fiel a los procesos logísticos reales.

9. Resultados y conclusiones

La ejecución del prototipo permitió validar que el diseño de la red y las decisiones técnicas funcionan en conjunto para ofrecer una cadena de suministro trazable, distribuida y confiable. Durante las pruebas se registraron múltiples transacciones que simulaban el ciclo completo de un bien: desde la extracción de materias primas hasta la manufactura, los envíos entre empresas y las acciones finales como el consumo o la destrucción. Cada una quedó registrada como una transacción firmada y, posteriormente, como parte de un bloque validado mediante DPoS.

Una vez que la red procesó todas las transacciones, cada nodo contó con la misma cadena de bloques y el mismo estado final. Esto confirma que la propagación de transacciones, la sincronización y el mecanismo de consenso funcionan correctamente, incluso cuando los nodos operan de manera independiente. Todas las pruebas de carga mostraron que la red se mantiene estable, sincronizada y consistente, sin depender de un servidor central.

Al consultar cualquier nodo con el script de visualización, se puede reconstruir el recorrido completo de cada shipment: quién lo creó, qué cantidad se manejó, dónde estuvo en cada etapa y cuál fue su última acción. Esto demuestra que el sistema logra su objetivo principal: ofrecer una trazabilidad clara y verificar que cada acción cumpla con las reglas establecidas.

Además, el uso de SHA-256, árboles de Merkle y firmas digitales añade una capa sólida de seguridad e integridad. La elección de Delegated Proof of Stake permitió mantener la eficiencia sin sacrificar descentralización, algo importante en una cadena de suministro donde se registran eventos constantemente.

En conclusión, los resultados muestran que este enfoque ofrece una alternativa viable para registrar la actividad de una cadena de suministro sin depender de intermediarios y con un nivel alto de transparencia y verificabilidad. El sistema mantiene coherencia entre nodos, protege la integridad de la información y permite auditar cualquier embarque paso a paso, lo cual cumple con las metas planteadas desde el inicio del proyecto.

10. Repositorio del proyecto

El código completo del sistema desarrollado se encuentra disponible en el siguiente repositorio de GitHub: https://github.com/G3rrry/proyectoFinalBlCh.git