IMPLEMENTACIÓN DE LA TECNOLOGÍA BLOCKCHAIN DENTRO DE LA CADENA DE SUMINISTRO: CASO APLICADO A PROCESO DE IMPORTACIÓN DE UNA EMPRESA DE MANUFACTURA

Trabajo de Grado para Optar al Título de Magister en Ingeniería

Presentado por:

Daniel Arroyave Martínez

Código: 1000008593

Asesores:

Juan Guillermo Lalinde Pulido

Diego Alexander Tobón Restrepo

UNIVERSIDAD EAFIT

MEDELLÍN

ESCUELA DE INGENERÍA

MAESTRÍA EN INGENIERÍA

2022

Contenidos

1. Abstract

The effects of globalization have made that supply chains (SC) became a major concern for most organizations. The number of actors involved within the SC make it an appealing target for cyber-attacks, corruption, and other threats that includes the flow of goods or information, therefore, there is an urgent need to start implementing actions that help to buffer the impact of such threats to keep a reliable SCs that guarantees a sustainable growth for all parties involved. Blockchain is one of the technologies that are part for the 4^{th} industrial revolution, which its main purpose is to make sure the information flow among certain parties can be highly reliable by creating blocks that are immutable as soon as they are approved to keep transparency among all organizations. In this order of ideas, applying blockchain may be a useful technology to combat against threats that may harm the flow of information. The purpose of this paper is to create a blockchain proof of concept applied in an import-export process of a manufacturing firm to demonstrate the benefits this technology can provide for the safety, reliability, and transparency of the information flow from the creation of the purchase order up to its arrival to the designated location.

2. Introducción:

La globalización ha hecho que la complejidad en la administración y coordinación de actividades relacionadas a la cadena de suministro aumente de manera significativa, en donde actores gubernamentales, navieras, puertos, compradores, vendedores, entre otros participantes deben alinearse para la realización efectiva del proceso. La gestión de la cadena de suministro abarca todo el flujo de producción desde la materia prima hasta la entrega del producto final (Perkins & Wailgum, 2017), por lo tanto, para poder entregar un producto, hay una serie de pasos que deben ser analizados e integrados para garantizar confiabilidad, transparencia y competitividad para poder sostenerse en el mercado y satisfacer las necesidades de la población.

En la actualidad, los medios tecnológicos convencionales como los correos, notificaciones o plataformas virtuales, los cuales trabajan de manera independiente, es decir, no se puede integrar la información entre todos los actores, aumentan el nivel de operatividad del

personal logístico para que se pueda garantizar el control, ejecución y trazabilidad de todos los eslabones de la cadena de suministro. Según Tradelens (2020), la gestión documental manual y operaciones hechas a papel reducen el potencial hasta un 15% de una empresa; esto se puede ejemplificar en el caso actual de compañías como Mitsubishi Electric en donde para garantizar todo el flujo de información del proceso de importación de una orden de compra, se debe invertir en una carga operativa para la ejecución de estas actividades.

El blockchain ha surgido como una herramienta que puede ayudar a la mejora de los procesos relacionados a la cadena de suministro, integrando a todos los actores de manera descentralizada, transparente, inmutable y segura. El blockchain es un libro mayor distribuido donde las transacciones están contenidas en bloques de manera cronológica (Para propósitos de esta tesis, el libro mayor tendrá nombre de *ledger*), las cuales están protegidas de alteraciones a través de la criptografía (Zhang, 2019). Aunque sea una tecnología reciente, se ven muchas posibilidades de aplicación en la cadena de suministro por sus características de flujo y por las necesidades actuales que se están presentando. Actualmente hay ejemplos implementados, como en Walmart y Tradelens, pero esta tecnología aún está en sus pasos iniciales. Este estudio presentará una prueba de concepto de la aplicación de blockchain en un proceso de importación y así encontrar las oportunidades que esta herramienta puede ofrecer.

3. Pregunta de investigación

¿Cómo aplicar la tecnología blockchain en la cadena de suministro internacional adaptado a las necesidades, legislación e infraestructura del comercio exterior en Colombia para una empresa una de manufactura (MELCOL).

4. Objetivo

Evaluar, mediante una prueba de concepto, la aplicación de la tecnología blockchain en el proceso de la cadena de suministro internacional, desde el momento de la orden de compra hasta su llegada al destino considerando los procesos físicos, financieros y documentales en una empresa de manufactura.

5. Objetivos específicos

- Diseñar flujo de comercio exterior de mercancías desde y hacia Colombia y éste como interactuaría a través de la tecnología blockchain.
- Implementar una prueba de concepto utilizando Hyperledger fabric
- Identificar los beneficios esperados a través de la aplicación de la tecnología blockchain en empresas importadoras y exportadoras de Colombia.
- Definir un perfil de empresa que sea beneficiada por la implementación de blockchain en la cadena de suministro internacional.

6. Cadena de suministro

6.1. ¿Qué es y cuál es su objetivo?

La cadena de suministro es el flujo que involucra a todos los participantes, de manera directa o indirecta, desde la materia prima hasta el cliente final, con el fin de satisfacer las necesidades del mercado objetivo. Entre las partes se encuentran proveedores, fabricantes, transportadores, bodegas y distribuidores y dentro de las actividades se incluyen las operaciones, las finanzas, las estrategias de distribución, entre otros (Chopra & Meindl, 2016).

El objetivo de la cadena de suministro es maximizar el valor agregado general, el cual es llamado superávit de la cadena de suministro (Chopra & Meindl, 2016), este consta de la diferencia entre el valor del consumidor y el costo de la cadena de suministro. Esto puede ser medido como el valor que un cliente está dispuesto a pagar contra los esfuerzos para cumplir con el requerimiento.

6.2. ¿Qué es comercio internacional?

El comercio internacional es una parte de la cadena de suministro internacional, la cual consta de la comercialización de bienes cruzando las fronteras (Pierre A., 2016). El método de entrada más común para esta actividad corresponde a la importación y exportación de bienes, por lo tanto, la gestión del comercio internacional abarca la administración de todos los riesgos, operaciones, obligaciones, deberes, entre otras actividades del proceso de importación y exportación (Pierre A., 2016).

6.3. Modelo SCOR para la gestión de la cadena de suministro internacional

El modelo SCOR hace referencia a sus siglas en inglés "*Supply chain Operations Reference"*, su objetivo es suministrar definiciones uniformes, conceptos, procesos y métricas para gestionar la cadena de suministro. Este facilita la comunicación y permite evaluar y comparar el desempeño entre empresas o industrias (Schwarz, 2008).

De acuerdo con (Hammadi, Souza de Cursi, Barbu, Ouahman, & Ibourk, 2018), el modelo SCOR posee tres niveles: Estructura de red de la cadena de suministro, configuración de pilares de cadena de suministro y configuración del proceso. Estos están compuestos de 5

procesos los cuales corresponden a la planificación, el aprovisionamiento, la fabricación, la distribución y la devolución (OBS Business School, 2021). La planificación incluye el planeamiento de suministros y administración de recursos, el aprovisionamiento considera la infraestructura de abastecimiento, la fabricación gestiona el proceso de manufactura, la distribución abarca pedidos, almacenaje y transporte y la devolución las reglas de negocio, inventario de cambio, devolución de productos, entre otros (ASCM, 2021).

En la ilustración 1 se describe el modelo SCOR adaptado a la cadena de suministro internacional de Mitsubishi Electric de Colombia. En el nivel 1, se define el alcance y contenido del SCOR, donde se visualizan de manera global los objetivos de rendimientos competitivos de la cadena de suministro (Pérez, 2021). En el nivel 2 se configuran los procesos de planificar, aprovisionar, fabricar y devolver en donde se muestra el flujo de interacción entre todas las partes para que el flujo pueda funcionar entre todos los participantes, los cuales están ilustrados en la tabla 1(Hammadi, Souza de Cursi, Barbu, Ouahman, & Ibourk, 2018). El nivel 3 genera un desglose detallado de las actividades de cada eslabón; estos son generalmente denominados inputs, outputs e información y materiales (Pérez, 2021).

Tabla 1. Participantes en el modelo SCOR

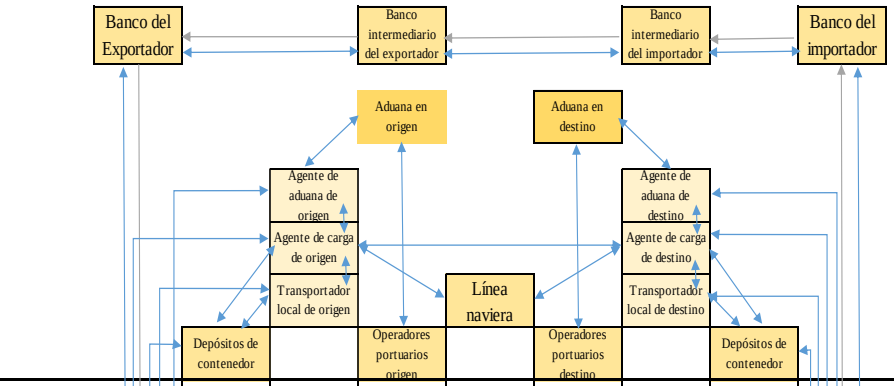
Categoría	Grupos de interés
Comercial	Importador y exportador
Organización	Forwarders, agente de la línea naviera, proveedores de servicios logísticos
Físicos	Operador de terminal portuaria, naviera, aerolínea, carriers, operador de depósitos de contenedor
Autorización	Aduana, autoridades aduaneras, policías en puerto, autoridades de inspección
Financiera	Banco, compañía de seguros

Tabla 1 *(Hammadi, Souza de Cursi, Barbu, Ouahman, & Ibourk, 2018)*

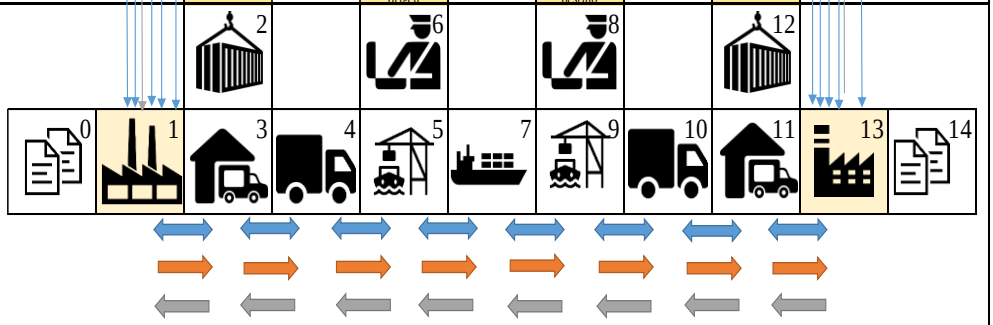
Para una gestión correcta del modelo SCOR, se debe involucrar posteriormente a la visión estratégica de la empresa, puesto que esta únicamente analiza el detalle de la cadena de suministro, pero no los otros puntos como ventas, finanzas, recursos humanos, entre otros.

MODELO SCOR

Nivel 2



Nivel 1



Flujo de información
Flujo de bienes
Flujo Financiero

Proceso
0. Creación de OC
1. Origen de mercancía (Exportador)
2. Recogida de contenedor vacío
3. Empaque de la mercancía
4. Transito local (Origen)
5. Proceso portuario (Origen)
6. Proceso Auduanero exportación
7. Tránsito internacional
8. Proceso aduanero importación
9. Manejo portuario
10. Transporte local
11. Recibo de mercancía
12. Entrega contenedor
13. Destino de mercancía (Importador)
14. Devolución por defecto

Nivel 3

Proceso clave	Confirmación de información	Preparación para la exportación	Recibo de mercancía	Transporte local	Proceso portuario y aduana en origen	Transporte internacional	Proceso portuario en destino	Transporte local	Recepción de la mercancía	Llegada de la mercancía	Análisis final
Funciones	Confirmación de fecha de despacho	Confirmación de documentación	Toma de registro de la mercancía	Tracking del estado de la mercancía	Notificación a agente de carga del importador	Tracking del estado de la mercancía	Notificación a agente de aduana para gestión.	Tracking del estado de la mercancía	Descargue de la mercancía	Notificación de llegada	Proceso de reclamación
	Segumiento a la fecha de envío	Reserva de transporte	Ubicación en el contenedor	Notificación a puerto de llegada de mercancía	Revisión de declaración de exportación	Notificación a puerto de llegada de mercancía	Generación de declaración de importación	Notificación a puerto de llegada de mercancía	Revisión de la documentación	Revisión del estado de los materiales	Revisión de indicadores
		Empaquetado de la mercancía			Confirmación de documentos		Confirmación de documentos			Entrada al sistema	
					Movimientos portuarios		Movimientos portuarios				
							Inspección física				
							Revisión de documentos				

Ilustración 1 Modelo SCOR inspirado de *(Hammadi, Souza de Cursi, Barbu, Ouahman, & Ibourk, 2018)*

6.4. Aspectos para considerar en el comercio internacional y la cadena de suministro internacional.

6.4.1. Método de entrada: Proceso de importación y exportación

De acuerdo con el artículo 3 del decreto 1165 de 2019, "*una importación es la introducción de mercancías de procedencia extranjera al territorio aduanero nacional cumpliendo con los términos y condiciones previstos en el presente decreto"* y *"una exportación es la salida de mercancías del territorio aduanero nacional con destino a otro país"*.

6.4.2. Infraestructura de transporte

En el campo de la logística, la palabra "infraestructura" es muy amplia, esta se refiere a todos los elementos dispuestos ya sean públicos o privados que facilitan el transporte. Es por esto, que estos no son únicamente los elementos de la comunicación y de transporte, sino también los servicios públicos, bancarios, canales de distribución, entre otros (Pierre A., 2016).

Para el proceso de importación, los modos de transporte más utilizados son el transporte marítimo, aéreo, terrestre, ferroviario y multimodal. La elección de cada uno de ellos dependerá de varios factores como la naturaleza del producto, distancia, embalaje, costo, entre otros (Diario del exportador, 2017).

Para tener una gestión correcta de la cadena de suministro internacional, es esencial comprender los retos que subyacen por el tema de la infraestructura internacional, puesto que cada país tiene diferentes estándares, desempeños, entre otras cosas que afectan al movimiento de bienes, documentos, dinero e información (Pierre A., 2016).

6.4.3. Términos comerciales

La manera en cómo se regula un proceso de comercio internacional de bienes es a través de las reglas Incoterms®, las cuales explican los términos de negociación reflejando contratos de compraventa de mercancía tanto doméstica como internacional (Procolombia, 2021). Los Incoterms®, son creados por la cámara de comercio internacional y definen obligaciones entre comprador y vendedor, adicionalmente, ayudan a establecer cómo se asignarán los costos y riesgos entre vendedores y compradores (DSV, 2020). Estos no son

vinculantes, pero son una metodología que se ha aceptado a nivel mundial puesto que su objetivo es tener un lenguaje común para todos (Enriquez Caro, 2015).

En la ilustración 2 se muestran los costos, obligaciones, riesgos y seguros que deben contemplar tanto los vendedores como los compradores al utilizar los Incoterms como medio de negociación de la cadena de suministro internacional (Cámara de comercio internacional, 2020).

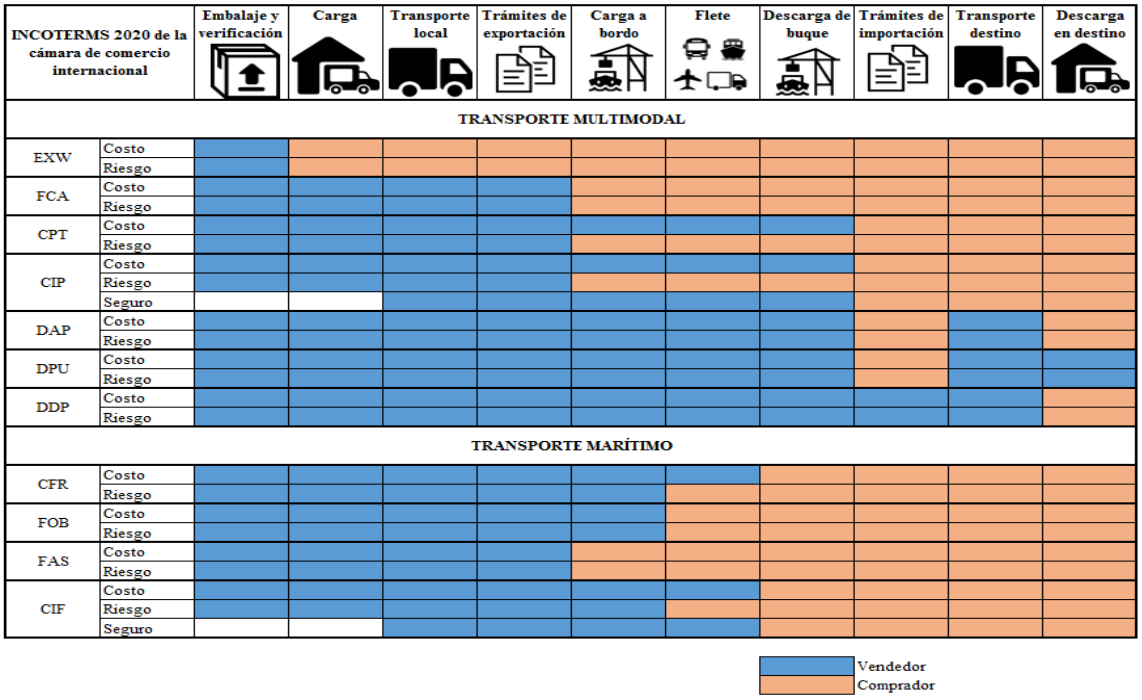


Ilustración 2 INCOTERMS 2020 (TIBA, 2020)

Es necesario destacar que los Incoterms no pueden reemplazar un contrato de compraventa, por ende, las especificaciones de los productos vendidos, el tiempo, lugar, moneda, sanciones, impuestos, aranceles o derechos de propiedad intelectual no hacen parte de la función de los Incoterms. Todos estos son pactados entre las partes en sus acuerdos en el contrato (Incoterms® 2020, 2020).

6.4.4. Condiciones de pago

Dentro de las condiciones de pago en el comercio internacional se destacan cinco formas: Pago por adelantado, carta de crédito, colección documental, cuenta abierta y consignación. La decisión de pago dependerá del nivel de riesgo que las partes estén dispuestas y estas son decisivas a la hora de realizar negocios entre los vendedores y compradores

(International Trade Administration, 2021). El nivel de riesgo es inversamente proporcional entre exportadores e importadores. En la ilustración 3, se presenta cuadro resumiendo el nivel de riesgos presentados por la administración de comercio internacional de los Estados Unidos en cuestión de las condiciones de pago.

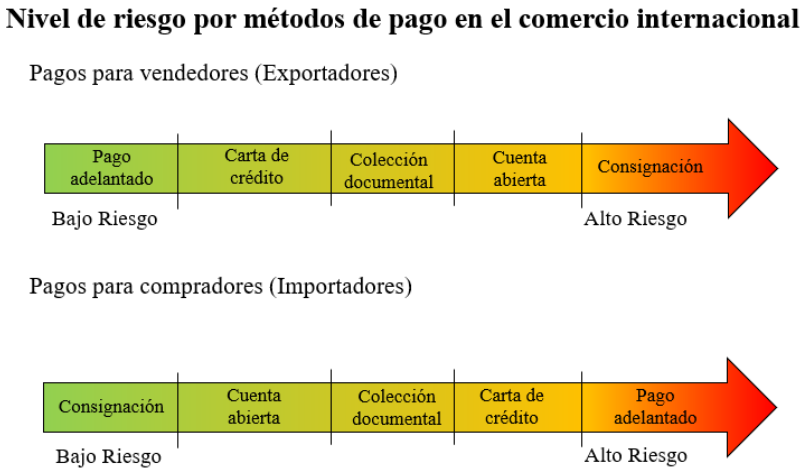


Ilustración 3 Nivel de riesgo por métodos de pago en el comercio internacional *(International Trade Administration, 2021)*

El pago adelantado corresponde a que el comprador pague el valor de la mercancía en su totalidad antes de enviar el producto. Las cartas de crédito son documentos emitidos por los bancos comerciales con el objetivo de garantizar que una mercancía sea pagada (Páez, 2020). La colección documental es un proceso en donde los exportadores van a un banco y presentan documentos de cobranzas correspondientes a la exportación, una vez los importadores paguen, el dinero se transfiere entre los bancos y finalmente a la empresa exportadora (Spiegato, 2022). La cuenta abierta corresponde a cualquier tipo de cuenta financiera donde algún tipo de crédito se extiende a un cliente. La consignación consiste en depositar dinero en la cuenta del vendedor quedando a disposición de cuando se vaya a utilizar (Gerencie, 2022) .

6.4.5. Documentación

Cada país requiere una serie de documentos para la exportación e importación de bienes, para el caso de Colombia, es establecen los siguientes documentos según el Artículo 177 del decreto 1165 de 2019:

- *"Registro o licencia de importación que ampare la mercancía, cuando hubiere lugar a ello hubiere lugar.*
- *Factura comercial, cuando hubiere lugar a ella.*
- *Documento de transporte.*

  La prueba de origen señalada en el respectivo acuerdo comercial y los documentos relativos a las condiciones de expedición directa, tránsito y/o transbordo, cuando a ello hubiere lugar; o certificación de origen no preferencial, cuando se requiera.
- *Certificado de sanidad y aquellos otros documentos exigidos por normas especiales, cuando hubiere lugar a ello.*
- *Lista de empaque, cuando hubiere lugar a ella.*
- *Mandato, cuando no exista endoso aduanero y la declaración de importación se presente a través de una agencia de aduanas o apoderado.*
- *Declaración andina del valor y los documentos justificativos de esta.*
- *Copia de la declaración de exportación o el documento que acredite la operación de exportación, en las modalidades de reimportación en el mismo estado y reimportación por perfeccionamiento pasivo, en los términos establecidos en el decreto.*
- *Las autorizaciones previas establecidas por la Unidad Administrativa Especial Dirección de Impuestos y Aduanas Nacionales (DIAN), para la importación de determinadas mercancías.*
- *Documento de constitución del consorcio o unión temporal cuando los documentos de transporte y demás documentos soporte de la operación de comercio exterior se consignen, endosen o expidan, según corresponda, a nombre de un consorcio o de una unión Temporal.*
- *Certificación de marcación física o electrónica expedida por el sistema técnico de control vigente (SUNIR), para los bienes sujetos al pago del impuesto al consumo de qué trata la Ley 223 de 1995. Este documento soporte solo será obligatorio una vez entre en producción la fase del SUNIR correspondiente a la obtención de información para cada industria.*

- *Documentos establecidos expresamente en disposiciones aduaneras o en normas especiales reguladas por otras autoridades, como soportes de la declaración de importación.”*

6.4.6. Seguridad en la logística internacional

Desde los atentados del 11 de septiembre en los Estados Unidos, se han creado medidas de seguridad que corroboren la integridad de la cadena de suministro internacional, por lo tanto, se han desarrollado técnicas y herramientas para la administración de seguridad que ayudan a disminuir la probabilidad de actos criminales que puedan afectar a las organizaciones (Pierre A., 2016). En la actualidad, es muy importante estar certificado para poder ser un cliente o proveedor atractivo. Algunas de las certificaciones son las siguientes:

OEA (Operador Económico Autorizado): Esta es una iniciativa liderada por la Organización Mundial de Aduanas (OMA) cuyo fin es garantizar que las operaciones realizadas en el comercio exterior en las organizaciones certificadas son seguras. El trámite en Colombia se hace a través de la DIAN (Procolombia, 2016).

- BASC (Business Alliance for Secure Commerce): Esta otra iniciativa certifica la confiabilidad de los asociados y de las alianzas estratégicas con el fin de brindar sostenibilidad en el comercio en beneficio de la sociedad en cooperación con gobiernos y organismos internacionales. Esta está liderada por el World BASC organization (WBO) (BASC, 2005).

6.4.7. Ciberseguridad en la cadena de suministro

Con base en (Gallagher & King, 2020) la ciberseguridad a lo largo de la cadena de suministro es definida como un sistema holístico que identifica tecnologías, procedimientos y personas con el objetivo de proteger la red, los dispositivos y todos los activos digitales que pueden ser afectados por daños, ataques o accesos no autorizados con el fin de afectar la cadena de suministro.

Actualmente, algunas de las metodologías para proteger los sistemas de ataques cibernéticos dentro de la cadena de suministro consisten en la instalación de firewalls,

programas de detección de virus, malware, entre otros (Griffin, 2021). Adicionalmente con actividades dentro de la organización como capacitación a empleados, contratación de expertos en ciberseguridad, colaboración con los proveedores, y con planes de contingencia dentro de los protocolos de plan de continuidad de negocio.

6.4.8. Autorización aduanal

Al entrar una mercancía al país, se debe pasar por un proceso de aduana el cual genera una serie de tributos que dependerán de una partida arancelaria la cual es el código de identificación mundial de dicho producto. Dentro de este tema se deben considerar los aranceles, los impuestos, la valuación, las reglas de origen, restricciones anti-dumping, entre otros (Pierre A., 2016).

6.5. Retos y problemas de la cadena de suministro internacional

Las cadenas de suministro en las organizaciones actualmente plantean retos y problemas que deben ser superados para el crecimiento sostenible de las empresas. Es por esto que la búsqueda de soluciones que permitan tener una cadena mucho más ligera es una necesidad latente para muchos. A continuación, se mencionan algunos de los casos:

- Adaptación al cambio: Hay una presión constante para responder a los cambios globales (Ben K, 2015). La toma de decisiones cada vez se complica más, especialmente cuando hay falta de recursos para analizar toda la información que sucede día a día.
- Transparencia en la información: Obtener e integrar información en tiempo real se ha convertido en un trabajo arduo. Uno de los primeros retos de la cadena de suministro consta de encontrar un recurso transparente y confiable para el intercambio de la información (Behera, 2017). La gran cantidad de integrantes hace que la operatividad de la gestión de la cadena de suministro aumente, el personal tenga más carga y se incrementen las probabilidades de error, adicionalmente, se presta para que la información sea modificada sin autorización.
- Medios desactualizados para compartir datos: En gran parte de las organizaciones, aún hay casos en que utilizan documentación física para las operaciones relacionadas a la cadena de suministro, como la solicitud de facturas originales para

liberación de carga en los puertos, conocimientos de embarques, entre otros (Zhang, 2019). Esto genera altas cargas de trabajo e ineficiencias en toda la cadena.

- Necesidad de cumplimiento y certificaciones: Las empresas en la actualidad deben demostrar en el mercado y a entidades gubernamentales que los procesos y actividades comerciales se están realizando de manera correcta sin afectar ningún asunto legal o ético. Es por esto que se debe tener un sistema de trazabilidad de todo el proceso de la cadena de suministro para garantizar todos los procesos. Hoy en día, muchas empresas colombianas están procediendo a certificarse como OEA (Operador Económico autorizado) y, para esto, deben tener trazabilidad desde la orden de compra hasta su llegada y certificar que sus proveedores cumplan con todos los requisitos.
- Hay urgencia para establecer estrategias de ciberseguridad dentro de las cadenas de suministro (Melnyk, y otros, 2021). Esto es un tema que apenas se está contemplando dentro de las organizaciones, pero es vital para las compañías garantizar que la información sea creíble y no sea fácilmente atacada o alterada. Adicionalmente, para el 2022 se considera que uno de los retos más considerables para la cadena de suministro es aumentar la seguridad de la información. Entre proveedores y clientes se debe cultivar una relación de confianza, no obstante, no hay organización que no esté expuesta a amenazas que pueden destruir a una compañía en su totalidad (Shea, 2022).

7. Blockchain

7.1. ¿Qué es blockchain?

El blockchain es un *ledger* inmutable y compartido que facilita el proceso de grabar transacciones y rastrear bienes en una red de negocios (Gupta, 2020). Esta metodología permite que los objetos de valor tanto tangibles como intangibles tengan una gestión menos riesgosa y reduce costos a todos los involucrados (IBM, 2021). Este *ledger* no responde a una autoridad central, sino que cada persona incluida en la comunidad puede visualizar la información. Una vez una transacción es realizada, se crea un bloque el cual es verificado por los participantes de la red para determinar si es válida o no (Peterson, y otros, 2019).

De acuerdo con Gupta (2020), Las características del blockchain son las siguientes:

- Consenso: Para que una transacción sea válida, los participantes deben estar de acuerdo con su validez. Estos mecanismos ayudan a garantizar que todos los nodos de la cadena estén sincronizados y sus transacciones sean legítimas (Joshi, 2019). Hay varios mecanismos de consenso que se pueden considerar y estos dependerán del tipo de red que se esté utilizando. Algunos de ellos son:
    - *Proof of work*: Cuando la información necesita ser añadida al blockchain, los mineros (creadores de nuevos bloques y verficiadores de los bloques añadidos a la cadena) ejecutan actividades computacionales para encontrar un bloque válido. Esto se hace encontrando un "*hash*" que satisfaga las condiciones necesarias (Bitcoin Suisse, 2021).
    - *Proof of stake*: Establece que una persona puede extraer o validar transacciones en bloque de acuerdo con el porcentaje de los recursos de la red que posee, por lo tanto, mientras más recursos posea un minero, más poder tendrá (Frankenfield, 2021).
    - *Byzantine fault tolerance*: Es la propiedad de un sistema de resistir las fallas derivadas del problema de los generales bizantinos, en otras palabras, este sistema puede seguir funcionando incluso si algunos de los nodos fallan o actúan de forma maliciosa (Binance, 2018).
- Procedencia: Los participantes conocen el origen de cada transacción y todo su historial.
- Inmutabilidad: A los participantes se les imposibilita modificar una transacción una vez esta sea grabada en el ledger. Esto se debe a que estas quedan guardadas en bloques de información que incluyen identificadores criptográficos, también llamados "*hash*". Cada bloque de la cadena tiene el *hash* del bloque anterior lo cual genera una cadena y une todo desde el primer bloque hasta el último. En caso de que algún bloque sea alterado dentro de la cadena, el *hash* cambia inmediatamente por lo que fácilmente se puede se puede identificar la ubicación del error (Peterson, y otros, 2019). En otras palabras, es más rentable participar legalmente del sistema que tratar de violentarlo.
- Finalidad: Un *ledger* compartido proporciona un lugar al que ir para determinar la propiedad de un activo o la finalización de una transacción.

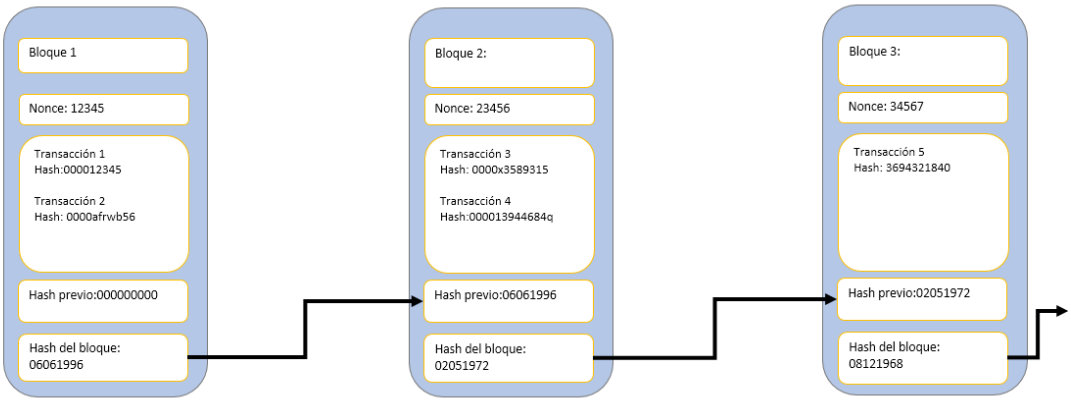


Ilustración 4 Flujo de Blockchain *(IBM, 2021)*

7.2. Elementos claves del blockchain

- Tecnología de *ledgers* distribuidos: Estos *ledgers* independientes usan nodos independientes para grabar, compartir y sincronizar las transacciones en cada nodo (World Bank, 2018).

- Contratos inteligentes: Los contratos inteligentes son códigos construidos para manipular automáticamente el movimiento de los bienes digitales entre las partes involucradas bajo ciertas condiciones (Ananthanarayanana & Arumugam, 2020). Para su uso se deben tener los permisos apropiados, y la transacción es exitosa si se cumplen todas las condiciones del contrato inteligente. Su uso ayuda a la optimización de procesos y por ende la reducción de tiempo en la operatividad.

- Registros inmutables: En este *ledger* permanece, permanente e inalterable, todo el historial de transacciones (Doubleday, 2018).

7.3. Bases de datos tradicionales versus blockchain

Las bases de datos hacen parte fundamental en las organizaciones; estas pueden contener listas de clientes, propiedades, información logística, precios, entre otros. Una base de datos es un conjunto organizado de información donde se puede encontrar y actualizar datos (Hyperledger, 2018).

La diferencia principal entre una base de datos tradicional y el blockchain es la centralización (IBM Blockchain Pulse, 2019). Las bases de datos tradicionales utilizan una

red de arquitectura enfocada al cliente y servidor (Tarasenko, 2019). Los usuarios en este caso pueden cambiar la información que es guardada en el servidor central puesto que el control prevalece en una autoridad designada, el cual autentica la información antes de ser agregada. No obstante, al tener un control único, si este es comprometido, la base de datos queda expuesta a modificaciones y violaciones. Adicionalmente, si se distribuye la base de datos entre varios participantes, pueden surgir problemas de consistencia e integridad. En el caso de blockchain, cada participante tiene una copia segura de todos los registros y cambios, por lo que cada usuario puede ver la procedencia de la información. Es por esto que, al momento de aparecer inconsistencias, esta tecnología permite identificar rápidamente el siniestro con el fin proteger la transparencia de la información.

A continuación, basándose en el desarrollo de Trust your supplier (Trust your supplier, 2021), se presentan las ventajas que tiene el blockchain en comparación de los sistemas convencionales.

Tabla 2. Sistemas convencionales vs Blockchain

Sistemas convencionales	Blockchain
Bases de datos múltiples: Cada participante tiene su base de datos propia lo que puede incrementar la posibilidad de fraude o de error.	**Ledger compartido:** Hay un *ledger* compartido entre todas las partes el cual es muy difícil de modificar y las transacciones no pueden ser alteradas.
Falta de seguridad en la información: Las bases de datos convencionales pueden ser atacadas fácilmente, destruyéndola o corrompiéndola lo cual se torna delicado para los participantes.	**Recurso único:** Suministra verificación de errores y validez de transacciones que no son obtenidas en bases de datos convencionales.
Procesos manuales: Generalmente cargada con procesos manuales, lo cual puede generar retrasos e ineficiencias.	**Proveniencia clara:** Se garantiza que los datos son válidos y conciliados con los datos en poder de los demás participantes en Blockchain.

Validación externa: La dependencia de validación a través de intermediarios puede crear ineficiencias en el proceso.	**Seguridad criptográfica:** Se graban las transacciones de manera criptográfica de inicio a fin para reducir posibilidades de ataques.

Tabla 2 (Trust your supplier, 2021)

7.4. Tipos de blockchain

De acuerdo con (IBM, 2021), existen los siguientes tipos de blockchain:

Blockchain público: Un blockchain público corresponde a una plataforma donde cualquier persona puede participar; este es el caso de Bitcoin.

Blockchain privado: A diferencia del blockchain público, hay una organización que gobierna la red, controla quienes son las personas que pueden participar, ejecuta el protocolo de consenso y mantiene el *ledger* compartido.

Blockchain de consorcio: En esta red hay múltiples organizaciones que comparten responsabilidades dentro del blokchain, son encargadas de acceder y suministrar la información.

7.5. Plataformas de Blockchain

7.5.1. Hyperledger fabric

Hyperledger fabric es una plataforma de código abierto que fue establecida por Linux Foundation diseñada para contextos empresariales, la cual ofrece unas capacidades diferenciales claves sobre otras plataformas de blockchain. Su alta capacidad de configuración y modularidad, hacen que Fabric sea usada en varias industrias como seguros, cadena de suministro, recursos humanos y salud (Hyperledger fabric, 2020).

Las características principales, según Hyperledger fabric (2020), son las siguientes:

- Es la primera plataforma DLT (Distributed ledger technology) que puede aplicarse contratos inteligentes en lenguajes de programación como Java, Go, Node Js, entre otros, lo que evita el aprendizaje de nuevos lenguajes de programación.
- La plataforma Hyperledger fabric es permisionada, por lo tanto, los participantes de la red son conocidos.

- Se pueden utilizar diferentes protocolos de consenso, lo que facilita la personalización para cada uno de los casos que se vayan a presentar dependiendo de la situación.
- Fabric soporta protocolos de consenso, para incentivar la minería de datos o la ejecución de contratos inteligentes, que no están basados en prueba de trabajo.

La combinación de estas características hace que sea una de las plataformas más atractivas para el procesamiento de datos, garantizando privacidad y confiabilidad de las transacciones y los contratos inteligentes.

7.5.2. Ethereum

Ethereum es una tecnología que permite a las personas enviar cripto monedas a cualquier persona cobrando una pequeña tarifa. Ofrece una red peer-to-peer segura para el comercio. Es utilizada principalmente en el sector financiero y es una plataforma de acceso público (Ethereum, 2021).

7.6. ¿Qué tipo de blockchain es más conveniente para la cadena de suministro?

En la ilustración 5 de (Blockchain training Alliance, 2018), se grafica el cuestionario que una organización debe aplicar con el fin de tomar la decisión de utilizar la tecnología Blockchain en sus procesos. Para el caso de la cadena de suministro, la ruta deseada es la resaltada en color naranjado, indicando que la tecnología más adecuada consiste en un Blockchain privado *(Permissioned)*.

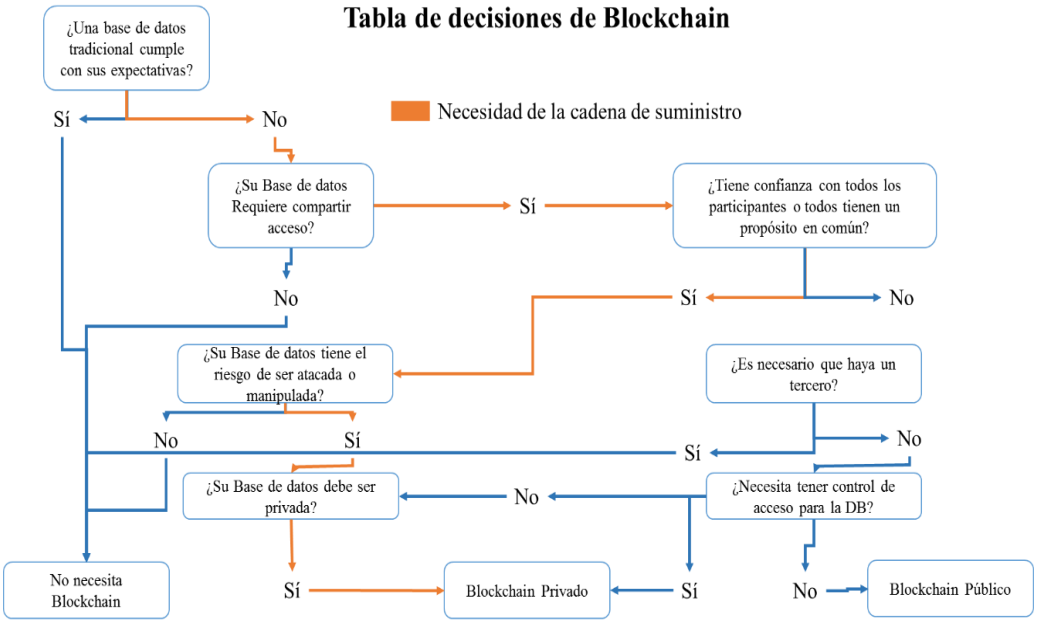


Ilustración 5 Tabla de decisiones de Blockchain (Hyperledger, 2018)

7.7. Beneficios del blockchain en la gestión de la cadena de suministro

En la actualidad las cadenas de suministro poseen un nivel de complejidad muy alto por lo que tener control de las operaciones en cada uno de los eslabones es un reto para las organizaciones. Con la incorporación del blockchain se han encontrado los siguientes beneficios:

- Mejora la comunicación entre todas las partes de la cadena de suministro (Gupta, 2020): Dado que todos los participantes están interconectados, este *ledger* descentralizado mejora la comunicación de todos al garantizar que la información de envíos, cantidades, tiempos, entre otros sean correctos y suministrados en tiempo real.
- Disminuye la carga operativa en la cadena de suministro: Según Tradelens (Tradelens, 2020) la gestión documental manual y operaciones hechas a papel reducen el potencial hasta un 15% de una empresa, lo que concluye claramente que las organizaciones deben migrar a un proceso automatizado que colaboren a la reducción de tiempo y costos, lo cual es una característica del blockchain. Adicionalmente la reducción de envío de correos o información a varios participantes de manera repetitiva.

- Reducción en gastos administrativos: Se generan ahorros por el envío internacional de documentos físicos, como pueden ser las facturas, conocimientos de embarques y certificados de origen, entre otros (Gupta, 2020).
- Incrementa la transparencia en la cadena de suministro: A través del ledger distribuido, los participantes autorizados podrán ver todas las actividades relacionadas a la cadena de suministro (IBM, 2021).
- Aumenta la confiabilidad para todos los participantes: El blockchain ayuda a eliminar los puntos de fricción tradicionales suministrando una plataforma con altos niveles de confianza (Gupta, 2020). En el caso de importación o exportación de mercancía, garantiza que todos los requerimientos solicitados por el gobierno del país estén completos y con toda la información necesaria.
- Seguridad en la información a través de plataformas permisionadas: dado que la información es imposible de modificar, hay muchas maneras de identificar los ataques a una cadena de suministro en los eslabones (Gupta, 2020).
- Optimización de inventario: Al tener información "al instante", las empresas no deben invertir mucho tiempo para la validación de datos y se puede actuar de una manera más ágil (Gupta, 2020). Para el caso de una cadena de suministro de la alimentación, se reducen los desechos e incrementa los productos posibles para su consumo.

7.8. Casos actuales de blockchain aplicados a la cadena de suministro

7.8.1. Trust your supplier:

Trust your supplier es un Desarrollo creado por Chainyard en conjunto con IBM. Consta de un sistema de gestión de información de proveedores cuyo objetivo es agilizar la búsqueda de proveedores yreducir el riesgo. Esto se realiza a través de la modernización de los sistemas de información relacionados con la gestión de proveedores y creación de nuevos estándares en la industria en cuestión de identificaciones digitales (IBM, 2021). De esta manera, se pueden reducir los tiempos correspondientes a la búsqueda, calificación, validación y gestión de los proveedores y mantenerse actualizado sin tener que hacer esfuerzos mayores.

Esta plataforma busca brindar beneficios a cada uno de los participantes dentro de la cadena de suministro. Los compradores tendrán acceso a información verificada de los requerimientos legales, de producto, cantidad, entre otros. Los auditores podrán acceder rápidamente a los datos que se necesiten verificar, los sistemas de registro tendrán la oportunidad de integrar la información del proveedor dentro de los sistemas existentes, los proveedores no tendrán que estar enviando información a todos los clientes cada vez que se requiere, por el contrario, simplemente se hará una vez a través de una llave digital, entre otros (Trust your supplier, 2021).

En la actualidad, empresas como IBM, Nokia, Vodafone, Lenovo y Jetblue utilizan esta plataforma para la gestión de proveedores (Trust your supplier, 2021).

7.8.2. *Tradelens*:

Tradelens es una plataforma de blockchain permisionada fundada entre IBM y Maersk orientada a la gestión de la cadena de suministro, especialmente para el proceso de importación o exportación de un producto. Los objetivos de esta es conectar a todos los ecosistemas como vendedores, compradores, agentes de carga, agentes de aduana, puertos, terminales, agentes del gobierno entre otros para automatizar los procesos logísticos, generar transparencia en la información, compartir información entre todos los involucrados, detalles de la carga, documentos y estados (Tradelens, 2020).

En la actualidad Tradelens posee grandes clientes como Hamburg Sud, Agility logistics y PSA Singapore (Apps Run The World, 2021), adicionalmente ha registrado más de dos mil millones de eventos, más de veinte millones de documentos y más de cuarenta millones de procesos de contenedores (Tradelens, 2020).

7.8.3. *Food trust*:

Food trust es una plataforma creada por IBM, apoyada del blockchain que proporciona a todos los participantes de la red rastreo, transparencia en las certificaciones relacionados a productos alimenticios y seguridad en la información (IBM, 2021). IBM *food trust* consiste en diferentes módulos para cada uno de los participantes en el entorno alimenticio, entre ellos productores, proveedores, manufactureras, distribuidores, entre otros. Este sistema

optimiza procesos de búsqueda, genera pronósticos en tiempo real y sincroniza de la cadena de suministro (IBM Food trust, 2021).

En la actualidad IBM Food Trust posee grandes clientes como Nestlé y Walmart (Apps Run the World, 2021).

8. Prototipo

8.1. Descripción del prototipo

La prueba de concepto consiste en crear un entorno blockchain, el cual se realiza a través de las instrucciones dadas por Hyperledger Foundation en su curso desarrollo Blockchain (HyperledgerFoundation, 2020), con el fin de que varias organizaciones, dentro de la cadena de suministro, puedan acceder y transaccionar la información de tracking de una orden de compra internacional, desde la aprobación de la orden de compra de cable de tracción de 12mm desde Corea del sur hasta la llegada al almacén en Medellín Colombia. Este considerará los factores de movimiento de mercancía, documentación y datos financieros para cubrir las 3 dimensiones principales de la cadena de suministro acuerdo al modelo SCOR.

8.2. Consideraciones para el prototipo que se va a realizar para la red de Hyperledger Fabric

A continuación, se presentan las consideraciones necesarias para poder realizar la red de esta prueba de concepto.

- Se debe tener una infraestructura instalada en el computador que permita desplegar una red de Hyperledger fabric. Para este caso se debe tener instalado Docker, WSL2 (Windows Subsystem for Linux version 2), Git, cURL, Go, JQ y Fabric samples, docker samples and binaries. Esta red se ejecutará en un computador cuyo sistema operativo es windows, pero se trabajará bajo la red Ubuntu. Adicionalmente, para el desarrollo del código se utilizará Visual Studio Code por su facilidad en la gestión del código.
- Se escogerá el dominio test.com para agrupar un conjunto de participantes de la red. Para facilidad y entendimiento del ejercicio se analizarán únicamente 3 organizaciones:

    - Organización 1 (Comprador): org1.test.com
    - Organización 2 (Gobierno): org2.test.com
    - Organización 3 (Agente logístico): org3.test.com
- Para que hyperledger fabric funcione se necesita el servicio de ordenamiento para la red, el cual ayuda a realizar transacciones y ordenar las solicitudes, peticiones o transacciones que generan las organizaciones.
- Se creará un canal de Hyperledger fabric que permita montar un canal seguro privado donde únicamente las 3 organizaciones podrán ver y generar la información que ocurre en el canal.
- Cada organización va a tener únicamente un nodo.
- Solo se utilizará una autoridad de certificación para la organización 1.
- Habrá un ledger distribuido que está presente para cada uno de los nodos.
- Se utilizará el lenguaje de programación Go para la creación del contrato inteligente (*Smart contract*)
- Para las políticas de respaldo de las transacciones, se indicará que únicamente la primera y la tercera organización podrán aprobar transacciones. La segunda organización solo tendrá visualización y no escritura.
- Contenerización: Cada componente está dockerizado, el servicio de ordenamiento será un servicio de docker

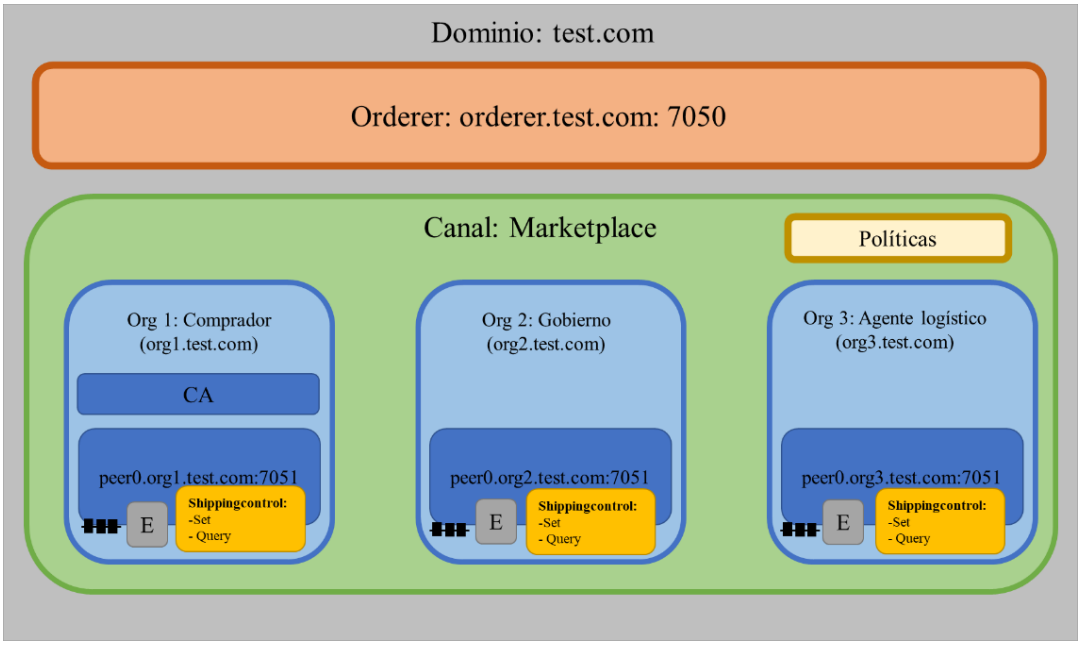


Ilustración 6 Diseño de prueba de concepto *(HyperledgerFoundation, 2020)*

8.3. Pasos

8.3.1. Paso 1: Creación de material criptográfico

Antes de comenzar, se debe crear una carpeta donde se van a almacenar todos los códigos. Para este ejercicio se llamará “pruebaconcepto”.

El primer paso consiste en crear el material criptográfico que contiene todos los certificados, las llaves privadas y usuarios. Para que esto funcione, se debe tener el programa instalado llamado “Cryptogen”.

Se crea un archivo llamado crypto-config.yaml, el cual contiene las definiciones del material criptográfico, en otras palabras las identificaciones de los componentes de alto nivel de la red.

8.3.1.1. Paso 1.1: Crear archivo crypto-config.yaml

Primera sección: Define cuál es el servicio de ordenamiento principal del consorcio

```
OrdererOrgs:
```

- `Name: Orderer` -> Define el Nombre

`Domain: test.com` -> Define cuál es el dominio principal del consorcio

`EnableNodeOUs: true` -> Habilita el NodeOUs

`Specs:`

`- Hostname: orderer`

`SANS:`

`- localhost` -> Se define el local host para poder operar en una red local para la prueba de concepto, no obstante, para el ámbito productivo se debe colocar el nombre del servidor correspondiente a la empresa.

Segunda sección

En la segunda sección se definen los pares (*peer*) que van a ser parte de la red de blockchain. La información es similar a la primera sección. A continuación, se explicarán las variables que cambian.

```
PeerOrgs:
  - Name: Org1
    Domain: org1.test.com
    EnableNodeOUs: true
    Template:
      Count: 1 -> Se define cuántos nodos se desean crear para cada organización.
      SANS:
        - localhost
    Users:
      Count: 1 -> Se define cuántos usuarios se deben crear para cada organización.
  - Name: Org2
```

```
    Domain: org2.test.com
    EnableNodeOUs: true
    Template:
      Count: 1
      SANS:
        - localhost
    Users:
      Count: 1
  - Name: Org3
    Domain: org3.test.com
    EnableNodeOUs: true
    Template:
      Count: 1
      SANS:
        - localhost
```

Para agregar más organizaciones, simplemente se debe generar para cada organización la información, similar a la que fue explicada arriba. De esta forma se pueden adicionar la cantidad de organizaciones, nodos y usuarios que se desee.

8.3.1.2. Paso 1.2: Utilizar programa Cryptogen

En la ubicación donde se está trabajando "~/pruebaconcepto/pruebaconcepto-network" se ejecuta utiliza el comando cryptogen para crear las organizaciones de acuerdo con la configuración definida en crypto-config.yaml, así:

```
cryptogen generate  --config=./crypto-config.yaml
```

y la respuesta que debe aparecer es la siguiente:

`org1.test.com`

`org2.test.com`

`org3.test.com`

Una vez ejecutado el comando correctamente, se creará una carpeta llamada crypto-config que contienen todos los certificados y llaves privadas del orderer y de los *peers* con sus respectivos certificados para las autoridades de certificación (CA), proveedores de servicios de afiliación (msp), y la comunicación segura (tls), como se puede ver en la imagen adjunta:

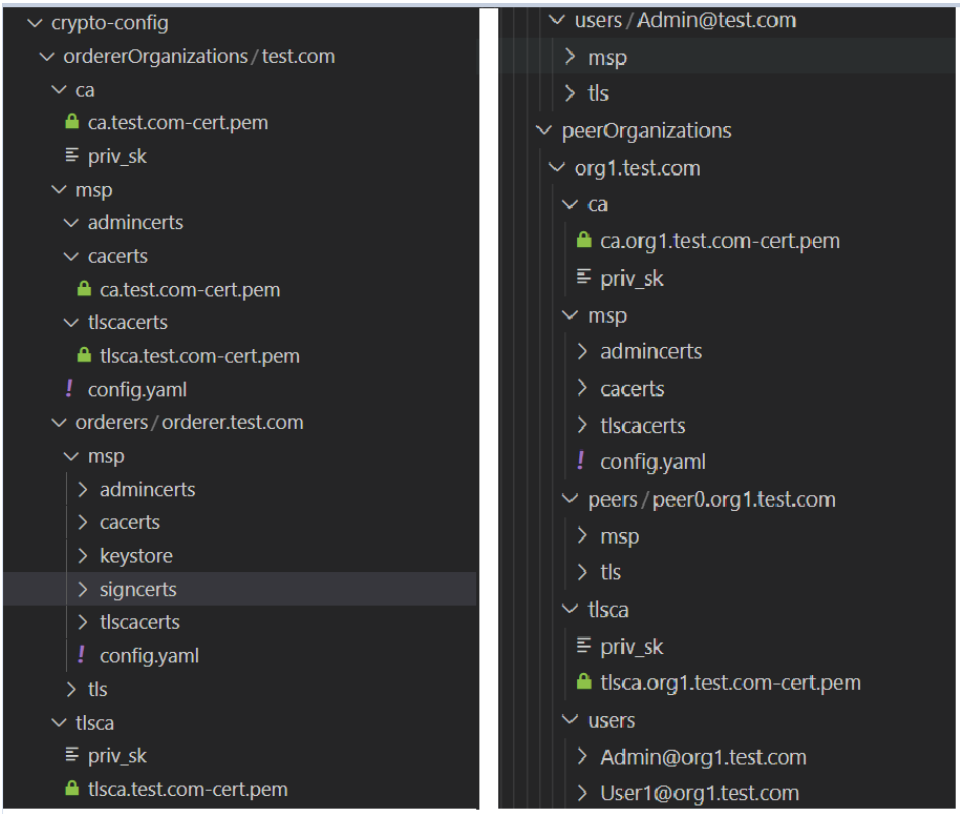


Ilustración 7 Resultado de Cryptogen (Creación propia)

8.3.1.3. Paso 1.3: Crear las transacciones de configuraciones en archivo configtx.yaml

El archivo configtx.yaml tiene como objetivo configurar, organizar, definir compatibilidad, políticas, servicios de ordenamiento y gestionar atributos de canales para generar el bloque inicial o bloque génesis.

Sección 1 de configtx.yaml: Organizaciones

`Organizations:` -> Define las organizaciones miembro de la red

```
- &OrdererOrg
    Name: OrdererOrg
    ID: OrdererMSP
    MSPDir: crypto-config/ordererOrganizations/test.com/msp
```

-> Define dónde está la configuración del membership service provider (MSP)

`Policies:` -> Se definen las políticas, para este caso se permiten todos los permisos.

```
        Readers:
            Type: Signature
            Rule: "OR('OrdererMSP.member')"
        Writers:
            Type: Signature
            Rule: "OR('OrdererMSP.member')"
        Admins:
            Type: Signature
            Rule: "OR('OrdererMSP.admin')"
```

`OrdererEndpoints:` -> Define el servicio que se va a levantar

- orderer.test.com:7050 -> Se define el puerto para levantar el servicio de ordenamiento dentro del contenedor de Docker

```
        - &Org1
            Name: Org1MSP
            ID: Org1MSP
            MSPDir: crypto-config/peerOrganizations/org1.test.com/msp
            Policies:
                Readers:
                    Type: Signature
                    Rule: "OR('Org1MSP.admin', 'Org1MSP.peer', 'Org1MSP.client')"
                Writers:
                    Type: Signature
                    Rule: "OR('Org1MSP.admin', 'Org1MSP.client')"
                Admins:
                    Type: Signature
                    Rule: "OR('Org1MSP.admin')"
                Endorsement:
                    Type: Signature
                    Rule: "OR('Org1MSP.peer')"
```

AnchorPeers: -> Se definen los anchor peers, el cual actuará como punto de contacto principal con toda la red.

- Host: peer0.org1.test.com -> Definir dominio

  Port: 7051 -> Definir el puerto en el local host

Para las siguientes organizaciones se procede al igual que con la primera

```
        - &Org2
            Name: Org2MSP
            ID: Org2MSP
            MSPDir: crypto-
config/peerOrganizations/org2.test.com/msp
            Policies:
                Readers:
                    Type: Signature
                    Rule: "OR('Org2MSP.admin', 'Org2MSP.peer',
'Org2MSP.client')"
                Writers:
                    Type: Signature
                    Rule: "OR('Org2MSP.admin', 'Org2MSP.client')"
                Admins:
                    Type: Signature
                    Rule: "OR('Org2MSP.admin')"
                Endorsement:
                    Type: Signature
                    Rule: "OR('Org2MSP.peer')"
            AnchorPeers:
```

```
                - Host: peer0.org2.test.com
                  Port: 7051


        - &Org3
            Name: Org3MSP
            ID: Org3MSP
            MSPDir: crypto-
config/peerOrganizations/org3.test.com/msp
            Policies:
                Readers:
                    Type: Signature
                    Rule: "OR('Org3MSP.admin', 'Org3MSP.peer',
'Org3MSP.client')"
                Writers:
                    Type: Signature
                    Rule: "OR('Org3MSP.admin', 'Org3MSP.client')"
                Admins:
                    Type: Signature
                    Rule: "OR('Org3MSP.admin')"
                Endorsement:
                    Type: Signature
                    Rule: "OR('Org3MSP.peer')"
            AnchorPeers:
```

```
            - Host: peer0.org3.test.com
              Port: 7051
```

Sección 2: Capabilities

Esta sección define las capacidades de la red, de manear que se garantiza la compatibilidad para nodos que ejecuten versiones diferentes de la misma. Define las características de cada una de las capas que hacen parte de una red de hyperledger.

```
Channel: &ChannelCapabilities
```

`V2_0: true` -> Habilita las capacidades de la versión 2.0 para los canales y capacidades del canal.

```
Orderer: &OrdererCapabilities
```

`V2_0: true` -> Habilita las capacidades de la versión 2.0 para el orderer y las capacidades del orderer.

```
Application: &ApplicationCapabilities
    V2_0: true
```

Sección 3: Application

Esta sección define las políticas de configuración para esta red.

```
Application: &ApplicationDefaults
    Policies:
        Readers:
            Type: ImplicitMeta
            Rule: "ANY Readers"
        Writers:
            Type: ImplicitMeta
```

```
            Rule: "ANY Writers"
        Admins:
            Type: ImplicitMeta
            Rule: "MAJORITY Admins"
        LifecycleEndorsement:
            Type: ImplicitMeta
            Rule: "MAJORITY Endorsement"
        Endorsement:
            Type: ImplicitMeta
            Rule: "MAJORITY Endorsement"


    Capabilities:
        <<: *ApplicationCapabilities
```

Sección 4: Orderer

En esta sección se define básicamente cuál es la implementación del servicio de ordenamiento por el cual se va a iniciar la red. Hay que tener en cuenta que hay 3 tipos de servicio de ordenamiento: El Solo, el cual es utilizado para pruebas, el Kafka que correspondía a la versión anterior a la 2, y el etc draft que corresponde a ámbitos productivos. Para la prueba de concepto se utilizará la versión “Solo” de la siguiente manera:

```
Orderer: &OrdererDefaults
    # Orderer Type: The orderer implementation to start
    OrdererType: solo
```

Nota: En caso de utilizar el etcdraft, se debe hacer de la siguiente forma:

```
   OrdererType: etcdraft

        EtcdRaft:

            Consenters:

            - Host: orderer.test.com

              Port: 7050
```

A continuación, se definen los protocolos seguros:

```
ClientTLSCert:  ../organizations/ordererOrganizations/test.com/ord
erers/orderer.test.com/tls/server.crt

ServerTLSCert:  ../organizations/ordererOrganizations/test.com/ord
erers/orderer.test.com/tls/server.crt


        Addresses:

            - orderer.test.com:7050

        BatchTimeout: 2s

        BatchSize:

            MaxMessageCount: 10

            AbsoluteMaxBytes: 99 MB

            PreferredMaxBytes: 512 KB


        Kafka:
```

```
            Brokers:
                - 127.0.0.1:9092
        Organizations:

        Policies:
            Readers:
                Type: ImplicitMeta
                Rule: "ANY Readers"
            Writers:
                Type: ImplicitMeta
                Rule: "ANY Writers"
            Admins:
                Type: ImplicitMeta
                Rule: "MAJORITY Admins"
            # BlockValidation specifies what signatures must be
included in the block
            # from the orderer for the peer to validate it.
            BlockValidation:
                Type: ImplicitMeta
                Rule: "ANY Writers"
```

Sección 5: Channel

Esta sección define las políticas que gobierna el nivel más alto de la configuración del canal.

```
Channel: &ChannelDefaults -> se definen políticas
        Policies:
            # Who may invoke the 'Deliver' API
            Readers:
                Type: ImplicitMeta
                Rule: "ANY Readers"
            # Who may invoke the 'Broadcast' API
            Writers:
                Type: ImplicitMeta
                Rule: "ANY Writers"
            # By default, who may modify elements at this config
level
            Admins:
                Type: ImplicitMeta
                Rule: "MAJORITY Admins"


        Capabilities:
            <<: *ChannelCapabilities
```

Sección 6 : Profile

En esta sección se deben definir 2 perfiles: El primer perfil define el bloque génesis del canal y el segundo perfil define cómo se va a crear el canal para las transacciones. En otras palabras, cómo está configurado el funcionamiento del canal de la red hyperledger fabric.

```
    Profiles:
```

Perfil 1:

ThreeOrgsOrdererGenesis:

<<: *ChannelDefaults -> Esto indica que se utilizará la información que se ha dejado en el canal.

Orderer:

<<: *OrdererDefaults -> Esto indica que se utilizará la información que se ha dejado en *Orderer*-

Organizations:

- *OrdererOrg -> Esto indica que se utilizará la información que se ha dejado en el orderer.

Capabilities:

<<: *OrdererCapabilities

Consortiums:

SampleConsortium:

Organizations:

- *Org1

- *Org2

- *Org3

Perfil 2:

ThreeOrgsChannel:

Consortium: SampleConsortium

`<<: *ChannelDefaults ->` Esto indica que se utilice las funciones de ChannelDefaults

```
Application:
    <<: *ApplicationDefaults
    Organizations: ->  Se indican Participantes
        - *Org1
        - *Org2
        - *Org3
    Capabilities:
        <<: *ApplicationCapabilities -> Utilizar las funciones de ApplicationCapabilities
```

8.3.1.4. Paso 1.4: Crear bloque génesis

Se crea una carpeta llamada "channel-artifacts" y se utiliza el programa “configtxgen” para la creación del bloque inicial (genesis) y del canal:

Transacción 1*:* `configtxgen -profile ThreeOrgsOrdererGenesis -channelID system-channel -outputBlock ./channel-artifacts/genesis.block`

Explicación de la transacción 1: crea el bloque inicial

`configtxgen -profile ThreeOrgsOrdererGenesis` (Indicar qué perfil voy a utilizar) `-channelID` (Indicar el canal por defecto que se llama system channel) `system-channel -outputBlock` (Se indica dónde se desea almacenar la información`)./channel-artifacts/genesis.block`

8.3.1.5. Paso 1.5: Crear y configurar el canal de transacciones

Para este paso se debe proceder con el siguiente comando:

```
configtxgen -profile ThreeOrgsChannel -outputCreateChannelTx./channel-artifacts/channel.tx -channelID marketplace
```

Explicación del comando

`configtxgen -profile ThreeOrgsChannel` (Indicar perfil) `-outputCreateChannelTx` (Generar la transacción de configuración de canal, que la cargue en el canal)`./channel-artifacts/channel.tx -channelID`(Escoger el nombre del canal) `Marketplace`

8.3.1.6. Paso 1.6: Crear archivos de para el anclaje de los pares (anchor peers)

Se debe proceder con el siguiente comando, para cada una de las organizaciones, para crear los anclajes que permitan a los pares tener acceso a las características avanzadas como información privada y descubrimiento de servicios.

`Configtxgen -profile ThreeOrgsChannel` (Indicar perfil) `-outputAnchorPeersUpdate ./channel-artifacts/Org1MSPanchors` (Cómo se va a llamar)`.tx -channelID` (Indicar nombre del canal) `marketplace -asOrg` (Para qué organización) `Org1MSP`

```
Configtxgen -profile ThreeOrgsChannel -outputAnchorPeersUpdate ./channel-artifacts/Org2MSPanchors.tx -channelID marketplace -asOrg Org2MSP
```

```
Configtxgen -profile ThreeOrgsChannel -outputAnchorPeersUpdate ./channel-artifacts/Org3MSPanchors.tx -channelID marketplace -asOrg Org3MSP
```

Al finalizar los pasos 1.4, 1.5 y 1.6 en la carpeta “channel-artifacts” se habrán creado los siguientes archivos: channel.tx, genesis.block, Org1MSPanchors.tx, Org2MSPanchors.tx

y Org3MSPanchors.tx como se puede observar en la imagen inferior. Esta es la muestra que los materiales criptográficos y los canales fueron creados exitosamente.

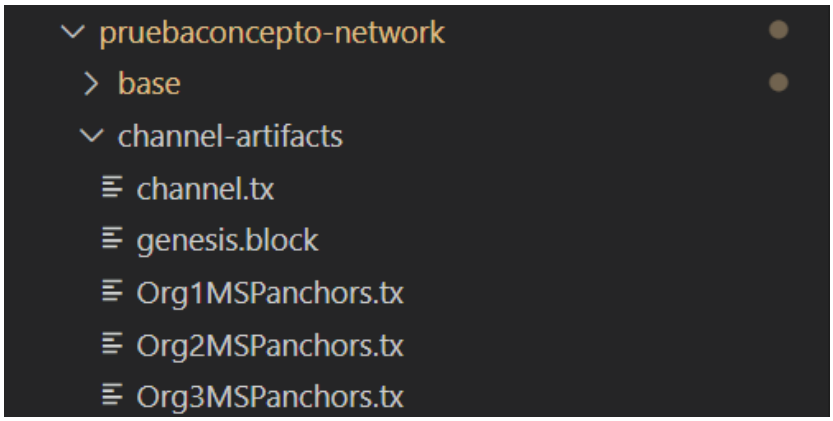


Ilustración 8 Resultado de Configtxgen (Creación propia)

Una vez se completen todos estos pasos, se da por terminado el primer paso de generación de material criptográfico.

8.3.2. Paso 2: Diseño de red

Lo que viene es diseñar la red y definir los servicios de Docker que se levantarán. Docker es un servicio de contenerización (Docker, 2022) que facilita la implementación del sistema en cualquier ambiente computacional. Para esta parte se utilizará Docker compose para ejecutar todos los servicios de esta aplicación.

8.3.2.1. Paso 2.1: Creación de la carpeta “base y archivo peer-base.yaml.

Este paso define el comportamiento principal de los peers de manera que, a partir de las definiciones que se coloque en este archivo, se puedan extender los servicios que se van a ofrecer.

En la carpeta del ejercicio, se crean la carpeta con los archivos necesarios como se ilustra en la imagen inferior

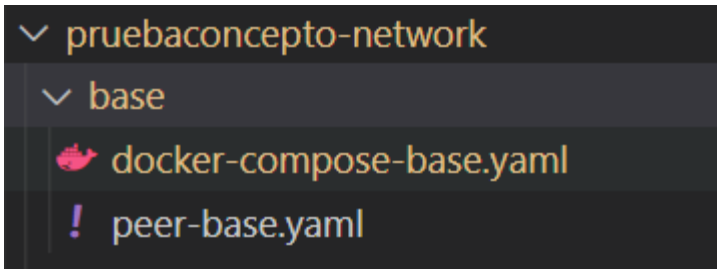


Ilustración 9 Contenido de carpeta "Base" (Creación propia)

`version: '2'` -> Se define la versión de docker compose

`services:`

`peer-base:` -> Se define el nombre del servicio, este puede tener cualquier nombre.

`image: hyperledger/fabric-peer:2.2.0` -> Es la imagen que lleva de Hyperledger Fabric. Esta imagen posee el programa y las configuraciones necesarias para montar un contenedor.

`environment:`

- `CORE_VM_ENDPOINT=unix:///host/var/run/docker.sock` -> Se define esta variable para que se pueda interactuar con el servicio contenerizado con docker.

- `CORE_VM_DOCKER_HOSTCONFIG_NETWORKMODE=pruebaconcepto-network_basic` -> Se define cuál es la red Docker que va a estar usando, para este ejercicio tendrá un nombre de pruebaconcepto-network_basic.

- `FABRIC_LOGGING_SPEC=INFO` -> Se define el nivel de log para este componente, tipo “información”.

`# - FABRIC_LOGGING_SPEC=DEBUG`

- `CORE_PEER_TLS_ENABLED=true` -> Se Indica si el peer se comunica en modo seguro, usando el protocolo TLS

- `CORE_PEER_GOSSIP_USELEADERELECTION=true` -> En esta sección se indican las propiedades del protocolo “Gossip”. El objetivo de este es conectarse el nodo más cercano para obtener de nuevo el estado actual en caso de alguna falla de la máquina.

- CORE_PEER_GOSSIP_ORGLEADER=false

- CORE_PEER_PROFILE_ENABLED=true

- CORE_PEER_TLS_CERT_FILE=/etc/hyperledger/fabric/tls/server.crt -> Se indica cuáles son los certificados que se estarán usando, para este caso será el "peer".

- CORE_PEER_TLS_KEY_FILE=/etc/hyperledger/fabric/tls/server.key -> Se indica cuáles son los certificados.

- CORE_PEER_TLS_ROOTCERT_FILE=/etc/hyperledger/fabric/tls/ca.crt -> Se indica dónde está el certificado raíz.

Nota importante: Para las indicaciones CORE_PEER_TLS_CERT_FILE, CORE_PEER_TLS_KEY FILY y CORE_PEER_TLS_ROOYCERT_FILE se debe indicar en qué directorio del Docker o de la máquina del contenedor va a estar ubicado. Lo que se debe hacer es montar los certificados que se tienen en carpeta host en el contenedor.

working_dir: /opt/gopath/src/github.com/hyperledger/fabric/peer -> Se especifica el directorio de trabajo por defecto donde se crea un contenedor

command: peer node start -> Se ejecuta un comando "peer node start" que levanta el fabric-peer:2.2.0.

8.3.2.2. Paso 2.2: Crear archivo Docker-compose-base.yaml

Este archivo ayuda a definir una red blockchain genérica, en otras palabras, son las definiciones de alto nivel de cómo se desea construir la red blockchain.

```
version: '2'
```

Sección 1: Activación de servicios, entorno, TLS y Volúmenes para el orderer

En esta parte se define el servicio de ordenamiento. Para el ejercicio solamente se activará una estancia, no obstante, se debe tener en cuenta que en un ambiente productivo se deben activar varias estancias para que este pueda trabajar en las diferentes máquinas y garantizar la disponibilidad de la red.

```
services:
```

`orderer.test.com:` -> Se define el servicio de ordenamiento

`container_name: orderer.test.com` -> Se define el nombre

`image: hyperledger/fabric-orderer:2.2.0` -> Se define que se va a utilizar la imagen de orderer

`environment:`

- `ORDERER_GENERAL_LOGLEVEL=debug` -> Se define el nivel de Log.

- `ORDERER_GENERAL_LISTENADDRESS=0.0.0.0` -> Se define la dirección de escucha

- `ORDERER_GENERAL_GENESISMETHOD=file` -> Se define el método del bloque génesis, el cual es tipo archivo

- `ORDERER_GENERAL_GENESISFILE=/var/hyperledger/orderer/orderer.genesis.block` -> Indica dónde está el bloque génesis.

- `ORDERER_GENERAL_LOCALMSPID=OrdererMSP` -> Se indica el ID de la organización

- `ORDERER_GENERAL_LOCALMSPDIR=/var/hyperledger/orderer/msp` -> Se indica el directorio de la msp.

- `ORDERER_GENERAL_TLS_ENABLED=true`

- `ORDERER_GENERAL_TLS_PRIVATEKEY=/var/hyperledger/orderer/tls/server.key` -> Se indica el directorio en donde está la llave privada

- ORDERER_GENERAL_TLS_CERTIFICATE=/var/hyperledger/orderer/tls/server.crt -> Se indica el directorio donde está el certificado para tls

- ORDERER_GENERAL_TLS_ROOTCAS=[/var/hyperledger/orderer/tls/ca.crt] -> Se indica el directorio donde está el Rootcas, o sea la autoridad certificadora raíz

working_dir: /opt/gopath/src/github.com/hyperledger/fabric -> Se indica el directorio de trabajo que en este caso es el programa de "orderer".

command: orderer

volumes:

- ../channel-artifacts/genesis.block:/var/hyperledger/orderer/orderer.genesis.block -> Indica en dónde se debe montar el contenedor en el directorio.

- ../crypto-config/ordererOrganizations/test.com/orderers/orderer.test.com/msp:/var/hyperledger/orderer/msp, el directorio ../crypto-config/ordererOrganizations/test.com/orderers/orderer.test.com/msp *se va a montar en* /var/hyperledger/orderer/msp

- ../crypto-config/ordererOrganizations/test.com/orderers/orderer.test.com/tls/:/var/hyperledger/orderer/tls

ports:

- 7050:7050 -> Se indican los puertos locales de la máquina host para que haga reenvío al puerto del contenedor 7050.

Sección 5: Configuración de peers

Para esta sección, se hace la configuración necesaria para cada uno de los peers.

`peer0.org1.test.com:` -> Se indica el nombre del servicio

`container_name: peer0.org1.test.com` -> Se define el nombre del contenedor

`extends:` -> Se extiende el archivo peer-base. Heredando las definiciones de este ledger

`file: peer-base.yaml`

`service: peer-base`

`environment:`

`- CORE_PEER_ID=peer0.org1.test.com` -> Este es el identificador único del peer

`- CORE_PEER_ADDRESS=peer0.org1.test.com:7051` -> Se indica la dirección de dónde se va a escuchar, es decir, en dónde se va a levantar dentro del contenedor 7051.

`- CORE_PEER_GOSSIP_EXTERNALENDPOINT=peer0.org1.test.com:7051` -> Se identifica el protocolo gossip para pasar transacciones entre los peers

`- CORE_PEER_GOSSIP_BOOTSTRAP=peer0.org1.test.com:7051` -> Se indica el bootstrap de peer

`- CORE_PEER_LOCALMSPID=Org1MSP` -> Se indica el nombre de a quién le pertenece este peer, el cual está definido en configtx.yaml, en partes de organizaciones Org1MSP

`volumes:` -> En esta parte se monta el material criptográfico que se creó para este peer. Es decir, en el archivo peer-base.yaml (CORE_PEER_TLS...), Lo que se hace es homologar el volumen /etc/hyperledger/fabric... del contenedor versus el directorio local donde se tiene el material criptográfico

`- /var/run/:/host/var/run/`

- ../crypto-config/peerOrganizations/org1.test.com/peers/peer0.org1.test.com/msp:/etc/hyperledger/fabric/msp -> Esto indica que lo que está dentro de la carpeta crypto.config/…/msp se montará en /etc/hyperledgerfabric/msp

- ../crypto-config/peerOrganizations/org1.test.com/peers/peer0.org1.test.com/tls:/etc/hyperledger/fabric/tls -> Indica dónde se van a montar los archivos ca.crt, server.crty server.key

ports: -> Se Define un conjunto de puertos que va a ser homologados.

- 7051:7051 -> Esto se entiende como si dentro del contenedor se levanta el puerto 7051, este va a ser homologado en el puerto de la máquina host.

- 7053:7053 -> Se entiende como realizar un match con la máquina host

Posteriormente se procede con la configuración para cada uno de los peers, se manera similar, así:

```
  peer0.org2.test.com:
    container_name: peer0.org2.test.com
    extends:
      file: peer-base.yaml
      service: peer-base
    environment:
      - CORE_PEER_ID=peer0.org2.test.com
      - CORE_PEER_ADDRESS=peer0.org2.test.com:7051
      - CORE_PEER_GOSSIP_EXTERNALENDPOINT=peer0.org2.test.com:7051
```

```
      - CORE_PEER_GOSSIP_BOOTSTRAP=peer0.org2.test.com:7051
      - CORE_PEER_LOCALMSPID=Org2MSP
    volumes:
      - /var/run/:/host/var/run/
      - ../crypto-config/peerOrganizations/org2.test.com/peers/peer0.org2.test.com/msp:/etc/hyperledger/fabric/msp
      - ../crypto-config/peerOrganizations/org2.test.com/peers/peer0.org2.test.com/tls:/etc/hyperledger/fabric/tls
    ports:
      - 8051:7051
      - 8053:7053
```

-> si se quisiera colocar en el ámbito productivo, se debe distribuir solamente esta información de cada organización, en este caso levantar información de material criptográfico de la segunda organización

ports: ->

- 8051:7051-> Todo lo que llega al puerto 8051 se reenviará al contenedor que se está levantando

- 8053:7053 -> Todo lo que llegue al puerto 8053 se reenviará con una copia al puerto 7053

```
  peer0.org3.test.com:
    container_name: peer0.org3.test.com
    extends:
      file: peer-base.yaml
      service: peer-base
    environment:
      - CORE_PEER_ID=peer0.org3.test.com
```

```
        - CORE_PEER_ADDRESS=peer0.org3.test.com:7051
        -
CORE_PEER_GOSSIP_EXTERNALENDPOINT=peer0.org3.test.com:7051
        - CORE_PEER_GOSSIP_BOOTSTRAP=peer0.org3.test.com:7051
        - CORE_PEER_LOCALMSPID=Org3MSP
      volumes:
          - /var/run/:/host/var/run/
          - ../crypto-
config/peerOrganizations/org3.test.com/peers/peer0.org3.test.com/
msp:/etc/hyperledger/fabric/msp
          - ../crypto-
config/peerOrganizations/org3.test.com/peers/peer0.org3.test.com/
tls:/etc/hyperledger/fabric/tls
      ports:
        - 9051:7051
        - 9053:7053
```

8.3.3. Paso 3: Contenedores Docker

Se deben tener en cuenta los siguientes parámetros para poder realizar de manera correcta la operación:

- Dentro de la máquina host del computador se activarán tres (3) contenedores. Posteriormente se activará un programa, escrito en el lenguaje de programación "Go", llamado peer. Este programa contiene la lógica y la configuración para poder realizar transacciones. Para este ejercicio, todos los peers se encontrarán en una máquina, aunque para un escenario de producción, deberían estar en máquinas diferentes.

- Para lograr que los contenedores puedan comunicarse, se necesita una red docker. Se utiliza un driver que permite crear una red que se llamará testnetwork_basic.
- La red Docker se debe definir en el archivo docker-compose-cli para que esta pueda crearse.
- Cada contenedor tendrá su propia dirección ip (internet protocol). Los contenedores siempre deben realizar las transacciones utilizando la red de Docker. Para este caso se utilizarán en el puerto 7051.

8.3.3.1. Paso 3.1: Definir couchdb creando el archivo docker-compose-cli-couchdb.yaml

```
version: '2'
```

`networks:` -> Si esta casilla se deja en blanco, va a coger por defecto el nombre de la red del directorio que se llama test-network basic, que se encuentra en peer_base.yaml/services

```
 basic:
```

Sección 1: Servicios

El primer servicio que se levanta es el servicio de ordenamiento

```
services:

  orderer.test.com:

    extends:

      file:   base/docker-compose-base.yaml
```
-> En esta parte se extienden las configuraciones creadas en el archivo base.

```
      service: orderer.test.com

    container_name: orderer.test.com

    networks:
```

`- basic` -> En esta parte se agrega este servicio a la red basic. Si no se agrega, se activa el contenedor sin conexión a la red y nadie lo podrá ver. Para poder realizar

transacciones, un servicio debe pertenecer a la misma red Docker; de lo contrario, no se ejecutará correctamente.

Sección 2: Peers

En esta parte de peers, se indica de manera explícita que se utilizará la base de datos Couchdb para los estados del peer. Couchdb es una base de datos NoSQL, de código abierto, desarrollada por la fundación apache y utilizada como base de datos en esta prueba de concepto.

`peer0.org1.test.com:` -> Se crea un servicio con el mismo nombre

`container_name: peer0.org1.test.com` -> Se indica el nombre del contenedor

`extends:` -> Se extiende lo que se definió en el archivo base.

`file: base/docker-compose-base.yaml`

`service: peer0.org1.test.com`

Sección 3: Entorno

En esta sección se definen las variables de ambiente utilizadas.

`Environment:`

`- CORE_LEDGER_STATE_STATEDATABASE=CouchDB` -> Se Indica que la base de datos estará en couchdb

`- CORE_LEDGER_STATE_COUCHDBCONFIG_COUCHDBADDRESS=couchdb0:5984` -> Se indica la dirección del couchdb. Esta dirección es la dirección dentro de la red de docker. Docker provee un DMS que permite invocar los servicios a través de su nombre

`- CORE_LEDGER_STATE_COUCHDBCONFIG_USERNAME=admin` -> Se indica usuario

`- CORE_LEDGER_STATE_COUCHDBCONFIG_PASSWORD=adminpw` -> Se indica contraseña

depends_on: -> Se indica de manera clara que un servicio espere que otros servicios se hayan activado correctamente para que este se active. Esto ayuda a garantizar el orden apropiado en la activación, de lo contrario puede producir un error

- orderer.test.com -> Primero se asegura que se active correctamente el orderer.test.com

- couchdb0 -> Luego se activa el Couchdb

networks: -> Se indica de manera clara que esté dentro de la red basic

- basic

Para el resto de los peers, se debe duplicar la información que se acaba de explicar en la parte superior. El único cambio sería en los nombres del Couchdb.

```
  peer0.org2.test.com:
    container_name: peer0.org2.test.com
    extends:
      file:  base/docker-compose-base.yaml
      service: peer0.org2.test.com
    environment:
      - CORE_LEDGER_STATE_STATEDATABASE=CouchDB
      -
CORE_LEDGER_STATE_COUCHDBCONFIG_COUCHDBADDRESS=couchdb1:5984
      - CORE_LEDGER_STATE_COUCHDBCONFIG_USERNAME=admin
      - CORE_LEDGER_STATE_COUCHDBCONFIG_PASSWORD=adminpw
    depends_on:
      - orderer.test.com
```

```
      - couchdb1
    networks:
      - basic
  peer0.org3.test.com:
    container_name: peer0.org3.test.com
    extends:
      file:  base/docker-compose-base.yaml
      service: peer0.org3.test.com
    environment:
      - CORE_LEDGER_STATE_STATEDATABASE=CouchDB
      -
CORE_LEDGER_STATE_COUCHDBCONFIG_COUCHDBADDRESS=couchdb2:5984
      - CORE_LEDGER_STATE_COUCHDBCONFIG_USERNAME=admin
      - CORE_LEDGER_STATE_COUCHDBCONFIG_PASSWORD=adminpw
    depends_on:
      - orderer.test.com
      - couchdb2
    networks:
      - basic
```

A continuación, se procede a crear el servicio para la autoridad de certificación (CA)

```
  ca.org1.test.com:
    image: hyperledger/fabric-ca:1.4.8
```
-> Se escoge la imagen de CA

```
    environment:
```

- FABRIC_CA_HOME=/etc/hyperledger/fabric-ca-server -> Se define un home

- FABRIC_CA_SERVER_CA_NAME=ca.org1.test.com -> Se define el nombre del servidor

- FABRIC_CA_SERVER_TLS_ENABLED=true -> Habilitar TLS

- FABRIC_CA_SERVER_TLS_CERTFILE=/etc/hyperledger/fabric-ca-server-config/ca.org1.test.com-cert.pem ->Se indica cuáles archivos criptográficos se van a usar

- FABRIC_CA_SERVER_TLS_KEYFILE=/etc/hyperledger/fabric-ca-server-config/priv_sk -> Indicar cuáles archivos criptográficos se van a usar

- FABRIC_CA_SERVER_CA_CERTFILE=/etc/hyperledger/fabric-ca-server-config/ca.org1.test.com-cert.pem -> Indicar cuáles archivos criptográficos se van a usar

- FABRIC_CA_SERVER_CA_KEYFILE=/etc/hyperledger/fabric-ca-server-config/priv_sk -> Indicar cuáles archivos criptográficos se van a usar

ports: -> Indicar cuáles son los puertos que se van a levantar

- "7054:7054"

command: sh -c 'fabric-ca-server start -b admin:adminpw' -> Este es el comando necesario para activar, cuando se cree el servidor, el fabric-ca-server.

volumes: -> Se indica el material criptográfico

- ./crypto-config/peerOrganizations/org1.test.com/ca/:/etc/hyperledger/fabric-ca-server-config

container_name: ca.org1.test.com

networks: -> Se agrega dentro de la red basic

```
      - basic
```

A continuación, se define el contenedor que permite la interacción por línea de comandos (command line interface – CLI), el cual se llama "fabric-tools". Este se personaliza con la identidad y los certificados del peer de la primera organización de manera que los programas que se ejecutan como miembro de la red de hyperledger fabric, utilizan la identidad del primer peer de la primera org (Org1).

```
  cli:
    container_name: cli
    image: hyperledger/fabric-tools:2.2
    tty: true
    stdin_open: true
    environment:
      - GOPATH=/opt/gopath
      - CORE_VM_ENDPOINT=unix:///host/var/run/docker.sock
      - FABRIC_LOGGING_SPEC=DEBUG
      - CORE_PEER_ID=cli
      # ---CHANGED--- peer0 from Org1 is the default for this CLI container
      - CORE_PEER_ADDRESS=peer0.org1.test.com:7051
```

-> Se hace énfasis de que se va a utilizar la identidad del peer0 de la organización 1

```
      -  CORE_PEER_LOCALMSPID=Org1MSP
```

-> Se indica que es de la primera organización

```
      - CORE_PEER_TLS_ENABLED=true
```

Se utilizan los materiales criptográficos de la organización 1

```
CORE_PEER_TLS_CERT_FILE=/opt/gopath/src/github.com/hyperledger/fa
```

```
bric/peer/crypto/peerOrganizations/org1.test.com/peers/peer0.org1
.test.com/tls/server.crt

CORE_PEER_TLS_KEY_FILE=/opt/gopath/src/github.com/hyperledger/fab
ric/peer/crypto/peerOrganizations/org1.test.com/peers/peer0.org1.
test.com/tls/server.key

CORE_PEER_TLS_ROOTCERT_FILE=/opt/gopath/src/github.com/hyperledge
r/fabric/peer/crypto/peerOrganizations/org1.test.com/peers/peer0.
org1.test.com/tls/ca.crt


CORE_PEER_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fa
bric/peer/crypto/peerOrganizations/org1.test.com/users/Admin@org1
.test.com/msp

    working_dir:
/opt/gopath/src/github.com/hyperledger/fabric/peer

    command: /bin/bash
```

Sección 4: Volúmenes

En la sección de los volúmenes se identifica un directorio en donde se va a estar generando el chaincode, ubicado en la carpeta "chaincode", por lo que todo lo que se monte de manera local será montado en la carpeta asignada. El chaincode es código que se ejecuta en el blockchain y es el responsable de inicializar y gestionar el estado del *ledger* a medida que las aplicaciones realizan transacciones.

```
 volumes:

        - /var/run/:/host/var/run/

        - ./../chaincode/:/opt/gopath/src/github.com/chaincode

        -                                                    ./crypto-
config:/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/
```

- ./channel-artifacts:/opt/gopath/src/github.com/hyperledger/fabric/peer/channel-artifacts -> Montar configuración del canal, anchor peers y bloque génesis

depends_on: -> Se Indica que este proceso no se levante hasta que no se levanten primero los orderer y los 3 org peers

- orderer.test.com
- peer0.org1.test.com
- peer0.org2.test.com
- peer0.org3.test.com

networks: -> Se Indica que se va a trabajar dentro de la red de hyperledger fabric basic

- basic

A continuación, se definirán los parámetros para las bases de datos couchdb.

couchdb0:

image: couchdb:3.1 -> Se define la imagen

environment:

- COUCHDB_USER=admin -> Usuario
- COUCHDB_PASSWORD=adminpw -> Contraseña

ports:

- 5984:5984 -> Definir en qué puerto se va a levantar,

container_name: couchdb0

networks:

- basic -> Hacer que este servicio funcione en la red basic

```
  couchdb1:
    image: couchdb:3.1
    environment:
      - COUCHDB_USER=admin
      - COUCHDB_PASSWORD=adminpw
    ports:
```

- `5985:5984` -> La máquina host no puede utilizar el mismo puerto, por lo tanto cambia a 5985 puesto que este será el encargado de reenviar a todas las peticiones del couchdb1

```
    container_name: couchdb1
    networks:
      - basic
  couchdb2:
    image: couchdb:3.1
    environment:
      - COUCHDB_USER=admin
      - COUCHDB_PASSWORD=adminpw
    ports:
      - 5986:5984
    container_name: couchdb2
    networks:
      - basic
```

8.3.3.2. Paso 3.2 Generar despliegue a través de Portainer para administrar Docker

Portainer es una aplicación que proporciona una interfaz gráfica (GUI) para la gestion de diferentes contenedores, incluidos Kubernetes y docker. Para genera el despliegue se utiliza el siguiente comando: `docker volume create portainer_data`

Respuesta: `portainer_data`

Posteriormente se despliega a través del siguiente comando: `docker run -d -p 8000:8000 -p 9000:9000 -v /var/run/docker.sock:/var/run/docker.sock -v potainer_data:/data portainer/portainer`

Una vez se ejecuten los comandos asignados, todo está preparado para abrir Portainer dentro de la máquina.

8.3.3.3. Paso 3.3: Activar Docker compose

En la carpeta del ejercicio se van a definir unas variables que van a ayudar a activar correctamente la red:

```
darroya8@Daniel:~/EjercicioTesis/Tesis-network$ export CHANNEL_NAME=marketplace
```

```
darroya8@Daniel:~/EjercicioTesis/Tesis-network$ export VERBOSE=false
```

```
darroya8@Daniel:~/EjercicioTesis/Tesis-network$ export FABRIC_CFG_PATH=$PWD
```

Siguiente paso: Levantar couchdb

Transacción*:* `CHANNEL_NAME=$CHANNEL_NAME docker-compose -f docker-compose-cli-couchdb.yaml up -d`

Explicación

`CHANNEL_NAME`(Definir variable channel name)`=$CHANNEL_NAME docker-compose` (Llamar comando de Docker compose) `-f` (el – f indica cuál es el archivo que se va a levantar) `docker-compose-cli-couchdb.yaml up` (Con up se indica que se levante)`-d` (Con -d se indica que se mantenga en background)

Respuesta:

```
darroya8@Daniel:~/pruebaconcepto/pruebaconcepto-network$ CHANNEL_NAME=$CHANNEL_NAME docker-compose -f docker-compose-cli-couchdb.yaml
 up -d

Starting orderer.test.com ... done
Starting couchdb2         ... done
Starting couchdb1         ... done
Starting couchdb0         ... done
Starting ca.org1.test.com    ... done
Starting peer0.org3.test.com ... done
Starting peer0.org1.test.com ... done
Starting peer0.org2.test.com ... done
Starting cli                 ... done
```

Ilustración 10 Respuesta de activación Docker-compose (Creación propia)

8.3.3.4. Paso 3.4: Entrar a portainer

Dar clic a la pestaña de contenedores y se podrán ver los contenedores que se levantaron.

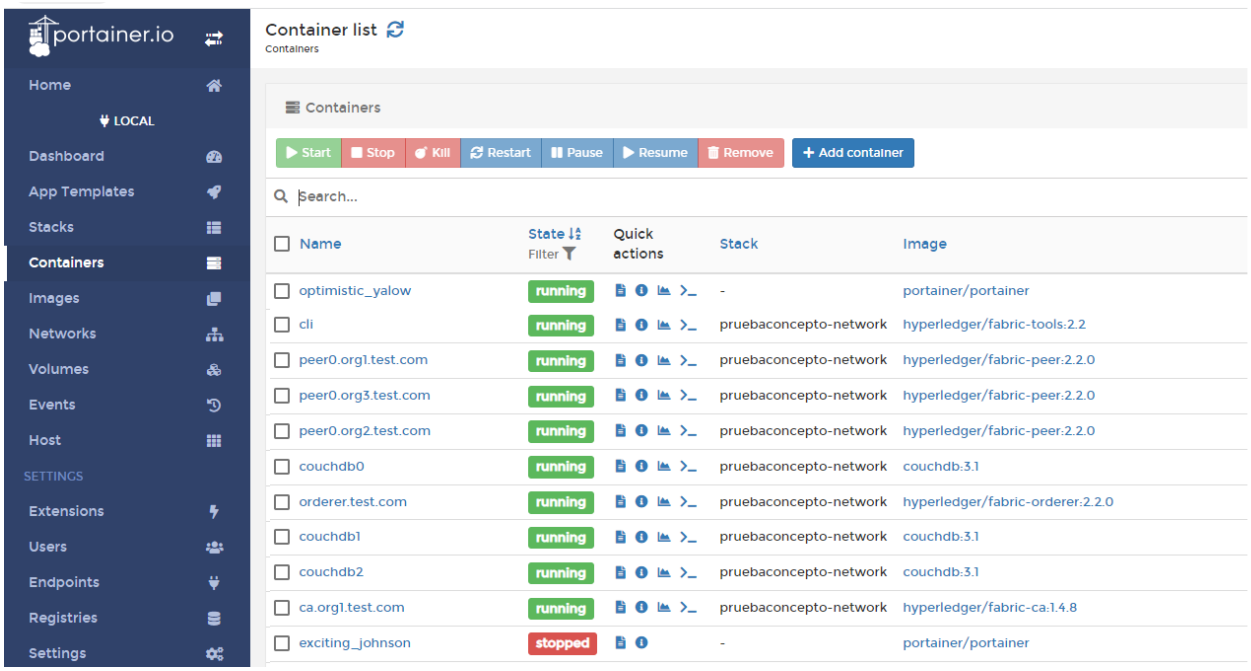


Ilustración 11 Portada de Portainer (Creación Propia)

8.3.3.5. Paso 3.5: Operaciones desde la línea de comandos (CLI)

Se debe dirigir hacia el comando cli y dar clic al ícono que se muestra en la imagen.

Ilustración 12 consola cli (Creación propia)

Una vez dado clic, se abrirá la siguiente consola:

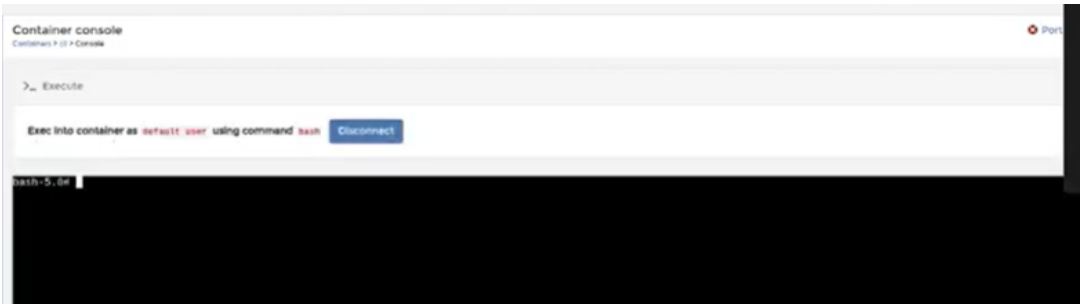


Ilustración 13 Consola de CLI (Creación propia)

En esta sección se creará el canal utilizando la configuración definida en channel.tx. Este archivo tiene las configuraciones necesarias para configurar el canal.

Se ejecutan los siguientes comandos:

```
export CHANNEL_NAME=marketplace
```

Comando peer

```
peer channel create -o orderer.test.com:7050 -c $CHANNEL_NAME -f ./channel-artifacts/channel.tx --tls true --cafile /opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrganizations/test.com/orderers/orderer.test.com/msp/tlscacerts/tlsca.test.com-cert.pem
```

Explicación

`peer channel create -o` (Indicar servicio de ordenamiento que se indicó en Docker-compose-yaml)`orderer.test.com:7050 -c` (Con el -c se indica lo que ya había indicado en la variable channel name)`$CHANNEL_NAME -f` (Con el -f se indica dónde está el directorio donde está la transacción del canal) `./channel-artifacts/channel.tx --tls true` (Se debe ejecutar esto dentro de un protocolo

seguro TLS) `--cafile` (Se indica dónde está el directorio cafile) `/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrganizations/test.com/orderers/orderer.test.com/msp/tlscacerts/tlsca.test.com-cert.pem`

Respuesta:

Ilustración 14 Respuesta de Peer channel Create (Creación Propia)

Como resultado se creó un bloque llamado Marketplace.block. Este es el bloque del canal.

Nota: En caso de que salgan errores, para efectos de la prueba de concepto, se deben borrar todos los contenedores del docker utilizando los siguientes comandos:

```
docker kill $(docker ps -q)
docker rm $(docker ps -a -q)
```

Siguiente paso: hacer que la organización sea parte del canal

Transacción: `peer channel join -b marketplace.block` (Como se está utilizando container cli, el cual está personalizado para conectarse con una organización, no se necesita especificar cuál organización es)

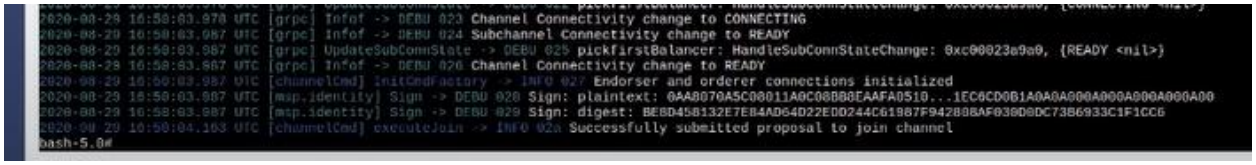


Ilustración 15 Respuesta de Peer channel join (Creación propia)

Ilustración 16 Visualización de CouchDb (Creación Propia)

Siguiente paso: unir las otras 2 organizaciones (Para unir a estas dos, se necesita cambiar la personalización del contenedor, mediante variables de ambiente, para que interactúe con la organización deseada, antes de ejecutar el comando)

```
CORE_PEER_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org2.test.com/users/Admin@org2.test.com/msp CORE_PEER_ADDRESS=peer0.org2.test.com:7051 CORE_PEER_LOCALMSPID="Org2MSP" CORE_PEER_TLS_ROOTCERT_FILE=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org2.test.com/peers/peer0.org2.test.com/tls/ca.crt peer channel join -b marketplace.block
```

Explicación

`CORE_PEER_MSPCONFIGPATH` (Indicar la información de la organización) `=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org2.test.com/users/Admin@org2.test.com/msp` (Suministrar identidades) `CORE_PEER_ADDRESS=peer0.org2.test.com:7051 CORE_PEER_LOCALMSPID="Org2MSP" CORE_PEER_TLS_ROOTCERT_FILE=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org2.test.com/peers/peer0.org2.test.com/tls/ca.crt` (Se indican certificados) `peer channel join -b marketplace.block`

```
CORE_PEER_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org3.test.com/users/Admin@org3.test.com/msp CORE_PEER_ADDRESS=peer0.org3.test.com:7051 CORE_PEER_LOCALMSPID="Org3MSP" CORE_PEER_TLS_ROOTCERT_FILE=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org3.test.com/peers/peer0.org3.test.com/tls/ca.crt peer channel join -b marketplace.block
```

Siguiente paso: Preparar organizaciones para que tengan configurado el anchor peer correspondiente y tengan acceso a los servicios especiales.

Comandos:

```
peer channel update -o orderer.test.com:7050 -c $CHANNEL_NAME -f ./channel-artifacts/Org1MSPanchors.tx --tls --cafile /opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrganizations/test.com/orderers/orderer.test.com/msp/tlscacerts/tlsca.test.com-cert.pem

CORE_PEER_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org2.test.com/users/Admin@org2.test.com/msp CORE_PEER_ADDRESS=peer0.org2.test.com:7051 CORE_PEER_LOCALMSPID="Org2MSP" CORE_PEER_TLS_ROOTCERT_FILE=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org2.test.com/peers/peer0.org2.test.com/tls/ca.crt peer channel update -o orderer.test.com:7050 -c $CHANNEL_NAME -f ./channel-artifacts/Org2MSPanchors.tx --tls --cafile /opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrganizations/test.com/orderers/orderer.test.com/msp/tlscacerts/tlsca.test.com-cert.pem

CORE_PEER_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org3.test.com/users/Admin@org3.test.com/msp CORE_PEER_ADDRESS=peer0.org3.test.com:7051 CORE_PEER_LOCALMSPID="Org3MSP" CORE_PEER_TLS_ROOTCERT_FILE=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org3.test.com/peers/peer0.org3.test.com/tls/ca.crt peer channel update -o orderer.test.com:7050 -c $CHANNEL_NAME -f ./channel-artifacts/Org3MSPanchors.tx --tls --cafile /opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/orderer
```

`Organizations/test.com/orderers/orderer.test.com/msp/tlscacerts/tlsca.test.com-cert.pem`

En este punto se finaliza la configuración de los diferentes componentes de la red de blockchain utilizando Docker y, por lo tanto, la red se encuentra lista para implementar contratos inteligentes.

8.3.4. Paso 4: Contratos inteligentes (Smart contracts)

Sección 1: Generación del código del Smart Contract

En esta sección se crea el contrato inteligente, el cual se programará en el lenguaje de programación “Go”. En Go es elemental definir una estructura base sobre el cual uno va a proveer un control y las funciones, adicionalmente se debe definir una estructura que implementa el “Contractapi” y todas las funciones que lo componen.

Para comenzar, se procede con la creación de una subcarpeta llamada “shipping” y posteriormente, en dicha carpeta se crea un archivo llamado “shipping.go”.

El alcance del contrato inteligente será un que este permita crear y consultar el archivo del blockchain. A continuación se explica el paso a paso del archivo “shipping.go”.

`package main` -> Se define un paquete

Con el valor "import" se define unas librerías necesarias para poder ejecutar el Smart contract.

```
import (
```

`"encoding/json"` -> librería 1

`"fmt"` -> Librería 2

`"github.com/hyperledger/fabric-contract-api-go/contractapi"` ->Librería 3: Módulo de control API para poder crear el contrato inteligente. Abstrae el uso y la implementación de “init” e “invoke”, adicionalmente, ofrece la posibilidad de implementar tantas funciones como queramos y esas funciones se exponen de manera directa.

)

`type SmartContract struct {` -> Se define el nombre, para este caso se llamará “smartContract”

```
	contractapi.Contract ->
```

}

A continuación, se definirán las funciones del contrato inteligente:

`type Shipping struct {`-> Se define los detalles básicos de la estructura

```
	Adate string `json:"adate"`
	Bbuyer  string `json:"bbuyer"`
	Cseller string `json:"cseller"`
	Dterms string `json:"dterms"`
	Ematerial string `json:"ematerial"`
	Fhscode string `json:"fhscode"`
	Gquantity string `json:"gquantity"`
	Hunitprice string `json:"hunitprice"`
	Ipricetotal string `json:"ipricetotal"`
	Jpaymentterms string `json:"jpaymentterms"`
	Kinvoice string `json:"kinvoice"`
	Lbl string `json:"lbl"`
	Mstatus string `json:"mstatus"`
	Nexitfactorydate string `json:"nexitfactorydate"`
	Oetd string `json:"oetd"`
	Prtd string `json:"prtd"`
```

```
	Qeta string `json:"qeta"`
	Rrta string `json:"rrta"`
	Scustomsdate string `json:"scustomsdate"`
	Tarrivaldate string `json:"tarrivaldate"`
}
```

En esta parte se crean las funciones que permita guardar en el blockchain la información de las transacciones.

```
func (s *SmartContract Set (ctx contractapi.TransactionContextInterface, shippingId string, adate string, bbuyer string, cseller string, dterms string, ematerial string, fhscode string, gquantity string, hunitprice string, ipricetotal string, jpaymentterms string, kinvoice string, lbl string, mstatus string, nexitfactorydate string, oetd string, prtd string, qeta string, rrta string, scustomsdate string, tarrivaldate string) error {
```

Explicación

`func (s *SmartContract)` *(*Parte de la estructura del smart contract) `Set` (número de posición)`(ctx contractapi.TransactionContextInterface`, (Definir un conjunto de propiedades que van a ser parte de la data que pide el cliente al momento de ejecutar el contrato inteligente) `shippingId string, adate string, bbuyer string, cseller string, dterms string, ematerial string, fhscode string, gquantity string, hunitprice string, ipricetotal string, jpaymentterms string, kinvoice string, lbl string, mstatus string, nexitfactorydate string, oetd string, prtd string, qeta string, rrta string, scustomsdate string, tarrivaldate string) error {`

En esta parte se arma el activo shipping

```
	shipping := Shipping{
```

```
      Adate: adate,
      Bbuyer:  bbuyer,
      Cseller: cseller,
      Dterms: dterms,
      Ematerial: ematerial,
      Fhscode: fhscode,
      Gquantity: gquantity,
      Hunitprice: hunitprice,
      Ipricetotal: ipricetotal,
      Jpaymentterms: jpaymentterms,
      Kinvoice: kinvoice,
      Lbl: lbl,
      Mstatus: mstatus,
      Nexitfactorydate: nexitfactorydate,
      Oetd: oetd,
      Prtd: prtd,
      Qeta: qeta,
      Rrta: rrta,
      Scustomsdate: scustomsdate,
      Tarrivaldate: tarrivaldate,
}
```

Es importante asegurarse que esta estructura se pueda transformar en un documento Json, y además se guarde como un conjunto de bytes para que esto sea compatible con lo definido para shipping state que es lo que va a quedar guardado en el blockchain.

Por esto, se debe proceder con lo siguiente:

```
shippingAsBytes, err := json.Marshal(shipping)
```

`shippingAsBytes` (declarar variable), `err := json.Marshal(shipping)` (Aquí lo que se hace es generar, a partir del elemento de shipping, una representación compacta en bytes que pueda ser almacenada en el blockchain.

```
if err != nil {
    fmt.Printf("Marshal error: %s", err.Error())
    return err
}
return ctx.GetStub().PutState (función que me permite guardar en el ledger)(shippingId, shippingAsBytes)
}
```

A continuación, se crea la función para la consulta:

```
func (s *SmartContract) Query(ctx contractapi.TransactionContextInterface, shippingId string) (*Shipping, error) {
```

`shippingAsBytes` (Se declara variable para poder proceder con el valor en bytes), `err := ctx.GetStub().GetState(shippingId)` -> Esto sirve para consultar el ledger

```
if err != nil {
```

return nil, fmt.Errorf("Failed to read from world state. %s", err.Error()) -> Validar no haya errores

}

if shippingAsBytes == nil { -> Validar si hay encontró el elemento buscado

return nil, fmt.Errorf("%s does not exist", shippingId)

}

shipping := new(Shipping) -> Se crea un nuevo elemento shipping vacío

err = json.Unmarshal(shippingAsBytes, shipping) -> Pasar de la representación en bytes de la red de bytes a la representación de go

if err != nil {

return nil, fmt.Errorf("Unmarshal error. %s", err.Error())

}

return shipping, nil

}

func main() { ->

chaincode, err := contractapi.NewChaincode(new(SmartContract)) -> Esta parte indica que se utilizará el API de los contratos para activar un nuevo chaincode .

Esta función define una estructura base de un fichero “Go” donde se va a escribir el contrato inteligente.

if err != nil {

```
            fmt.Printf("Error create shipping chaincode: %s",
err.Error())
            return
      }
      if err := chaincode.Start(); err != nil {
            fmt.Printf("Error starting shipping chaincode: %s",
err.Error())
      }
}
```

Otros datos importantes:

Es necesario tener un archivo llamado "go.mod", donde se indiquen cuáles son las versiones de los módulos que se están utilizando dentro del proyecto y se define la versión de go.

Igualmente, se debe tener un archivo llamado "go.sum", el cual se crea dentro de las transacciones posteriores y este indica que se van a usar exactamente las mismas librerías y compilaciones de esta red.

Sección 2: Despliegue del contrato inteligente

Para el despliegue del contrato inteligente se debe volver a la interfaz de comandos en línea CLI.

El primer paso es generar las siguientes variables de ambiente:

`export CHANNEL_NAME=marketplace` -> nombre del canal

`export CHAINCODE_NAME=shipping` -> Nombre del chaincode

`export CHAINCODE_VERSION=1` -> nombre de la versión

`export CC_RUNTIME_LANGUAGE=golang` -> Lenguaje de ejecución del contrato inteligente

`export CC_SRC_PATH="../../../chaincode/$CHAINCODE_NAME/"` -> permite definir el path del chaincode

`cat /opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrganizations/test.com/orderers/orderer.test.com/msp/tlscacerts/tlsca.test.com-cert.pem` -> Se define el certificado de la CA

`export ORDERER_CA=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrganizations/test.com/orderers/orderer.test.com/msp/tlscacerts/tlsca.test.com-cert.pem`

El siguiente paso consiste en empaquetar en un archivo en tar.gz el chaincode que luego se instalará en cada uno de los peers.

`peer lifecycle chaincode package ${CHAINCODE_NAME}.tar.gz --path ${CC_SRC_PATH} --lang ${CC_RUNTIME_LANGUAGE} --label ${CHAINCODE_NAME}_${CHAINCODE_VERSION} >&log.txt`

Al ejecutar el comando de arriba, se crea una carpeta shipping.tar.gz y se procede con las siguientes funciones: Lo primero que se realiza, es instalarse en cada una de las organizaciones.

`peer lifecycle chaincode install shipping.tar.gz`

Mensaje:

`shipping_1:d8f6879cd632a43e259fe09d8dc70c2049c790d3dd4f211ffd2c6320c9cfefe1` -> El hash permite identificar el chaincode. Debe ser el mismo en todas las organizaciones pues actúa como una especie de huella digital que cambia cada vez que el archivo es modificado. Si el hash cambia es porque el contrato ha sido modificado y lo invalida.

Para las otras organizaciones se debe especificar cuál organización debe ser el siguiente:

```
CORE_PEER_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org2.test.com/users/Admin@org2.test.com/msp CORE_PEER_ADDRESS=peer0.org2.test.com:7051 CORE_PEER_LOCALMSPID="Org2MSP" CORE_PEER_TLS_ROOTCERT_FILE=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org2.test.com/peers/peer0.org2.test.com/tls/ca.crt peer lifecycle chaincode install ${CHAINCODE_NAME}.tar.gz

shipping_1:d8f6879cd632a43e259fe09d8dc70c2049c790d3dd4f211ffd2c6320c9cfefe1

CORE_PEER_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org3.test.com/users/Admin@org3.test.com/msp CORE_PEER_ADDRESS=peer0.org3.test.com:7051 CORE_PEER_LOCALMSPID="Org3MSP" CORE_PEER_TLS_ROOTCERT_FILE=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org3.test.com/peers/peer0.org3.test.com/tls/ca.crt peer lifecycle chaincode install ${CHAINCODE_NAME}.tar.gz

shipping_1:d8f6879cd632a43e259fe09d8dc70c2049c790d3dd4f211ffd2c6320c9cfefe1
```

Cuando se realicen todos los pasos de la parte de arriba con sus mensajes respectivos, se puede confirmar que ya quedó instalado el contrato. Lo que sigue es establecer las políticas de aprobación que se hayan definido para el chaincode. Para este chaincode, la Org 1 y la Org 3 son capaces de aprobar transacciones, de firmar transacciones.

Se debe ejecutar lo siguiente:

Organización 1

```
peer lifecycle chaincode approveformyorg --tls --cafile $ORDERER_CA --channelID $CHANNEL_NAME --name $CHAINCODE_NAME --
```

```
version $CHAINCODE_VERSION --sequence 1 --waitForEvent --signature-policy "OR ('Org1MSP.peer', 'Org3MSP.peer')" --package-id shipping_1:d8f6879cd632a43e259fe09d8dc70c2049c790d3dd4f211ffd2c6320c9cfefe1
```

Respuesta:

```
txid [6634339f41efb9f451914dc777d11cf8a629b0372e3f8fe25f754ab24ab0e4cb] committed with status (VALID) at peer0.org1.test.com:7051
```

Organización 3

```
CORE_PEER_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org3.test.com/users/Admin@org3.test.com/msp CORE_PEER_ADDRESS=peer0.org3.test.com:7051 CORE_PEER_LOCALMSPID="Org3MSP" CORE_PEER_TLS_ROOTCERT_FILE=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org3.test.com/peers/peer0.org3.test.com/tls/ca.crt peer lifecycle chaincode approveformyorg --tls --cafile $ORDERER_CA --channelID $CHANNEL_NAME --name $CHAINCODE_NAME --version $CHAINCODE_VERSION --sequence 1 --waitForEvent --signature-policy "OR ('Org1MSP.peer', 'Org3MSP.peer')" --package-id shipping_1:d8f6879cd632a43e259fe09d8dc70c2049c790d3dd4f211ffd2c6320c9cfefe1
```

Respuesta:

```
txid [fb8f1add529a8ba522f777daf896f9b0714deb51859f3a2af8142836ad427199] committed with status (VALID) at peer0.org3.test.com:7051
```

El siguiente paso es realizar un “commit” que en otras palabras significa poner en marcha el chaincode

```
peer lifecycle chaincode commit -o orderer.test.com:7050 --tls --cafile $ORDERER_CA --peerAddresses peer0.org1.test.com:7051 --tlsRootCertFiles /opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org1.test.com/peers/peer0.org1.test.com/tls/ca.crt --peerAddresses peer0.org3.test.com:7051 --tlsRootCertFiles /opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org3.test.com/peers/peer0.org3.test.com/tls/ca.crt --channelID $CHANNEL_NAME --name $CHAINCODE_NAME --version $CHAINCODE_VERSION --sequence 1 --signature-policy "OR ('Org1MSP.peer', 'Org3MSP.peer')"
```

Respuesta:

```
2022-01-26 00:59:15.735 UTC [chaincodeCmd] ClientWait -> INFO 06a txid [1e4edb7446663dd654e50d13f18964366e1ea23cf2d80f56e9a8b8b0bb755b1d] committed with status (VALID) at peer0.org3.test.com:7051

2022-01-26 00:59:15.735 UTC [chaincodeCmd] ClientWait -> INFO 06b txid [1e4edb7446663dd654e50d13f18964366e1ea23cf2d80f56e9a8b8b0bb755b1d] committed with status (VALID) at peer0.org1.test.com:7051
```

Una vez todas estas transacciones se encuentren completas, ya se encuentra lista la red de Hyperledgerfabric para poder ejecutar contratos inteligentes.

8.3.5. Paso 5: Creación de transacciones en la red de blockchain en Hyperledger Fabric

Para este ejercicio se va a hacer una serie de 6 transacciones relacionadas con la trazabilidad de la carga desde el punto de inicio hasta su llegada para cumplir con la resolución 67 de 2016 de la DIAN, punto 6.3, la cual se debe cumplir para obtener la certificación OEA: *“Tener herramientas que le permitan garantizar la trazabilidad de la*

carga, desde el punto de llenado en el exterior hasta la sede del importador o el punto de distribución". En la imagen inferior, se ilustra cada una de las etapas que se van a incluir dentro de esta red para la importación de un contenedor de 20 pies de Cable de tracción de 12 milímetros desde el puerto de Busán, Corea del sur hasta la llegada de la mercancía a las instalaciones de una fábrica ubicada en la ciudad de Medellín, Colombia.

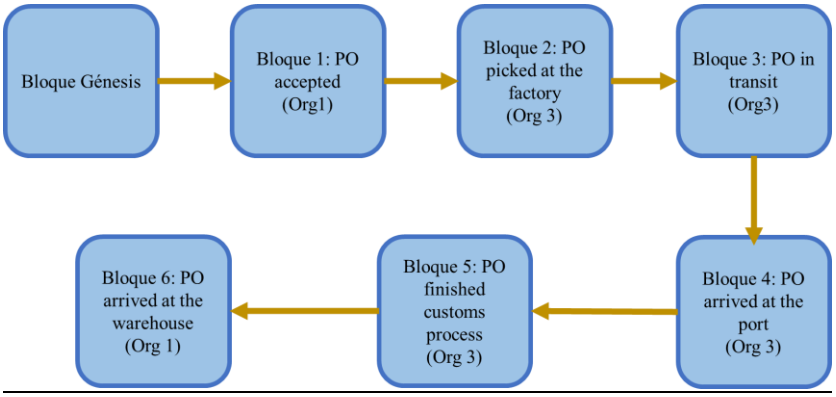


Ilustración 17 Transacciones de la prueba de concepto Creación propia

A continuación, se presenta el detalle de cada uno de los bloques

Tabla 3. Detalle de las transacciones

Función	**Definición**	**Bloque 1**	**Bloque 2**	**Bloque 3**	**Bloque 4**	**Bloque 5**	**Bloque 6**
	Organización creadora del bloque	Org1	Org 3	Org 3	Org 3	Org 3	Org 1
Adate	Fecha de ejecución	22/11/2021	21/01/2022	30/01/2022	05/03/2022	25/03/2022	28/03/2022
Bbuyer	Nombre del comprador	DanielCompany	DanielCompany	DanielCompany	DanielCompany	DanielCompany	DanielCompany

Cseller	Nombre del vendedor	SupplierCompany	SupplierCompany	SupplierCompany	SupplierCompany	SupplierCompany	SupplierCompany
Dterms	Incoterms	FOB	FOB	FOB	FOB	FOB	FOB
Ematerial	Nombre del material	12mm Wire rope	12mm Wire rope	12mm Wire rope	12mm Wire rope	12mm Wire rope	12mm Wire rope
Fhscode	Partida Arancelaria	80.50.20.00.00	80.50.20.00.00	80.50.20.00.00	80.50.20.00.00	80.50.20.00.00	80.50.20.00.00
Gquantity	Cantidad	32000	32000	32000	32000	32000	32000
Hunitprice	Valor Unitario (USD)	2	2	2	2	2	2
Ipricetotal	Valor total	64000	64000	64000	64000	64000	64000
Jpayment terms	Tipo de pago	90 days after BL	90 days after BL	90 days after BL	90 days after BL	90 days after BL	90 days after BL
Kinvoice	Factura		Invoice12345	Invoice12345	Invoice12345	Invoice12345	Invoice12345
Lbl	Número de BL			BLDAM123	BLDAM123	BLDAM123	BLDAM123
Mstatus	Estado de la órden de compra	PO accepted	PO picked at the factory	PO in transit	PO arrived at the port	PO finished customs process	PO arrived at the warehouse

Nexitfactorydate	Salida de la fábrica	21/01/2022	21/01/2022	21/01/2022	21/01/2022	21/01/2022
Oetd	Fecha estimada de envío desde puerto de Busán	28/01/2022	28/01/2022	28/01/2022	28/01/2022	28/01/2022
Prtd	Fecha real de envío desde puerto de Busán		30/01/2022	30/01/2022	30/01/2022	30/01/2022
Qeta	Fecha estimada de llegada a puerto de Buenaventura	04/03/2022	04/03/2022	04/03/2022	04/03/2022	04/03/2022
Rrta	Fecha real de llegada a puerto de Buenaventura			05/03/2022	05/03/2022	05/03/2022

Scustoms date	Fecha de nacionalización	25/03/2022	25/03/2022
Tarrivaldate	Fecha de llegada a planta		28/03/2022

Tabla 3 Detalle de las transacciones (Creación propia)

8.3.5.1. Paso 5.1: Invocar transacciones

Transacción 1

```
peer chaincode invoke -o orderer.test.com:7050 --tls --cafile $ORDERER_CA -C $CHANNEL_NAME -n $CHAINCODE_NAME -c '{"Args":["Set","PO:TEST1","22-11-2021","DanielCompany","SupplierCompany","FOB","12 mm Wire Rope","80.50.20.00.00","32000","2 USD","64000USD","90 days after BL","","","PO accepted","","","","","","","","]}'
```

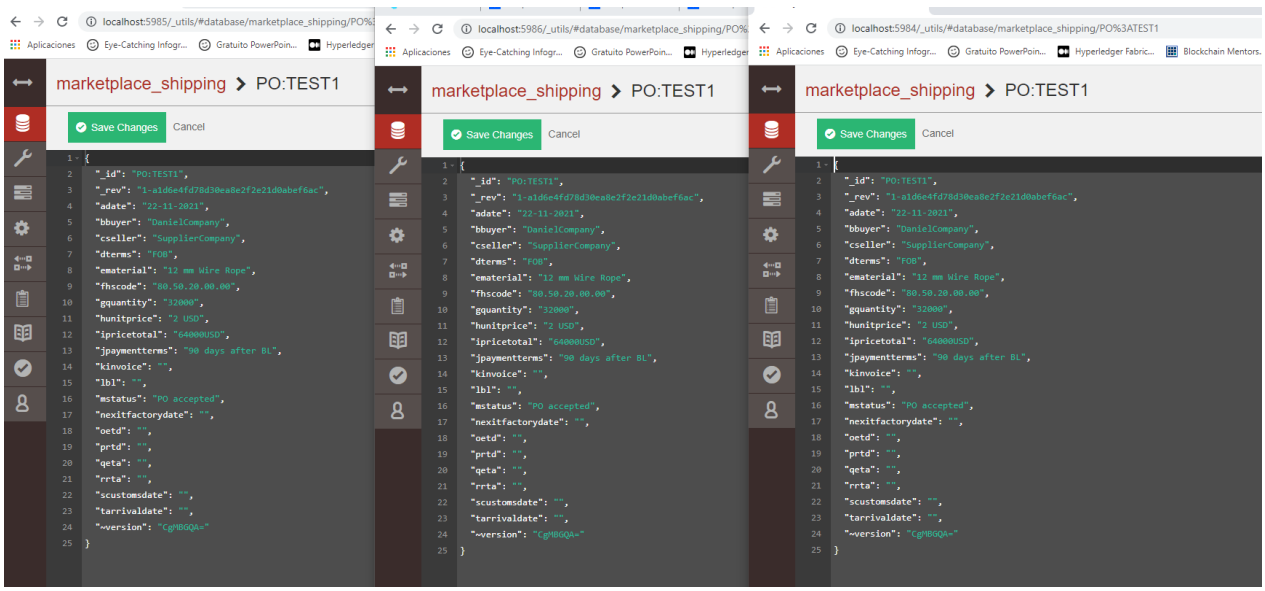



Ilustración 18 Transacción 1 (Creación Propia)

Transacción 2

```
CORE_PEER_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org3.test.com/users/Admin@org3.test.com/msp CORE_PEER_ADDRESS=peer0.org3.test.com:7051 CORE_PEER_LOCALMSPID="Org3MSP" CORE_PEER_TLS_ROOTCERT_FILE=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org3.test.com/peers/peer0.org3.test.com/tls/ca.crt peer chaincode invoke -o orderer.test.com:7050 --tls --cafile $ORDERER_CA -C $CHANNEL_NAME -n $CHAINCODE_NAME -c '{"Args":["Set","PO:TEST1","21-01-2022","DanielCompany","SupplierCompany","FOB","12 mm Wire Rope","80.50.20.00.00","32000","2 USD","64000USD","90 days after BL","Invoice12345","","PO picked at the factory","21-01-2022","28-01-2022","","04-03-2022","","",""]}'
```

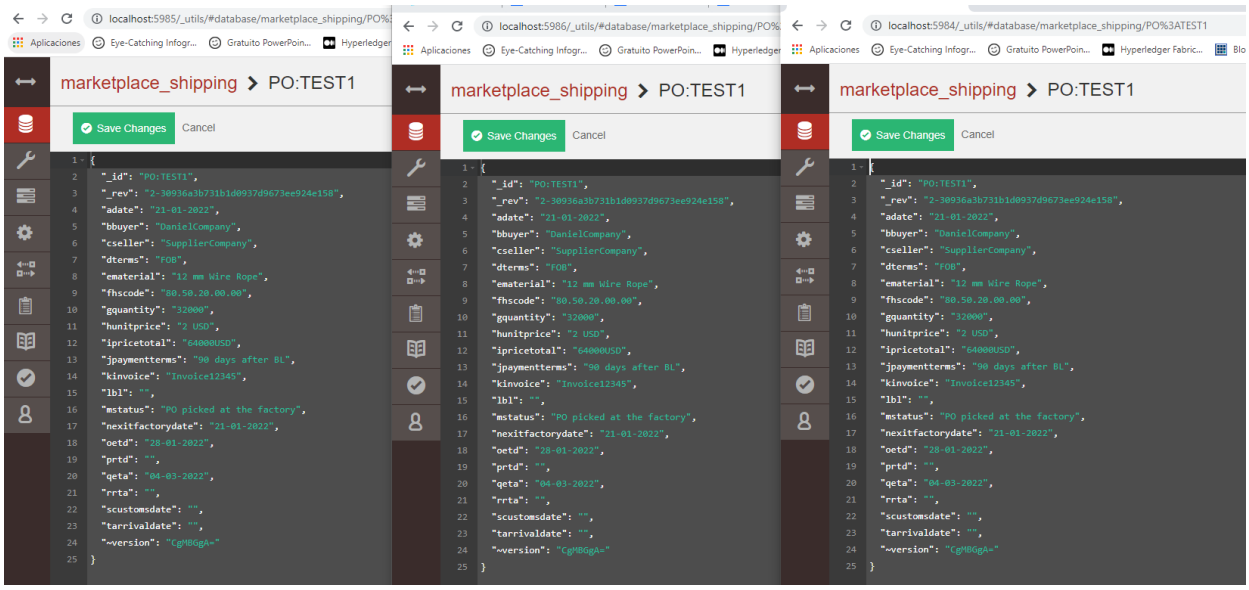


Ilustración 19 Transacción 2 (Creación Propia)

Transacción 3

```
CORE_PEER_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org3.test.com/users/Admin@org3.test.com/msp CORE_PEER_ADDRESS=peer0.org3.test.com:7051 CORE_PEER_LOCALMSPID="Org3MSP"
```

CORE_PEER_TLS_ROOTCERT_FILE=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org3.test.com/peers/peer0.org3.test.com/tls/ca.crt peer chaincode invoke -o orderer.test.com:7050 --tls --cafile $ORDERER_CA -C $CHANNEL_NAME -n $CHAINCODE_NAME -c '{"Args":["Set","PO:TEST1","30-01-2022","DanielCompany","SupplierCompany","FOB","12 mm Wire Rope","80.50.20.00.00","32000","2 USD","64000USD","90 days after BL","Invoice12345","","PO in transit","21-01-2022","28-01-2022","30-01-2022","04-03-2022","","",""]}'

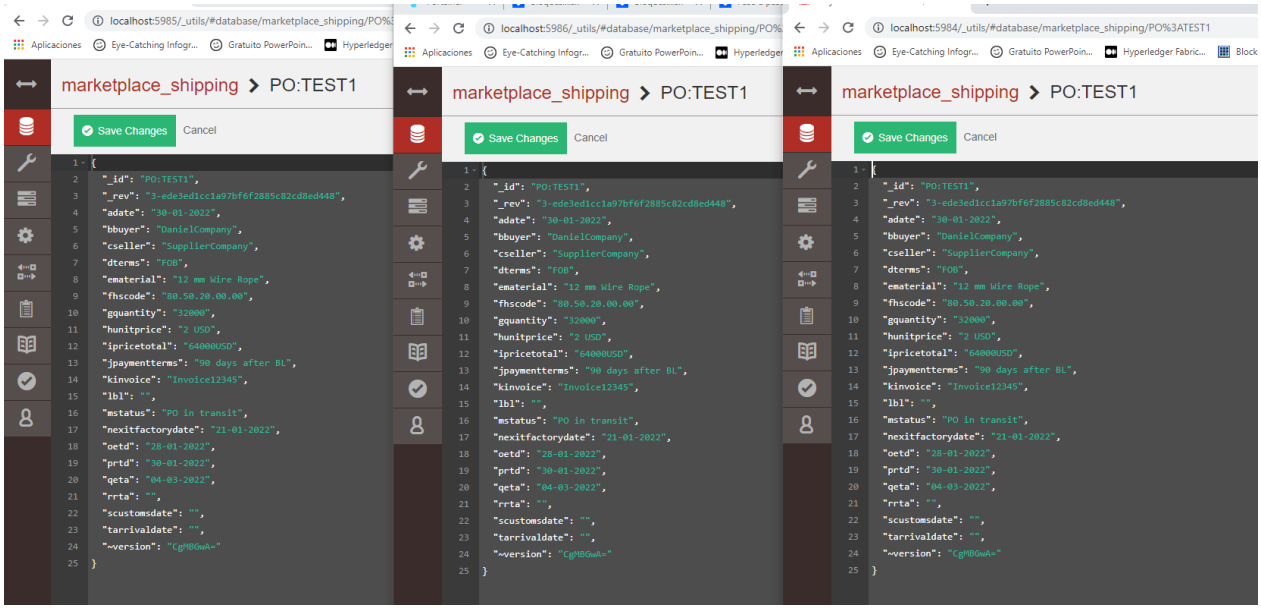


Ilustración 20 Transacción 3 (Creación Propia)

Transacción 4

CORE_PEER_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org3.test.com/users/Admin@org3.test.com/msp CORE_PEER_ADDRESS=peer0.org3.test.com:7051 CORE_PEER_LOCALMSPID="Org3MSP" CORE_PEER_TLS_ROOTCERT_FILE=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org3.test.com/peers/peer0.org3.test.com/tls/ca.crt peer chaincode invoke -o

orderer.test.com:7050 --tls --cafile $ORDERER_CA -C $CHANNEL_NAME -n $CHAINCODE_NAME -c '{"Args":["Set","PO:TEST1","05-03-2022","DanielCompany","SupplierCompany","FOB","12 mm Wire Rope","80.50.20.00.00","32000","2 USD","64000USD","90 days after BL","Invoice12345","","PO arrived at the port","21-01-2022","28-01-2022","30-01-2022","04-03-2022","05-03-2022","",""]}'

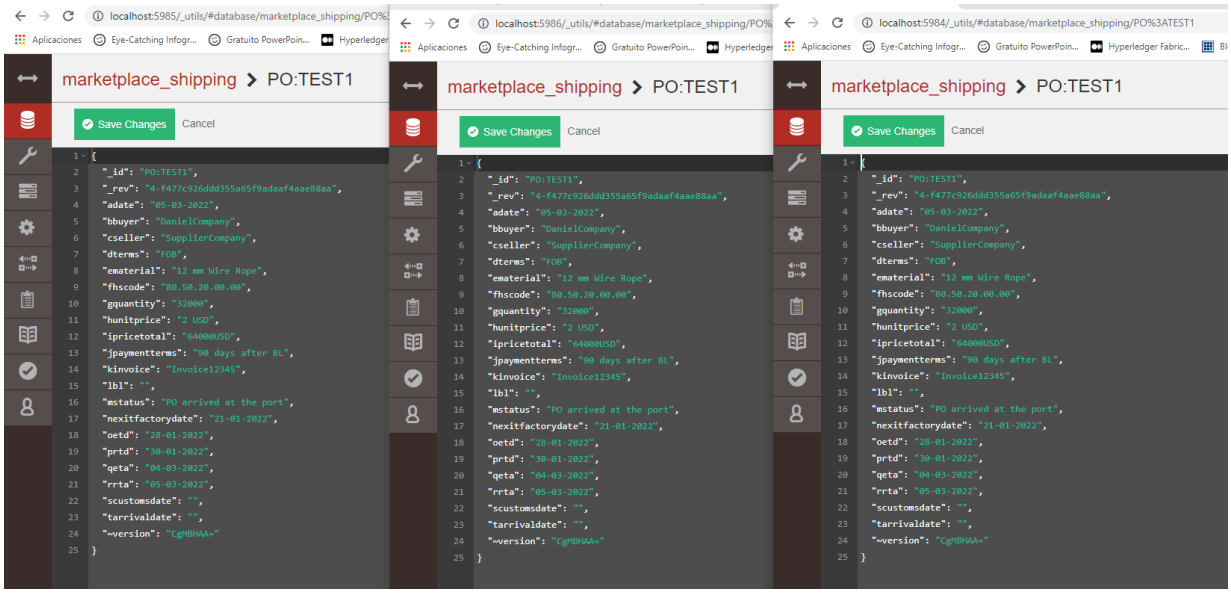


Ilustración 21 Transacción 4 (Creación Propia)

Transacción 5

CORE_PEER_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org3.test.com/users/Admin@org3.test.com/msp CORE_PEER_ADDRESS=peer0.org3.test.com:7051 CORE_PEER_LOCALMSPID="Org3MSP" CORE_PEER_TLS_ROOTCERT_FILE=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org3.test.com/peers/peer0.org3.test.com/tls/ca.crt peer chaincode invoke -o orderer.test.com:7050 --tls --cafile $ORDERER_CA -C $CHANNEL_NAME -n $CHAINCODE_NAME -c '{"Args":["Set","PO:TEST1","25-03-2022","DanielCompany","SupplierCompany","FOB","12 mm Wire Rope","80.50.20.00.00","32000","2 USD","64000USD","90 days after

```
BL","Invoice12345","","PO finished customs processes","21-01-2022","28-01-2022","30-01-2022","04-03-2022","05-03-2022","25-03-2022",""]}'
```

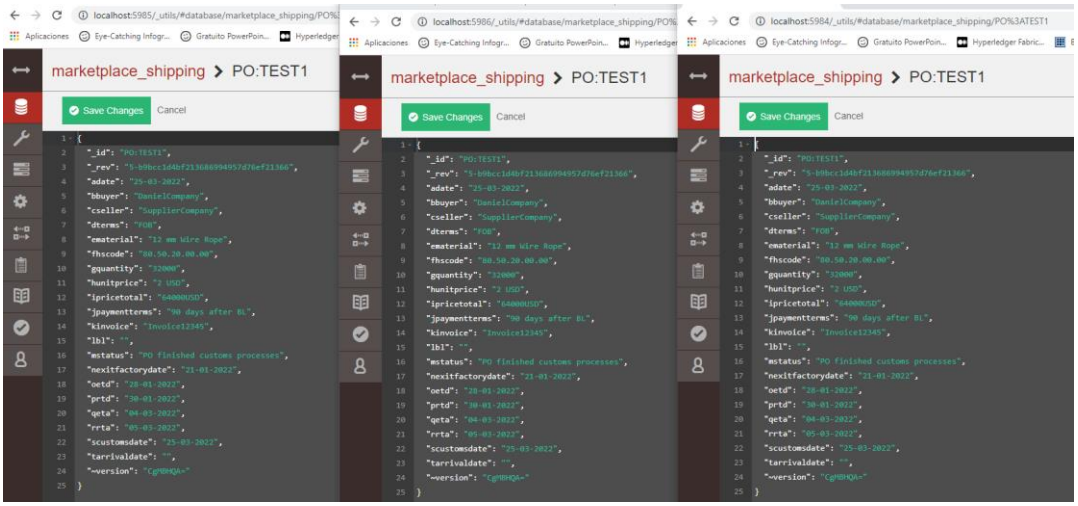


Ilustración 22 Transacción 5 (Creación Propia)

Transacción 6

```
peer chaincode invoke -o orderer.test.com:7050 --tls --cafile $ORDERER_CA -C $CHANNEL_NAME -n $CHAINCODE_NAME -c '{"Args":["Set","PO:TEST1","28-03-2022","DanielCompany","SupplierCompany","FOB","12 mm Wire Rope","80.50.20.00.00","32000","2 USD","64000USD","90 days after BL","Invoice12345","","PO arrived at the warehouse","21-01-2022","28-01-2022","30-01-2022","04-03-2022","05-03-2022","25-03-2022","28-03-2022"]}'
```

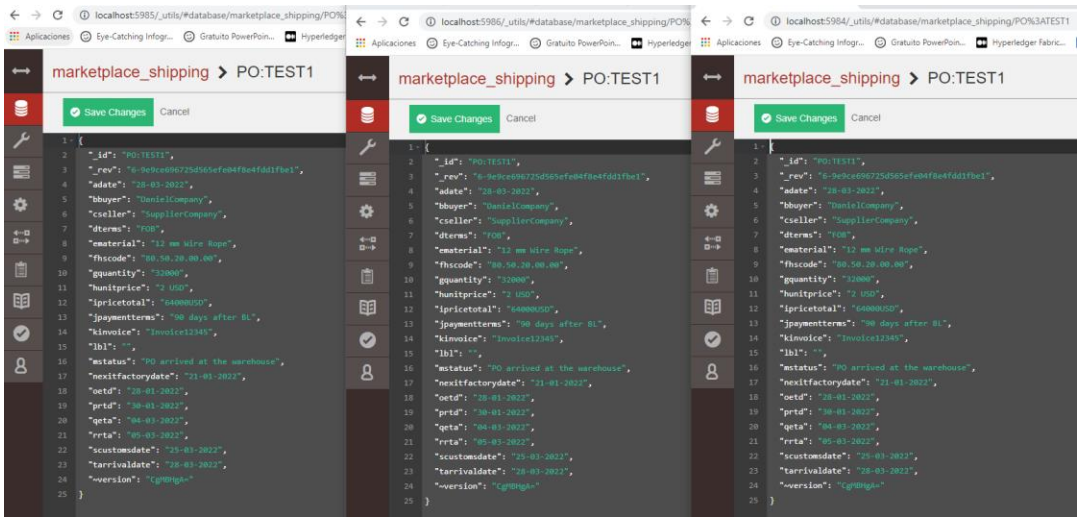


Ilustración 23 Transacción 6 (Creación Propia)

8.3.5.2. Paso 5.2: Consultar tracking

Desde la interfaz de comandos de línea – CLI –, se procede a utilizar la segunda función que se incluye en el smart contract llamada "query", la cual permite visualizar el estado de la orden de compra. En este caso se demuestra que ya terminó con el proceso de importación puesto que

```
peer chaincode query -C $CHANNEL_NAME -n $CHAINCODE_NAME -c '{"Args":["Query","PO:TEST1"]}'
```

```
2022-02-28 01:23:02.140 UTC [msp.identity] Sign -> DEBU 03b Sign: digest: 4560498B7E3BA9300
6C78B3135EEE6D01FF147614D3D62F908B1FE8DFF0D8608
{"adate":"28-03-2022","bbuyer":"DanielCompany","cseller":"SupplierCompany","dterms":"FOB","
ematerial":"12 mm Wire Rope","fhscode":"80.50.20.00.00","gquantity":"32000","hunitprice":"2
 USD","ipricetotal":"64000USD","jpaymentterms":"90 days after BL","kinvoice":"Invoice12345"
,"lbl":"","mstatus":"PO arrived at the warehouse","nexitfactorydate":"21-01-2022","oetd":"2
8-01-2022","prtd":"30-01-2022","qeta":"04-03-2022","rrta":"05-03-2022","scustomsdate":"25-0
3-2022","tarrivaldate":"28-03-2022"}
```

Ilustración 24 Respuesta de tracking (Creación Propia)

8.3.5.3. Paso 5.3: Otros - Revisar si la Organización 2 puede generar cambios.

Con base a las políticas que se establecieron en el smart contract, la organización 2 no podría escribir. Para comprobar esta política se procederá con el siguiente comando:

```
CORE_PEER_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org2.test.com/users/Admin@org2.test.com/msp CORE_PEER_ADDRESS=peer0.org2.test.com:7051 CORE_PEER_LOCALMSPID="Org2MSP" CORE_PEER_TLS_ROOTCERT_FILE=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org2.test.com/peers/peer0.org3.test.com/tls/ca.crt peer chaincode invoke -o orderer.test.com:7050 --tls --cafile $ORDERER_CA -C $CHANNEL_NAME -n $CHAINCODE_NAME -c '{"Args":["Set","PO:TEST1","31-03-2022","DanielCompany","SupplierCompany","FOB","12 mm Wire Rope","80.50.20.00.00","32000","2 USD","64000USD","90 days after BL","Invoice12345","","PO does not exist","","","","","","","","]}'
```

Al generar la transacción, se presenta el error que aparece en la imagen inferior, lo cual evidencia que esto no puede ser realizado puesto que la organización únicamente posee el permiso de visualización.

```
2022-02-28 01:33:12.996 UTC [msp] setupSigningIdentity -> DEBU 01e Signing identity expires
 at 2032-01-19 00:34:00 +0000 UTC
Error: error getting endorser client for invoke: failed to load config for PeerClient: unab
le to load peer.tls.rootcert.file: open /opt/gopath/src/github.com/hyperledger/fabric/peer/
crypto/peerOrganizations/org2.test.com/peers/peer0.org3.test.com/tls/ca.crt: no such file o
r directory
```

Ilustración 25 Respuesta a error (Creación Propia)

9. Conclusiones y apreciaciones finales

La aplicación de Blockchain en la cadena de suministro es una oportunidad valiosa que aplicarse por los siguientes motivos:

- Como se puede observar desde la prueba de concepto, la aplicación del Blockchain privado garantiza que únicamente los actores que se indiquen puedan aprobar y realizar transacciones dentro de la red.
- La aplicación de Blockchain les permitiría a las empresas colombianas con deseo de certificarse como Operador Económico Autorizado, garantizar que las ley se

aplique a través de garantizar toda la trazabilidad informática desde el inicio de la orden de compra, hasta el pago y llegada de la mercancía.

- La aplicación de esta tecnología permitirá a agentes externos auditores que puedan verificar la veracidad de la información de una manera más ágil.
- Para aplicar la tecnología Blockchain privada se necesita un acuerdo entre todas las partes que deseen visualizar y participar dentro de las transacciones para que esto pueda aplicarse en el ámbito empresarial.

Dado que el alcance de esta tesis consta de realizar una prueba de concepto para demostrar su aplicabilidad dentro de la cadena de suministro, para futuros estudios se podrían contemplar los siguientes escenarios:

- Generar un código que contemple experiencia al usuario para que se pueda observar dentro de un ambiente empresarial el impacto que este tipo de tecnología podría brindar.
- Analizar las variables económicas y financieras que justifiquen la inversión de este tipo de tecnologías en una organización.

10. Referencias

Ananthanarayanana, B., & Arumugam, U. (Agosto de 2020). Development of a reliable supply chain system using blockchain. *Journal of Intelligent & Fuzzy Systems*, págs. 1-11.

Apps Run the World. (15 de Enero de 2021). *List of IBM Food Trust Customers*. Obtenido de https://www.appsruntheworld.com/customers-database/products/view/ibm-food-trust#:~:text=Companies%20using%20IBM%20Food%20Trust,revenues%20of%20%2498.26%20billions%2C%20Golden

Apps Run The World. (15 de Enero de 2021). *List of IBM TradeLens Customers*. Obtenido de https://www.appsruntheworld.com/customers-database/products/view/ibm-tradelens#:~:text=Companies%20using%20IBM%20TradeLens%20for,of%20%245.10%20billions%2C%20APM%20Terminals

ASCM. (2021). *Introduction*. Obtenido de SCOR digital standard: https://scor.ascm.org/processes/introduction

BASC. (2005). *Business Alliance for secure commerce*. Obtenido de Quiénes somos: https://www.wbasco.org/es/pagina-institucional/quienes-somos

Behera, R. K. (2017). Big Data Analytics in Real Time – Technical Challenges and its Solutions. *International Conference on Information Technology (ICIT), IEEE*, 30-35.

Ben K, D. (2015). Big data and analytics in higher education: opportunities and challenges. *British Journal of Educational Technology*, Vol. 46 No. 5, pp. 904-920.

Binance. (5 de Diciembre de 2018). *Articles*. Obtenido de Byzantine Fault Tolerance Explained: https://academy.binance.com/en/articles/byzantine-fault-tolerance-explained

Bitcoin Suisse. (8 de Marzo de 2021). *What is Proof of Work?* Obtenido de Battle-tested blockchain security: https://www.bitcoinsuisse.com/fundamentals/what-is-proof-of-work

Blockchain training Alliance. (2018). *LinuxFoundationX LFS170x*. Obtenido de https://learning.edx.org/course/course-v1:LinuxFoundationX+LFS170x+1T2020/block-v1:LinuxFoundationX+LFS170x+1T2020+type@sequential+block@1bc1bb174ea04f549ec7cad967a5ecb1/block-v1:LinuxFoundationX+LFS170x+1T2020+type@vertical+block@46a2909492f44034b4457a34

Cámara de comercio internacional. (2020). *Incoterms® 2020 practical free wallchart*. Obtenido de https://iccwbo.org/publication/incoterms-2020-practical-free-wallchart/

Chopra, S., & Meindl, P. (2016). *Supply Chain Management: Srategy, planning, and Operation.* Harlow: Pearson Education Limited.

Deloitte. (2020). *Deloitte's 2020 Global Blockchain Survey.* Deloitte Insights.

DIAN. (01 de Enero de 2021). *Consulta de Arancel*. Obtenido de https://muisca.dian.gov.co/WebArancel/DefResultadoConsNomenclaturas.faces

Diario del exportador. (2017). *Logística & Transporte*. Obtenido de Modos y medios de transporte para la importación y exportación: https://www.diariodelexportador.com/2017/08/modos-y-medios-de-transporte-para-la.html

Docker. (2022). *What is a container?* Obtenido de Use containers to Build, Share and Run your applications: https://www.docker.com/resources/what-container/

Doubleday, K. (27 de Noviembre de 2018). *Blockchain Immutability — Why Does it Matter?* Obtenido de https://medium.com/fluree/immutability-and-the-enterprise-an-immense-value-proposition-98cd3bf900b1

DSV. (2020). *Tipos de Incoterms® 2020*. Obtenido de https://www.dsv.com/es-es/ayuda/faq/tipos-de-incoterms-2020

Enriquez Caro, R. (23 de Diciembre de 2015). *Los Incoterms: ¿Es obligatorio su uso?* Obtenido de https://taemperuconsulting.com/es-obligatorio-el-uso-de-los-incoterms-en-el-comercio-internacional/

Ethereum. (8 de Julio de 2021). *WHAT IS ETHEREUM?* Obtenido de The foundation for our digital future: https://ethereum.org/en/what-is-ethereum/

Frankenfield, J. (21 de Abril de 2021). *Cryptocurrency strategy and education*. Obtenido de Proof of stake (PoS): https://www.investopedia.com/terms/p/proof-stake-pos.asp

Gallagher, M., & King, A. (2020). Cyberspace Solium Commission Final Report. *United States of America Cyberspace*.

Gerencie. (14 de Enero de 2022). *Pago por consignación*. Obtenido de https://www.gerencie.com/en-que-consiste-el-pago-por-consignacion-segun-normas-del-codigo-civil.html

Griffin, T. (26 de Enero de 2021). *C.L Services*. Obtenido de How to Protect Your Supply Chain from Cyber Attacks: https://www.clservicesinc.com/2021/01/26/protect-supply-chain-cyber-attacks/

Gupta, M. (2020). *Blockchain for dummies.* Hoboken, NJ: John Wiley & Sons, Inc.

Hammadi, L., Souza de Cursi, E., Barbu, V. S., Ouahman, A. A., & Ibourk, A. (Septiembre de 2018). A SCOR model for customs supply chain process design . *World Customs Journal*, págs. 95-103.

Hyperledger. (2018). *An introduction to hyperledger.* Hyperledger.

Hyperledger fabric. (2020). *Introduction*. Obtenido de https://hyperledger-fabric.readthedocs.io/en/latest/whatis.html#hyperledger-fabric

HyperledgerFoundation. (1 de Diciembre de 2020). *Blockchain Foundation*. Obtenido de Curso Desarrollo Blockchain: https://wiki.hyperledger.org/display/CP/Curso++Hyperledger+Fabric

IBM. (2021). *Blockchain for supply chain solutions*. Obtenido de https://www.ibm.com/blockchain/industries/supply-chain?lnk=BK4D

IBM. (2021). *IBM Food Trust*. Obtenido de https://www.ibm.com/co-es/products/food-trust

IBM. (2021). *Topics*. Obtenido de What is blockchain technology?: https://www.ibm.com/topics/what-is-blockchain

IBM. (2021). *Trust Your Supplier solution transforms supplier management*. Obtenido de https://www.ibm.com/blockchain/solutions/trust-your-supplier

IBM Blockchain Pulse. (3 de Marzo de 2019). *What's the difference between a blockchain and a database?* Obtenido de https://www.ibm.com/blogs/blockchain/2019/01/whats-the-difference-between-a-blockchain-and-a-database/

IBM Food trust. (2021). *Focus on supply chain efficiency.* IBM food trust.

Incoterms® 2020. (2020). *Incoterms® 2020 introduction.* Paris: International Chamber of commerce.

International Trade Administration. (2021). *International trade Administration*. Obtenido de Methods of payment: https://www.trade.gov/methods-payment

Joshi, N. (23 de Abril de 2019). *Allerin*. Obtenido de 8 blockchain consensus mechanisms you should know about: https://www.allerin.com/blog/8-blockchain-consensus-mechanisms-you-should-know-about

Melnyk, S. A., Schoenherr, T., Speier-Pero, C., Peters, C., Chang, J. F., & Friday, D. (2021). New challenges in supply chain management: cybersecurity across the supply chain. *International Journal of Production Research*, 162-183.

OBS Business School. (21 de Marzo de 2021). Obtenido de Modelo SCOR: definición, procesos, ejemplo, pros y contras: https://www.obsbusiness.school/blog/modelo-scor-definicion-procesos-ejemplo-pros-y-contras

Páez, G. (14 de Febrero de 2020). *Economipedia*. Obtenido de Carta de crédito: https://economipedia.com/definiciones/carta-de-credito.html

Pérez, A. (23 de Marzo de 2021). *OBS business school*. Obtenido de Modelo SCOR: definición, procesos, ejemplo, pros y contras: https://www.obsbusiness.school/blog/modelo-scor-definicion-procesos-ejemplo-pros-y-contras

Perkins, B., & Wailgum, T. (2017). What is supply chain management (SCM)? Mastering logistics end to end. *CIO*. Obtenido de https://www.cio.com/article/2439493/what-is-supply-chain-management-scm-mastering-logistics-end-to-end.html

Peterson, B., George, R., Yaros, O., Beam, D., Dibbell, J., & Moore, R. (2019). Blockchain for business. *Journal of investment compliance*, págs. 17-21.

Pierre A., D. (2016). *Logística Internacional: La administración de las operaciones de comercio internacional.* Ciudad de México: Cengage Learning, Inc.

Procolombia. (24 de Octubre de 2016). *Conozca los beneficios de ser Operador Económico Autorizado*. Obtenido de https://procolombia.co/actualidad-internacional/agroindustria/conozca-los-beneficios-de-ser-operador-economico-autorizado

Procolombia. (2021). *Herramientas y servicios para el exportador*. Obtenido de https://www.colombiatrade.com.co/herramientas-del-exportador/logistica/incoterms-2020?__cf_chl_jschl_tk__=f729423ee35c823952f0a973dcba799c964a19e9-1620357682-0-AVhKMoDRZnujFfe_uetGlja5jp_Eykfp0TMCr8elUuN4qjT6K4lBCXrw2_1FyD5OVpvsWU69AmjAvwlf9zYCbX_edxUjpKe

Purplesec. (2020). *The Ultimate List Of Stats, Data & Trends.* Obtenido de 2020 Cyber Security Statistics: https://purplesec.us/resources/cyber-security-statistics/#:~:text=In%202018%20there%20were%2080%2C000,30%20million%20attacks%20per%20year.&text=Ransomware%20attacks%20are%20growing%20more,businesses%20with%20under%201%2C000%20employees.

Robinson, A. (2018). *[INFOGRAPHIC] The Evolution and History of Supply Chain Management.* Obtenido de Cerasis: https://cerasis.com/history-of-supply-chain-management/

Schwarz, L. B. (2008). Reviewed Work(s): Application of the SCOR Model in Supply Chain Management by Rolf G. Poluha. *Interfaces*, 414-416.

Shea, S. (Marzo de 2022). *Techtarget*. Obtenido de Top 7 enterprise cybersecurity challenges in 2022: https://www.techtarget.com/searchsecurity/tip/Cybersecurity-challenges-and-how-to-address-them

Spiegato. (Junio de 2022). *¿Qué es una colección documental?* Obtenido de https://spiegato.com/es/que-es-una-coleccion-documental

Tarasenko, E. (27 de Diciembre de 2019). *PRIVATE BLOCKCHAIN VS TRADITIONAL CENTRALIZED DATABASE*. Obtenido de https://merehead.com/blog/private-blockchain-vs-traditional-centralized-database/

TIBA. (21 de Enero de 2020). *Incoterms 2020*. Obtenido de https://www.tibagroup.com/blog/incoterms-2020

Tradelens. (2020). *Tradelens*. Obtenido de https://www.tradelens.com/

Trust your supplier. (2021). *About TYS Trust Your Supplier*. Obtenido de https://trustyoursupplier.com/about-us/about-tys/

Trust your supplier. (2021). *Blockchain.* Obtenido de Blockchain-Enabled Supplier Information Management: https://trustyoursupplier.com/about-us/blockchain/

UNCITRAL. (1980). *Convención de las Naciones Unidas sobre los Contratos de Compraventa Internacional de Mercaderías (Viena, 1980)*. Obtenido de https://uncitral.un.org/es/texts/salegoods/conventions/sale_of_goods/cisg

World Bank. (12 de Abril de 2018). *Blockchain & Distributed Ledger Technology (DLT)*. Obtenido de https://www.worldbank.org/en/topic/financialsector/brief/blockchain-dlt

WTO. (6 de Junio de 2021). *Data.* Obtenido de https://data.wto.org/

Zhang, J. (Mayo de 2019). *Deploying Blockchain tecnhology in the Supply Chain.* Obtenido de Research gate: https://www.researchgate.net/publication/333685407_Deploying_Blockchain_Technology_in_the_Supply_Chain?enrichId=rgreq-286ac115e0407b064c10005f6898911e-XXX&enrichSource=Y292ZXJQYWdlOzMzMzY4NTQwNztBUzo4MTk3ODcyNDg3MjYwMTdAMTU3MjQ2MzkxOTMxNw%3D%3D&el=1_x_2&_