Metodología de implementación Blockchain para la mejora en rendimiento, seguridad y transparencia de las transacciones en el sector bancario – Caso de uso

Juan Camilo Naranjo Marin

TRABAJO DE GRADO

Asesor
Juan Guillermo Lalinde Pulido

UNIVERSIDAD EAFIT
Ciencias Aplicadas e Ingeniería
Maestría en Ingeniería
MEDELLIN, COLOMBIA
Marzo 2024

TABLA DE CONTENIDO

LISTA DE FIGURAS

RESUMEN

El presente trabajo se enfoca en analizar y presentar una estrategia metodológica para implementar la tecnología blockchain y cómo puede esto mejorar el rendimiento, la seguridad y la en las transacciones financieras del sector bancario mediante un caso de uso. La tecnología blockchain ha emergido como solución prometedora para abordar diversos retos en varios sectores como salud, cadena de suministros, entre otras. Ayudando en términos de confianza y seguridad, en los cuales el sector financiero podría mejorar. La hipótesis planteada establece que la implementación de la tecnología blockchain puede resolver estos problemas y optimizar aún más las operaciones.

Para abordar este objetivo, se identificarán los problemas actuales en el sector financiero, específicamente en las transacciones financieras. Posteriormente, se analizará cómo la tecnología blockchain puede resolver estos problemas y entregar los prometedores resultados que se han obtenido alrededor del mundo en términos de eficiencia, transparencia y seguridad. Se propondrá un modelo base de implementación basado en los hallazgos obtenidos.

Palabras clave: Blockchain, Hyperledger, Ciberseguridad, Descentralización, Transacciones, Banca.

JUSTIFICACIÓN

El sector financiero juega un papel fundamental en la economía y la sociedad, y en la búsqueda continua por mejorar la transparencia y seguridad en las transacciones financieras, se han identificado oportunidades mediante la aplicación de nuevas tecnologías. Entre estas tecnologías, la tecnología blockchain ha demostrado ser especialmente prometedora.

La tecnología blockchain ofrece un sistema descentralizado de registros inmutables que permite inscribir y verificar transacciones de forma transparente y segura.

Esta investigación se justifica al analizar el potencial de la tecnología blockchain en el sector financiero para mejorar la transparencia y seguridad en transacciones y pagos. En particular, se busca desarrollar una estrategia metodológica que permita su implementación práctica por parte de los bancos.

El impacto de los resultados de esta investigación se manifiesta en el fortalecimiento de la confianza en las operaciones financieras, la reducción de intermediarios, la mejora en tiempos de procesamiento y la prevención de fraudes y errores. La implementación exitosa de blockchain en el sector bancario puede transformar la forma en que se realizan las transacciones y pagos, beneficiando tanto a las instituciones financieras como a los usuarios.

Asimismo, la relevancia de este estudio tiene el potencial de generar una transformación significativa en la industria bancaria, mejorando la eficiencia y seguridad de sus procesos.

La justificación de esta investigación se fundamenta en la necesidad práctica de afrontar los desafíos actuales del sector financiero en términos de seguridad, rendimiento y transparencia. Al ofrecer una estrategia metodológica basada en la tecnología blockchain, se busca resolver una problemática relevante y avanzar hacia un futuro financiero más confiable y eficiente.

ANTECEDENTES

¿Qué es Blockchain?

Realizando una similitud cercana, visualiza un libro de contabilidad gigante, compartido y digital donde se registran todas las transacciones que se realizan en una red. Este libro es blockchain, también conocido como cadena de bloques. En lugar de tener un solo administrador, la información se distribuye entre varios o miles de computadoras, nodos o participantes en todo el mundo, lo que la hace inmutable (imposible de modificar) y segura.

¿Cómo funciona?

La información se organiza en bloques, como si fueran páginas del libro. Cada bloque contiene:

- **Información de la transacción**: Fecha, hora, participantes, valor(es), etc.
- **Enlace al bloque anterior**: Crea una cadena irrompible pues, además del enlace, incluye el hash del bloque anterior.
- **Un código único (hash)**: Si se modifica cualquier dato del bloque, el hash cambia, alertando a toda la red.

Para agregar un nuevo bloque:

- Se verifican las transacciones por los miembros de la red utilizando el algoritmo de consenso.
- Se crea un nuevo bloque con la información de las transacciones válidas.
- Se calcula el hash del nuevo bloque. El bloque incluye el hash del bloque anterior, por lo que el cálculo del hash garantiza que cualquier cambio en la cadena se pueda detectar.
- El nuevo bloque se añade a la cadena.

¿Qué mecanismos existen para garantizar la integridad y seguridad de la cadena de bloques, las propiedades y los datos?

Inmutabilidad:

- **Cadena irrompible**: Modificar un bloque exitosamente implicaría modificar todos los bloques posteriores, lo que es casi imposible por la cantidad de

computadoras, nodos o integrantes que verifican la información. Esto se debe a la forma como se encadenan los bloques: cada uno incluye el hash del anterior.
- **Criptografía**: Los hashes garantizan la integridad de los datos.

Seguridad:

- **Descentralización**: No hay un punto único de ataque.
- **Autenticidad**: La identidad de los participantes se valida mediante criptografía de clave pública, la cual utiliza algoritmos robustos.
- **Consenso**: Los participantes de la red verifican que las transacciones se ejecuten de manera que, si alguien quiere alterar la construcción del consenso, tiene que controlar la gran mayoría de los nodos de la red que participan en este proceso.

La tecnología blockchain, desde que fue concebida, se estableció como una red de bloques estáticos de información administrados de forma descentralizada por múltiples computadores o servidores que no son propiedad de una sola entidad, estos bloques son vinculados unos a otros usando Hash. La blockchain, por lo tanto, no es una red que se encuentre dependiente de algún tipo de autoridad central. La información que viaja dentro de estos bloques se comparte entre la red y su registro y la trazabilidad es completamente visible para cualquier miembro de la red. Adicionalmente, dicha información no puede ser alterada ni eliminada sin que esta acción sea detectada, logrando así una característica de transparencia. (Salem et al., 2021)

La red permite transacciones de información sin intermediaciones, ha sido la entrada a múltiples oportunidades de mejora en diferentes sectores e industrias. Sin embargo, desde su creación, su uso ha sido poco visible a nivel mundial a través de todos estos sectores.

A continuación, se mencionarán algunos ejemplos y usos de los sectores más susceptibles a potencializar su eficiencia y eficacia tanto operacional como de cualquiera otra índole usando blockchain para el desarrollo de sus funciones.

I. **Medicina:** En el campo de la medicina, un uso muy frecuente es el almacenamiento de los datos recopilados de los pacientes en bases de datos convencionales, los cuales contienen generalmente uno o varios administradores, los cuales cuentan con todos los privilegios como lectura, escritura y ejecutar casi todo tipo de acciones, esto es un problema debido a se tiene un control centralizado sobre estos registros médicos y existe la posibilidad de un mal uso, manipulación o incluso perdida de los mismos, lo cual conlleva a un fallo inminente en el sistema. Es por esto por lo que la implementación de la tecnología blockchain podría eliminar estos riesgos sirviendo como puente directo entre el personal de salud y el paciente, asegurando toda la información médica para que

no sea alterada. Si se hace uso adicional de criptografía, se puede dar al paciente el control para definir qué información se comparte. (Ibor et al., 2023)

II. **Cadena de Suministro:** La cadena de suministro históricamente ha tenido una seria de problemas, entre ellas se encuentra la falta de trazabilidad, la falta de transparencia de la información y la gran cantidad de procesos manuales. Por lo tanto, las cadenas de suministro serían excelentes casos de estudio para la implementación de la tecnología Blockchain.

Una aplicación puede ser la tratada en (Liu et al., 2021) presentando una plataforma inteligente de seguimiento y localización basada en Blockchain para una cadena de suministro de medicamentos, aquí el Blockchain traería unos beneficios de transparencia y trazabilidad proporcionando registros inmutables y transparentes de todas las transacciones y eventos que sucedan, esto permite un seguimiento exacto de cada etapa del proceso, desde la producción del producto hasta la entrega final al cliente o consumidor. Posibilita, además, la prevención del fraude ya que reduce la posibilidad de medicamentos falsificados o con baja calidad, ya que, al tener una trazabilidad completa y detallada, resulta ser más difícil para actores malintencionados introducir fallas, errores o productos defectuosos. Otra arista sería la seguridad ya que la descentralización y la criptografía que ofrece Blockchain aumenta la seguridad de los datos y transacciones ayudando a proteger la integridad de la información que se almacene. Y por último podemos destacar también la automatización y eficiencia que traen consigo los contratos inteligentes, permitiendo una automatización de procesos en la cadena de suministro. Ejecutando automáticamente una serie de eventos con ciertas condiciones predefinidas como podría ser liberación de pagos, alertas, notificaciones, entre otros.

III. **Aseguradoras:** La implementación de la tecnología Blockchain en el sector de las aseguradoras, puede ser posible en cualquier tipo de red Blockchain. En el ejemplo de implementación presentado en (Aleksieva et al., 2020a) se puede apreciar como una red privada de Blockchain basada en Hyperledger Fabric puede lograr una digitalización completa de los procesos, protección de datos contra ataques maliciosos, y control de visibilidad de la información a personas autorizadas. Además de lo anterior, también permite la creación de pólizas, determinar el riesgo de los seguros, ver su estado (trazabilidad), ejecutar reclamaciones y automatizar cualquier otra lógica de negocio necesaria.

IV. **Protección de Propiedad Intelectual:** La tecnología Blockchain tiene el potencial de revolucionar los derechos de autor y propiedad intelectual en diferentes sectores como la música, el cine, el arte, literatura, patentes e invenciones, marcas únicas, investigaciones científicas, contenido digital, moda y diseño, software y hasta credenciales y certificaciones de educación.

Es por esto por lo que tomamos de referencia la industria de la música, en donde se tienen diferentes investigaciones acerca de la implementación del blockchain y sus beneficios. Con los principios de diseño para sistemas DRM (Gestión de derechos digitales) propuestos en (Ciriello et al., 2023) se busca tener una solución integrada que resuelva los desafíos históricos de los DRM tradicionales centralizados que tienen poca inclusión de propietarios de derechos, creadores y consumidores, teniendo como resultado con Blockchain.

- El almacenamiento de los metadatos de los derechos musicales en una cadena de bloques pública.
- Validar los metadatos musicales con mecanismos de consenso en una cadena de bloques autorizada y así poder asignar identificadores únicos a los propietarios.
- Pagos eficientes y transparentes automatizados de regalías mediante contratos inteligentes con monedas estables.

Un sistema descentralizado usando Blockchain de estas características, en contraste con un sistema centralizado tradicional, permite aumentar la transparencia, la coherencia y la eficiencia significativamente.

V. **Gobierno:** En las ciudades inteligentes, un servicio fundamental que puede existir es la votación electrónica, para poder así facilitar el servicio de votación a los ciudadanos, este servicio implementado en una red blockchain ayudaría no solo a crear un sistema descentralizado y seguro, sino también transparente y preciso. Es cierto que al tener este sistema se pueden tener algunas brechas con vulnerabilidades como un ataque Sybil, permitiendo que un votante cree más de una entidad y vote más de una vez, pero si se contempla la unión de un registro único de ciudadanos manejado por un administrador y un contrato inteligente que permita solo un único voto por usuario, se podría consolidar como un servicio efectivo con las cualidades mencionadas. (Chentouf & Bouchkaren, 2023)

VI. Banca / Sector Financiero: En los servicios financieros es muy factible que el uso de Blockchain pueda resolver múltiples problemáticas presentes en la actualidad. Cuatro grandes áreas mencionadas en (Al-Jeshi et al., 2022) que podrían mejorar significativamente en los servicios financieros son:

- **Ejecución Comercial y Acuerdos Financieros:** Con el uso de la tecnología Blockchain se podría erradicar la problemática, por ejemplo, de las grandes cantidades de tiempo que toman los pagos transfronterizos eliminando la intermediación de redes permitiendo una ejecución más rápida y segura.

- **Registro e Intercambio de Activos:** Una de las grandes ventajas de la red Blockchain es que puede almacenar cualquier tipo de activo digital, incluyendo activos físicos, propiedad intelectual o bienes raíces. Por lo tanto, el registro de estos se puede llevar a cabo sin problemas.

- **Administración de Cadenas de Suministros:** Mejora el rastreo en tiempo real del movimiento de las mercancías, lograda a través de la verificación mediante un protocolo de consenso.

- **Contratos Inteligentes para Acciones Automatizadas:** Estos contratos pueden ser programados de acuerdo con la necesidad, y es por esto el gran potencial que representan a la hora de automatizar procesos, múltiples acciones pueden ser ejecutadas con uno o múltiples disparadores cuando el bloque entre a la red. El contrato inteligente permite transformar las reglas de negocio en comandos programables y la ejecución automatizada de los mismos, eliminando así cualquier tipo de practica manual o intervención de intermediarios.

En los últimos años ha estado cobrando un poco más de relevancia en el sector financiero y la banca debido a su gran cantidad de ventajas y beneficios que representan a sus usuarios una mejor experiencia en las transacciones y la seguridad. Existen diferentes enfoques de implementación y tipos de blockchain para su uso dependiendo del tipo de red que necesitemos.

Tipos de Blockchain

Los primeros tres tipos de Blockchain, y lo más usados generalmente, explicados de forma clara en (Sheth & Dattani, 2019) son los siguientes.

- **Pública**: Se define una red Blockchain de tipo pública cuando no hay ninguna barrera sobre quien pueda usarlo, cualquier puede ejecutar un nodo, realizar minería, se puede tener acceso a una billetera, realizar transacciones, entre otras acciones. Son Blockchains abiertas y transparentes para cualquiera en internet. Este tipo de red escalona a varias criptomonedas más reconocidas y utilizadas en el mercado, por ejemplo, el Bitcoin, Litecoin, Cardano, Ethereum, entre otras.

- **Privada**: Es el tipo de Blockchain que posee un ecosistema cerrado, donde el acceso es restringido y no permite el ingreso libremente a la red. Por lo tanto, no es posible ver el historial de transacciones, realizar una transacción o cualquier otro tipo de acción. Estas redes pertenecen a un particular o a una entidad central, la cual se encarga de conceder los permisos necesarios para que los autores ingresen, también debe velar por la existencia de un mecanismo de consenso, ya sea existente (el mismo de la red pública) o creando mecanismos propios.

- **Consorcio**: En este tipo se elimina el poder total de la Blockchain que se le otorga a una persona o entidad, en su lugar, se distribuyen los permisos a un grupo de personas o entidades para que todos tengan toma de decisión sobre la misma.

Estos grupos formados se denominan consorcios. Como ejemplo tenemos Quorum, Hyperledger, Corda.

Es posible también tener una combinación de dos tipos de redes blockchain, lo que facilita el funcionamiento de diversas funcionalidades que demandan tanto interacción cerrada como abierta. Estas redes son las consideradas Hibridas.

- **Hibrida**: pueden aprovechar las ventajas de ambos tipos de Blockchain públicas y privadas. Este enfoque permite tener un gran equilibrio entre la descentralización necesaria para salir al mercado y al mundo y, a su vez, tener el control selectivo de los procesos e información que requerimos que se mantengan de manera sensible y privada.

Plataformas Blockchain

Para poder entrar en las plataformas de Blockchain, es pertinente de manera anticipada mencionar los protocolos de consenso, estos son utilizados por las plataformas de blockchain y se caracterizan por realizar una función de validación. Existen múltiples tipos de protocolos de consenso, entre ellos, los más destacados son **PoW** (Power of Work) en donde participan todos los nodos de la red, de forma anónima y a su vez compiten entre sí mismos para lograr consolidar y registrar un bloque en la red. A esta labor se le conoce como minería, con esto el primer nodo que adquiera esta consolidación del bloque obtiene una recompensa brindada en la criptomoneda nativa de la plataforma publica, y por otro lado tenemos el protocolo **PoA** (Power of Authority) que, a diferencia del protocolo anterior, los validadores no tienen generalmente interés en obtener recompensas por consolidar y registrar el bloque, sino que su interés se basa en demostrar su reputación como validadores para verificar la información del nodo y validar si la información es correcta, pertinente y necesaria para entrar en la red. PoA es muy útil para redes privadas, ya que ofrece altas velocidades y es altamente adecuado para empresas y se puede ver claramente los resultados positivos obtenidos en (Monrat et al., 2020)

Para dar un contexto más amplio acerca del funcionamiento de las otras plataformas (públicas) de las redes Blockchain, además de lo anteriormente mencionado, se presenta el concepto de pagar por los recursos computacionales necesarios para llevar a cabo transacciones y operaciones en estas redes. Esto conlleva a un costo determinado en un valor único de su propia criptomoneda nativa, por ejemplo, en Ethereum se cobra una tarifa llamada GAS (que se cobra en Ether), en la Binance Smart Chain la tarifa es el BNB fee (cobra en BNB), en Solana es el SOL fee, en Cardano se denomina ADA fee, en Polkadot es DOT fee, y así sucesivamente con todas las redes Blockchain públicas.

El costo aproximado está atado a múltiples variables como por ejemplo (Choi et al., 2023) la congestión en la red, el valor de la moneda nativa, la alta complejidad o la poca optimización de los contratos inteligentes, los pocos validadores o "mineros" en la red y la alta demanda de estos, los mecanismos de ajustes de tarifas por parte de las mismas redes por la oferta y la demanda, entre otros.

Existen varias plataformas de Blockchain con diferente tipo de enfoque, una comparación entre las principales existentes para el interés de este articulo (privadas) se muestran en la Tabla 1, tomada de (Monrat et al., 2020)

Blockchain Platforms	Ethereum	Hyperledger Fabric	Quorum	Corda
Type	Public	Enterprise	Enterprise	Enterprise
Purpose	Cross-Industry	Cross-Industry	Cross-Industry	Financial Services
Purpose	Business to Client (B2C)	Business to Business (B2B)	Financial Service Industry	Financial Service Industry
Smart Contract Programming Language	Solidity	GoLang, NodeJs	Solidity	Kotlin, Java
Currency	Ether	Can be built using chaincodes	Ether	No native Cryptocurrency (Corda coin)
Governance	DAO	Linux Foundation	Ethereum Foundation	R3
Consensus Algorithm	PoW	PBFT	RAFT	Pluggable Consensus
Throughput	A few 100s	more than 200 tps	200 tps	200 tps

Tabla 1. Comparación entre Plataformas Blockchain permisionadas

Podemos apreciar que existen opciones muy factibles para las Blockchain Privadas permitiendo, por ejemplo, un número mayor de transacciones por segundo y un algoritmo de consenso que no requiere tanto consumo de recursos como lo es el PoW.

Es de importante relevancia entender algunos de los conceptos como los tokens, los estándares de tokens y los contratos inteligentes.

Tokens Comúnmente utilizados por Blockchains Públicas

- **ERC-20:** Estándar de Ethereum para token fungibles, los cuales permiten que cada Token sea exactamente igual (en tipo y valor) a otro Token. Por ejemplo, un Token ERC-20 funciona de manera similar a una criptomoneda, 1 token representativo a una criptomoneda es y siempre será igual a los demás tokens. (Corwin Smith et al., n.d.)

- **ERC-721:** Estándar de Ethereum para tokens no fungibles, a diferencia del token anterior, este es único y puede tener un valor diferente que otro token proveniente del mismo contrato inteligente. (Corwin Smith et al., n.d.)

- **ERC-1155:** Estándar de Ethereum para que los contratos inteligentes puedan administrar múltiples tipos de tokens, por lo tanto, un solo contrato desplegado puede incluir cualquier combinación de token fungible o no fungible. (Corwin Smith et al., n.d.)

- **BEP-20:** Estándar de Binance para representar tokens que, al igual que los ERC-20 en Ethereum, sean tokens exactamente iguales. Creado especialmente para la Binance Smart Chain y puede representar cualquier cosa, desde acciones de un negocio hasta dólares almacenados en una bóveda de banco (stablecoins). (Binance Academy, n.d.-a)

- **Stablecoins:** Es un tipo de token que representa una conexión con las monedas fiat, ya que sus precios están vinculados a un activo de reserva como el dólar, euro, yen e, incluso, oro y petróleo, su propio nombre lo establece como un activo digital con dos características, la estabilidad de una moneda tradicional y la flexibilidad de los activos digitales. Se consideran como un token, una criptomoneda o un activo, y representan una paridad 1:1 con alguna moneda fiat existente, por ejemplo, existen múltiples stablecoins respaldadas con esta misma proporción con el dólar estadounidense, entre las cuales están los BUSD, USDC, USDT. Una stablecoin permite al poseedor retener ganancias y pérdidas, y transferir valor a un precio estable en redes Blockchain peer-to-peer. (Binance Academy, n.d.-b)

Comúnmente utilizados por Blockchains Privadas

- **Chaincode Tokens:** Define un activo o activos y las instrucciones de la transacción para modificar el activo o activos; en otras palabras, es la lógica del negocio. Se encarga de hacer cumplir las reglas para leer o modificar los pares clave-valor y otra información de la base de datos de estado. (Hyperledger Fabric, 2020)

- **Channel Tokens:** Es una ruta de comunicación privada entre dos o más miembros de una red Hyperledger Fabric o una Blockchain administrada por Amazon (AMB). Cada transacción en la red Hyperledger Fabric ocurre en un canal. Es necesario tener al menos un canal para una red. (Amazon, 2023)

Contratos Inteligentes

Los contratos inteligentes son programas almacenados en una cadena de bloques que se ejecutan cuando se cumplen ciertas condiciones predeterminadas. Generalmente se usa

para realizar automatización de procesos o ejecuciones de un acuerdo para que los participantes en una transacción puedan estar seguros del resultado de esta sin necesidad de participación de ningún intermediario o grandes brechas de tiempo innecesarias. (IBM, 2023)

Diferentes tipos de contratos inteligentes:

- Pago y Transferencia
- Apuestas o Juegos
- Tokenizacion (NTFs)
- Gobernanza
- Oráculos
- Finanzas Descentralizadas (DeFi)

El uso la red Blockchain conlleva a un uso de recursos proporcionales a ciertas variables y situaciones, por lo tanto, también se deben aclarar varios conceptos fundamentales para el entendimiento del funcionamiento de toda la red.

A pesar de todas las virtudes y bondades que nos puede ofrecer la Blockchain, como cualquier otra tecnología, trae sus retos, los cuales deberemos tener presentes a la hora de su implementación y seguimiento una vez se logré su desarrollo. A continuación, se presentarán algunos temas de consideración importantes presentados en el Framework Cost, Benefit, Risks, and Opportunity Analysis mencionado en (Osmani et al., 2021)

Costos: Si bien utilizar la tecnología Blockchain promueve la reducción de los costos operativos, también se debe de tener en cuenta que hay costos asociados a las transacciones, por otro lado, el procesamiento de la red requiere uso generalmente significativo de energía, y, además, unos costos también por parte del almacenamiento debido a que las bases de datos de la Blockchain deben almacenar datos indefinidamente, por lo que se espera que crezcan sustancialmente a lo largo del tiempo.

Beneficios: Entre la lista de los beneficios más comunes al implementar Blockchain son:

- Privacidad ya que eliminamos intermediación de terceros.
- Transparencia, ya que todas las transacciones pueden ser visibles y compartidos en toda la red.
- Mejora de seguridad, los datos no estarían expuestos solo en una o unas cuantas bases de datos centralizadas.
- Eficiencia en gastos generales.
- Inmutabilidad, las transacciones no pueden ser modificadas ni alteradas una vez se crean en la red.
- Transacciones rápidas (en las blockchain privadas o de consorcio)
- Confianza.

Riesgos:

- Aunque es cierto que se menciona una mejora en seguridad, es una característica que no es posible asegurarla en su totalidad, por lo tanto, se debe de tener en cuenta los posibles riesgos de seguridad.
- La escalabilidad, en comparación a la red bancaria tradicional que permite realizar cientos de transacciones por segundo, la Blockchain puede procesar un número muy inferior de transacciones en el mismo tiempo.
- Reversibilidad debido a que las instituciones financieras o bancos no pueden reversar transacciones por haber cometido algún error en la mismas.
- Interoperabilidad, es de resaltar que la interoperabilidad entre diferentes tipos de redes Blockchain es limitada, lo cual puede perjudicar a las instituciones financieras en el caso de que necesiten transar entre ellas.
- Riesgos regulatorios, es de conocimiento que las aplicaciones de blockchain que se usan para pagos, prestamos o inversión actualmente están en fases muy preliminares y exploratorias, por lo tanto, es muy compleja la adopción y entendimiento por parte de todas las instituciones financieras.

Oportunidades:

- Para los bancos, Blockchain provee varias ventajas competitivas en términos de contratos financieros basados en reducción de costos, transparencia en la información, un control de riesgos operacionales y costos de transacciones.
- Se puede tener una gran cantidad de ideas para realizar nuevos servicios, aquí Blockchain podría ayudar significativamente a los bancos en un futuro.

La tecnología Blockchain tiene la capacidad de solucionar en gran medida los problemas de confianza, seguridad y control sobre los datos en los servicios financieros, tal como se describe en la definición del problema en (Zhu et al., 2019) donde se presentan dificultades con la legitimidad y seguridad de la administración de documentación, la confianza en los sistemas, el tema de las validaciones en cambios, la necesidad de un enfoque transparente y rastreable y, además, los ataques y actividades maliciosas.

PLANTEAMIENTO DEL PROBLEMA

En el ámbito de la industria bancaria, surge la necesidad de establecer una relación clara entre la implementación de la tecnología blockchain y su impacto en la seguridad y transparencia de las transacciones financieras.

¿En qué medida la adopción de la tecnología blockchain podría mejorar la falta de rendimiento, seguridad y transparencia en las transacciones bancarias actuales? Esta pregunta busca abordar cómo la implementación de la tecnología blockchain influye en la disminución de fraudes y errores, la velocidad de las transacciones, la mejora en la confianza de los usuarios, la eficiencia en las transacciones y los costos operacionales de las transacciones en cada ámbito.

PREGUNTA DE INVESTIGACIÓN

¿Como Implementar de forma adecuada la Tecnología Blockchain para mejorar la Seguridad, Rendimiento y Transparencia en las Transacciones Financieras en el sector bancario actual?

OBJETIVOS

Objetivos Generales:

- Desarrollar una estrategia metodológica sólida para la implementación efectiva de la tecnología blockchain en el sector bancario.
- Utilizar un caso de uso existente en el sector bancario para implementar esta metodología.

Objetivos Específicos:

- Analizar los desafíos actuales relacionados con la seguridad, rendimiento y transparencia en las transacciones bancarias.

- Evaluar la efectividad de la tecnología blockchain en la reducción de fraudes y errores en las transacciones financieras.

METODOLOGIA

I. Consideraciones

El enfoque metodológico del presente trabajo contiene un componente mixto con resultados cuantitativos, representando los atributos más representativos que se deben tener en cuenta para la implementación y cualitativos, en contraste con las diferentes perspectivas que se pueden adquirir de los empleados, clientes y otros actores interesados.

Teniendo como fundamento principal los conceptos vistos anteriormente en los antecedentes, es prudente considerar una implementación de una Blockchain privada basada en Hyperledger Fabric, esta puede ser una elección sólida para el sector financiero (especialmente para un banco, como lo es nuestro caso de estudio) debido a sus ventajas en términos de privacidad, seguridad, rendimiento y control de datos. Sin embargo, es necesario tener presente que la decisión final en un contexto de aplicación real debe basarse en una evaluación exhaustiva de las necesidades, requisitos, consideraciones operativas y técnicas específicas de la entidad interesada en implementar la Blockchain.

Por lo tanto, una alternativa a la presente metodología sería la implementación de una Blockchain publica, teniendo en consideración todo lo que esto conlleva, sus factores de regulaciones y cumplimiento, los costos, la escalabilidad, el control sobre los consensos de los protocolos, entre muchas otras variables.

II. Tipos de investigación

La metodología que se presentará a continuación contiene una investigación experimental, esta nos ayudaría a verificar como se comportaría el sistema resultante manipulando o alterando ciertos parámetros en la creación de la metodología, como técnicas de recolección de datos tenemos los resultados que arrojen algunos experimentos controlados o comparación de resultados de investigaciones externas.

III. Determinación de Población y Muestra

- **Población**: Grupo especifico de usuarios activos del sector bancario que participen en transacciones financieras.
- **Muestra**: Muestra representativa de usuarios y sus transacciones para analizar cómo la implementación de la tecnología blockchain impacta la seguridad, rendimiento y transparencia en las transacciones financieras.

- **Justificación**: Esta muestra permitiría evaluar cómo la adopción de blockchain afecta directamente a los usuarios y cómo los cambios se reflejan en las transacciones.

IV. Técnicas de análisis

- **Desempeño**: Métricas y Gráficos de rendimiento.
- **Análisis de eficiencia**: Uso de recursos, eficiencia energética.
- **Análisis de eficacia**: Trazabilidad y auditoria y cumplimiento de contratos inteligentes.
- **Análisis comparativo**: Benchmarking con otros tipos de configuraciones o de redes Blockchain.

V. Implementación

Pasos previos al desarrollo técnico:

Paso 1: Definición de Requisitos

Requisitos funcionales:

- **Registro de transacciones**: el sistema debe ser capaz de registrar de manera segura y transparente todas las transacciones realizadas por los diferentes actores.

- **Gestión de identidad**: Debe existir un sistema de gestión de identidad seguro y eficiente que permita a los validadores autenticarse de manera segura en la red.

- **Creación y ejecución de contratos inteligentes**: el sistema debe permitir la ejecución y gestión de contratos inteligentes para automatizar procesos y garantizar su cumplimiento.

- **Seguridad:** se debe de tener control de no alteración de las transacciones realizadas ni de los contratos inteligentes desplegados.

- **Escalabilidad**: La solución debe ser capaz de manejar un alto volumen de transacciones sin degradar el rendimiento o la seguridad.

Requisitos no funcionales:

- **Rendimiento**
  La plataforma debe ser capaz de manejar una tasa mínima de 1000 transacciones por segundo para acomodar la demanda en momentos de alta actividad.

El tiempo requerido para que un bloque de transacciones se confirme y se agregue a la cadena debe ser consistente y no superar 10 segundos para garantizar la eficiencia en la confirmación de transacciones.

El tiempo de respuesta para consultar el estado de una transacción o acceder a datos históricos en la cadena de bloques no debe exceder 2 segundos, asegurando una experiencia fluida para los usuarios.

- **Latencia**
  La latencia de red entre nodos de la blockchain debe mantenerse dentro de un rango aceptable, como 2000 milisegundos, para garantizar una comunicación rápida y estable entre los participantes de la red.

- **Disponibilidad**
  La plataforma debe estar disponible el 99.00% del tiempo para evitar interrupciones en los servicios bancarios.

- **Seguridad**
  La blockchain debe cumplir con los más altos estándares de seguridad, incluyendo cifrado robusto y protección contra ataques.

- **Capacidad de auditoria**
  Debe ser posible auditar y rastrear todas las transacciones y cambios realizados en la plataforma.

Paso 2: Diseñar la Arquitectura de la Blockchain

Se define la arquitectura de Blockchain tipo Hyperledger Fabric, incluyendo la cantidad de nodos, la topología de la red y la configuración de los canales privados.

Entities: Personas, grupos, organizaciones o cualquier otro tipo de entidad caracterizada por ser autoridades y que interactúan con uno o varios nodos.
Channels: Son canales que sirven como medio de comunicación privada entre los nodos.
Founder: Personas, grupos, organizaciones o cualquier otro tipo de entidad que realizaron el aprovisionamiento inicial de la red blockchain estableciendo el sistema de Chaincodes y de Ledgers.
Ledger: Libro de contabilidad único de cada nodo, también se puede relacionar como si fuera una base de datos propia del nodo.
System Ledger: Este libro de contabilidad contiene el *Actual World State*, que contiene el estado actual del sistema entero y almacena los datos en una base de datos de clave-valor.
Peers: Son los Nodos pertenecientes la red.

Anchor: Validador de contratos inteligentes.
Endorser Validador de consensos y firmas, estos pueden ser los mismos o diferentes nodos.
Ordering Service: encargado de recibir las solicitudes de transacciones, la creación de los bloques y la distribución de estos en la red.
Client: sería un actor fundamental y representa ese desencadenador de acciones a la red por medio de los canales que existan, los cuales a su vez activan los nodos con sus lógicas programadas en los contratos inteligentes.

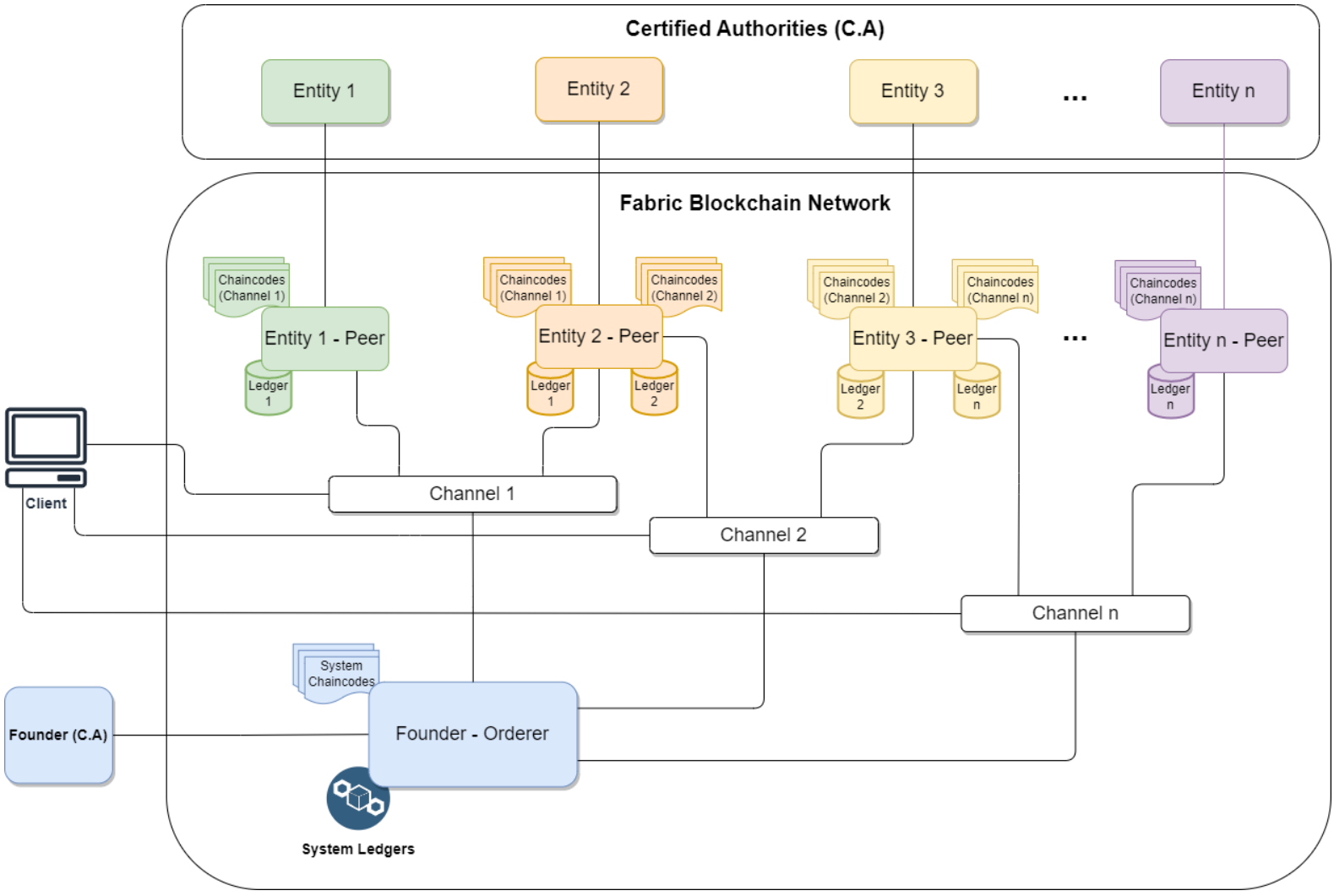


Figura 1. Adaptación de la arquitectura general de una red Blockchain basada en Hyperledger Fabric tomada en (Hyperledger Foundation, 2023)

A continuación, se describen dos posibles implementaciones de esta arquitectura para nuestro caso de estudio del sector financiero, específicamente en un banco.

- **Uso de blockchain internamente en la compañía**: En esta implementación se podría tomar al banco como entidad *Founder* de la red, y a su vez, las entidades serían departamentos, ecosistemas o grupos específicos internos dentro del banco, como por ejemplo el departamento de ventas, el de análisis de créditos, el departamento legal, el área de riesgos o contabilidad. Cada uno de estos departamentos o áreas se consideran autoridades certificadas para interactuar en la red por medio de sus propios nodos.

  El Cliente sería la página web o el aplicativo móvil realizando *querys/updates/requests*, En este caso en específico sería el usuario realizando por ejemplo una solicitud de préstamo al banco. Las entidades son los diferentes departamentos o áreas internas del banco.

- **Uso de blockchain para interactuar con entidades externas a la compañía**: Aquí lo que se busca es que el banco sea también entidad *Founder* como el anterior caso, pero las diferentes entidades serían otras organizaciones externas al banco, como por ejemplo un ente gubernamental, otra empresa o incluso otro banco. De esta manera podríamos tener una red blockchain privada y segura con actores diferentes a nuestro propio banco o empresa.

  El Cliente sería la página web o el aplicativo móvil realizando *querys/updates/requests*, En este caso en específico sería un usuario del banco solicitando una transferencia a otro banco. Las dos entidades existentes en este caso sería el banco A (nuestro banco) y el banco B (banco externo)

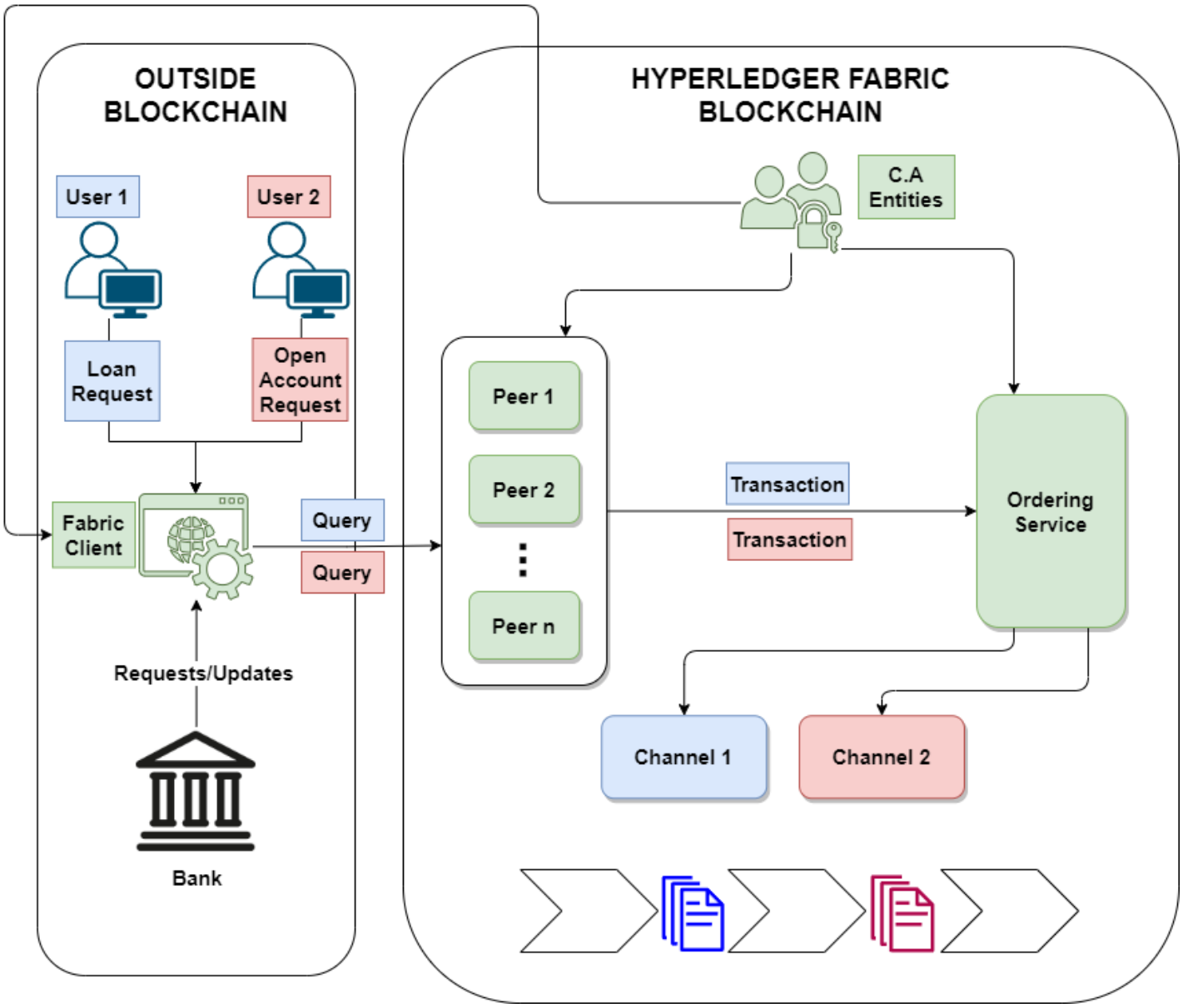


Figura 2. Arquitectura del aplicativo usando Blockchain Hyperledger Fabric para uso interno del banco

Seguidamente, procedemos con el diseño de los *smart contracts* (contratos inteligentes) que automatizarán los procesos financieros y las reglas de negocio mediante el código.

Consensus Mechanism: Existen varias políticas populares de consenso en los *Smart contracts*, como por ejemplo el 50% + 1 de los nodos para aprobar las transacciones, o 1 nodo por entidad validadora.

Cryptocurrency: Ninguna, Hyperledger Fabric no posee una criptomoneda nativa.

Paso 3: Configurar la Infraestructura

Se procede a instalar todas las herramientas necesarias de Hyperledger Fabric y configurar los nodos de la red.

Se puede decidir ejecutar la red *onpremise* en servidores locales de la organización o ejecutarlo con computación en nube y aprovechar sus beneficios (Habib et al., 2022) como la descentralización, mejora en protección de datos, seguimiento de servicios mejorado y la tolerancia a errores. También es posible realizar una combinación de ambos.

Paso 4: Desarrollar y Probar Smart Contracts

Desarrollamos los *smart contracts* utilizando el lenguaje de programación compatible con Hyperledger Fabric se podría escoger entre los lenguajes de programación: *Chaincode* en Go o Node.js.

Implementamos las 3 fases (Aleksieva et al., 2020b) del desarrollo de contratos inteligentes.

- **Analizar**: Necesitamos identificar cual o cuales son los problemas para resolver mediante la lógica programada en los contratos inteligentes, así como los vínculos entre ellos.
- **Diseñar**: Basados en la información obtenida en la fase anterior, desarrollamos un esquema de objetos. Estos contendrán las variables y las interacciones con otros objetos.
- **Implementar**: Los componentes de los contratos inteligentes son implementados y distribuidos a través de la red.

Realizamos pruebas de los *smart contracts* en un entorno de desarrollo controlado para garantizar su correcto funcionamiento y seguridad.

Paso 5: Configurar la Identidad y la Autenticación

En este paso realizamos la implementación de la autenticación de los participantes en la red mediante certificados digitales y mecanismos de identidad.

Se utiliza Hyperledger Fabric CA (Certificate Authority) para emitir y gestionar certificados.

Paso 6: Implementar y Configurar Canales Privados

Se configuran canales privados para permitir transacciones y comunicación selectiva entre los participantes específicos.

Paso 7: Realizar Pruebas de Carga y Seguridad

Aquí se pretende evaluar la capacidad de la red para manejar un alto volumen de transacciones por segundo.

También se realiza pruebas de seguridad para identificar posibles vulnerabilidades y garantizar la protección de los datos financieros sensibles.

Paso 8: Implementar la Solución y Capacitar a los Usuarios

Desplegar la red blockchain en el entorno de producción de manera controlada.

Se recomienda altamente proporcionar una guía detallada de uso para los usuarios en caso de ser necesario, y también una capacitación a los empleados del banco, proveedores y otros actores o participantes sobre cómo interactuar con la plataforma.

Paso 9: Monitorear y Mantener

Se puede establecer un sistema de monitoreo continuo para asegurarse de que la red funcione sin problemas y que los *smart contracts* se estén ejecutando correctamente mediante la herramienta **Hyperledger Explorer** (Hyperledger Explorer & Hyperledger Foundation, 2019). Para ello se debe clonar el repositorio de la herramienta en su página oficial de github, después de esto se accede al directorio clonado y se realiza la configuración necesaria en el archivo ***config.json***. Seguidamente, se generan los archivos de conexión usando las herramientas proporcionadas por Hyperledger Fabric, como la herramienta ***configtxgen*** para generar los archivos ***connection-profile*** y ***config.json.*** Finalmente, se edita en el directorio del Hyperledger explorer el archivo docker-compose.yaml para asegurarse de que todo se encuentre configurado correctamente.

Al realizar todo el proceso anterior, se puede ejecutar el docker compose y acceder a la interfaz que proporciona la herramienta en el navegador web.

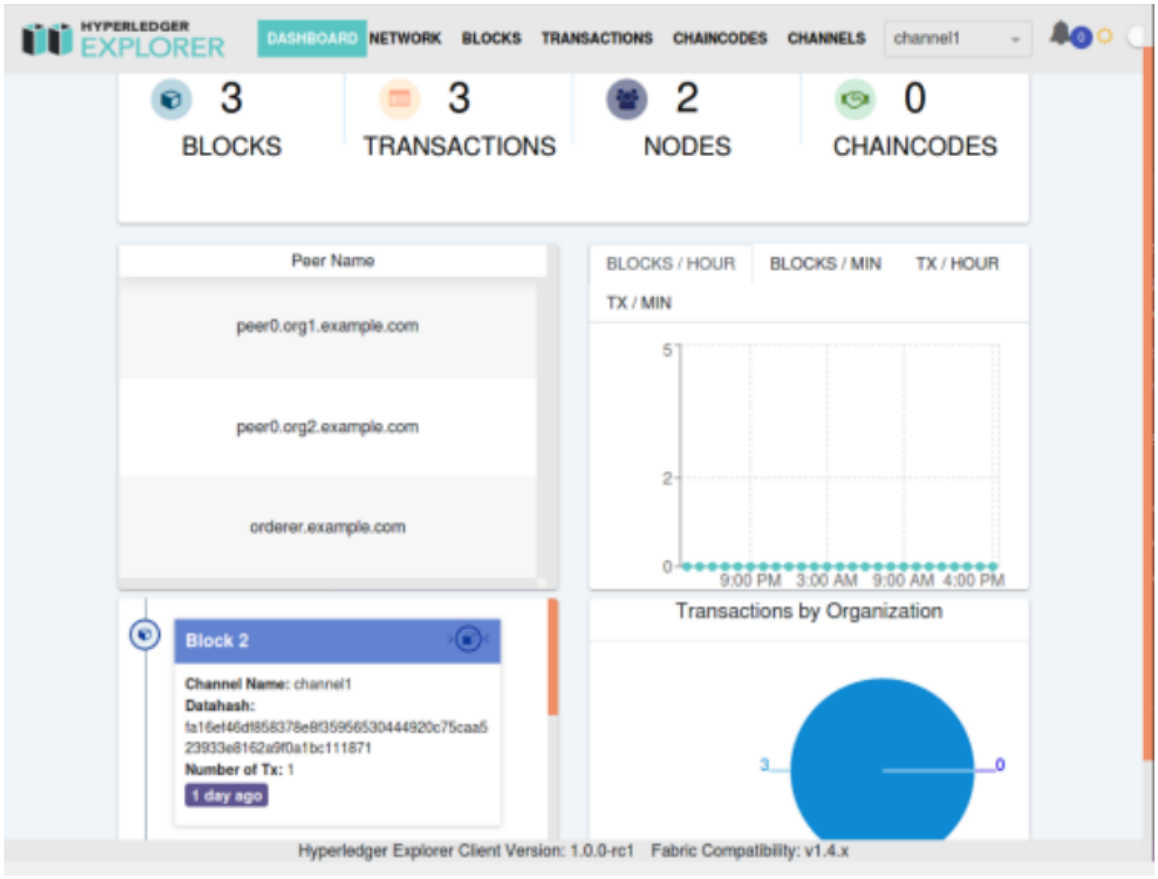


Figura 3. Hyperledger Explorer tomada de la figura 10 de (Aleksieva et al., 2020b)

CASO DE USO

Las transferencias transfronterizas generalmente son lentas, costosas y propensas a errores, con un alto riesgo de fraude y sin visibilidad en tiempo real para los clientes. Los bancos pueden mejorar drásticamente estas transferencias transfronterizas al implementar un contrato inteligente en una plataforma Blockchain privada con Hyperledger Fabric.

Esto agiliza las transacciones, reduce costos, aumenta la seguridad y la transparencia, y ofrece al cliente visibilidad en tiempo real, beneficiando a la institución financiera y a sus usuarios.

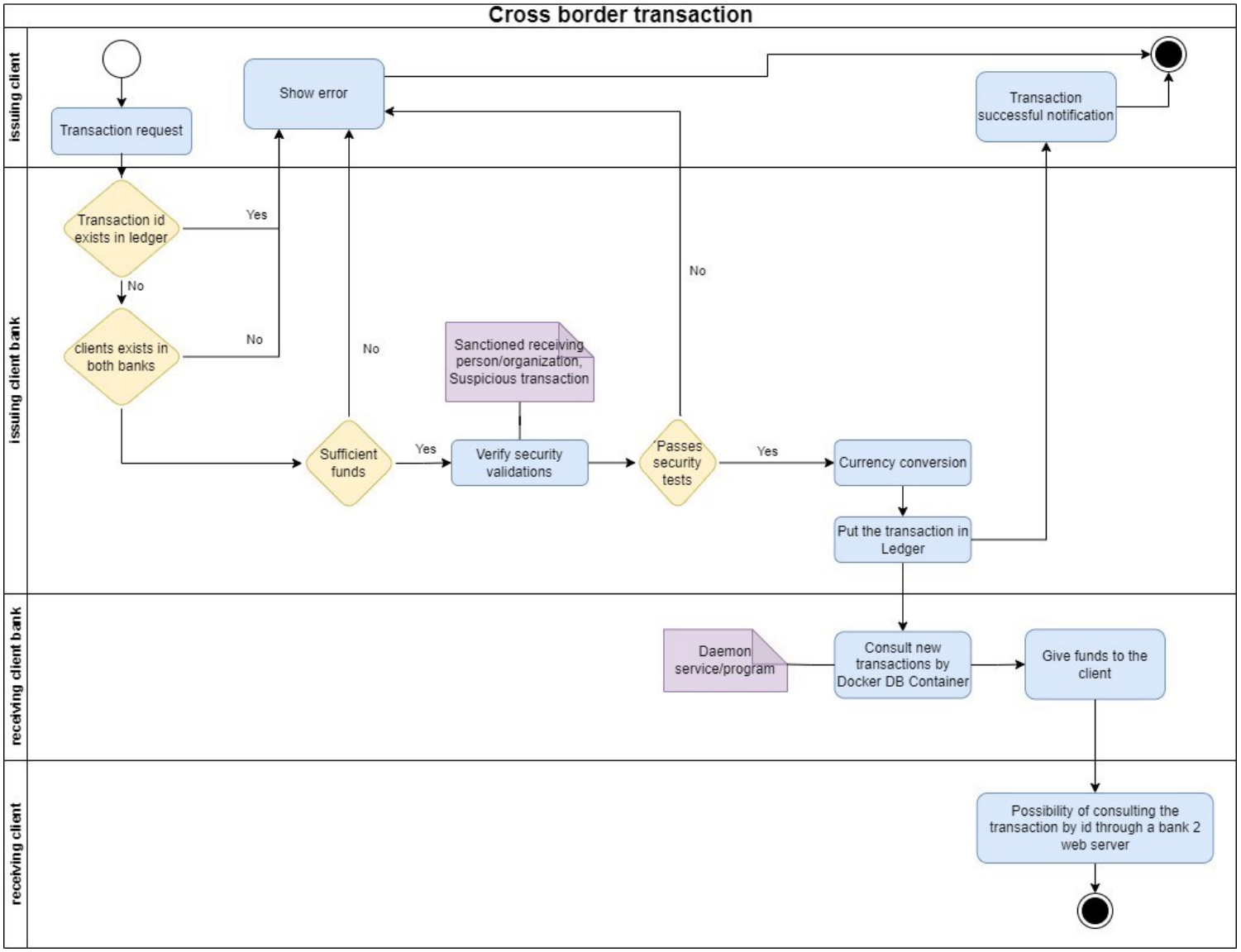


Figura 4. Flujo de secuencia BPMN para el caso de uso en un banco

Teniendo en cuenta los pasos previos, se procede a realizar el desarrollo técnico

La Blockchain Hyperledger Fabric se destaca por su diseño modular, lo que la hace muy flexible y adaptable a diferentes necesidades. Si se requiere aumentar su capacidad de procesamiento (escalado horizontal), se recomienda usar Docker y Kubernetes.

Para este caso de uso se tiene la siguiente distribución

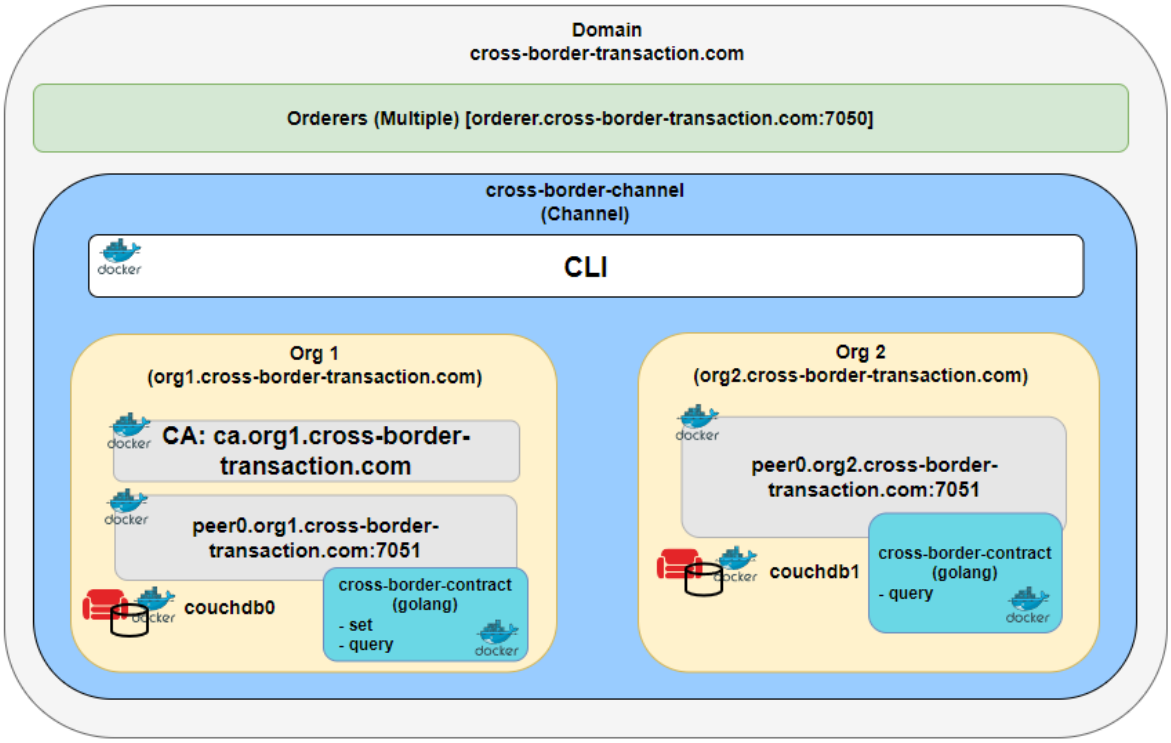


Figura 5. Distribución de las tecnologías

1. Tener prerreq.sh y ejecutarlo para instalar los prerrequisitos necesarios para la red Blockchain.

    - Utilidades como unzip, wget, git, tree, telnet, httping, tcpping
    - Java
    - Maven
    - Docker, Docker compose
    - Node

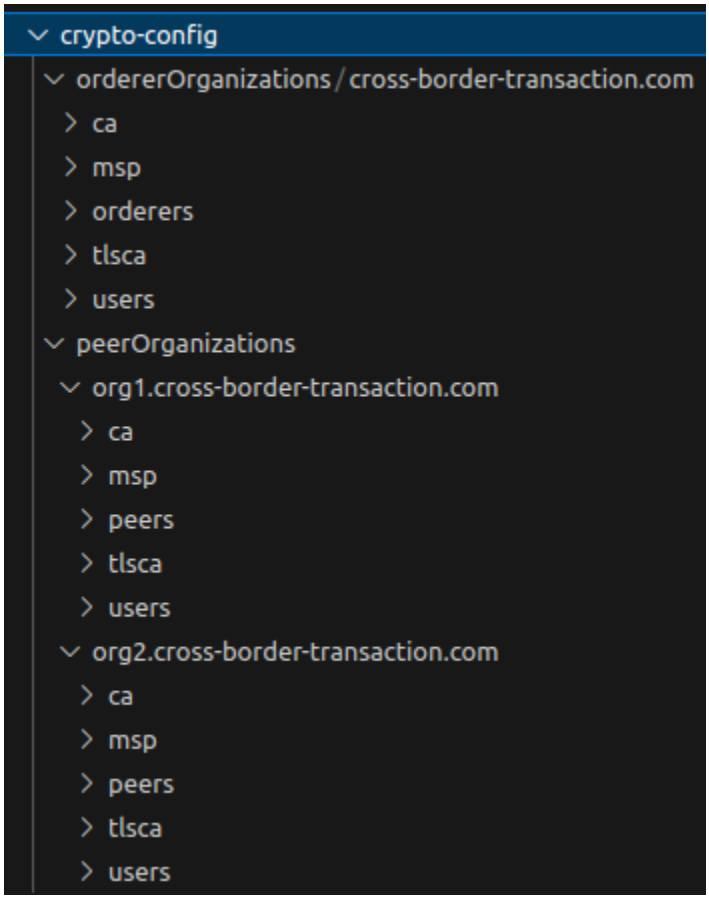
- Golang
- Hyperledger Fabric (para poder incluir en nuestro PATH los archivos y programas binarios necesarios)

2. Se configura el crypto-config.yaml para establecer todo el material criptográfico necesario para la red. Es aquí donde se indican cuantos peers va a tener cada organización y cuantos usuarios además del administrador.

```yaml
OrdererOrgs:
  - Name: Orderer
    Domain: cross-border-transaction.com
    EnableNodeOUs: true
    Specs:
      - Hostname: orderer
        SANS:
          - localhost
PeerOrgs:
  - Name: Org1
    Domain: org1.cross-border-transaction.com
    EnableNodeOUs: true
    Template:
      Count: 1
      SANS:
        - localhost
    Users:
      Count: 1
  - Name: Org2
    Domain: org2.cross-border-transaction.com
    EnableNodeOUs: true
    Template:
      Count: 1
      SANS:
        - localhost
    Users:
      Count: 1
```

3. Ejecutar el programa cryptogen con el comando:
   - `cryptogen generate –config=./crypto-config.yaml`

   Esto crea el ordererOrganization y los peerOrganizations en la carpeta crypto-config ubicada en la raíz



4. Crear archivo configtx.yaml
   Orderer.cross-border-transaction.com:7050
   En este archivo se configuran varias características, entre ellas se encuentran nombres, políticas y tipos de ordenamientos:

   - Configurar el ordererOrg y los peers por cada organización
   - Capacidades, para establecer compatibilidad entre versiones
   - Políticas para la aplicación
   - Tipo de ordenamiento/Orderer type (aquí tener en consideración las diferencias entre los tipos de ordenamiento existentes)

Existen diferentes tipos de ordenadores que se pueden utilizar. Cada tipo de ordenador tiene sus propias ventajas y desventajas, por lo que es importante elegir el tipo adecuado para un caso de uso especifico.

Tipos de ordenadores

La siguiente tabla describe algunos los diferentes tipos de ordenadores disponibles en Hyperledger Fabric:

Tipo de ordenador	Descripción	Ventajas	Desventajas
Solo	Un único ordenador valida las transacciones y ordena el bloque.	Simple de configurar y administrar.	No es tolerante a fallos. Si el ordenador falla, la red se detiene.
Kafka	Utiliza Apache Kafka como sistema de mensajería para distribuir las transacciones a los ordenadores.	Escalable y tolerante a fallos. Puede manejar un alto volumen de transacciones.	Más complejo de configurar y administrar que Solo.
Raft	Implementa el algoritmo Raft para la replicación de datos y el consenso.	Escalable, tolerante a fallos y ofrece alta disponibilidad.	Más complejo de configurar y administrar que Solo.
Etcdraft	Implementa el algoritmo etcdraft para la replicación de datos y el consenso.	Similar a Raft, pero con menor latencia y mayor eficiencia en el uso de recursos.	Menos tolerante a fallos que Raft.

Tabla 2. Comparación entre algunos tipos de ordenadores en Hyperledger Fabric

- Configuración de políticas a nivel del canal.
- Y por último la configuración de los perfiles (el *orderer genesis* y el canal)

5. Se crea la carpeta para el canal privado

   - `mkdir channel-artifacts`

6. Se crea el bloque genesis de la siguiente manera ejecutando el comando:

   - `configtxgen -profile TwoOrgsOrderGenesis – channelID system-channel -outputBlock ./channel-artifacts/genesis.block`

7. Se hace algo similar para poder crear la configuración del canal ejecutando el comando:

- `configtxgen -profile TwoOrgsChannel  -outputCreateChannelTx ./channel-artifacts/channel.tx -channelID cross-border-channel`

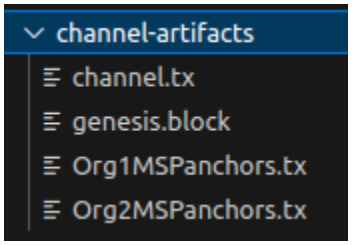
8. Se crean los archivos de configuración de los *anchorPeers*
   Para ello se ejecuta una transacción de actualización del *anchorPeer* para una organización en particular ejecutando el comando:

   - `configtxgen -profile TwoOrgsChannel -outputAnchorPeersUpdate ./channel-artifacts/Org1MSPanchors.tx -channelID cross-border-channel -asOrg Org1MSP`

   Y se realiza lo mismo para las otras organizaciones, en este caso para la segunda organización.

   - `configtxgen -profile TwoOrgsChannel -outputAnchorPeersUpdate ./channel-artifacts/Org2MSPanchors.tx -channelID cross-border-channel -asOrg Org2MSP`

9. Se visualiza, tras estos últimos 4 pasos anteriores, la creación de los artefactos del canal.



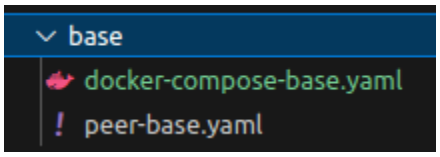
10. Se crea la carpeta /base en la raíz con el comando.
    - `mkdir base`

    Y dentro de la misma se configura el archivo peer-base.yaml
    - Se asigna una imagen con una versión de Hyperledger
    - Se tienen variables de ambiente
    - El directorio del peer base
    - Y el comando que hará que se ejecute el peer

```yaml
version: '2'

services:
  peer-base:
    image: hyperledger/fabric-peer:2.2.0
    environment:
      - CORE_VM_ENDPOINT=unix:///host/var/run/docker.sock
      # the following setting starts chaincode containers on the same
      # bridge network as the peers
      # https://docs.docker.com/compose/networking/
      # ---CHANGED---
      - CORE_VM_DOCKER_HOSTCONFIG_NETWORKMODE=cross-border-transaction-network_basic
      - FABRIC_LOGGING_SPEC=INFO
     # - FABRIC_LOGGING_SPEC=DEBUG
      - CORE_PEER_TLS_ENABLED=true
      - CORE_PEER_GOSSIP_USELEADERELECTION=true
      - CORE_PEER_GOSSIP_ORGLEADER=false
      - CORE_PEER_PROFILE_ENABLED=true
      - CORE_PEER_TLS_CERT_FILE=/etc/hyperledger/fabric/tls/server.crt
      - CORE_PEER_TLS_KEY_FILE=/etc/hyperledger/fabric/tls/server.key
      - CORE_PEER_TLS_ROOTCERT_FILE=/etc/hyperledger/fabric/tls/ca.crt
    working_dir: /opt/gopath/src/github.com/hyperledger/fabric/peer
    command: peer node start
```

11. En esta misma carpeta base se crea docker-compose-base.yaml
    Aquí se especifica la configuración que se quiere del *orderer* y de los *peers* de cada organización con sus dominios, variables, volúmenes y puertos dentro del contenedor.



    Se debe tener en cuenta la conexión que se hará a través de los contenedores que se van a levantar en docker para cada *peer*

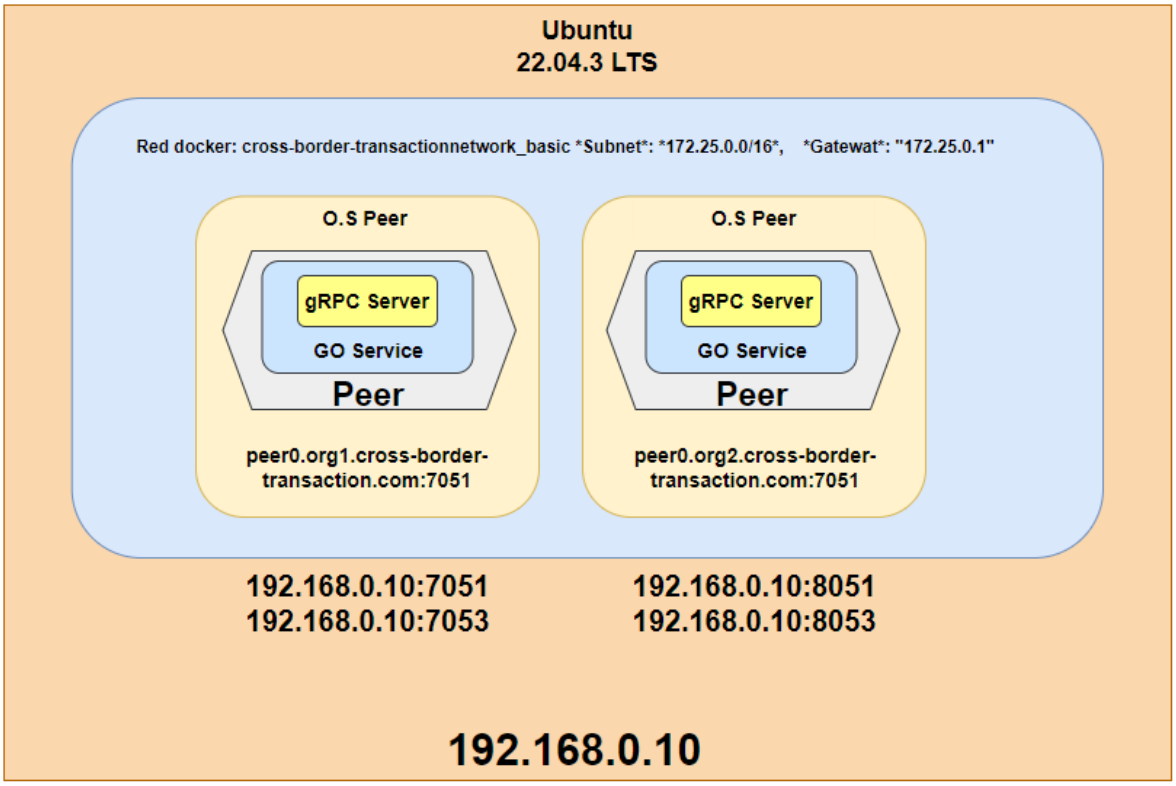


Figura 6. Conexión de contenedores con sus puertos

12. Se creamos el archivo docker-compose-cli-couchdb.yaml para poder crear los servicios dockerizados de:

    - El *orderer*
    - El *peer0* de la organización1
    - El *peer0* de la organizacion2
    - Autoridad de certificación (organizacion1)
    - CLI (en donde por defecto se carga la identidad de la organizacion1 que es la Autoridad de Certificación, si se desea utilizar estos comandos con otras organizaciones, se debe de modificar la variable de entorno que se define en el servicio en tiempo de ejecución)
    - *couchdb0* para la organizacion1
    - *couchdb1* para la organizacion2

    Nota: se decide usar *couchdb*, sin embargo, se pueden usar otras tecnologías para poder guardar el estado global/world state como por ejemplo leveldb, etc.

13. Se levanta una herramienta docker llamada *portainer_data* con el comando.

- `docker volumen create portainer_data`

Luego se levanta el contenedor con esta herramienta

- `docker run -d -p 8000:8000 -p 9000:9000 -v /var/run/docker.sock:/var/run/docker.sock -v portainer_data:/data portainer/portainer`

En caso de que no se tenga esta herramienta al ejecutar el comando, docker descarga la última versión y levanta el contenedor.

```
camilo@camilo:~/hyperledger-fabric-first-step/cross-border-transaction-network$ docker run -d -p 8000:8000 -p 9000
:9000 -v /var/run/docker.sock:/var/run/docker.sock -v portainer_data:/data portainer/portainer
Unable to find image 'portainer/portainer:latest' locally
latest: Pulling from portainer/portainer
772227786281: Pull complete
96fd13befc87: Pull complete
0bad1d247b5b: Pull complete
b5d1b01b1d39: Pull complete
Digest: sha256:47b064434edf437badf7337e516e07f64477485c8ecc663ddabbe824b20c672d
Status: Downloaded newer image for portainer/portainer:latest
c2b1ef7e5d44b36dfb76c06bd0926e197b4bf810694656979e5373fe9c0c4d1d
camilo@camilo:~/hyperledger-fabric-first-step/cross-border-transaction-network$
```

14. Se ingresa al localhost:9000 utilizando un navegador para dar un usuario administrador con su respectiva contraseña y configurar *portainer*. Esto debido a que, si no se configura esto en un lapso corto de tiempo, ya no permitirá configurar *portainer*.

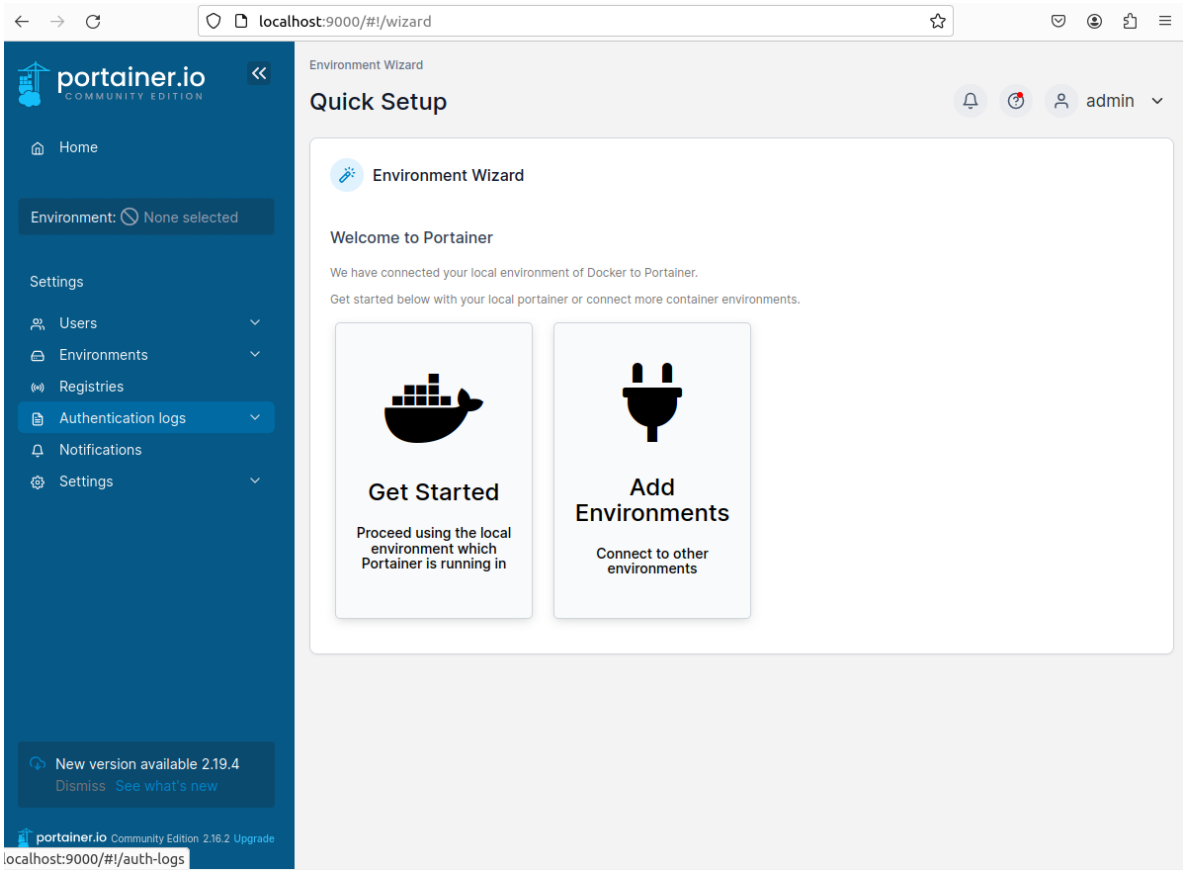


Figura 7. Wizard inicial de la herramienta Portainer

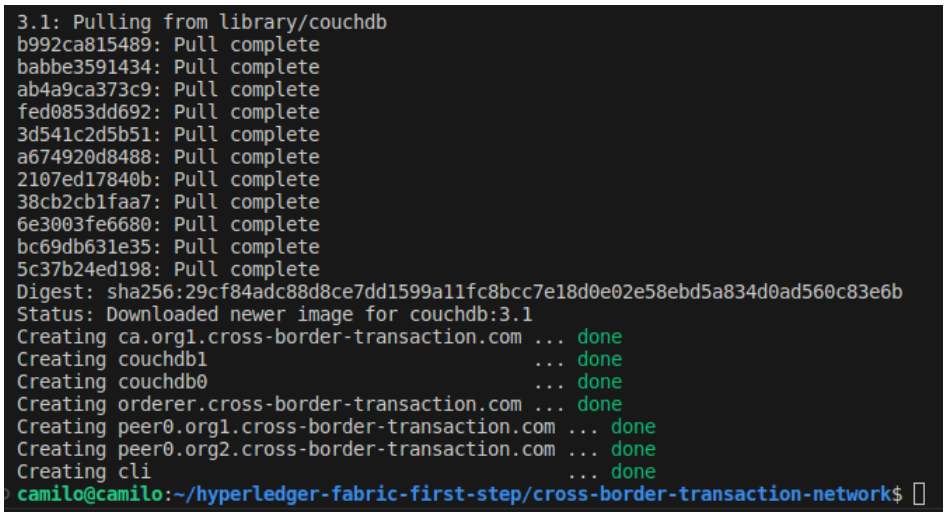
15. Se regresa a la terminal y se definen las variables de ambientes que ayudaran a levantar la red. Para estoy hay que ubicarse en la raíz de la carpeta cross-border-transaction-network en la consola.

    - `export CHANNEL_NAME=cross-border-channel`
    - `export VERBOSE=false`
    - `export FABRIC_CFG_PATH=$PWD`

16. Se levanta *couchdb* con el comando.

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

Al finalizar el comando se puede ver como levanta los diferentes servicios.



También se puede visualizar en la herramienta *portainer* que se abrió en el navegador, revisando los contenedores.

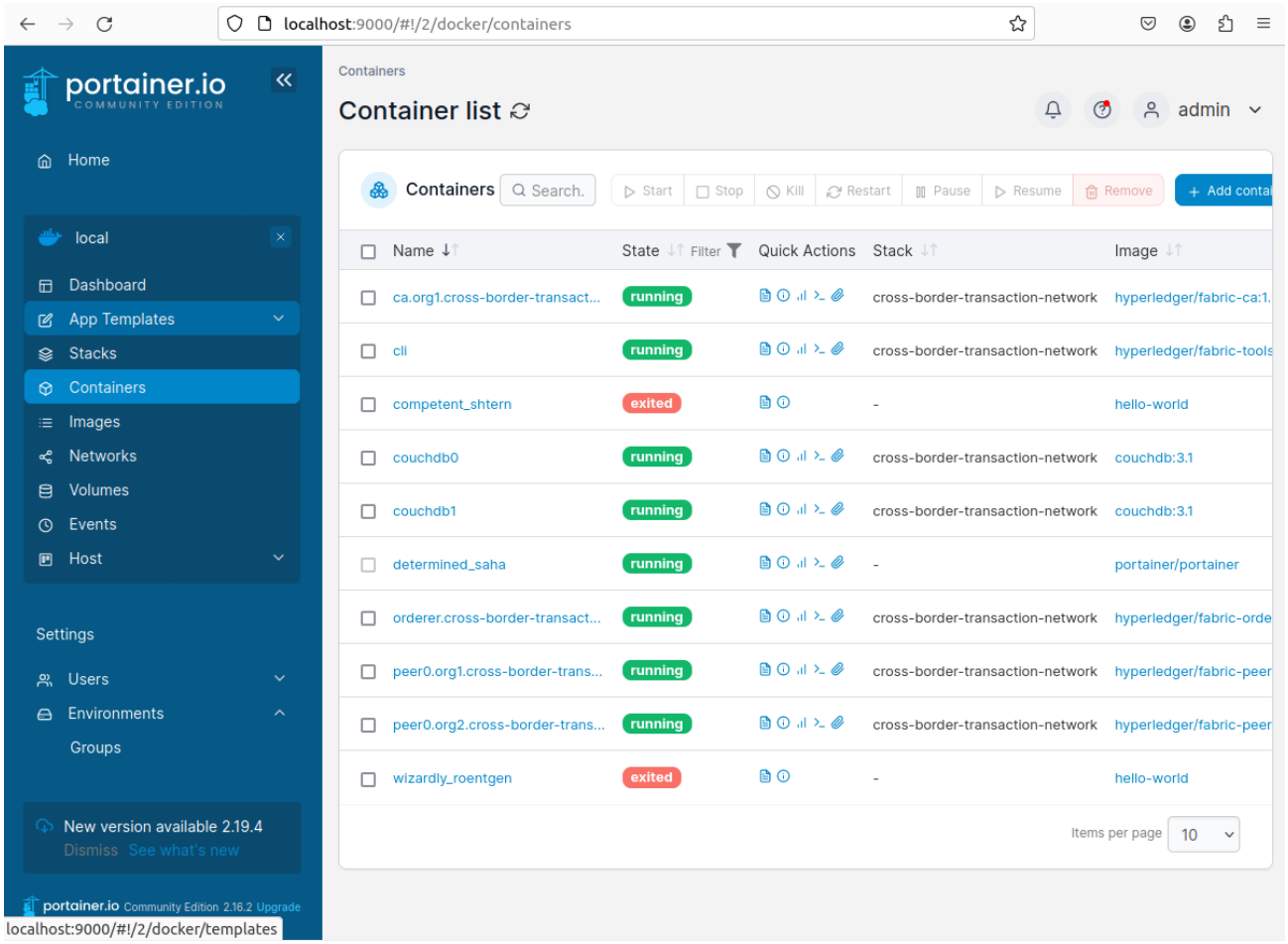


Figura 8. Lista de contenedores en Portainer

17. Hay que conectarse a la consola de comandos del contenedor cli para poder crear el canal, en este caso se hizo utilizando portainer mediante el ícono “Exec Console” en la columna Quick Actions. Se da clic en conectar y aparecerá la consola del contenedor cli.

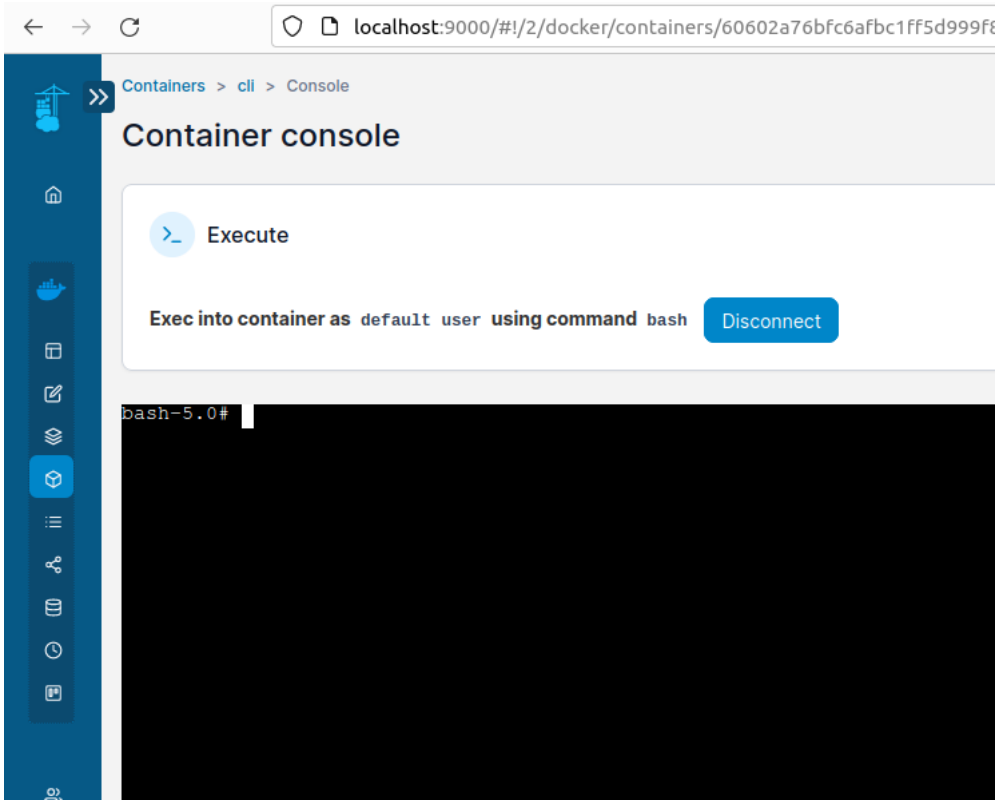


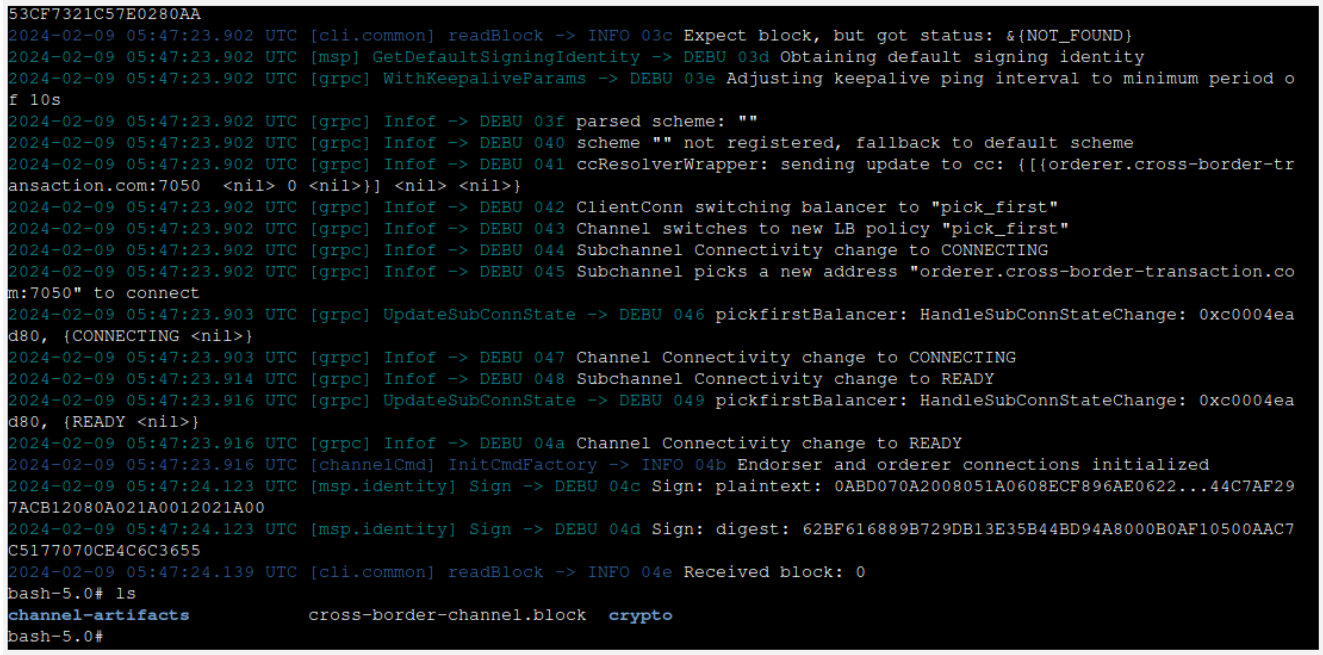
Figura 9. Consola del contenedor CLI en Portainer

Como se posee la información de channel-artifacts (ya que se encuentra en ese directorio home) se puede crear el canal con las configuraciones que se proporcionaron en los pasos anteriores. Para esto se procede a ejecutar los siguientes comandos.

- `export CHANNEL_NAME=cross-border-channel`

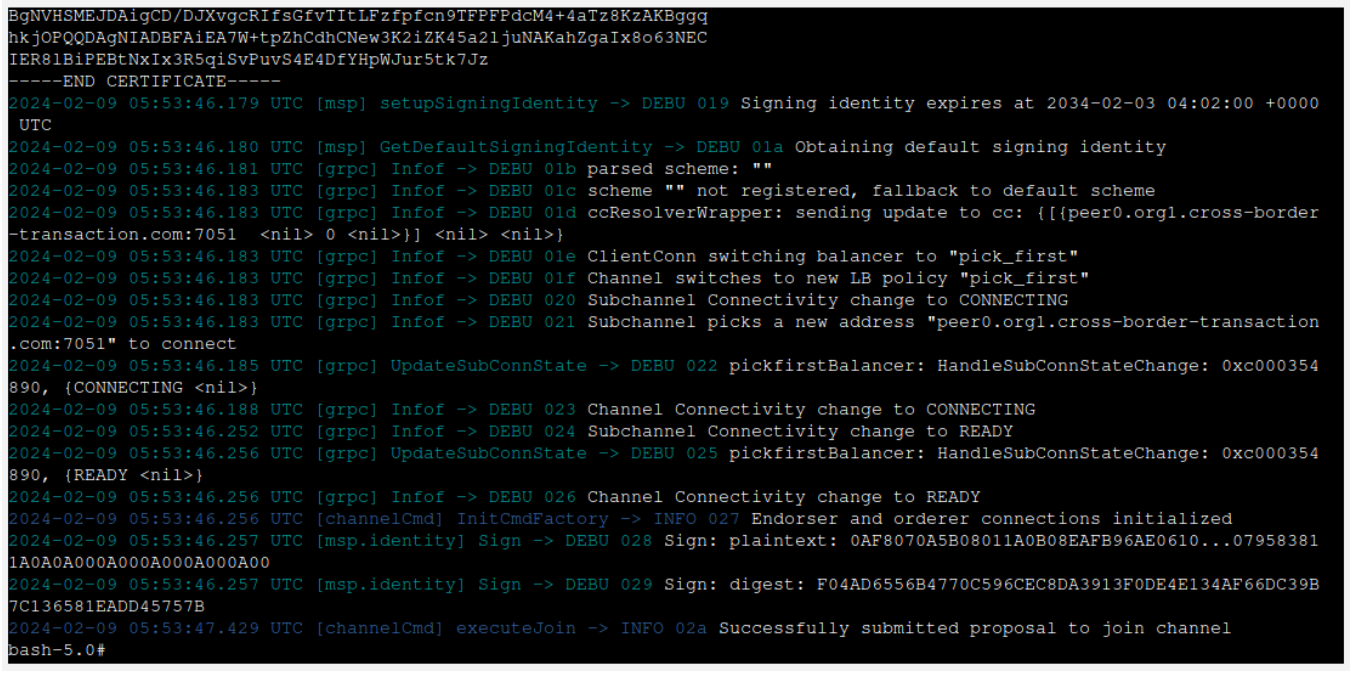
- ```
  peer channel create -o orderer.cross-border-
  transaction.com:7050 -c $CHANNEL_NAME -f ./channel-
  artifacts/channel.tx --tls true --cafile
  /opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/orde
  rerOrganizations/cross-border-
  transaction.com/orderers/orderer.cross-border-
  transaction.com/msp/tlscacerts/tlsca.cross-border-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

Al ejecutarlos se verifica que se crea un bloque llamado cross-border-channel.block. Este será el bloque del canal y fue creado exitosamente.



18. Ahora se debe integrar la primera organización a este canal. Para ello se ejecutan los siguientes comandos.

    - `peer channel join -b cross-border-channel.block`



19. Para unir la Segunda organización o cualquier otra aparte de la organización principal o CA, se debe indicar una metadata adicional al comando anterior, esto se logra con el siguiente comando.

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

```
KwYDVR0jBCQwIoAgchgOPIF5CNpWiGdAYXu4qHJHicVSMahuC88tuNCX1LAwCgYI
KoZIzj0EAwIDSAAwRQIhALCzdQWvIvihdvZiWmYecchXKMUYqc9Yc55uc4JAcoMF
AiATScAzzVV6MQYq5UoClzI1n0XfbaXYTK4sexQBOFuaUQ==
-----END CERTIFICATE-----
2024-02-09 06:12:09.107 UTC [msp] setupSigningIdentity -> DEBU 019 Signing identity expires at 2034-02-03 04:02:00 +000
0 UTC
2024-02-09 06:12:09.120 UTC [msp] GetDefaultSigningIdentity -> DEBU 01a Obtaining default signing identity
2024-02-09 06:12:09.135 UTC [grpc] Infof -> DEBU 01b parsed scheme: ""
2024-02-09 06:12:09.137 UTC [grpc] Infof -> DEBU 01c scheme "" not registered, fallback to default scheme
2024-02-09 06:12:09.141 UTC [grpc] Infof -> DEBU 01d ccResolverWrapper: sending update to cc: {[{peer0.org2.cross-borde
r-transaction.com:7051  <nil> 0 <nil>}] <nil> <nil>}
2024-02-09 06:12:09.141 UTC [grpc] Infof -> DEBU 01e ClientConn switching balancer to "pick_first"
2024-02-09 06:12:09.141 UTC [grpc] Infof -> DEBU 01f Channel switches to new LB policy "pick_first"
2024-02-09 06:12:09.141 UTC [grpc] Infof -> DEBU 020 Subchannel Connectivity change to CONNECTING
2024-02-09 06:12:09.142 UTC [grpc] Infof -> DEBU 021 Subchannel picks a new address "peer0.org2.cross-border-transactio
n.com:7051" to connect
2024-02-09 06:12:09.144 UTC [grpc] UpdateSubConnState -> DEBU 022 pickfirstBalancer: HandleSubConnStateChange: 0xc00016
ea70, {CONNECTING <nil>}
2024-02-09 06:12:09.150 UTC [grpc] Infof -> DEBU 023 Channel Connectivity change to CONNECTING
2024-02-09 06:12:09.156 UTC [grpc] Infof -> DEBU 024 Subchannel Connectivity change to READY
2024-02-09 06:12:09.156 UTC [grpc] UpdateSubConnState -> DEBU 025 pickfirstBalancer: HandleSubConnStateChange: 0xc00016
ea70, {READY <nil>}
2024-02-09 06:12:09.156 UTC [grpc] Infof -> DEBU 026 Channel Connectivity change to READY
2024-02-09 06:12:09.156 UTC [channelCmd] InitCmdFactory -> INFO 027 Endorser and orderer connections initialized
2024-02-09 06:12:09.157 UTC [msp.identity] Sign -> DEBU 028 Sign: plaintext: 0AFC070A5B08011A0B08B98497AE0610...0795838
11A0A0A000A000A000A000A00
2024-02-09 06:12:09.157 UTC [msp.identity] Sign -> DEBU 029 Sign: digest: 8ED1A16936CDEA4A05A715DEB6EECC94FD66861F024B6
94DD3F134DB632DBDF7
2024-02-09 06:12:10.660 UTC [channelCmd] executeJoin -> INFO 02a Successfully submitted proposal to join channel
bash-5.0#
```

20. Ahora se hace la configuración para que todas las organizaciones de la red tengan establecidos los *anchorPeer* correspondientes utilizando los *anchorPeers* que se encuentran en la carpeta de channel-artifacts. Se realiza utilizando el siguiente comando.

    - `peer channel update -o orderer.cross-border-transaction.com:7050 -c $CHANNEL_NAME -f ./channel-artifacts/Org1MSPanchors.tx --tls true --cafile`

```
/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/orde
rerOrganizations/cross-border-
transaction.com/orderers/orderer.cross-border-
transaction.com/msp/tlscacerts/tlsca.cross-border-
transaction.com-cert.pem
```

Y ya se puede ver que el *anchorPeer* para la organización 1 está establecido.

```
2024-02-09 14:38:02.518 UTC [msp] GetDefaultSigningIdentity -> DEBU 01a Obtaining default signing identity
2024-02-09 14:38:02.518 UTC [channelCmd] InitCmdFactory -> INFO 01b Endorser and orderer connections initialized
2024-02-09 14:38:02.518 UTC [msp.identity] Sign -> DEBU 01c Sign: plaintext: 0AFB060A074F7267314D535012EF062D...41646D69
6E7312002A0641646D696E73
2024-02-09 14:38:02.518 UTC [msp.identity] Sign -> DEBU 01d Sign: digest: 290B4191E6B178574F3D47DC9458D4FE1B397B5FA51D77
0B13643FD96B753275
2024-02-09 14:38:02.518 UTC [msp.identity] Sign -> DEBU 01e Sign: plaintext: 0ABD070A2008021A0608CAF198AE0622...471A653F
75EA71D59EE57F46792D4157
2024-02-09 14:38:02.518 UTC [msp.identity] Sign -> DEBU 01f Sign: digest: 2E8C566473DBF80EB4A25A2949981406412E8492690755
677FF06DB1455EE71E
2024-02-09 14:38:02.525 UTC [grpc] WithKeepaliveParams -> DEBU 020 Adjusting keepalive ping interval to minimum period o
f 10s
2024-02-09 14:38:02.525 UTC [grpc] Infof -> DEBU 021 parsed scheme: ""
2024-02-09 14:38:02.525 UTC [grpc] Infof -> DEBU 022 scheme "" not registered, fallback to default scheme
2024-02-09 14:38:02.539 UTC [grpc] Infof -> DEBU 023 ccResolverWrapper: sending update to cc: {[{orderer.cross-border-tr
ansaction.com:7050  <nil> 0 <nil>}] <nil> <nil>}
2024-02-09 14:38:02.539 UTC [grpc] Infof -> DEBU 024 ClientConn switching balancer to "pick_first"
2024-02-09 14:38:02.539 UTC [grpc] Infof -> DEBU 025 Channel switches to new LB policy "pick_first"
2024-02-09 14:38:02.539 UTC [grpc] Infof -> DEBU 026 Subchannel Connectivity change to CONNECTING
2024-02-09 14:38:02.539 UTC [grpc] Infof -> DEBU 027 Subchannel picks a new address "orderer.cross-border-transaction.co
m:7050" to connect
2024-02-09 14:38:02.540 UTC [grpc] UpdateSubConnState -> DEBU 028 pickfirstBalancer: HandleSubConnStateChange: 0xc0005da
580, {CONNECTING <nil>}
2024-02-09 14:38:02.540 UTC [grpc] Infof -> DEBU 029 Channel Connectivity change to CONNECTING
2024-02-09 14:38:02.553 UTC [grpc] Infof -> DEBU 02a Subchannel Connectivity change to READY
2024-02-09 14:38:02.554 UTC [grpc] UpdateSubConnState -> DEBU 02b pickfirstBalancer: HandleSubConnStateChange: 0xc0005da
580, {READY <nil>}
2024-02-09 14:38:02.554 UTC [grpc] Infof -> DEBU 02c Channel Connectivity change to READY
2024-02-09 14:38:02.573 UTC [channelCmd] update -> INFO 02d Successfully submitted channel update
bash-5.0#
```

Se realiza lo mismo con la organización 2 pero especificándole las variables necesarias antes del update.

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

```
2024-02-10 20:45:36.978 UTC [msp] GetDefaultSigningIdentity -> DEBU 01a Obtaining default signing identity
2024-02-10 20:45:36.979 UTC [channelCmd] InitCmdFactory -> INFO 01b Endorser and orderer connections initialized
2024-02-10 20:45:36.979 UTC [msp.identity] Sign -> DEBU 01c Sign: plaintext: 0AFF060A074F7267324D535012F3062D...65616465
727312002A0641646D696E73
2024-02-10 20:45:36.979 UTC [msp.identity] Sign -> DEBU 01d Sign: digest: B232CA6E316AB3FFA3E11C539680C615B45C7A63F247FF
227C299B8F4F6E3FCF
2024-02-10 20:45:36.979 UTC [msp.identity] Sign -> DEBU 01e Sign: plaintext: 0AC1070A2008021A0608F0C09FAE0622...1B823CCF
293FB3031653B8232C38949F
2024-02-10 20:45:36.979 UTC [msp.identity] Sign -> DEBU 01f Sign: digest: F8F2EA174072C3ABFEEEC60B560921F60A441F023BB2F8
D53A411C1606901A4D
2024-02-10 20:45:36.980 UTC [grpc] WithKeepaliveParams -> DEBU 020 Adjusting keepalive ping interval to minimum period o
f 10s
2024-02-10 20:45:36.980 UTC [grpc] Infof -> DEBU 021 parsed scheme: ""
2024-02-10 20:45:36.980 UTC [grpc] Infof -> DEBU 022 scheme "" not registered, fallback to default scheme
2024-02-10 20:45:36.983 UTC [grpc] Infof -> DEBU 023 ccResolverWrapper: sending update to cc: {[{orderer.cross-border-tr
ansaction.com:7050  <nil> 0 <nil>}] <nil> <nil>}
2024-02-10 20:45:36.983 UTC [grpc] Infof -> DEBU 024 ClientConn switching balancer to "pick_first"
2024-02-10 20:45:36.983 UTC [grpc] Infof -> DEBU 025 Channel switches to new LB policy "pick_first"
2024-02-10 20:45:36.983 UTC [grpc] Infof -> DEBU 026 Subchannel Connectivity change to CONNECTING
2024-02-10 20:45:36.983 UTC [grpc] Infof -> DEBU 027 Subchannel picks a new address "orderer.cross-border-transaction.co
m:7050" to connect
2024-02-10 20:45:36.983 UTC [grpc] UpdateSubConnState -> DEBU 028 pickfirstBalancer: HandleSubConnStateChange: 0xc00071b
6a0, {CONNECTING <nil>}
2024-02-10 20:45:36.985 UTC [grpc] Infof -> DEBU 029 Channel Connectivity change to CONNECTING
2024-02-10 20:45:37.033 UTC [grpc] Infof -> DEBU 02a Subchannel Connectivity change to READY
2024-02-10 20:45:37.034 UTC [grpc] UpdateSubConnState -> DEBU 02b pickfirstBalancer: HandleSubConnStateChange: 0xc00071b
6a0, {READY <nil>}
2024-02-10 20:45:37.034 UTC [grpc] Infof -> DEBU 02c Channel Connectivity change to READY
2024-02-10 20:45:37.055 UTC [channelCmd] update -> INFO 02d Successfully submitted channel update
bash-5.0#
```

En este punto ya se tiene toda la red configurada, lista para poder anexar y desplegar contratos inteligentes. Esta es una red mínima para ejemplificar el caso de uso. Cuando se pase a producción se debe de tener en cuenta la seguridad y escalabilidad. Se debe tener también en consideración que en entornos productivos se deben tener roles encargados de arquitectura, el área de infraestructura y desarrolladores de los contratos inteligentes, además también, de las personas de negocio.

21. Se procede entonces a crear el contrato inteligente, el cual contiene 3 funciones principales: PoblarBD, CrearTransaccion y ConsultarTransaccion.

El archivo que representa al contrato inteligente se llama cross-border-conctract, y se debe situar dentro de una carpeta llamada /chaincode/cross-border-contract/cross-border-contract.go

Se crea también dentro de esta carpeta el archivo go.mod para especificar los módulos, la versión de go y la versión del *contract api*
El archivo go.mod se verá de la siguiente manera:

```
module                    github.com/jnaran24/hyperledger-fabric-first-
step/chaincode/cross-border-contract
go 1.13
require github.com/hyperledger/fabric-contract-api-go v1.1.0
```

22. Al tener ya listo el contrato inteligente, se regresa a la consola del *cli* en el *portainer* y ejecutamos los siguientes comandos para asignar variables de entorno necesarias para los futuros comandos.

```
- export CHANNEL_NAME=cross-border-channel
- export CHAINCODE_NAME=cross-border-contract
- export CHAINCODE_VERSION=1
- export CC_RUNTIME_LANGUAGE=golang
- export CC_SRC_PATH="../../../chaincode/$CHAINCODE_NAME/"
- export
  ORDERER_CA=/opt/gopath/src/github.com/hyperledger/fabric/peer/
  crypto/ordererOrganizations/cross-border-
  transaction.com/orderers/orderer.cross-border-
  transaction.com/msp/tlscacerts/tlsca.cross-border-
  transaction.com-cert.pem
```

Ahora se empaqueta el contrato inteligente para poderlo instalar en las diferentes organizaciones. Se realiza con el siguiente comando:

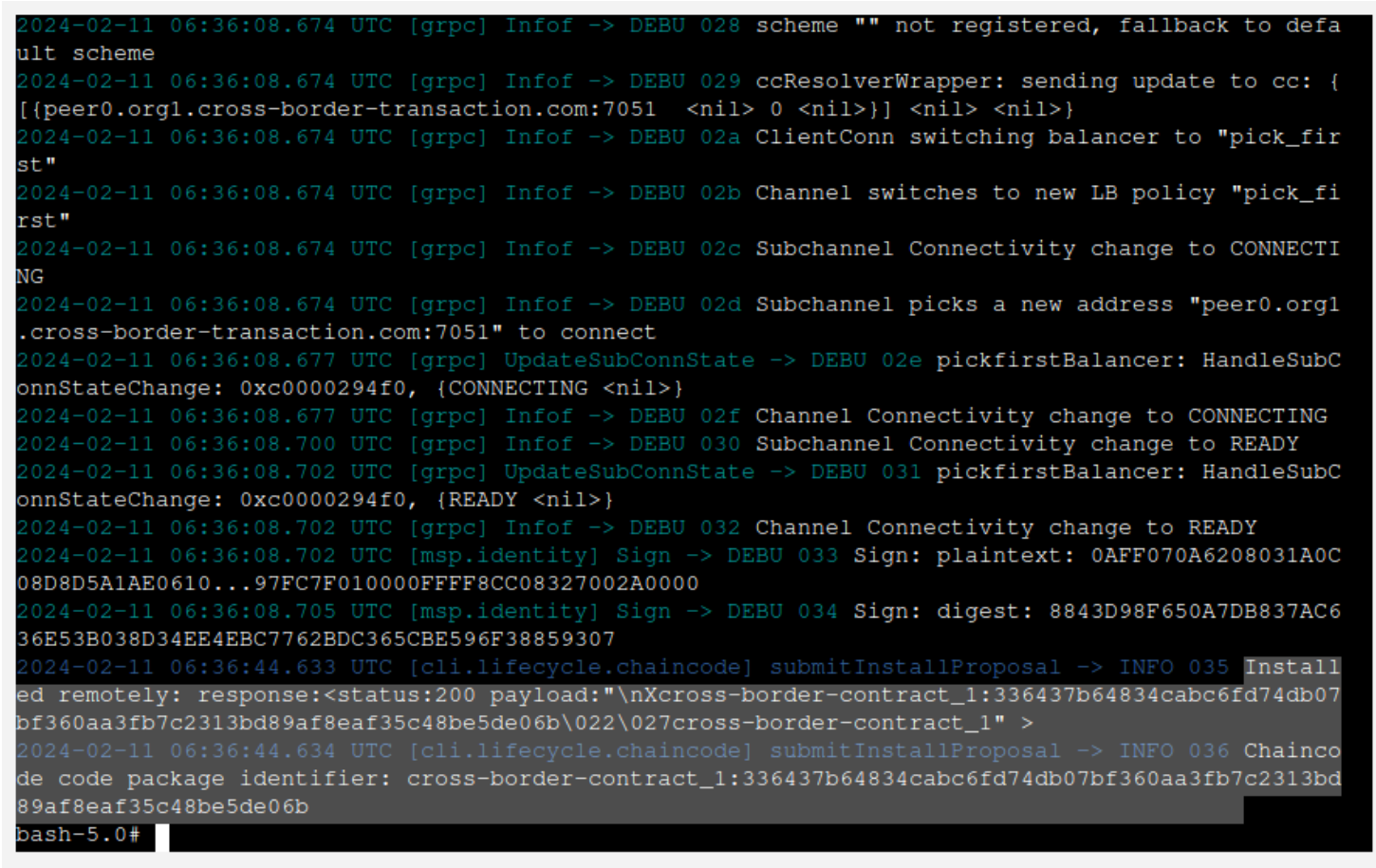
```
- peer lifecycle chaincode package ${CHAINCODE_NAME}.tar.gz --
  path ${CC_SRC_PATH} --lang ${CC_RUNTIME_LANGUAGE} --label
  ${CHAINCODE_NAME}_${CHAINCODE_VERSION} >&log.txt
```

De esta manera se logra tener empaquetado todo lo correspondiente al contrato inteligente en el *cli*

23. Se procedemos a instalar el contrato inteligente para el primer *peer* de la org1. Para ello se ejecuta el comando:

```
- peer lifecycle chaincode install cross-border-contract.tar.gz
```

Y como resultado se tiene que se ejecutó y se instaló el contrato remotamente con su respectivo hash.



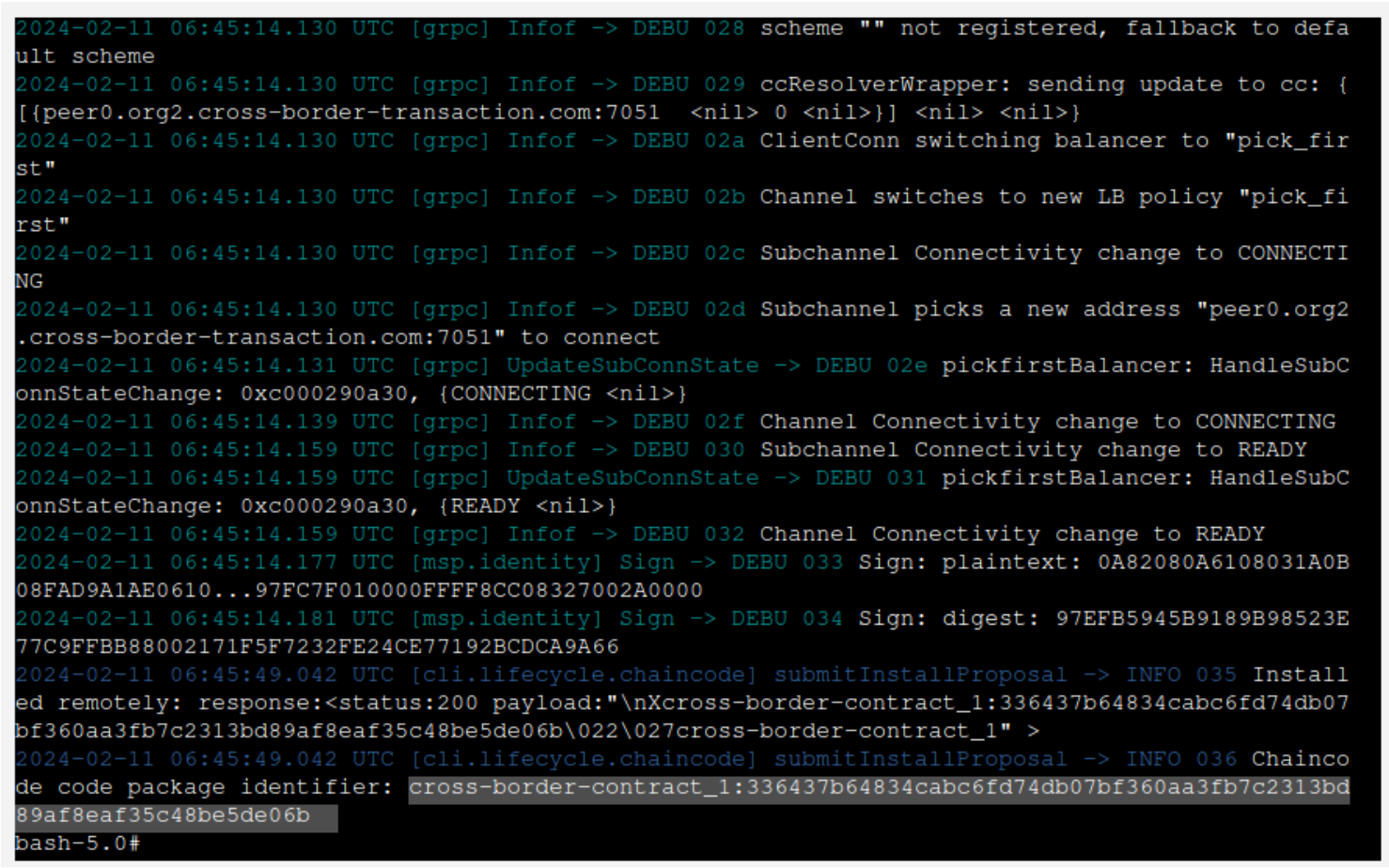
Es indispensable tener en cuenta el *code package identifier* resultante, que en este caso fue:

```
cross-border-
contract_1:336437b64834cabc6fd74db07bf360aa3fb7c2313bd89af8eaf35c48be5d
e06b
```

24. Se procede a instalar en las otras organizaciones, en este caso, en la organización 2, dando como premisas en el comando las identificaciones de la organización 2 con el comando:

    - ```
      CORE_PEER_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger
      /fabric/peer/crypto/peerOrganizations/org2.cross-border-
      transaction.com/users/Admin\@org2.cross-border-
      transaction.com/msp CORE_PEER_ADDRESS=peer0.org2.cross-border-
      transaction.com:7051 CORE_PEER_LOCALMSPID="Org2MSP"
      CORE_PEER_TLS_ROOTCERT_FILE=/opt/gopath/src/github.com/hyperle
      dger/fabric/peer/crypto/peerOrganizations/org2.cross-border-
      transaction.com/peers/peer0.org2.cross-border-
      transaction.com/tls/ca.crt peer lifecycle chaincode install
      cross-border-contract.tar.gz
      ```
      ```

Y se puede ver que da como resultado el mismo identificador que el anterior.



25. Se procede entonces a establecer las políticas de aprobación que se hayan definido para el contrato inteligente con dicho identificador anterior.

Estas políticas son únicas del contracto inteligente, no de la red ni del servicio de ordenamiento. Sin embargo, antes de realizarlo es clave aclarar que la organización 1 es la única que puede realizar aprobaciones. La Segunda no contará con estos permisos.

- `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')" --package-id cross-border-contract_1:336437b64834cabc6fd74db07bf360aa3fb7c2313bd89af8eaf35c48be5de06b`

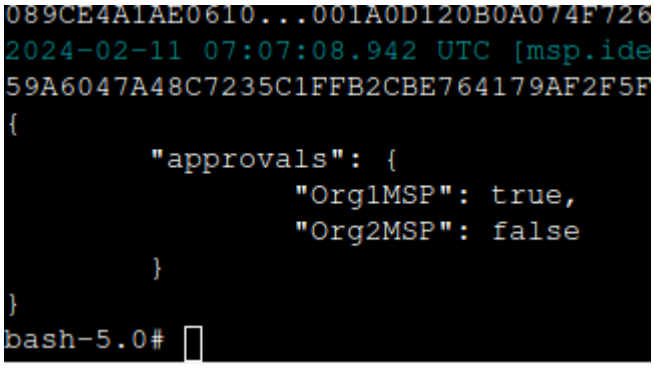
Si se deben añadir más organizaciones para que tengan este rol de aprobador, se agregan en el “OR” dentro de comillas simples, separados por coma. Por ejemplo: "OR ('Org1MSP.peer', 'Org3MSP.peer')"

```
2024-02-11 06:59:08.314 UTC [grpc] Infof -> DEBU 0ae ClientConn switching balancer to "pick_fir
st"
2024-02-11 06:59:08.314 UTC [grpc] Infof -> DEBU 0af Channel switches to new LB policy "pick_fi
rst"
2024-02-11 06:59:08.315 UTC [grpc] Infof -> DEBU 0b0 Subchannel Connectivity change to CONNECTI
NG
2024-02-11 06:59:08.315 UTC [grpc] Infof -> DEBU 0b1 Subchannel picks a new address "orderer.cr
oss-border-transaction.com:7050" to connect
2024-02-11 06:59:08.315 UTC [grpc] UpdateSubConnState -> DEBU 0b2 pickfirstBalancer: HandleSubC
onnStateChange: 0xc00043baa0, {CONNECTING <nil>}
2024-02-11 06:59:08.315 UTC [grpc] Infof -> DEBU 0b3 Channel Connectivity change to CONNECTING
2024-02-11 06:59:08.333 UTC [grpc] Infof -> DEBU 0b4 Subchannel Connectivity change to READY
2024-02-11 06:59:08.333 UTC [grpc] UpdateSubConnState -> DEBU 0b5 pickfirstBalancer: HandleSubC
onnStateChange: 0xc00043baa0, {READY <nil>}
2024-02-11 06:59:08.333 UTC [grpc] Infof -> DEBU 0b6 Channel Connectivity change to READY
2024-02-11 06:59:08.333 UTC [msp.identity] Sign -> DEBU 0b7 Sign: plaintext: 0A95080A7808031A0C
08BCE0A1AE0610...656166333563343862653564653036 62
2024-02-11 06:59:08.333 UTC [msp.identity] Sign -> DEBU 0b8 Sign: digest: 746EDD52A6B08BA39C295
A82C57EBA6E1A4019A71DA835ED40EF659DF59B2497
2024-02-11 06:59:09.731 UTC [msp.identity] Sign -> DEBU 0b9 Sign: plaintext: 0A95080A7808031A0C
08BCE0A1AE0610...D2AF6BD76617FB8481028506 24ADDC98
2024-02-11 06:59:09.731 UTC [msp.identity] Sign -> DEBU 0ba Sign: digest: 7FF6613B8E27A0EA5FBBC
8843FBFBC66D3036F5453D481338E04CF75B4A4E2BE
2024-02-11 06:59:09.741 UTC [msp.identity] Sign -> DEBU 0bb Sign: plaintext: 0ABD070A2008051A06
08BDE0A1AE0622...00120D1A0B08FFFFFFFFFFFFFFFFFF01
2024-02-11 06:59:09.741 UTC [msp.identity] Sign -> DEBU 0bc Sign: digest: EBCE33DC4CB8CDE7E0B41
1BD285821B7FC034ADB0D2555C1388E5D1CC9C0D846
2024-02-11 06:59:12.499 UTC [chaincodeCmd] ClientWait -> INFO 0bd txid [0da00b605474f91e767d60e
5ad40e1628d79072658dcb299edb6480dba6f3594] committed with status (VALID) at
bash-5.0#
```

Con esto se verifica que para la organización 1 se aprueban estas políticas de aprobación. Si se debe añadir otra organización para que apruebe las políticas, se debe ejecutar el mismo comando anterior, pero añadiéndole las variables identificadoras de esa organización al inicio.
Se puede validar mediante el siguiente comando cuáles son las organizaciones que se encuentran como aprobadoras.

```
peer lifecycle chaincode checkcommitreadiness --channelID $CHANNEL_NAME
--name $CHAINCODE_NAME --version $CHAINCODE_VERSION --sequence 1 --
signature-policy "OR ('Org1MSP.peer')" --output json
```

Aquí se pueden ver en formato json las organizaciones validadoras que han a su vez aprobado estas políticas.



26. Se procede a realizar la misma aprobación de políticas para las demás organizaciones aprobadoras, sea la 2, 3, etc. Es decir, que dichas organizaciones estén de acuerdo con estas políticas.

    - `CORE_PEER_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org2.cross-border-transaction.com/users/Admin\@org2.cross-border-transaction.com/msp CORE_PEER_ADDRESS=peer0.org2.cross-border-transaction.com:7051 CORE_PEER_LOCALMSPID="Org2MSP" CORE_PEER_TLS_ROOTCERT_FILE=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org2.cross-border-transaction.com/peers/peer0.org2.cross-border-transaction.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')" --package-id cross-border-contract_1:336437b64834cabc6fd74db07bf360aa3fb7c2313bd89af8eaf35c48be5de06b`

la política de firma especificada es OR ('Org1MSP.peer'). Esto significa que las transacciones del chaincode solo puede ser aprobada si es aprobada por al menos uno de los peers de la organización 1 (Org1MSP). La organización 2 (Org2MSP) no está incluida en esta política de firma.

Por lo tanto, en este caso, la organización 2 no tiene el poder de aprobar transacciones según esta política de firma. Solo la organización 1 tiene ese poder.

```
2024-02-11 22:29:11.877 UTC [grpc] Infof -> DEBU 0ac scheme "" not registered, fallback to default scheme
2024-02-11 22:29:11.877 UTC [grpc] Infof -> DEBU 0ad ccResolverWrapper: sending update to cc: {[{orderer.cross-border-tr
ansaction.com:7050  <nil> 0 <nil>}] <nil> <nil>}
2024-02-11 22:29:11.877 UTC [grpc] Infof -> DEBU 0ae ClientConn switching balancer to "pick_first"
2024-02-11 22:29:11.877 UTC [grpc] Infof -> DEBU 0af Channel switches to new LB policy "pick_first"
2024-02-11 22:29:11.877 UTC [grpc] Infof -> DEBU 0b0 Subchannel Connectivity change to CONNECTING
2024-02-11 22:29:11.877 UTC [grpc] Infof -> DEBU 0b1 Subchannel picks a new address "orderer.cross-border-transaction.co
m:7050" to connect
2024-02-11 22:29:11.878 UTC [grpc] UpdateSubConnState -> DEBU 0b2 pickfirstBalancer: HandleSubConnStateChange: 0xc00047c
fd0, {CONNECTING <nil>}
2024-02-11 22:29:11.878 UTC [grpc] Infof -> DEBU 0b3 Channel Connectivity change to CONNECTING
2024-02-11 22:29:11.893 UTC [grpc] Infof -> DEBU 0b4 Subchannel Connectivity change to READY
2024-02-11 22:29:11.893 UTC [grpc] UpdateSubConnState -> DEBU 0b5 pickfirstBalancer: HandleSubConnStateChange: 0xc00047c
fd0, {READY <nil>}
2024-02-11 22:29:11.893 UTC [grpc] Infof -> DEBU 0b6 Channel Connectivity change to READY
2024-02-11 22:29:11.893 UTC [msp.identity] Sign -> DEBU 0b7 Sign: plaintext: 0A99080A7808031A0C08B794A5AE0610...65616633
3563343862653564653036 62
2024-02-11 22:29:11.893 UTC [msp.identity] Sign -> DEBU 0b8 Sign: digest: 5C4252A462BC0264CA830FD50F7CB72935E8E856A393D7
41D78D9D9C67AC169F
2024-02-11 22:29:12.621 UTC [msp.identity] Sign -> DEBU 0b9 Sign: plaintext: 0A99080A7808031A0C08B794A5AE0610...112D1F77
E2BFE25317780572DC73F16F
2024-02-11 22:29:12.622 UTC [msp.identity] Sign -> DEBU 0ba Sign: digest: A87D47B9B51D1D812D989AFE146C12F88185F8A680EBE9
1FFFE1E59BC303A141
2024-02-11 22:29:12.623 UTC [msp.identity] Sign -> DEBU 0bb Sign: plaintext: 0AC1070A2008051A0608B894A5AE0622...00120D1A
0B08FFFFFFFFFFFFFFFFF01
2024-02-11 22:29:12.623 UTC [msp.identity] Sign -> DEBU 0bc Sign: digest: A0329257754890B5092DCCD77B1E457AB02BB8099BBCC3
A2090FB2BE27DDE7A9
2024-02-11 22:29:15.158 UTC [chaincodeCmd] ClientWait -> INFO 0bd txid [966b6c787e6ecf3b1d5111a16f668ee0cba7244d7377e659
d8775778060d0389] committed with status (VALID) at
bash-5.0#
```

27. Se realizamos el *commit* del contrato inteligente definiendo los *peer addresses* de los archivos de certificación de las organizaciones que pertenecen al canal, para que de esta forma estén de acuerdo con las políticas definidas.

    - `peer lifecycle chaincode commit -o orderer.cross-border-transaction.com:7050 --tls --cafile $ORDERER_CA --peerAddresses peer0.org1.cross-border-transaction.com:7051 --tlsRootCertFiles /opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org1.cross-border-transaction.com/peers/peer0.org1.cross-border-transaction.com/tls/ca.crt --peerAddresses peer0.org2.cross-border-transaction.com:7051 --tlsRootCertFiles /opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org2.cross-border-transaction.com/peers/peer0.org2.cross-border-transaction.com/tls/ca.crt --channelID $CHANNEL_NAME --name $CHAINCODE_NAME --version $CHAINCODE_VERSION --sequence 1 --signature-policy "OR ('Org1MSP.peer')"`

En este caso únicamente son 2 organizaciones en el canal, y 1 validadora, pero si en otro caso fueran más, se deben agregar los *path* del *peer* de la organización con --*peerAddresses*. Además de su ca.crt con la bandera --tlsRootCertFiles y por último añadir también los MSP.peer de las organizaciones validadoras al final del comando en la bandera --signature-policy

```
2024-02-11 22:38:54.641 UTC [grpc] Infof -> DEBU 052 Subchannel picks a new address "orderer.cross-border-transaction.co
m:7050" to connect
2024-02-11 22:38:54.648 UTC [grpc] UpdateSubConnState -> DEBU 053 pickfirstBalancer: HandleSubConnStateChange: 0xc000365
5c0, {CONNECTING <nil>}
2024-02-11 22:38:54.648 UTC [grpc] Infof -> DEBU 054 Channel Connectivity change to CONNECTING
2024-02-11 22:38:54.665 UTC [grpc] Infof -> DEBU 055 Subchannel Connectivity change to READY
2024-02-11 22:38:54.665 UTC [grpc] UpdateSubConnState -> DEBU 056 pickfirstBalancer: HandleSubConnStateChange: 0xc000365
5c0, {READY <nil>}
2024-02-11 22:38:54.665 UTC [grpc] Infof -> DEBU 057 Channel Connectivity change to READY
2024-02-11 22:38:54.666 UTC [msp.identity] Sign -> DEBU 058 Sign: plaintext: 0A95080A7808031A0C08FE98A5AE0610...001A0D12
0B0A074F7267314D53501003
2024-02-11 22:38:54.666 UTC [msp.identity] Sign -> DEBU 059 Sign: digest: 1020DCBD0F0D63A58BE847E41AEA3D30727BA25BCC008D
42EC556D01907A0A3D
2024-02-11 22:38:55.056 UTC [msp.identity] Sign -> DEBU 05a Sign: plaintext: 0A95080A7808031A0C08FE98A5AE0610...575D2FEA
EC14EF18815515BD86834BBE
2024-02-11 22:38:55.056 UTC [msp.identity] Sign -> DEBU 05b Sign: digest: 1C10A5F91B5ACD452FD57F29110CF574A586EAFABB89ED
75B5FEDAE3CA7DFF17
2024-02-11 22:38:55.059 UTC [msp.identity] Sign -> DEBU 05c Sign: plaintext: 0ABD070A2008051A0608FF98A5AE0622...00120D1A
0B08FFFFFFFFFFFFFFFFFF01
2024-02-11 22:38:55.059 UTC [msp.identity] Sign -> DEBU 05d Sign: digest: DD3BD3F20870437BF51C57FDC4758E6897055DD2438572
D18D0FDC10416BE348
2024-02-11 22:38:55.061 UTC [msp.identity] Sign -> DEBU 05e Sign: plaintext: 0ABD070A2008051A0608FF98A5AE0622...00120D1A
0B08FFFFFFFFFFFFFFFFFF01
2024-02-11 22:38:55.063 UTC [msp.identity] Sign -> DEBU 05f Sign: digest: AD8399F51B9FB26012ECAEA8D6D2287586BFB858B4B44D
A37E5E9153EEF80FC9
2024-02-11 22:38:57.917 UTC [chaincodeCmd] ClientWait -> INFO 060 txid [f91c07765f0425334ccbf333820cc445520d5d744b613ca7
9a79b4a22bd6798c] committed with status (VALID) at peer0.org2.cross-border-transaction.com:7051
2024-02-11 22:38:58.048 UTC [chaincodeCmd] ClientWait -> INFO 061 txid [f91c07765f0425334ccbf333820cc445520d5d744b613ca7
9a79b4a22bd6798c] committed with status (VALID) at peer0.org1.cross-border-transaction.com:7051
bash-5.0#
```

Se pueden ver en portainer dos nuevos contenedores correspondientes a el *chaincode* desplegado en cada una de las organizaciones.

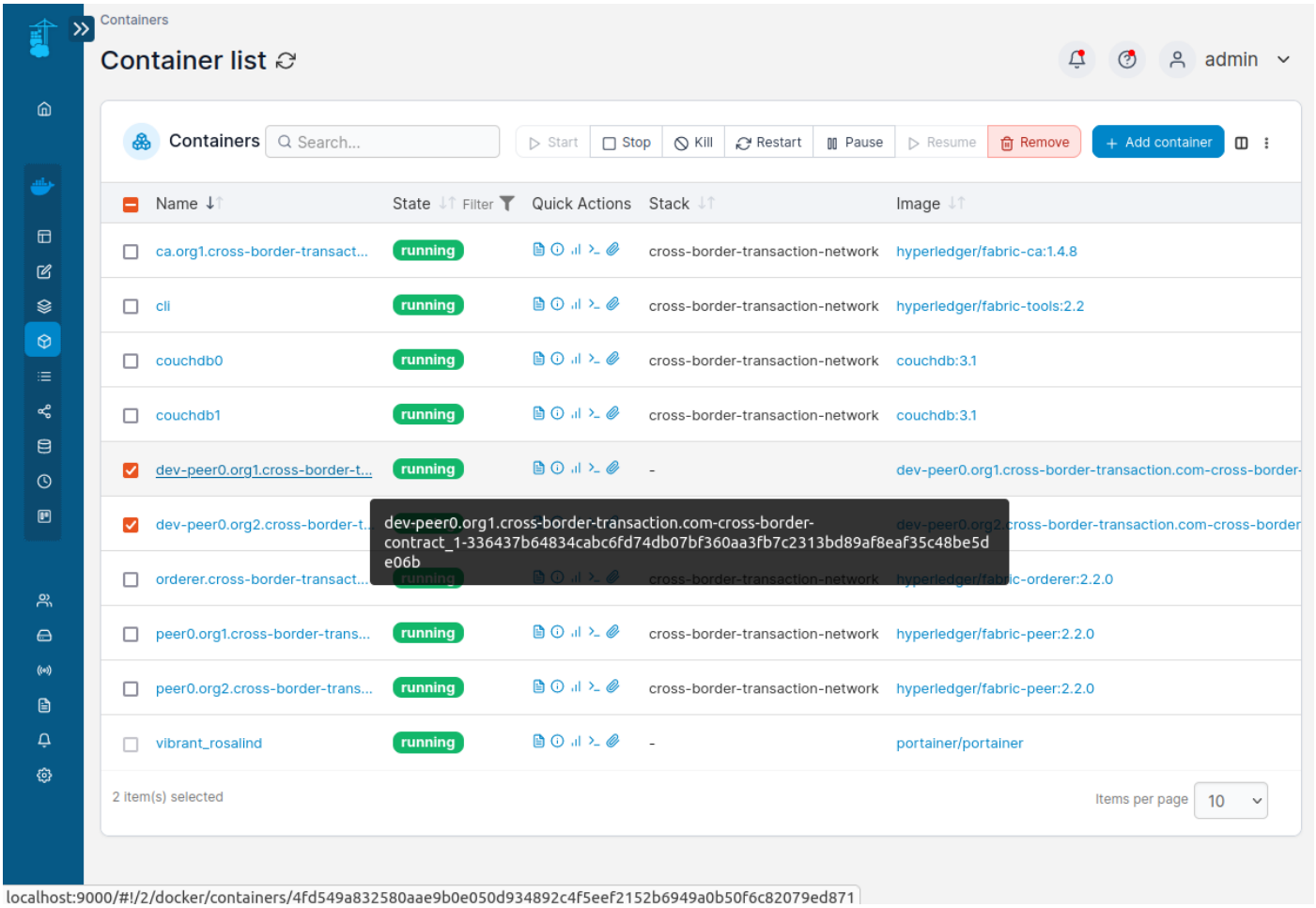


Figura 10. Evidencia de los contenedores del contrato inteligente

28. Se puede en este momento probar el contrato inteligente. Primero, se ingresan a la base de datos los clientes de los bancos y las entidades sancionadas con el comando:

- `peer chaincode invoke -o orderer.cross-border-transaction.com:7050 --tls --cafile $ORDERER_CA -C $CHANNEL_NAME -n $CHAINCODE_NAME -c '{"Args":["PoblarBD"]}'`

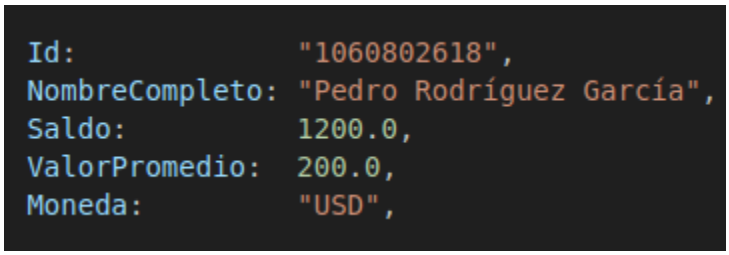
```
ature:"0E\002!\000\252\230\305 \201_P\033\231\"\236\205M\330E@$\370A_`\023y\311\245g\334b\030\304 \337\002 \026\374\036}L
\363\240\333\r`[\374F/\231\333Sb\210#\005\331G\"@\370\030\2070\315\343\214" >
2024-03-02 03:40:00.211 UTC [chaincodeCmd] chaincodeInvokeOrQuery -> INFO 045 Chaincode invoke successful. result: statu
s:200
bash-5.0#
```

Ahora si se puede realizar una transacción:

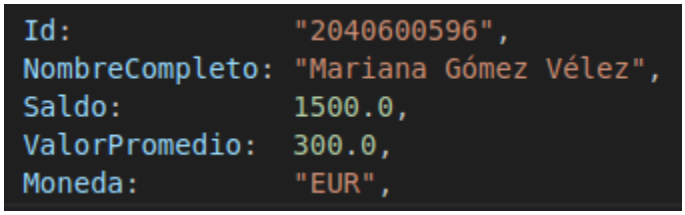
- `peer chaincode invoke -o orderer.cross-border-transaction.com:7050 --tls --cafile $ORDERER_CA -C $CHANNEL_NAME -n $CHAINCODE_NAME -c '{"Args":["CrearTransaccion","1060802618","150.0","EUR","2040600596","id:22285"]}'`

En este comando se crea una transacción, ejecutada por el banco 1, en donde el usuario con id 1060802618, quien está registrado como cliente del banco 1 y con tipo de moneda “USD”, realiza una transferencia de 150 euros al cliente del banco 2 con id 2040600596, el cual tiene como destino el tipo de moneda “EUR”. Esta transacción se crea con el id único “id:22284”

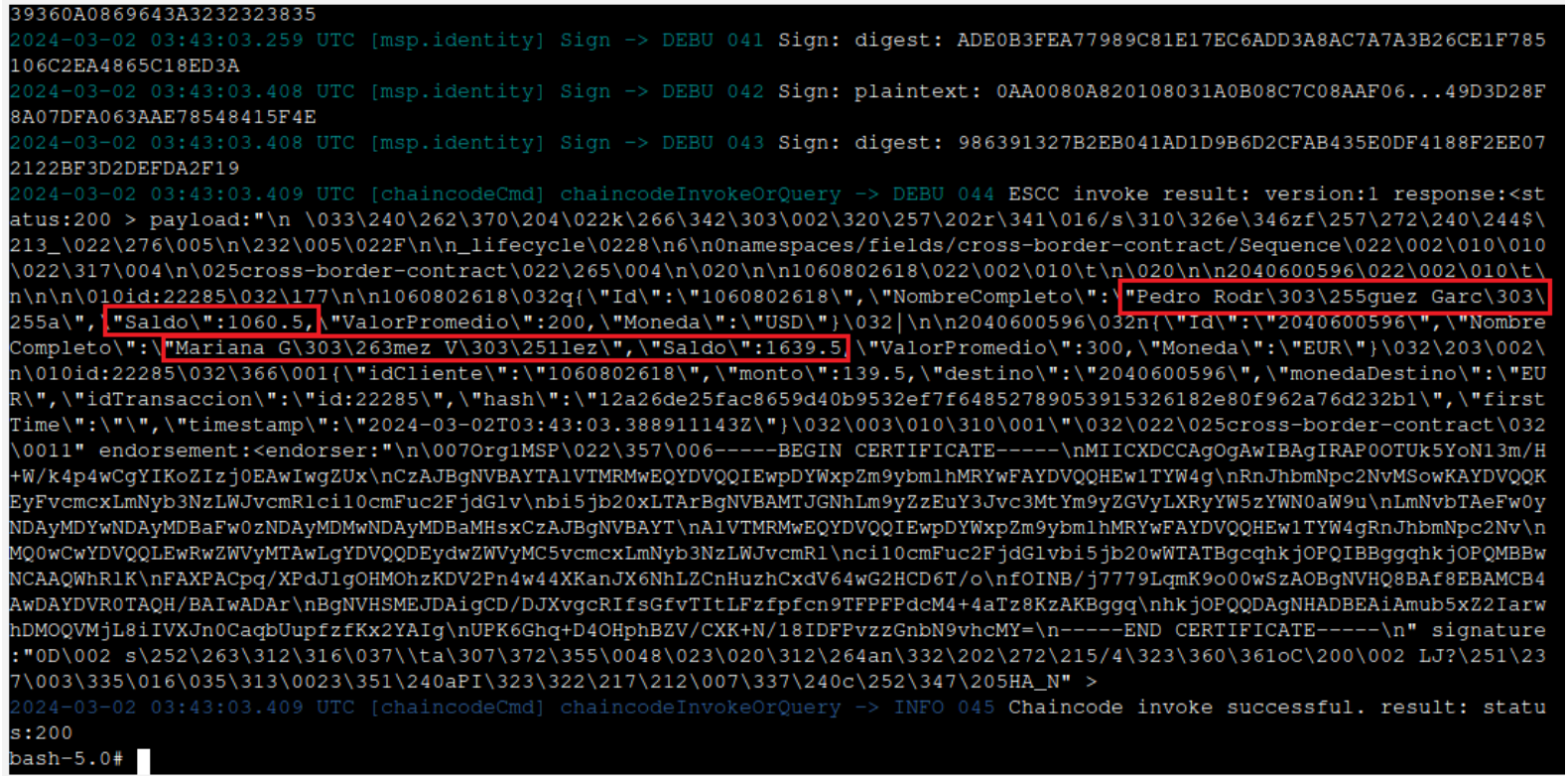
El usuario 1060802618 del banco 1 tenía un saldo inicial de 1200 USD.



Y el Usuario 2040600596 del banco 2 tenía un saldo inicial de 1500 EUR.

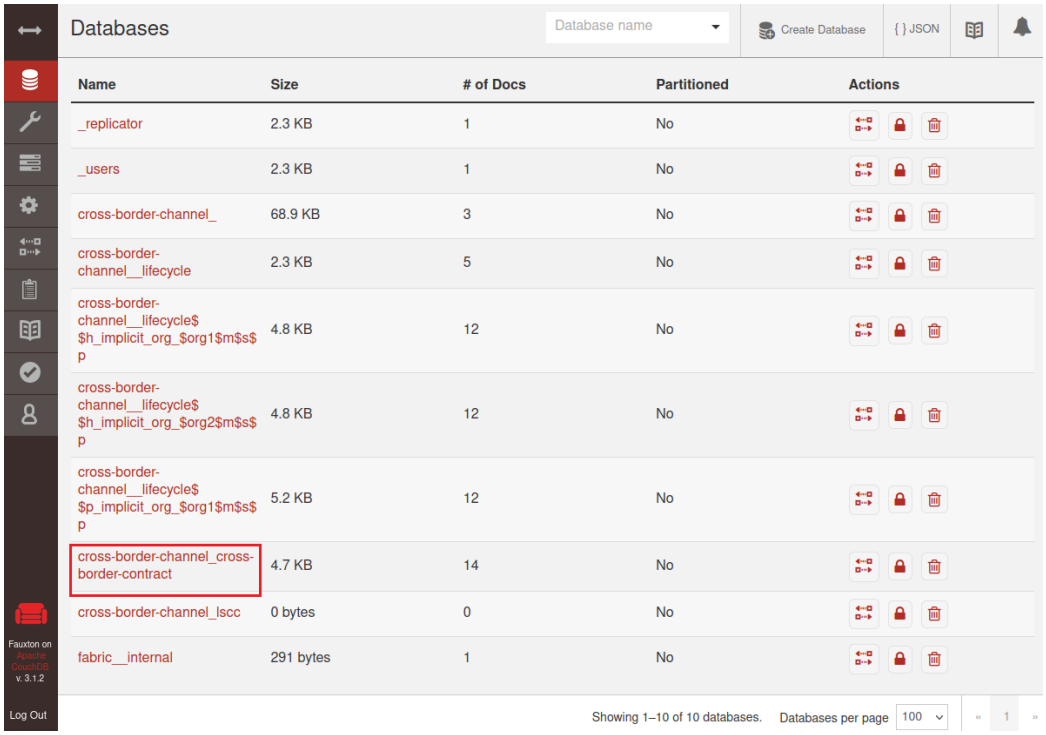


Tras la transacción, la cual se ejecutó correctamente con status 200, se puede ver la trazabilidad de los saldos actualizados una vez se aprueba una transacción que paso todas las validaciones.

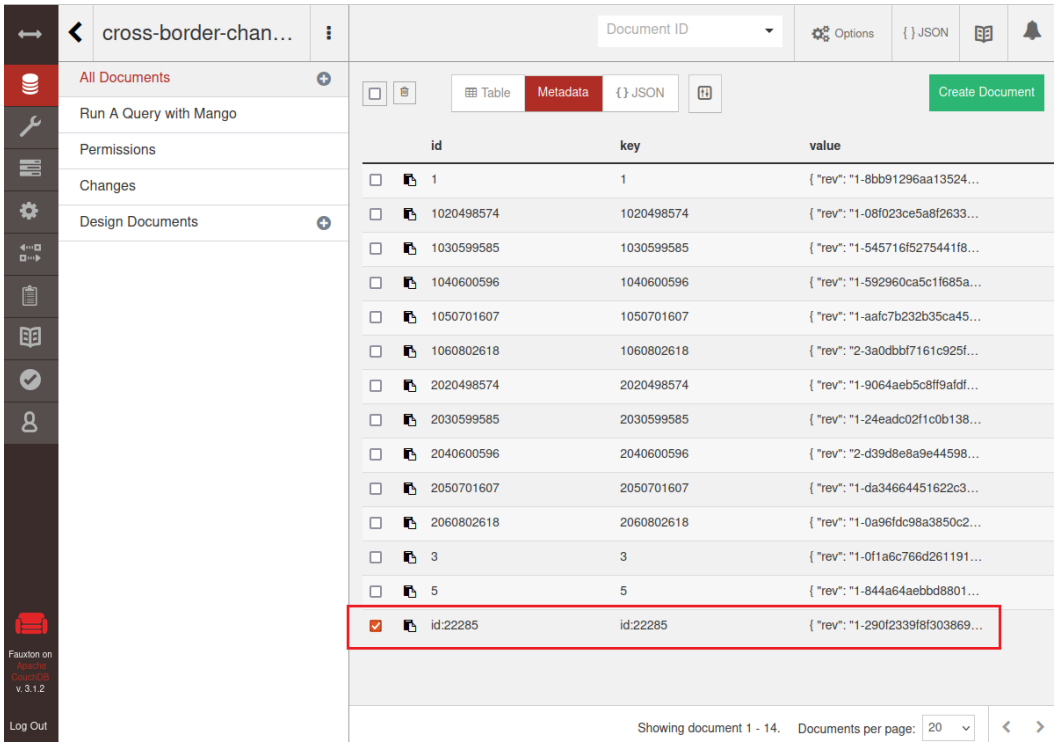


Se ve claramente que los saldos de los 2 clientes se actualizaron, Pedro quedo con 1060.5 USD y Mariana con 1639.5 EUR.

29. Se valida en la base de datos dbcouch0 de la organización 1, dentro de la carpeta correspondiente al canal cross-border-channel_cross-border-contract



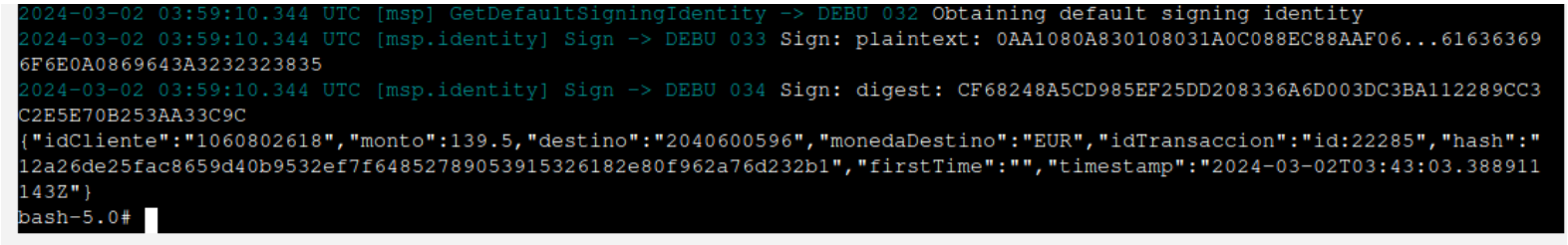
Aquí se ve el estado del libro distribuido, de tipo clave-valor, en donde el valor es el json con toda la data de la transacción.



30. Una vez creada la transacción, es posible consultarla de igual manera con el comando que se creó para el *query*. Mediante el comando:

    - `peer chaincode query -C $CHANNEL_NAME -n $CHAINCODE_NAME -c '{"Args":["ConsultarTransaccion","id:22285"]}'`

Da como resultado toda la información de esta transacción, incluido el hash.

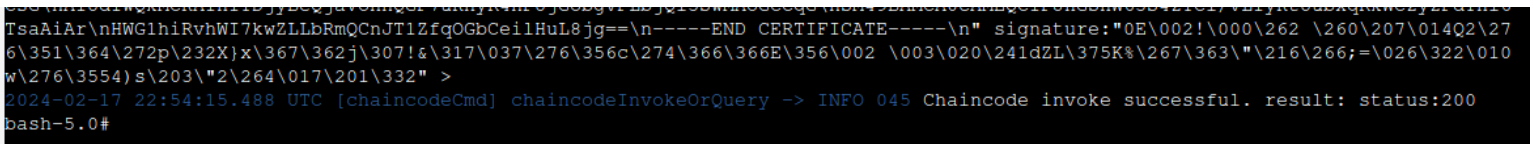


31. Si se intenta crear una transacción desde la organización 2, dicha transacción no se creará satisfactoriamente, y se puede evidenciar en la base de datos. Por ejemplo, si se realiza este intento de transacción en nombre de la organización 2, se verifica que, al ejecutarse el comando, no se muestra dicha transacción en la base de datos.

    - `CORE_PEER_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org2.cross-border-transaction.com/users/Admin\@org2.cross-border-transaction.com/msp CORE_PEER_ADDRESS=peer0.org2.cross-border-transaction.com:7051 CORE_PEER_LOCALMSPID="Org2MSP" CORE_PEER_TLS_ROOTCERT_FILE=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org2.cross-border-transaction.com/peers/peer0.org2.cross-border-transaction.com/tls/ca.crt peer chaincode invoke -o orderer.cross-border-transaction.com:7050 --tls --cafile $ORDERER_CA -C $CHANNEL_NAME -n $CHAINCODE_NAME -c '{"Args":["CrearTransaccion","1020498574","150.0","EUR","2030599585","id:95851"]}'`

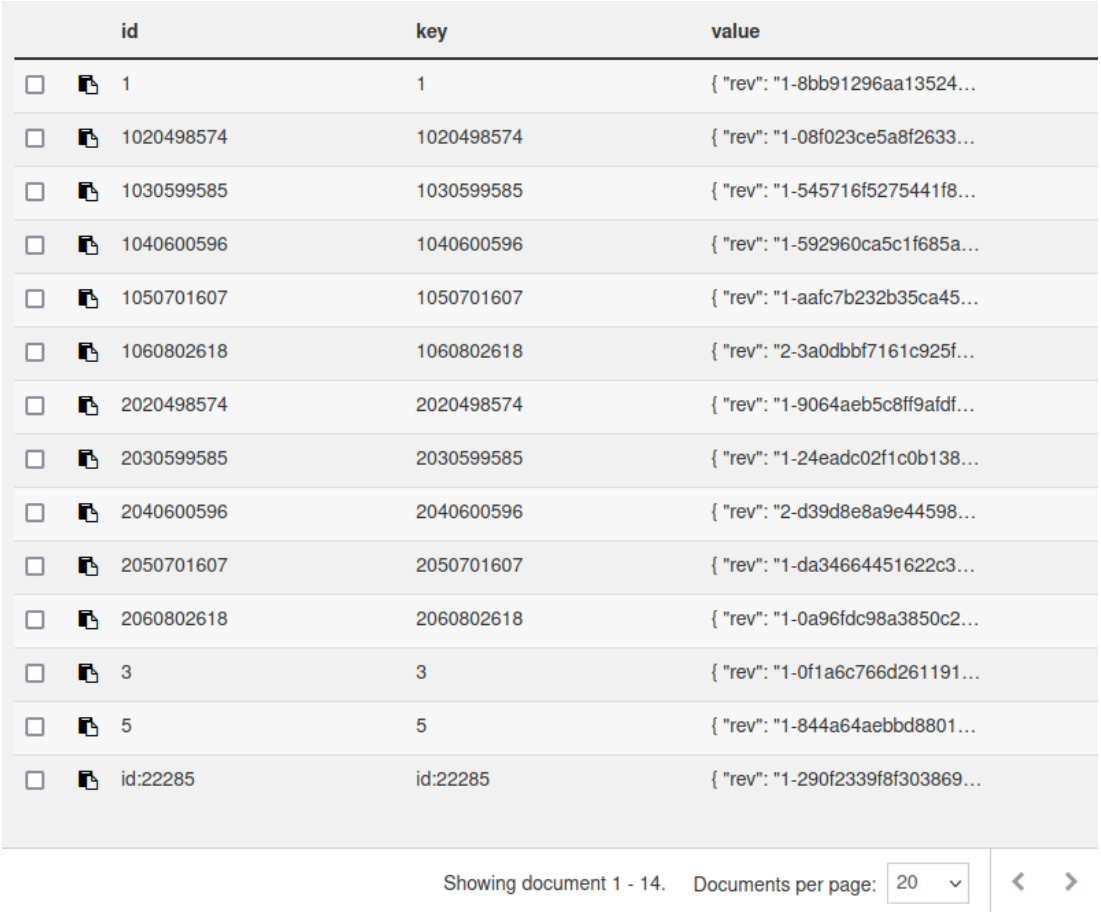
Una transacción del cliente del banco 1 con id 1020498574 con saldo 1000 EUR al cliente del banco 2 con id 2030599585 y saldo 800 USD. Con id de transacción 95851.

La consola dirá que se realizó satisfactoriamente la transacción.



pero como este banco no tiene los permisos y no es una entidad certificadora, al verificar en la base de datos, se puede verificar que no se creó la transacción y que los saldos de los clientes de los 2 bancos siguen igual.

La transacción se hizo con id único “id:95851”, pero se verificó que no se encuentra en las transacciones del canal en la base de datos; únicamente está la transacción realizada a nombre de la organización 1, la cual es la aprobadora y la que puede realizar la escritura.



id	key	value
1	1	{ "rev": "1-8bb91296aa13524…
1020498574	1020498574	{ "rev": "1-08f023ce5a8f2633…
1030599585	1030599585	{ "rev": "1-545716f5275441f8…
1040600596	1040600596	{ "rev": "1-592960ca5c1f685a…
1050701607	1050701607	{ "rev": "1-aafc7b232b35ca45…
1060802618	1060802618	{ "rev": "2-3a0dbbf7161c925f…
2020498574	2020498574	{ "rev": "1-9064aeb5c8ff9afdf…
2030599585	2030599585	{ "rev": "1-24eadc02f1c0b138…
2040600596	2040600596	{ "rev": "2-d39d8e8a9e44598…
2050701607	2050701607	{ "rev": "1-da34664451622c3…
2060802618	2060802618	{ "rev": "1-0a96fdc98a3850c2…
3	3	{ "rev": "1-0f1a6c766d261191…
5	5	{ "rev": "1-844a64aebbd8801…
id:22285	id:22285	{ "rev": "1-290f2339f8f303869…

Showing document 1 - 14. Documents per page: 20

Figura 11. Transacciones hechas con llave-valor vistas desde la base de datos

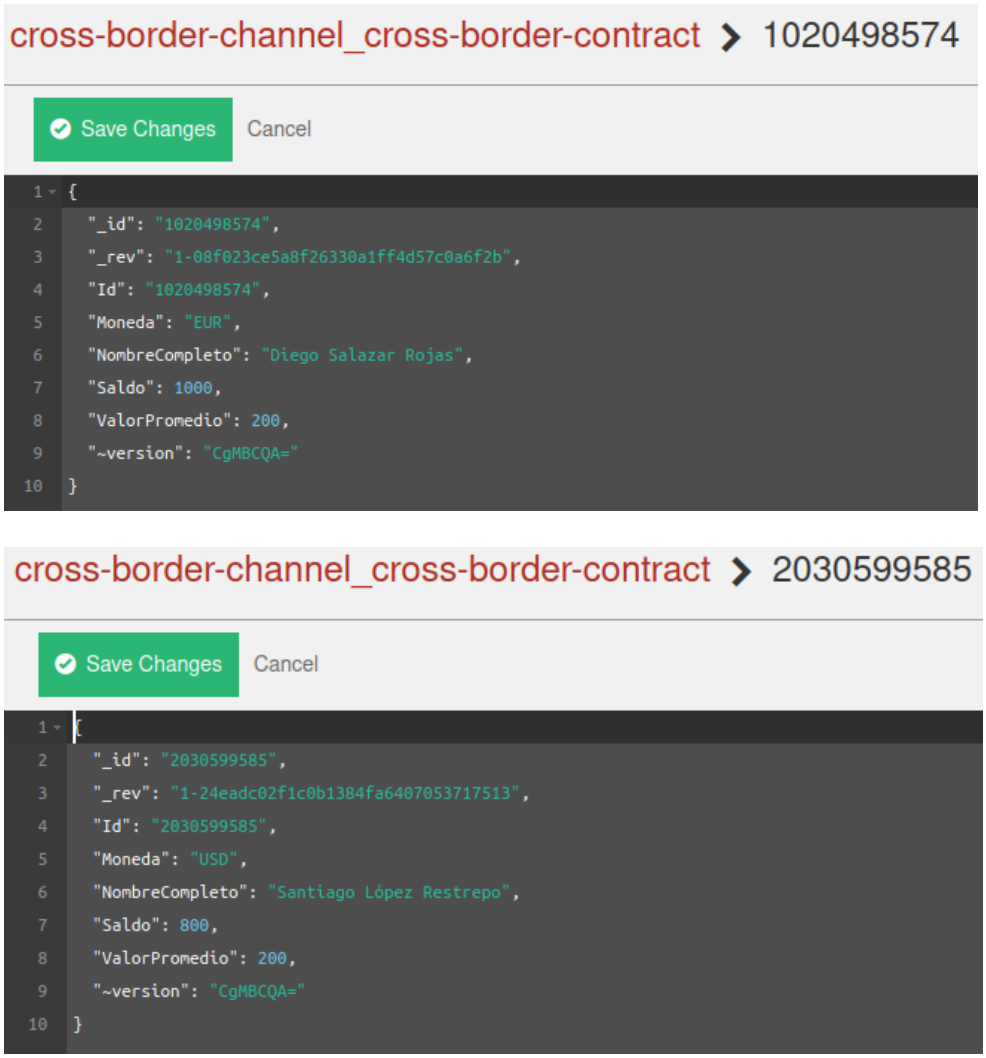


Figura 12. Detalles de la información de usuarios en la base de datos

Y se ve que los saldos siguen iguales.

Sin embargo, la organización 2 si tiene acceso a lectura y puede realizar *query's* para consultar transacciones existentes en la Blockchain. Por ejemplo, usando el siguiente comando se puede ver la transacción que se creó anteriormente con la organización 1.

- `CORE_PEER_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org2.cross-border-transaction.com/users/Admin\@org2.cross-border-transaction.com/msp CORE_PEER_ADDRESS=peer0.org2.cross-border-transaction.com:7051 CORE_PEER_LOCALMSPID="Org2MSP" CORE_PEER_TLS_ROOTCERT_FILE=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org2.cross-border-transaction.com/peers/peer0.org2.cross-border-transaction.com/tls/ca.crt peer chaincode query -C`

```
$CHANNEL_NAME -n $CHAINCODE_NAME -c
'{"Args":["ConsultarTransaccion","id:22285"]}'
```

```
6F6E0A0869643A3232323835
2024-03-02 04:13:18.955 UTC [msp.identity] Sign -> DEBU 034 Sign: digest: 42FABACCF3240210B87D691049B108665094C0B3C522D4
A9467C05065DBDC998
{"idCliente":"1060802618","monto":139.5,"destino":"2040600596","monedaDestino":"EUR","idTransaccion":"id:22285","hash":"
12a26de25fac8659d40b9532ef7f6485278905391532618 2e80f962a76d232b1","firstTime":"","timestamp":"2024-03-02T03:43:03.388911
143Z"}
bash-5.0#
```

IMPLEMENTACIÓN DE HYPERLEDGER EXPLORER

Hyperledger Explorer es una herramienta de código abierto que permite visualizar, interactuar y analizar la red blockchain de Hyperledger Fabric de manera intuitiva mediante un navegador web, permitiendo explorar la red y acceder a información crucial como:

- **Bloques**: Visualiza los detalles de cada bloque, como su número, marca de tiempo, transacciones y datos adicionales.
- **Transacciones**: Examina las transacciones individuales, incluyendo su estado, detalles operativos y participantes involucrados.
- **Canales**: Supervisa la actividad en cada canal de la red, incluyendo el recuento de bloques y transacciones.
- **Organizaciones**: Observa la implicación de las organizaciones en la red, así como sus roles y autorizaciones.
- **Contratos inteligentes**: Explora e interactúa con los contratos inteligentes (*chaincodes*) desplegados en la red.

La inclusión del Explorador de Hyperledger en la red Fabric conlleva diversos beneficios:

- **Transparencia y visibilidad**: Facilita la comprensión del funcionamiento de la red, permitiendo a los usuarios visualizar y analizar la actividad en tiempo real.
- **Monitoreo y análisis**: Proporciona un seguimiento preciso de las transacciones, bloques y contratos inteligentes, facilitando la detección de errores y la identificación de tendencias.
- **Resolución de problemas y depuración**: Ofrece herramientas para diagnosticar y resolver problemas en la red de forma eficiente.
- **Gestión de permisos**: Permite el control del acceso a la información de la red, definiendo roles y permisos específicos para diferentes usuarios.
- **Interfaz intuitiva**: Ofrece una interfaz gráfica de usuario fácil de usar, ideal para usuarios no técnicos.

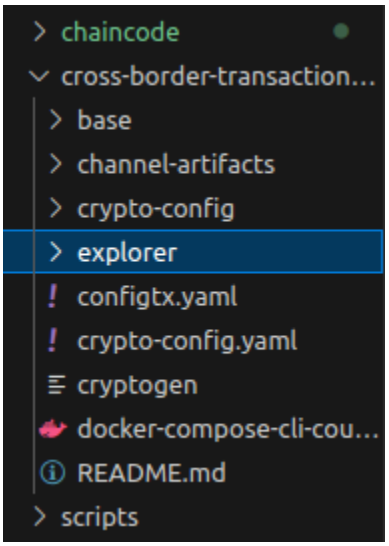
Algunos casos de uso específicos de algunos roles donde el Explorador de Hyperledger puede ser útil, son:

Auditores: Pueden utilizar el Explorador de Hyperledger para verificar la integridad de las transacciones y los datos en la red.
Desarrolladores: Pueden utilizar el Explorador de Hyperledger para depurar contratos inteligentes y monitorear el rendimiento de la red.

Gerentes de producto: Pueden utilizar el Explorador de Hyperledger para obtener información sobre el uso de la red y tomar decisiones sobre su evolución.
Verificadores del cumplimiento de la regulación: El Explorador de Hyperledger puede ayudar a las empresas a cumplir con las regulaciones que exigen la trazabilidad de las transacciones.

Implementación de Explorer en Hyperledger fabric

1. Para implementar Explorer en Hyperledger fabric se deben tener en cuenta los prerrequisitos necesarios: Docker y Docker Compose.
2. Se procede a crear una nueva carpeta /explorer dentro de la carpeta de red, en este caso dentro de la carpeta cross-border-transaction-network



3. Dentro de esta carpeta se descargan los archivos necesarios para Hyperledger explorer.

   - `wget https://raw.githubusercontent.com/hyperledger/blockchain-explorer/main/examples/net1/config.json`
   - `wget https://raw.githubusercontent.com/hyperledger/blockchain-explorer/main/examples/net1/connection-profile/test-network.json -P connection-profile`
   - `wget https://raw.githubusercontent.com/hyperledger/blockchain-explorer/main/docker-compose.yaml`

4. Se configura editando el archivo docker-compose.yaml

Para el *name* de la *network* mynetwork.com se coloca el nombre de la red. Se puede ver el nombre de las redes utilizando el comando:

   - `docker network ls`

allí podemos ver el nombre de la *network* creada en pasos previos, en este caso es cross-border-transaction-network_basic

También, si es necesario, se cambian las rutas de los volumes en donde se especifica dónde se encuentra el archivo config.json, la carpeta connection-profile y la información de las organizaciones (peerOrganizations). En este caso los volumes quedarían de la siguiente manera:

```
volumes:
  - ./config.json:/opt/explorer/app/platform/fabric/config.json
  - ./connection-profile:/opt/explorer/app/platform/fabric/connection-profile
  - ../crypto-config/peerOrganizations:/tmp/crypto
  - walletstore:/opt/explorer/wallet
ports:
  - 8080:8080
depends_on:
  explorerdb.mynetwork.com:
    condition: service_healthy
networks:
  - mynetwork.com
```

5. Se configura el archivo test-network.json con la información de la red, proporcionando los canales con sus respectivos peers, las organizaciones con su mspid con la llave privada del usuario administrador y la llave de certificación, y por ultimo los peers con sus tlsCACerts.

En este caso se configuró de la siguiente manera:

```
{
    "name": "test-network",
    "version": "1.0.0",
    "client": {
        "tlsEnable": true,
        "adminCredential": {
            "id": "exploreradmin",
            "password": "exploreradminpw"
        },
        "enableAuthentication": true,
        "organization": "Org1MSP",
        "connection": {
            "timeout": {
                "peer": {
                    "endorser": "300"
                },
```

```
                "orderer": "300"
            }
        }
    },
    "channels": {
        "cross-border-channel": {
            "peers": {
                "peer0.org1.cross-border-transaction.com": {},
                "peer0.org2.cross-border-transaction.com": {}
            }
        }
    },
    "organizations": {
        "Org1MSP": {
            "mspid": "Org1MSP",
            "adminPrivateKey": {
                "path": "/tmp/crypto/org1.cross-border-
transaction.com/users/Admin@org1.cross-border-
transaction.com/msp/keystore/priv_sk"
            },
            "peers": ["peer0.org1.cross-border-transaction.com"],
            "signedCert": {
                "path": "/tmp/crypto/org1.cross-border-
transaction.com/users/Admin@org1.cross-border-
transaction.com/msp/signcerts/Admin@org1.cross-border-
transaction.com-cert.pem"
            }
        },
        "Org2MSP": {
            "mspid": "Org2MSP",
            "adminPrivateKey": {
                "path": "/tmp/crypto/org2.cross-border-
transaction.com/users/Admin@org2.cross-border-
transaction.com/msp/keystore/priv_sk"
            },
            "peers": ["peer0.org2.cross-border-transaction.com"],
            "signedCert": {
                "path": "/tmp/crypto/org2.cross-border-
transaction.com/users/Admin@org2.cross-border-
transaction.com/msp/signcerts/Admin@org2.cross-border-
transaction.com-cert.pem"
            }
        }
```

```
    },
    "peers": {
        "peer0.org1.cross-border-transaction.com": {
            "tlsCACerts": {
                "path": "/tmp/crypto/org1.cross-border-
transaction.com/peers/peer0.org1.cross-border-
transaction.com/tls/ca.crt"
            },
            "url": "grpcs://peer0.org1.cross-border-
transaction.com:7051"
        },
        "peer0.org2.cross-border-transaction.com": {
            "tlsCACerts": {
                "path": "/tmp/crypto/org2.cross-border-
transaction.com/peers/peer0.org2.cross-border-
transaction.com/tls/ca.crt"
            },
            "url": "grpcs://peer0.org2.cross-border-
transaction.com:7051"
        }
    }
}
```

6. Con los archivos ya configurados se puede levantar el docker compose. Para esto hay que ubicarse dentro de la carpeta /explorer

Se valida que la base de datos que utiliza Hyperledger Explorer este vacía, y se ejecutan los siguientes comandos dentro de carpeta /explorer:

- `sudo docker-compose down`
- `sudo docker volume rm explorer_pgdata`

Y ahora si se procede a levantarlo:

- `sudo docker-compose up`

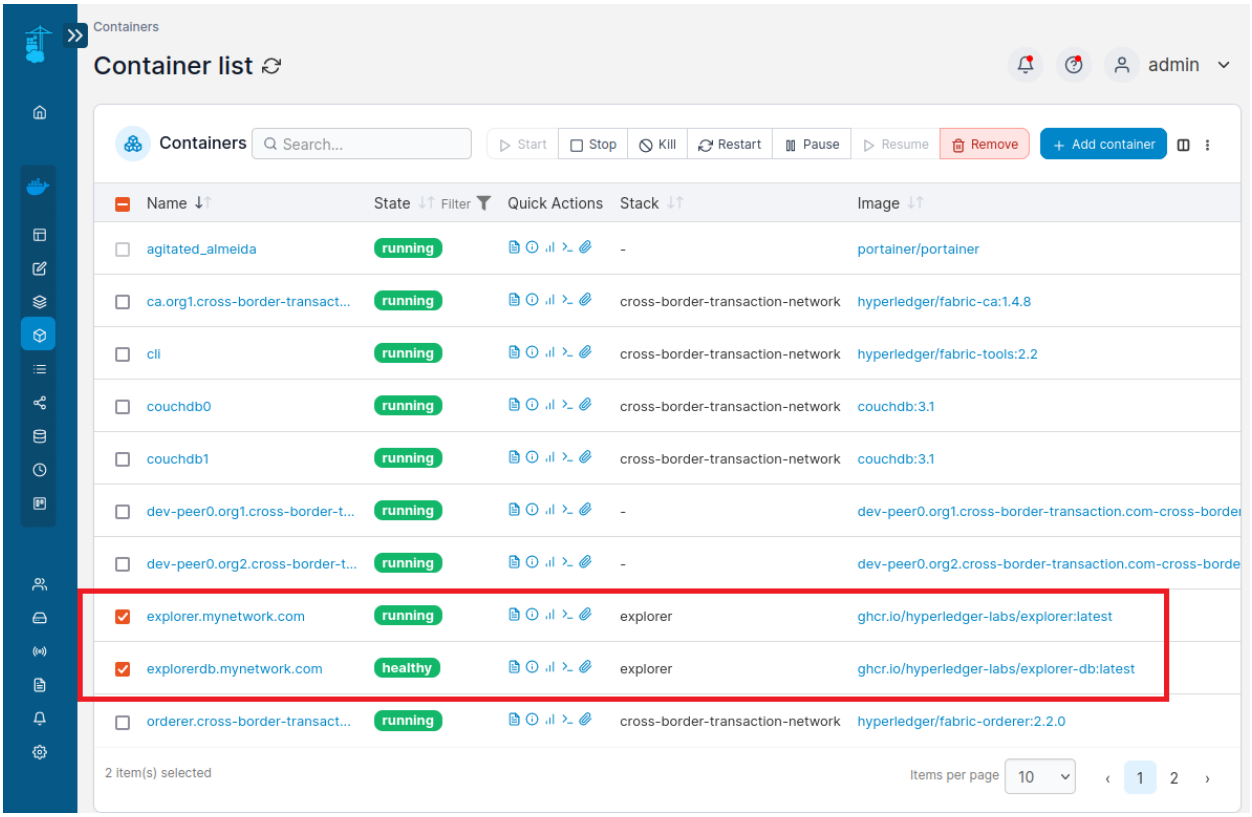
o si se quiere liberar la consola:

- `sudo docker-compose up -d`

Si es la primera vez ejecutando el comando, se descargará todo lo necesario para levantar el servidor web, recuerde que se utilizan varias tecnologías, como es el caso de postgresql, entre otras.

Se puede también seguir las instrucciones en caso de algún error, en el readme del repositorio oficial en https://github.com/hyperledger-labs/blockchain-explorer

7. Si el comando anterior fue exitoso y ejecuto todo sin problemas se puede verificar también la existencia de nuevos contenedores docker corriendo en el sistema. Mediante portainer se pueden ver estos dos contenedores nuevos, el explorer-db y



explorer.

Figura 13. Evidencia de los contenedores del Hyperledger Explorer en Portainer

8. Se utiliza un navegador para abrir el sitio http://localhost:8080/#/login. Allí se solicitan las credenciales de administrador del portal Hyperledger Explorer, entonces se ve el siguiente login.

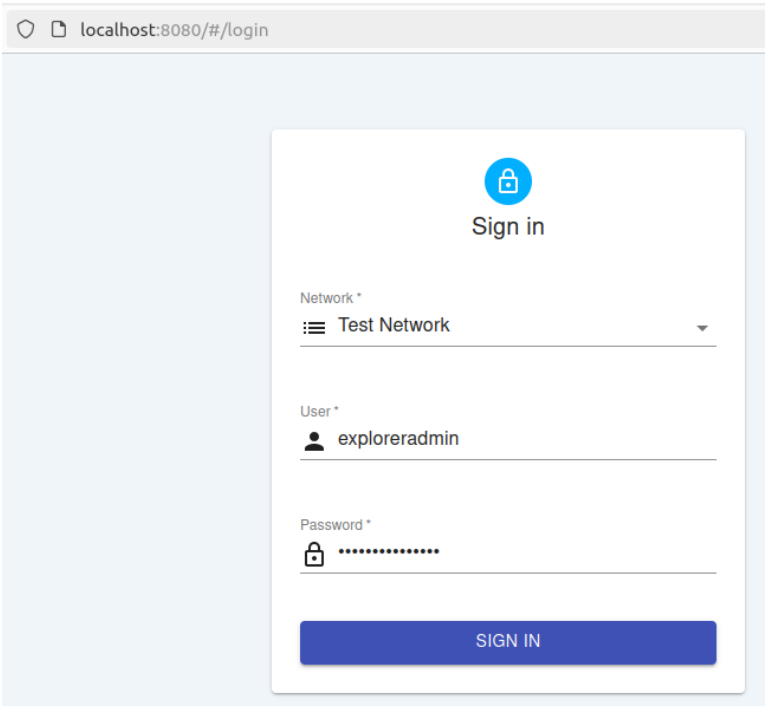


Figura 14. Login del Hyperledger Explorer en ambiente local

El usuario es exploreradmin y la contraseña por defecto exploreradminpw tal como lo especificaba el adminCredential en el archivo test-network.json.

9. Una vez se ingresa con las credenciales, se tiene acceso al portal de Hyperledger Explorer con todas sus funcionalidades.

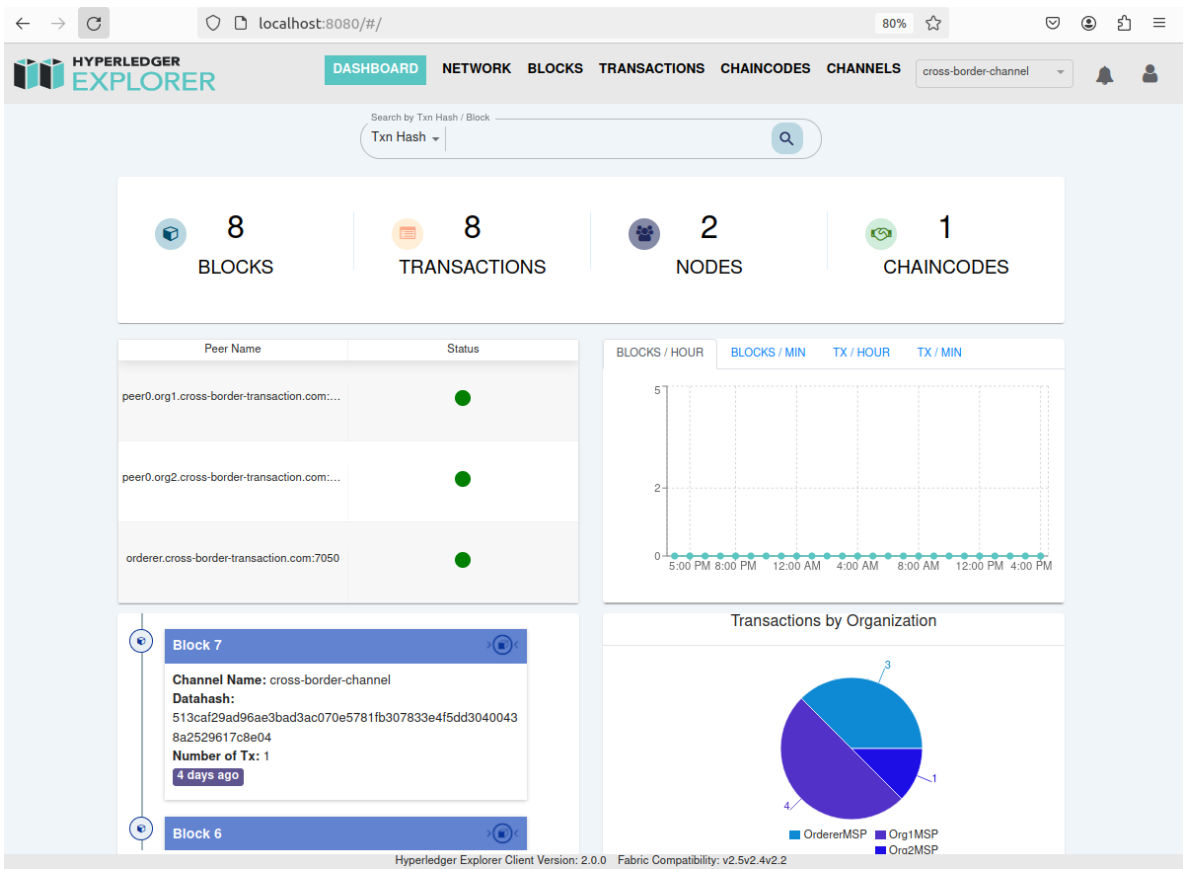


Figura 15. Dashboard del Hyperledger Explorer con la información de la red

Si se quiere bajar los contenedores subidos, parar y eliminar las imágenes creadas y el volume de portainer_data se puede hacer con el siguiente comando:

- `CHANNEL_NAME=$CHANNEL_NAME docker-compose -f docker-compose-cli-couchdb.yaml down && docker stop $(docker ps -a -q) && docker rm $(docker ps -a -q) && docker volume rm portainer_data`

Y se podría volver al paso 13 una vez se quiera subir de nuevo la red con las configuraciones o modificaciones que se hayan realizado.

REPOSITORIO DE IMPLEMENTACIÓN

https://github.com/jnaran24/hyperledger-fabric-first-step

LISTA DE REFERENCIAS

Aleksieva, V., Valchanov, H., & Huliyan, A. (2020a). Implementation of Smart Contracts based on Hyperledger Fabric Blockchain for the Purpose of Insurance Services. *2020 International Conference on Biomedical Innovations and Applications (BIA)*, 113–116. https://doi.org/10.1109/BIA50171.2020.9244500

Aleksieva, V., Valchanov, H., & Huliyan, A. (2020b). Implementation of smart-contract, based on hyperledger fabric blockchain. *2020 21st International Symposium on Electrical Apparatus and Technologies, SIELA 2020 - Proceedings*. https://doi.org/10.1109/SIELA49118.2020.9167043

Al-Jeshi, S., Tarfa, A., Al-Aswad, H., Elmedany, W., & Balakrishna, C. (2022). A Blockchain Enabled System for Enhancing Fintech Industry of the Core Banking Systems. *2022 International Conference on Innovation and Intelligence for Informatics, Computing, and Technologies (3ICT)*, 209–213. https://doi.org/10.1109/3ICT56508.2022.9990875

Amazon. (2023). *Amazon Managed Blockchain (AMB) Hyperledger Fabric Developer Guide*. Https://Docs.Aws.Amazon.Com/Managed-Blockchain/Latest/Hyperledger-Fabric-Dev/Hyperledger-Work-with-Channels.Html.

Binance Academy. (n.d.-a). *BEP-20*. Https://Academy.Binance.Com/Es/Glossary/Bep-20.

Binance Academy. (n.d.-b). *¿What is a stablecoin?* Https://Academy.Binance.Com/Es/Articles/What-Is-a-Stablecoin.

Chentouf, F. Z., & Bouchkaren, S. (2023). Security and privacy in smart city: a secure e-voting system based on blockchain. *International Journal of Electrical and Computer Engineering*, *13*(2), 1848–1857. https://doi.org/10.11591/ijece.v13i2.pp1848-1857

Choi, W., Woo, J., & Hong, J. W.-K. (2023). Gas Cost Analysis of Fractional NFT on the Ethereum Blockchain. *2023 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, 1–6. https://doi.org/10.1109/ICBC56567.2023.10174920

Ciriello, R. F., Torbensen, A. C. G., Hansen, M. R. P., & Müller-Bloch, C. (2023). Blockchain-based digital rights management systems: Design principles for the music industry. *Electronic Markets*, *33*(1), 5. https://doi.org/10.1007/s12525-023-00628-5

Corwin Smith, Johannes Schulz, Vardhan Shorewala, Jithil P Ponnan, Paul Wackerow, Sina Pilehchiha, Matthieu SCARSET, Ira Ko, Abdulhakeem Almidan, Ensar Yusuf Yılmaz, slightlyfloating, ryancreatescopy, Sam Richards, & Artur Gontijo. (n.d.). *ERC-20 TOKEN STANDARD*. Https://Ethereum.Org/En/Developers/Docs/Standards/Tokens/Erc-20/.

Habib, G., Sharma, S., Ibrahim, S., Ahmad, I., Qureshi, S., & Ishfaq, M. (2022). Blockchain Technology: Benefits, Challenges, Applications, and Integration of Blockchain Technology with Cloud Computing. *Future Internet*, *14*(11). https://doi.org/10.3390/fi14110341

Hyperledger Explorer, & Hyperledger Foundation. (2019). *Hyperledger Explorer*. Https://Wiki.Hyperledger.Org/Display/Explorer.

Hyperledger Fabric. (2020). *Hyperledger Fabric Model*. Https://Hyperledger-Fabric.Readthedocs.Io/En/Latest/Fabric_model.Html.

Hyperledger Foundation. (2023). *Hyperledger Foundation Projects - HYPERLEDGER FABRIC*. Https://Www.Hyperledger.Org/Projects/Fabric.

IBM. (2023). *What are smart contracts on blockchain?* Https://Www.Ibm.Com/Topics/Smart-Contracts.

Ibor, A., Edim, E., & Ojugo, A. (2023). Secure Health Information System with Blockchain Technology. *Journal of the Nigerian Society of Physical Sciences*, 992. https://doi.org/10.46481/jnsps.2023.992

Liu, X., Barenji, A. V., Li, Z., Montreuil, B., & Huang, G. Q. (2021). Blockchain-based smart tracking and tracing platform for drug supply chain. *Computers & Industrial Engineering*, *161*, 107669. https://doi.org/https://doi.org/10.1016/j.cie.2021.107669

Monrat, A. A., Schelén, O., & Andersson, K. (2020). Performance Evaluation of Permissioned Blockchain Platforms. *2020 IEEE Asia-Pacific Conference on Computer Science and Data Engineering (CSDE)*, 1–8. https://doi.org/10.1109/CSDE50874.2020.9411380

Osmani, M., El-Haddadeh, R., Hindi, N., Janssen, M., & Weerakkody, V. (2021). Blockchain for next generation services in banking and finance: cost, benefit, risk and opportunity analysis. *Journal of Enterprise Information Management*, *34*(3), 884–899. https://doi.org/10.1108/JEIM-02-2020-0044

Salem, A., Tareq, M., & Siam, S. (2021). *Report of Blockchain Techniques and Applications*. https://www.researchgate.net/publication/348559459

Sheth, H., & Dattani, J. (2019). Overview of Blockchain Technology. *Asian Journal For Convergence In Technology (AJCT) ISSN -2350-1146*. https://asianssr.org/index.php/ajct/article/view/728

Zhu, L., Wu, Y., Gai, K., & Choo, K.-K. R. (2019). Controllable and trustworthy blockchain-based cloud data management. *Future Generation Computer Systems*, *91*, 527–535. https://doi.org/https://doi.org/10.1016/j.future.2018.09.019