GRADO EN INGENIERÍA INDUSTRIAL

TRABAJO FIN DE GRADO

TRAZABILIDAD DE ACTIVOS CON TENCOLOGÍA BLOCKCHAIN

Autor: Diego Mendoza Pérez

Director: Atilano Fernández-Pacheco Sánchez-Migallón

Madrid

Julio de 2022

Declaro, bajo mi responsabilidad, que el Proyecto presentado con el título

Trazabilidad de activos con tecnología blockchain

en la ETS de Ingeniería - ICAI de la Universidad Pontificia Comillas en el

curso académico 2021-2022 es de mi autoría, original e inédito y

no ha sido presentado con anterioridad a otros efectos. El Proyecto no es plagio de otro, ni total ni parcialmente y la información que ha sido tomada

de otros documentos está debidamente referenciada.

Fdo.: Diego Mendoza Pérez Fecha: 25/ 08/ 2022

Autorizada la entrega del proyecto

EL DIRECTOR DEL PROYECTO

71216314B ATILANO FERNÁNDEZ-PACHECO

Firmado digitalmente por 71216314B ATILANO FERNÁNDEZ-PACHECO
Fecha: 2022.08.25 11:15:23 +02'00'

Fdo.: Atilano Fernández-Pacheco Sánchez-Migallón Fecha: 25/ 08/ 2022

Agradecimientos

A mi familia y a mis profesores. Especialmente, a mi director de proyecto.

TRAZABILIDAD DE ACTIVOS CON TECNOLOGÍA BLOCKCHAIN

Autor: Mendoza Pérez, Diego.

Director: Fernández-Pacheco Sánchez-Migallón, Atilano.

Entidad Colaboradora: ICAI – Universidad Pontificia Comillas

RESUMEN DEL PROYECTO

La necesidad de las empresas de realizar una transformación digital sirve de excusa perfecta para la introducción de nuevas tecnologías que generan valor por sí mismas y por su aplicación de forma que generan oportunidades de negocio y ventajas competitivas. En este aspecto, destaca Blockchain, cuyo aporte es idóneo para la transparencia e inmutabilidad de los datos, vitales en tiempos donde hay una desconfianza generalizada sobre los mismo.

1. Introducción

Gracias a la innovación que supuso Bitcoin como forma de moneda descentralizada, y a la tecnología empleada para ello, surgió una nueva forma de realizar transferencias monetarias sin la necesidad de depositar la confianza en una entidad o persona. Las aplicaciones que tienen el concepto de transferencia de valor sin necesidad de confianza abarcan un gran abanico de posibilidades y más en un contexto donde el dato, su veracidad y su inmutabilidad cobran una gran importancia.
El proyecto se divide en dos partes fundamentales. La primera, consiste en realizar una investigación a fondo sobre todo el ecosistema blockchain y comprender su aplicabilidad y ventajas a la trazabilidad de productos. Descubrir cómo puede esta tecnología ayudarnos a la hora de verificar el origen de distintos activos y los distintos procesos que han sido llevados a cabos a través de la cadena de suministros. La segunda, consiste en el desarrollo de una aplicación que realice dicho traqueo, desplegando una red blockchain localmente en hyperledger, programando los Smart contracts y, finalmente, conectándola con una aplicación a través de una API REST.

2. Metodologías

La aplicación consiste en traquear el origen y los procesos de calidad por los que ha ido pasando de un activo, en concreto del aceite de oliva. El cliente podría verificar de esta forma el origen del producto, con la confianza de que los datos proporcionados son transparentes e inmutables. Así mismo, también podría comprobar todos los pasos que ha ido dando en la cadena de suministros.

Se desplegará una red blockchain de forma local, para lo cual se usará la tecnología ofrecida por Hyperledge Fabric. La aplicación conectará a través de una API REST con la red de Fabric. Todo podrá ser guardado y consultado a través de una base de datos de CouchDB, proporcionada por Hyperledger. Con la red se interactuará gracias a un Smart contract con el que se podrá hacer consultas, agregar nuevos activos y transferirlos.
Esta es una muestra de cómo funciona la red:

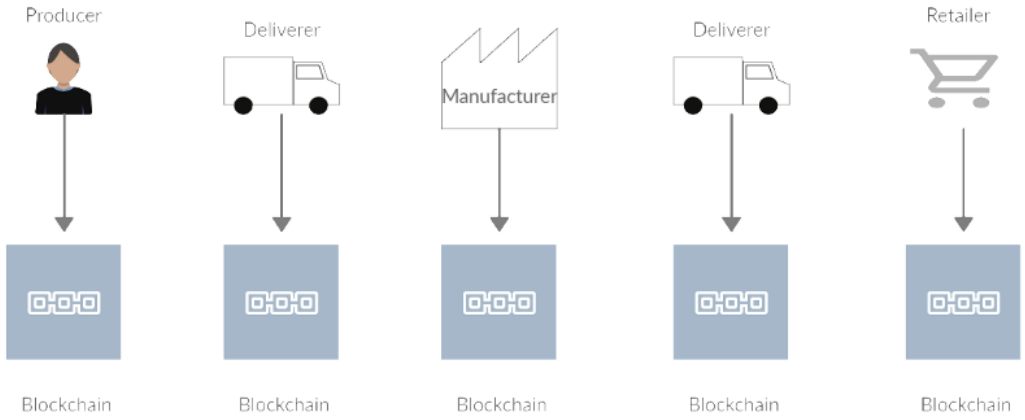

Fig. 1: Representación gráfica de la lógica de negocio

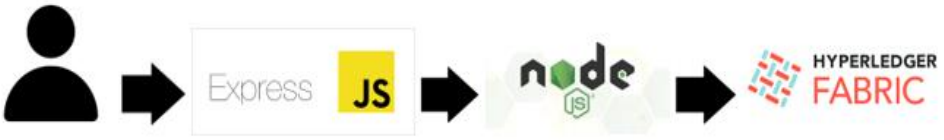

Fig. 2: Arquitectura de la aplicación

3. Resultados

En primer lugar, se observa cómo se ha desplegado con éxito una red privada basada en Hyperledger Fabric. Esta red contiene cuatro organizaciones que representa una cadena de suministros donde participan cuatro (productor, transportista, fabricante y minorista). Se crean las claves privadas de cada uno y se configura qué tipo de consenso se llevará a cabo. Cada unidad queda almacenada en un contenedor Docker. Posteriormente, se instala un Smart contract (o chaincode, como es llamado por los creadores de Hyperledger) que es capaz de interactuar con la red a través de peticiones realizadas por una aplicación externa. La interacción de la aplicación externa basada en Javascript y el SDK de hyperldeger, se hace con una api construido con Express. El chaincode permite realizar diversas funciones como son: una consulta masiva del ledger, consulta del estado de un activo a través de su identificador, agregar nuevos activos a la red, actualizarlos y transferirlos.

Se realizan diversas pruebas donde se comprueba que todas las funciones del Smart contract funcionan correctamente, se muestran a continuación un par de ellas, pero se exponen todos los resultados en el capítulo 4. Estas funciones son llevadas a cabo en una aplicación ofrecida por Hyperledger ya que al intentar establecer la conexión a través de la api de nuestra red, hay un error que no se ha podido resolver. Sin embargo,

se puede ver cómo es el funcionamiento con la aplicación en JavaScript que ofrece Hyperledger. En la red del caso de uso sería igual, solo que en el ejemplo proporcionado por Hyperledger es una red más simple (dos organizaciones) y no personalizada (no tienen nombres autoexplicativos, son simplemente Org1 y Org2) mientras que en la red desarrollada sí, hay cuatro organizaciones y sus respectivos nombres son Producer, Deliverer, Manufacturer y Retailer. Se puede apreciar la red personalizada mirando los nombres de los contenedores Docker:

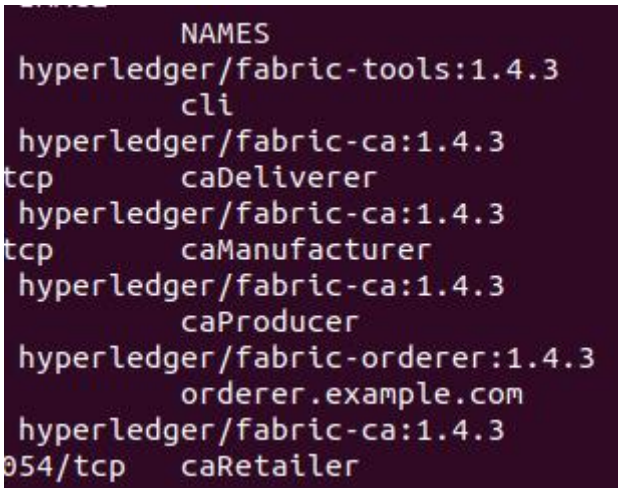

Fig. 3: Contenedores Docker

Aquí se pueden apreciar los nombres de los contenedores de los Certificados de autoridad de las organizaciones. El cli (comand line interface) y el orderer son elementos para el buen funcionamiento de la red.

En la Fig. 4, se muestra una consulta masiva del ledger.

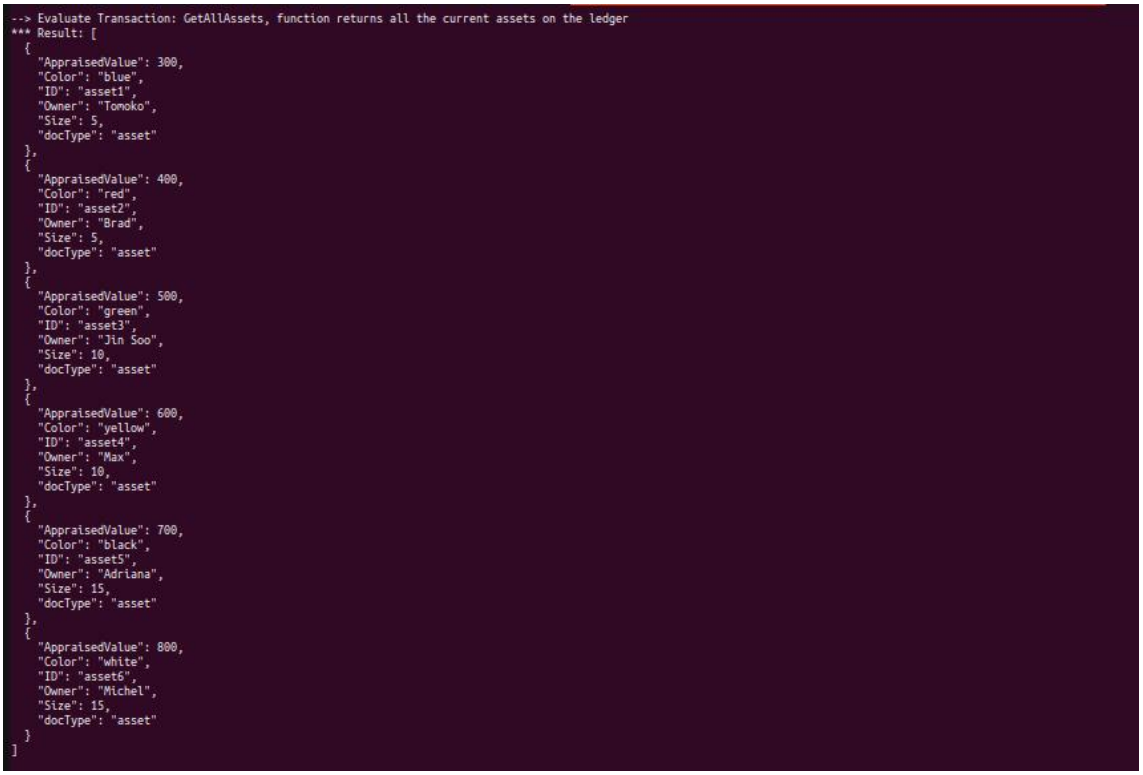

Fig. 4: Captura de pantalla de respuesta al llamar a la función GetAllAssets.

En la Fig. 5, se muestra cómo se transfiere un activo con éxito.

Fig. 5: Captura de pantalla de respuesta al llamar a la función TransferAsset.

Al hacer la petición POST en nuestra red de añadir un activo desde la cuenta del Productor, proporcionando sus llaves privadas, hay un error que no se ha podido resolver.

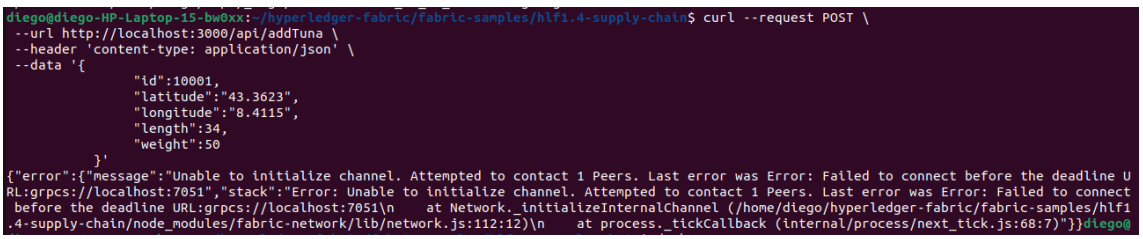

Fig. 6:Petición POST fallida

4. Conclusiones

Tras la realización del proyecto, se puede apreciar un correcto funcionamiento del sistema en general, habiendo conseguido el objetivo principal de representar una cadena de suministros en la blockchain, con los beneficios de trazabilidad, seguridad y automatización que se comentan en el capítulo 2. Lamentablemente, a la hora de conectarse a la red desde una aplicación externa personalizada, distinta a la aplicación cliente ofrecida por Fabric, hay un fallo en la conexión de nuestra aplicación cliente con el SDK a través de la API.

Finalmente, no ha sido posible conectar la aplicación con un front-end, dadas las complicaciones técnicas que exige Hyperledger (entre ellas, un alto conocimiento de JavaScript). En un principio, la idea era usar una red de testeo de Ethereum, que cuenta con un gran número de herramientas de interfaces gráficas. Finalmente, tras investigar más acerca del tema, se consideró interesante desarrollar una cadena de suministros de estas características (que debe ser lo suficientemente privada) en una blockchain privada, para lo cual Hyperledger ofrecía grandes ventajas. Al ser una blockchain menos desarrollada que su competidor Ethereum, el número de herramientas para el desarrollo de una interfaz gráfica es mucho más reducido. Se intentó realizar un front-end con la herramienta Hyperledger Composer, que permite gran flexibilidad para este fin, pero esta herramienta está obsoleta y no es compatible con las últimas versiones de Hyperledger Fabric.

5. Referencias

[1] "Writing your first application", *Hyperledger Fabric*, 2020. [Online]. Available: https://hyperledger-fabric.readthedocs.io/en/release-2.2/write_first_app.html

[2] "Commercial paper tutorial", *Hyperledger fabric*, 2020. [Online]. Available: https://hyperledger-fabric.readthedocs.io/en/release-2.2/tutorial/commercial_paper.html

TRACEABILITY OF ASSET USING BLOCKCHAIN TECHNOLOGY

ABSTRACT

The digital transformation the companies are doing make this moment the perfect one to introduce new technologies that generates value by themselves and their usage can generate business opportunities and competitive advantages. In this aspect, blockchain bright up. Its usage is perfect for the transparency and immutability of the data, a critical point in today´s world where there a generally spread mistrust on them.

1. Introduction

Thanks to the innovation brought by Bitcoin as a decentralized coin, and the technology used for that purpose, emerged a new way to make monetary transference with no need to trust any entity or person. There is several applications for the transference of value with no trust needed concept. It becomes crucial in the current world, where the data, its veracity and its immutability have a huge relevance.
The Project is divided in two principal parts. The first one consists in do reasearches about the blockchain environment and understand their use case and advantages it can offer in terms of asset´s traceability. To find out how this technology allows us to verify the origin of the assets and the processes it has gone through the supply chain. The second one consists in the development of an application which can track an asset, deploying locally an Hyperledger, programming the smart contracts and connecting it to an application through an API REST.

2. Metodology

The application can track the origin and quality processes suffered by an asset throught the supply chain. The client will be able to verify the origin of the product, knowing that the data given is transparent and immutable. The client will also able to see all the stages the product has gone through the supply chain. An hyperledger network will be deployed locally. Through an API REST, the app will connect to a Fabric network. Everything will be stored in a CouchDB data base. Thanks to a smart contract, it will be possible to consult, add new assets, update them or transfer them.
Here it is shown how the net works:

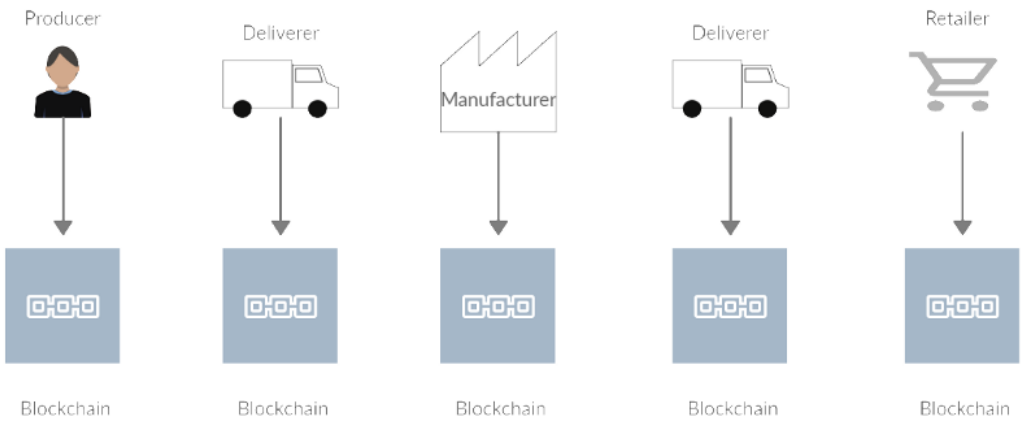

Fig. 1: Representación gráfica de la lógica de negocio.

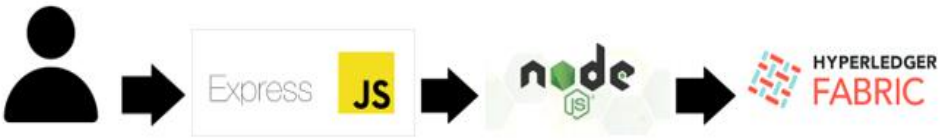

Fig. 2: Arquitectura de la aplicación

3. Results

First, it is possible to appreciate the development of an Hyperledger Fabric private network successfully. This network is formed by two organizations which represent a small supply chain where there is two participants, the farmer and the seller. It is created the private keys of each of the participants and set up the type of consensus that the net will have. After that, a smart contract (also called chain code by Hyperledger creators) is installed and it is able to ineract to with the network through petitions made by an external application. The chaincode allow the user to carry out different functions as a massive consult of the ledger, consult about a determined asset based on its identification number, add new assets to the network, update them or transfer them to another owner.

Here it is shown the personalized net taking a look at docker containers names:

```
          NAMES
hyperledger/fabric-tools:1.4.3
          cli
hyperledger/fabric-ca:1.4.3
tcp       caDeliverer
hyperledger/fabric-ca:1.4.3
tcp       caManufacturer
hyperledger/fabric-ca:1.4.3
          caProducer
hyperledger/fabric-orderer:1.4.3
          orderer.example.com
hyperledger/fabric-ca:1.4.3
054/tcp   caRetailer
```

Fig. 3: Contenedores Docker

Different tests are carried out and they show how every function works correctly. Next, a couple of pictures showing how two different function works correctly although all of them working well will be showed in chapter 4.

In the Fig. 4, it is showed a massive consultation to the ledger.

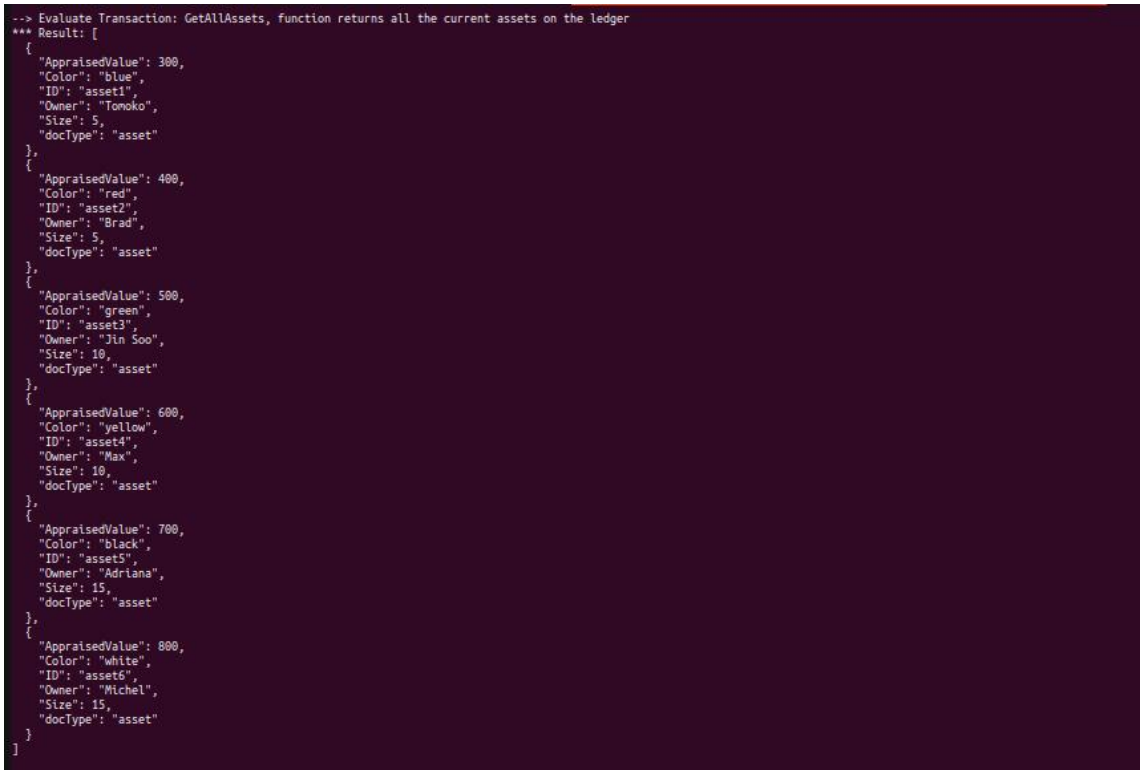

Fig. 4: Screenshot after calling GetAllAssets function.

In the Fig. 5, it is showed how an asset is transferred to another owner successfully.

Fig. 5: Screenshot after calling TransferAsset function.

Al hacer la petición POST en nuestra red de añadir un activo desde la cuenta del Productor, proporcionando sus llaves privadas, hay un error que no se ha podido resolver.

When a POST request is made in the net to add an asset from Producer account, giving its private keys, there is an error which cannot be fixed:

Fig. 6: Petición POST fallida.

4. Conclusions

After doing the Project, it is possible to see how the system works correctly. The principal goal has been achieved which was to represent a supply chain in a blockchain with the benefits it brings: trazability, security and automatizations that are commented in chapter 2.

Finally, it is has not been possible to connect the application with a front-end given the technical difficulties demanded by Hyperledger(between them, an advanced knowledge of JavaScript). In the beginning, the idea was to use an Ethereum testnet which counts with good number of front-end tools. Finally, after being investigating about the topic, it was considered interesting to develop a supply chain like this (which needs to have a high degree of privacy) to develop a private supply chain and for this purpos Hyperledger offered great advantages. Being a less famous technology than its competitor Ethereum, the number of tools to the development of a graphic interface was much lower. It was tried to develop a front-end using Hyperledger Composer (which

allows a great flexibility to this end) but was found out that this tool is obsolete and would not match with the last versions of Fabric.

5. References

[1] "Writing your first application", *Hyperledger Fabric*, 2020. [Online]. Available: https://hyperledger-fabric.readthedocs.io/en/release-2.2/write_first_app.html

[2] "Commercial paper tutorial", *Hyperledger fabric*, 2020. [Online]. Available: https://hyperledger-fabric.readthedocs.io/en/release-2.2/tutorial/commercial_paper.html

Índice de la memoria

Lista de figuras

Lista de tablas

Capítulo 1: Introducción

Las cadenas de suministros actualmente presentan grandes pérdidas debidas a la falta de información entre las distintas partes que la conforman. Blockchain es capaz de arreglar mucho de los problemas que presenta la cadena de suministros actualmente. Sobre todo, en términos de integridad y visibilidad de los datos. Muchas organizaciones no están preparadas para corregir los problemas que hay o puede haber en la cadena de suministros debido principalmente a la falta de confianza en los datos de estas. La falta de intercambio de información cliente-comprador, la integridad y la visibilidad de esta, provoca que haya dificultados a la hora de tomar decisiones. Con blockcain, es posible tener una información de confianza en tiempo real con la que tomar decisiones acertadas.

La cadena de suministros de la mayoría de las empresas está poco digitalizada, basándose una gran parte de ellas en el uso de papel. Los directores de grandes compañías afirman que una de sus principales prioridades es mejorar la eficiencia (69%) y digitalizar la cadena de suministros (59%) [1]. Una mejora en la información obtenida en la cadena de suministros es traducida en unas decisiones más acertadas. Este proceso se ha visto acelerado en los últimos años gracias a la pandemia COVID-19. El porcentaje de cadenas de suministros totalmente digitalizadas se espera que incremente hasta 6 veces en los próximos dos años. Esto crea una gran oportunidad de negocio para aquellos que sean capaces de ofrecer una solución técnica idónea.

Muchos directores miran a blockchain como una gran oportunidad para mejorar sus procesos, pero la parte de blockchain es solo el pico del iceberg. El uso de blockchain trae consigo una serie de retos a superar: acuerdos en flujos de procesos, desarrollar un modelo de gobernanza apropiado o establecer los acuerdos legales requeridos son solo algunos de ellos. Para explotar el potencial de blockchain es necesario un alto grado de colaboración.

A continuación, serán nombradas algunas de las ventajas competitivas que blockchain puede llegar a ofrecer: integridad de los datos, automatización de procesos, una única fuente de datos, tokenización de activos digitales y físicos, una mejora en cuenta saber donde un activo se encuentra constantemente, asegurar que las condiciones en las que se encuentra el producto son adecuadas en todo momento, exactitud de los datos…

Blockchain es una de las tecnologías que más ilusión ha creado en el mercado y que menos entendida ha sido pero que, tras toda la cortina de fuegos artificiales que la rodea, hay un valor real que puede ser añadido a la empresa. Se advierte de vital importancia formar a las nuevas generaciones en este asunto de cara a poder solventar las necesidades presentes y futuras de las empresas. Resulta necesario separar la tecnología blockchain de su evidente relación con las criptomonedas y la especulación inherente a ellas.

Capítulo 2: Descripción de las tecnologías

A continuación, se describirán las tecnologías usadas en este proyecto, incidiendo especialmente en las más importantes.

2.1 BLOCKCHAIN

2.1.1 Introducción a blockchain

En las siguientes líneas se pretende hacer una breve introducción a qué es blockchain y la tecnología que lo sustenta sin entrar en muchos detalles técnicos.

Blockchain, como conjunto de herramientas que la forma, fue creada en 2008 con Bitcoin, un sistema que aspira a ser una moneda descentralizada, fuera del alcance de los gobiernos, en la que la confianza es eliminada como factor elemental en el intercambio de transacciones.

Blockchain no deja de ser una base de datos descentralizada capaz de almacenar un historial de transacciones de manera transparente, segura e inmutable.

Si atendemos a su traducción al español, cadena de bloques, se puede contemplar que es una cadena de bloques de información. Bloques porque un conjunto de transacciones es almacenado en un único elemento que recibe el nombre de bloque. Cadena porque estos bloques se van depositando de forma secuencial, en la que cada bloque nuevo presenta un registro de cual es el bloque que lo procede.

Para entender un poco mejor esto, se explicará de forma breve cómo funciona Bitcoin. Los usuarios lanzan transacciones que son recogidas por los distintos mineros formando un conjunto de transacciones (no todos los mineros recogen las mismas transacciones, recogen una pequeña parte de las muchas que se están realizando en ese momento, principalmente aquellas que mayores comisiones ofrezcan). Los mineros, mediante el uso de potencia computacional resolviendo un numéricamente complicado problema matemático, forman un bloque. El primer minero capaz de formar un bloque bajo las condiciones necesarias (condiciones cada vez más exigentes para asegurar la red ante el incremento de mineros en la misma), obtiene el privilegio de añadir el bloque a la cadena tras haber sido considerado como bueno por la mayoría de la red. Posteriormente, todos los nodos que forman la red incluyen este bloque en su “ledger” (libro mayor) o cadena. El minero recibe una recompensa en forma de bitcoins por su buena labor, la de haber ordenado y encriptado las transacciones y haber realizado el procedimiento de forma correcto. La idea del dinero digital no es algo novedoso, era algo sobre lo que se llevaba hablando mucho tiempo, pero que no podía llevarse a cabo hasta que blockchain consiguió el problema del doble gasto (la posibilidad de copiar y pegar una transacción varias veces).

El proceso es **seguro** porque la información está encriptada, asegurando el contenido y la anonimidad (mediante la identificación a través de una clave pública) de la transacción. Cada bloque creado está relacionado con el anterior a través de un Hash. Un Hash es la encriptación de la información mediante un algoritmo, resultando en una cadena alfanumérica. Al introducir el identificador del bloque anterior, las transacciones contenidas y su fecha se genera una cadena alfanumérica. En el caso concreto de Bitcoin, la prueba de trabajo (PoW, por sus siglas en inglés) consiste en encontrar un parámetro (nonce) que, al hacer el hash a todo el bloque, el valor sea inferior (con un número determinado de ceros a la izquierda) a la dificultad establecida por la red en ese momento. Por cómo es la lógica del algoritmo que hashea el bloque, no es posible calcular estos valores analíticamente, siendo necesario el uso de fuerza bruta para conseguirlo. En otras palabras, un pequeño cambio en el bloque desemboca en un cambio caótico en el valor final del hash. Bitcoin establece que el tiempo medio de transacción sean diez minutos, por tanto, cuando crece el número de mineros y este tiempo se ve reducido, aumenta la dificultad de la prueba (exige que el hash final tenga mas ceros a la izquierda). De esta forma, la dificultad del proceso es ajustable. Gracias a este mecanismo, el coste para un hacker de crear un bloque que invierta las transacciones de los bloques inferiores es cercano a infinito [2]. En la Fig. 7 se muestra el funcionamiento de la creación de bloques de la red Bitcoin.

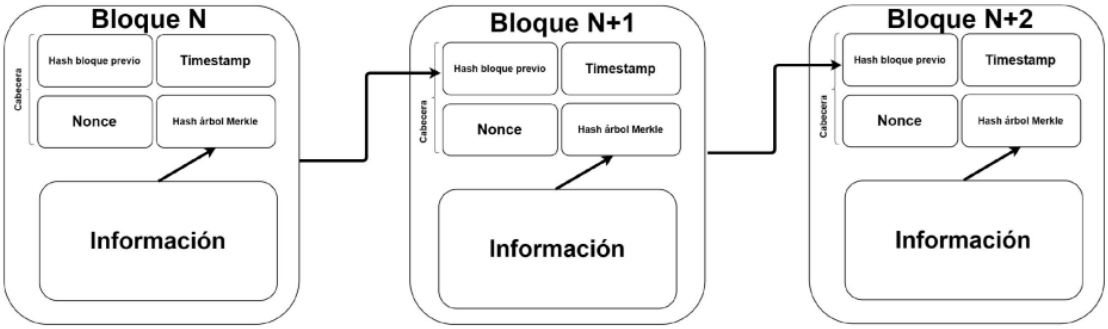

Fig. 7. Funcionamiento de la red Bitcoin [19].

El proceso es **descentralizado** porque todos los nodos de la red tienen el mismo poder y la misma importancia (cuando hablamos de blockchain públicas) y para llevar a cabo una transacción es necesaria la aprobación de la mayoría de la red. Sin embargo, en redes privadas es posible alcanzar cierto grado de descentralización, aunque haya nodos con más permisos que otros, esto se discutirá en próximas páginas.

El proceso permite **eliminar la confianza**. Permite el intercambio de valor entre organizaciones sin la necesidad de confiar en las otras organizaciones con las que se va a realizar el intercambio. Esto es un gran salto, pues siempre que ha habido un intercambio, ha sido necesario depositar la confianza de que la otra persona tenía garantías suficientes. Por ejemplo, en el sistema bancario actual, depositas la confianza de que un banco es retenedor de tu dinero. Otro ejemplo es cuando haces un pedido online, cuando pagas, confías en que, si no te llegase el producto, la tienda te devolverá tu dinero.

El protocolo de consenso es la parte fundamental para eliminar la confianza y es distinto según el proyecto. Más tarde, serán descritos los principales protocolos de consenso.

2.1.2 El verdadero valor de blockchain

A pesar de tener desarrollados sistemas que aportan soluciones altamente transaccionales desde hace mucho tiempo, antes de que llegase blockchain era difícil crear un sistema que permitiera compartir información y que una vez compartido, ésta fuera totalmente transparente y además poder comprobar que esa información sea inmutable. También soluciona un problema de los sistemas distribuidos, que era comprobar que el sistema del estado fuera el mismo en todos los nodos de la red. Sin blockchain, había una gran dificultad para comprobar que todos los nodos de la red tuvieran en todo momento la misma información.

Una de las grandes aplicaciones es cuando se necesita una única fuente de verdad. Por ejemplo, en el caso de importaciones de productos, los que usan para traquear los activos son todavía muy rudimentarios. Las transacciones se van registrando en papel, pudiendo cada parte tener versiones distintas sobre qué trayecto ha seguido un activo o sobre el punto en el qué se encuentra. Es un ejemplo perfecto donde blockchain puede tener un gran potencial, haciendo que la información que tiene cada una de las partes sea coherente.

2.1.3 Algoritmos de consenso

Como se trata de un sistema descentralizada, se necesita de alguna forma para poder asegurar que esta información que está repartida de manera descentralizada sea consistente (la misma en todos los nodos, evitando añadir transacciones falsas) y para eso necesitamos un mecanismo llamado consenso.

PoW

`Proof work´ o prueba de trabajo consiste en la resolución de un problema para poder añadir un nodo a la red.

El problema de PoW es que necesita realizar un alto gasto energético para asegurar la robustez de la red. Sin embargo, se hipotetiza que es precisamente éste gasto lo que hace imbatible la red. Pensemos. Si los mineros realizan una gran inversión para minar (con el incremento de la dificultad de la prueba, es necesario realizar una gran inversión para poder minar), serán los primeros interesados en que la información fuese verídica. Si intentasen introducir información falsa, los nodos del resto de la red no lo aceptarían y le denegarían el bloque. Esto conllevaría en no rentabilizar la inversión realizada y perder dinero. Es entonces justo aquí, en el coste de oportunidad, la verdadera fortaleza. Los mineros son aliados, no enemigos.

PoS

`Proof of Stake´ prueba de participación es un protocolo que surge para intentar corregir el problema del gasto energético (que, como hemos visto, quizás es necesario). Consiste en que la probabilidad de ser el encargado de minar el bloque es proporcional al número de criptomonedas que poseas. Esto puede llegar a dar un problema de centralización, pues si un usuario tuviera demasiadas criptomonedas, tendría un gran control en la red.

DPos

`Delegated proof of stake´ viene a arreglar el problema de la centralización sufrida en PoS. Todos los participantes de la red tienen capacidad de voto proporcional a la cantidad de criptomonedas que posean. Con su voto, deciden qué nodo será el ejecutor del bloque. De esta forma, el proceso de voto sigue siendo descentralizado. Realmente si los nodos ejecutores se pusieran de acuerdo, podrían no llevar a cabo una transacción (nunca modificarla), pero simplemente en la siguiente ronda, estos nodos ejecutores no serían reelegidos y los nuevos nodos ejecutores lo harían. De esta forma se compensa a los nodos 100% honestos. Mejora la escalabilidad de la red.

Byzantine Fault Tolerance

Se plantea el problema de los generales bizantinos como metáfora para entender el problema que se da cuando un conjunto de elementos informáticos debe actuar en concordancia.

Se trata de varios grupos formados por generales y soldados, que tienen que tomar una decisión estratégica y decidir sobre si atacar un castillo retirarse. Cualquiera de las dos opciones será buena siempre y cuando todos los grupos hagan lo mismo (haya una única verdad).

El problema es que la comunicación puede no ser confiable (puede haber un traidor en las tropas): el general puede dar una orden errónea o un teniente puede transmitir información contraria a la orden dada por un general leal.

Blockchain permite resolver este problema, permite verificar las transacciones correctamente incluso con la presencia de nodos malintencionados en la red.

Un ejemplo muy gráfico de esto puede ser un avión. El avión debe estar preparado en caso de fallo de uno de sus sensores para seguir funciónando correctamente.

2.1.4 Blockchain vs. Base de datos

Las bases de datos se encuentran por todas partes. Una base de datos no es más que un lugar donde se encuentra almacenada una información y es de rápido acceso. Hace más de 50 años, las empresas empezaron a implementar esta tecnología y poco a poco la

demanda por unas bases de datos más potentes hicieron avanzar hacia bases de datos relacionales.

Con el tiempo y la interconectividad actual, ante la necesidad de varias empresas de acceder a la misma información, se crearon las bases de datos distribuidas. Con esto, se posibilitó una mejor coordinación y eficiencia, permitiendo a más de una persona acceder a la misma información. Sin embargo, estas bases de datos presentan algunos problemas:

- ¿Quién es confiable para compartir tu información?

- ¿Cómo detectar las identidades de cada uno?

- ¿Cómo limitar las acciones a realizar por cada uno de los participantes?

- ¿Cómo identificar a participantes mal intencionados?

- ¿Qué hacer ante desacuerdos de la información?

Algunos argumentan que blockchain es simplemente una base de datos distribuida. Nada más lejos de la realidad, pues es mucho más que eso. Blockchain soluciona los problemas presentados por las bases de datos distribuidas, eliminando a una autoridad central y consiguiendo **eliminar la confianza.** Confianza puede significar muchas cosas, por ejemplo, se puede referir a la legitimidad de todos los participantes para leer/escribir en la base de datos. O a evitar que los hackers roben información comprometida. O evitar pérdidas de rendimientos por distintos ataques. [3]

La información está protegida criptográficamente y se asegura la descentralización mediante los distintos protocolos de consenso que puede haber.

A diferencia de base de datos, se muestra el histórico de las transacciones (en una base de datos normalmente solo se muestra la última modificación).

Aun así, es cierto que sigan existiendo casos de uso donde una base de datos es más apropiada que una blockchain. Hay varios parámetros que son importantes cuando construyes una aplicación informática (rendimiento, latencia, quienes van a leer, escribir, quien lo va a gestionar…). Una base de datos es superior a blockchain en términos de capacidad. Por tanto, el tiempo de latencia va a ser menor. El tiempo de latencia de las blockchains públicas, normalmente por cómo es la infraestructura, es muy alto. Incluso en algunas redes con muchísimos nodos (Bitcoin, Ethereum...) el tiempo de latencia (el tiempo que hay entre la confirmación del primer y último nodo de la red) es indeterminado.

Otra cosa para destacar es que una base de datos nunca va a proporcionarte acceso a la escritura de base de datos si tú no eres confiable (ninguna empresa te va a dejar escribir en su base de datos si tú no eres confiable, obviamente); normalmente está centralizado (ejemplo, Oracle). Cualquier usuario no confiable, no se le permite interactuar con una base de datos. Pero en blockchains públicas, puede haber escritores en la blockchain no confiables y la red y el consenso determinará qué se escribe y qué no se escribe. En una

red privada es improbable que haya escritores no confiables, si los hubiera estarían muy reconocidos.

´

2.1.5 Tipos de blockchains

A pesar de que todo comenzó con blockchains públicas, se fueran creando diferentes formas de aplicar esta tecnología. Los diferentes tipos de blockchains se puede dividir en públicas o privadas y en permisionadas o no permisionadas.

Públicas: Son aquellas en las que cualquier persona puede montar un nodo y unirse como participantes de la red, con acceso a las transacciones y capaces de validar las mismas participando en el protocolo de consenso. Sus principales ventajas son la transparencia, accesibilidad, descentralización y seguridad.

Privadas: El validador de las operaciones ha de tener una licencia que lo acredite como tal.

Permisionadas: Los participantes de la red han de tener permisos para poder participar en la red.

No permisionadas: Aplica tanto a blockchains públicas como privadas. Cualquier participante puede leer transacciones. En el caso de blockchains privadas, se puede pensar como que cualquier persona puede acceder a las transacciones (para auditarlas, por ejemplo), pero solo ciertas personas pueden validarlas.

En la Tabla 1 se realiza una comparación de los atributos entre las distintas tipos de blockchain.

Tabla 1. Comparación de blockchains permisionadas, no permisionadas y bases de datos [20].

	Permissionless Blockchain	Permissioned Blockchain	Central Database
Throughput	Low	High	Very High
Latency	Slow	Medium	Fast
Number of readers	High	High	High
Number of writers	High	Low	High
Number of untrusted writers	High	Low	0
Consensus mechanism	Mainly PoW, some PoS	BFT protocols (e.g. PBFT [5])	None
Centrally managed	No	Yes	Yes

2.1.6 Conectando esta tecnología con la empresa

En un conjunto de participantes de una organización, se suele estar compartiendo información constantemente. Para ello, se suele usar un sistema informático que lo permita. Blockchain permite a los participantes mediante un proceso descentralizado gestionar la información a través de contratos inteligentes, asegurando inmutabilidad, seguimiento y veracidad de la información transaccionada.

Las empresas tienen necesidades que como veremos puede chocar con el uso de blockchains públicas, principalmente en términos de privacidad. En una empresa:

-Los participantes han de estar definidos claramente.

-Las redes deben ser permisionadas.

-Es necesario que haya un alto rendimiento de transacciones, acompañado de una baja latencia.

-Privacidad y confidencialidad de los datos comerciales; aunque la información esté encriptada, las empresas no quieren que la información de la misma sea de acceso público.

Normalmente, en el mundo actual nos encontramos ante un conjunto de empresas u organizaciones que necesitan interactuar entre ellas para poder trabajar. Lo suelen hacer a través de APIs o mensajería más tradicional como el correo electrónico. El caso es que cada empresa ejecuta un proceso que les permite compartir información. El problema viene cuando cada empresa tiene una versión de la verdad distinta, un repositorio de información cuyo contenido no es coherente con el de las otras empresas. Estas diferencias pueden venir de formatos, modelos o tecnologías distintas. De esta forma, cuando se necesite consultar el estado del mundo o "world state" es, en un gran número de veces, información distinta. Esto no es eficiente; para serlo, es necesario tener un sistema que permita transaccionar información en tiempo real y que además en cada transacción sea posible verificar un único registro.

En general, los modelos centralizados son más costosos porque no solo tienen que satisfacer los modelos/infraestructuras de su organización sino del conjunto de organizaciones con las que interactúa y cuyas infraestructuras difieren de la propia. Esto es ineficiente y presenta el problema comentado anteriormente, en el que cada organización tiene su propio registro. Esto presenta problemas de posibles manipulaciones de los registros.

Con blockchain, se logra que todos los participantes se pongan de acuerdo para generar transacciones y la inmutabilidad queda resuelta gracias a que los registros no pueden ser cambiados ni eliminados y, en general, es sencillo demostrar la propiedad de los registros porque cada pieza de información tiene asociado un origen. Con esto, logramos resolver:

-Múltiples versiones de la verdad

-Reducir errores, fraudes, disputas…

-Reducir ineficiencias de intermediarios.

2.2 ETHEREUM

2.2.1 Introducción a Ethereum

En 2014, de la mano de Vitalik Butterin, nació Ethereum. Su creador, intentó convencer sin éxito a la comunidad Bitcoin del momento a implementar Smart contracts sobre la red de Bitcoin. Esto fue algo que no prosperó porque el objetivo de Bitcoin era convertirse en modo de pago y la implementación de estos Smart contracts eran fuente de hackeos y pérdida de confianza en la moneda. Por ello, Vitalik decidió crear Ethereum, una plataforma similar a Bitcoin, pero sobre la que se podían desplegar contratos inteligentes, dando lugar a las hoy conocidas como DApps (Decentralized Applications). Hay varias diferencias entre Ethereum y Bitcoin (en el encriptado, minado, cantidades emitidas…).

2.2.2 Qué trajo Ethereum al panorama: Los smart contracts

Un contrato inteligente no es más que una aplicación, un lenguaje de programación con un conjunto de reglas que van a gestionar el ciclo de vida de un activo que necesita ejecutarse en un entorno descentralizado. Son similares a los contratos tradiciones con las ventajas de que no se necesita un intermediario (un notario) y no admite interpretaciones. Si se dan unas condiciones determinadas, se ejecutarán unas órdenes u otras. Esto permite una mayor velocidad de ejecución y una importante reducción de costes.

2.2.2.1 Casos de uso de los Smart contract

Los usos donde de momento se mira a blockchain como una gran alternativa son:

-Banca con finanzas descentralizadas (DeFi). Son contratos financieros (préstamo e inversión principalmente) basados en blockchain, eliminando el intermediario (hasta ahora, el banco), creando un sistema financiero más transparente, seguro y barato. Como suelen funcionar estas plataformas es de la siguiente manera: unos usuarios hacen un depósito de sus criptomonedas para recibir intereses a cambio. Otros usuarios, con necesidad de financiación, toman prestadas estas criptomonedas, pagando un interés por ellas.

-Registro y almacenamiento de datos. Con esta tecnología es posible registrar cualquier evento del mundo real en una base de datos.

-Digitalización de títulos de propiedad. Muchas de las propiedades del mundo real son tokenizables, pudiendo registrar su autoría.

-Creación de contratos de seguros. Gracias a esta tecnología, es posible que los Smart contracts interactúen con un oráculo, que haría de intermediario entre las partes, para todos los temas relacionados con reclamaciones, pagos, cobros…

-Seguimiento en la cadena de suministros y certificación de autenticidad. Muy en línea con el tema a tratar en este documento, la tecnología blockchain puede mostrar la trazabilidad de un activo a través de la cadena de suministros y verificar el origen del mismo.

-Gobernanza y votaciones.

-Identidad digital.

Estas son las primeras aplicaciones de los Smart contracts, los sectores donde han encontrado una aplicación inmediata. Sin embargo, a lo largo de los años irán surgiendo nuevas oportunidades y se creará una industria en torno a los Smart contracts [4].

2.3 HYPERLEDGER

2.3.1 Introducción a Hyperledger

Hyperledger nació en el año 2015 con el objetivo de aunar las necesidades de una empresa con las ventajas aportadas por blockchain. IBM e Intel, al darse cuenta de que era muy difícil desarrollar soluciones por su propia cuenta, donaron sus proyectos cripto para que la fundación Hyperledger tratara de desarrollar soluciones. Cabe destacar que no existe relación entre Hyperledger e IBM o Intel. Actualmente, se encuentran hospedados por Linux Foundation. Hyperledger trata de ofrecer un conjunto de frameworks y herramientas de carácter empresarial para implementar redes blockchain.

Es de código abierto, con todas las ventajas que eso conlleva: soluciones de alta gama (interoperabilidad, mejorando la red), corrección rápida de errores y personalización para la mejora (cada desarrollador personalizará el código acorde a sus necesidades, promoviendo creatividad e innovación), reducción de costes (no es lo mismo tener que crear una aplicación apropiada para tu negocio que usar un código abierto y personalizarlo).

2.3.2 Arquitectura

-Diseño modular: Hyperledger trabaja con marcos modulares extensibles. Esto quiere decir que los mismos pueden ser reutilizados en distintos proyectos. Esto es una gran característica pues se puede trabajar separadamente por bloques para luego combinarlos y crear una aplicación. Así mismo, se pueden reutilizar bloques que forman otras aplicaciones, ahorrando tiempo y siendo altamente eficientes en la construcción de nuevos registros distribuidos.

-Interoperable: Hasta ahora, uno de los grandes problemas que presentan en general los proyectos blockchains es la interoperabilidad entre los mismos. Hyperledger decide desde el primer momento ser interoperable de forma que todo lo desarrollado en ella, sea exportado a otras redes blockchains, consiguiendo así una mayor velocidad en la adopción de esta tecnología.

- “Criptomoneda-agnósta”: Hyperledger afirma que nunca emitirá su propia criptomoneda, no quiere verse metido en todo el ecosistema de inflación y especulación que existe en torno a las criptos. El sistema no necesita una criptomoneda para operar, aún así, da la opción a las empresas de crearla porque puede llegar a tener cierto sentido en algunos entornos [3].

2.3.3 Hyperledger: Frameworks y herramientas

La comunidad de Hyperledger ha creado distintos frameworks y herramientas para satisfacer distintas necesidades. En la Fig. 8, se muestra un resumen a alto nivel de ellas.

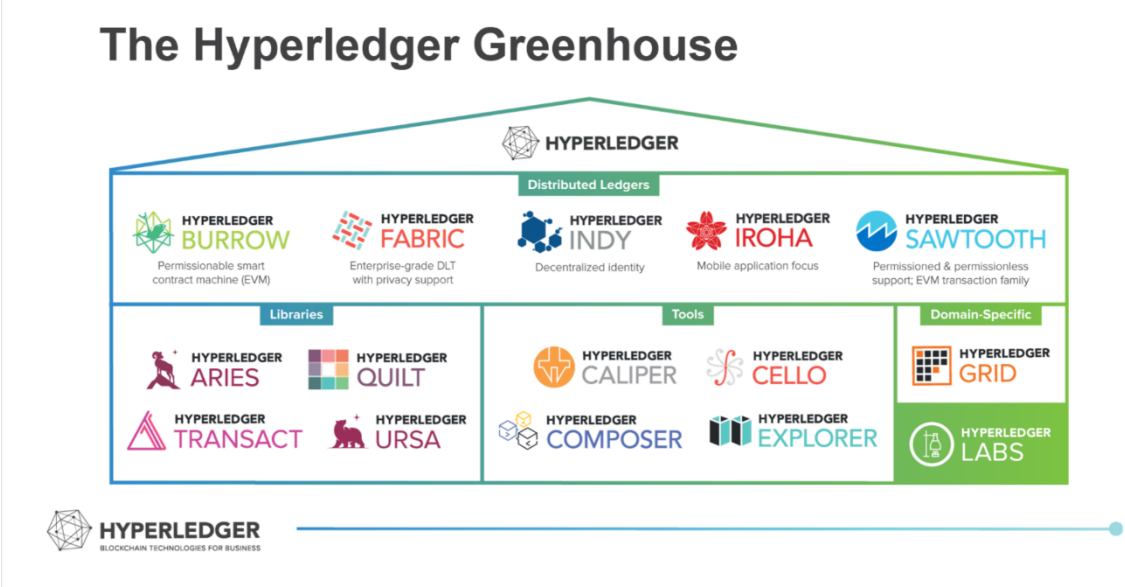

Fig. 8. Tecnologías que forman Hyperledger [21].

A continuación, se describirán brevemente algunos de los frameworks y herramientas proporcionadas por Hyperledger, ahondando más en aquella que finalmente se elegirá para el desarrollo: Hyperledger Fabric.

2.3.3.1 Hyperledger SAWTOOTH

El logo de Hyperledger Sawtooth se muestra en la Fig. 9.

Fig. 9. Logo Hyperledger Sawtooth [22].

Hyperledger Sawtooth es una solución empresarial para construir, implementar libros distribuidos. Es altamente modular y flexiblepara realizar modelos de basados en transacciones actualizados para compartir entre partes donde no exista confianza [5].

2.3.3.2 Hyperledger IROHA

El logo de Hyperledger Iroha se muestra en la Fig. 10.

Fig. 10. Logo Hyperedger Iroha [23].

Hyperledger Iroha es una plataforma para implementar una red blockchain. Está enfocada en modelos empresariales para crear modelos simples. No se requiere de Smart contracts y sirve para que los desarrolladores puedan testar modelos rápidamente [6].

2.3.3.3 Hyperledger INDY

El logo de Hyperledger Indy se muestra en la Fig. 11

Fig. 11. Logo Hyperledger Indy [24].

Hyperledger Indy ofrece herramientas y librerías para proveer identidad digital a través de blockchain [7].

2.3.3.4 Hyperledger BURROW

El logo de Hyperledger Burrow se muestra en la Fig. 12.

Fig. 12. Logo Hyperledger Burrow [25].

Hyperledger Burrows desarrolla una máquina de contrato inteligente permitida basada en Ethereum [8].

2.3.3.5 Hyperledger FABRIC

El logo de Hyperledger Fabric se muestra en la Fig. 13.

Fig. 13. Logo Hyperledger Fabric [26].

Hyperledger Fabric es una tecnología que consigue crear un libro de cuentas permisionado distribuido. Con ello, puedes crear tu propia blockchain sin la necesidad de usar Ethereum u otras blockchains existentes. Las grandes ventajas que presenta son [9]:

-Es modular. Esto quiere decir que partiendo de un modelo general, puedes customizar distintas acciones dentro de la red (el protocolo de consenso, el manejo del servicio de base de datos, el servicio de ordenamiento…).

-Los Smart contracts pueden ser escritos en distintos lenguajes de programación ya conocidos (Go, Java,JavaScript…), aunque el lenguaje “natural” sobre el que está desarrollado esta tecnología es Go.

-Es una red permisionada, no todo el mundo puede correr un nodo y participar, sino que debe tener permisos.

2.3.3.6 Hyperledger QUILT

El logo de Hyperledger Quilt se muestra en la Fig. 14.

Fig. 14. Logo Hyperledger Quilt [27].

Hyperledger Quil ofrece interoperabilidad entre frameworks de registros mediante la ejecución de ILP, que es básicamente una convección de pagos parciales [10].

2.3.3.7 Hyperledger EXPLORER

El logo de Hyperledger Explorer se muestra en la Fig. 15.

Fig. 15. Logo Hyperledger Explorer [28].

Hyèrledger Explorer es una herramienta de análisis. Te permite ver ,consultar, transmitir y desplegar transacciones [10].

2.3.3.8 Hyperledger CELLO

El logo de Hyperledger Cello se muestra en la Fig. 16.

Fig. 16. Logo Hyperledger Cello [29].

Hyperledger Cello es un suministro de blockchain y sistema de operaciones, ayudando a un desarrollo de la red más rápido y eficiente. Las ventajas que ofrece son:

-Manejo del ciclo de vida de una red blockchain.

-Da soporte para crear una red blockchain personalizada (tamaño de la red, tipo de consenso…)

-Avanzadas característica como el monitoreo, loggeo y análisis de capacidad.

El gran aporte de esta herramienta es que permite levantar la red blockchain, implementar contratos inteligentes, testear la red y comprobar el estado de la red desde un dashboard [11].

2.3.3.9 Hyperledger COMPOSER

El logo de Hyperledger Composer se muestra en la Fig. 17.

Fig. 17.Logo Hyperledger Composer [30].

Hyperledger Composer es una herramienta que te ayuda a crear redes blockchainde una forma más fácil. A la hora de crear una blockchain, tienes que pensar quién la va a usar y qué permisos tiene cada participante. Utilizaremos el ejemplo ofrecido por la fuente [12] para explicar unan serie de términos. Se sitúa en un ejemplo de un restaurante de comida rápida. Existirían los siguientes conceptos:

-Pariticpantes: Quién puede poseer algo o realizar una acción. En nuestro ejemplo, el dueño, un cocinero o un cliente.

-Activo: El token de algo con valor. Una hamburguesa o dinero.

- Transacción. Cuando uno de los participantes transacciona activos. Un cliente le da dinero al vendedor, quién le ofrece la hamburguesa.

-Evento: Cuando hay un cambio en el estado de un activo. Por ejemplo, la creación de una hamburguesa.

-Carpeta CTO: La extensión del binario con el que declaramos el modelo de participantes, activos, transacciones y eventos.

-Funciones de transacción: Código que define la lógica de la transacción.

-Carpeta ACL: La extensión del binario donde definimos reglas sobre qué puede hacer cada participante o a dónde puede acceder.

-Registro de activos: El lugar donde se encuentran los datos. Por ejemplo, cuántas hamburguesas hay en un momento dado.

-GetFactory(): Se llama a esta función cuando se quiera crear un nuevo elemento. (activo, evento, participante o transacción).

-Namespace: La forma en la que organizas las cosas en grupos. Por ejemplo, nuestro restaurante se puede llamar "casaPaco", asique todos los participantes o activos que pertenezcan a ese restaurante, tienen ese namespace (la blockchain quizás podría estar formada por varios participantes de comida rápida). Incluso podríamos hacerlo más específico, los cocineros podrían pertenecer al namespace "casaPaco.cocineros".

-Queries: Instrucciones similares a las de SQL para ver cuantos activos un participante tiene. Pr ejemplo, para ver cuantas hamburguesas un restaurante tiene. Se escriben en una carpeta separada con extensión qry. Se usa principalmente para que las APIs puedan consultar datos de la cadena de forma más rápida.

-Business Network Definition: Todas tus carpetas se junta en una carpeta Business Netowrk Definition (.bna), que representa tu blockchain. En la Fig. 18 se aprecia el uso que tiene esta tecnología.

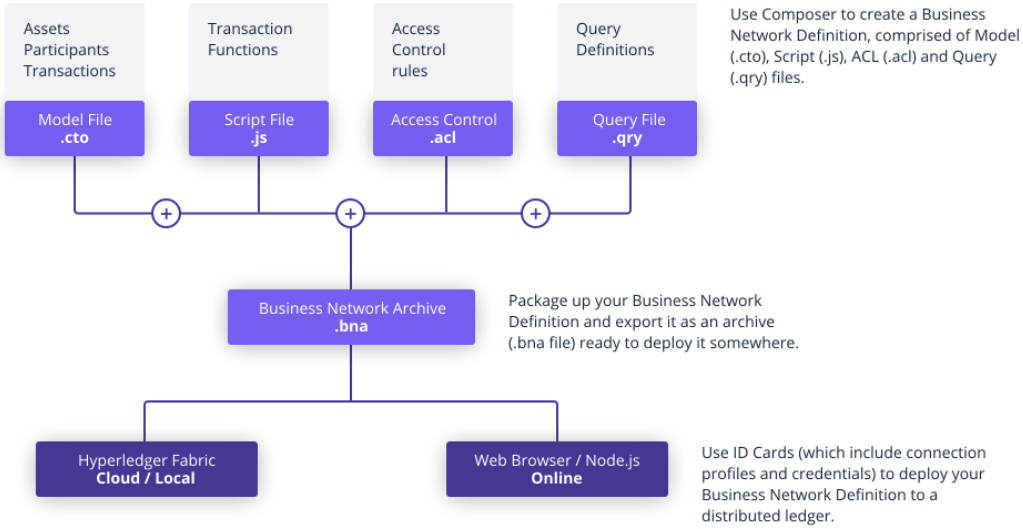

Fig. 18. Interacción de Composer con Fabric y Node [31].

Tabla 2. Comparativa de los distintos Frameworks de Hyperledger [32].

Título	Descripción	Estado
HYPERLEDGER BURROW	Es una plataforma modular de blockchain con integración de contrato inteligente autorizada. Se realizó a lo largo de la especificación de la Máquina Virtual Ethereum (EVM).	Incubación
HYPERLEDGER FABRIC	El tejido es una solución de registro distribuido con diseño modular que permite alos desarrolladores crear una aplicación de alta calidad para cualquier propósito.	Activo
HYPERLEDGER INDY	Indy es una plataforma de registro distribuido que ofrece una variedad de bibliotecas, herramientas y componentes reutilizazbles para ayudar a construir un sistema descentralziado basados en la identidad.	Incubación
HYPERLEDGER IROHA	Este es un marco de blockchai para una fácil integración de blockchain en arquitecturas empresariales.	Activo
HYPERLEDGER SAWTOOTH	Sawtooth es un traje de blockchain para correr, desplegar y construir registros distribuidos. Ofrece un nuevo protocolo de consenso. Prueba de tiempo transcurrido.	Activo

2.3.4 Hyperledger Fabric en profundidad

Es el framework más popular de Hyperledger. Al pertenecer a Hyperledger y presentar un arquitectura modular, permite construir redes DLT (Distributed Ledger Technology) sobre una arquitectura establecida. Sus contratos inteligentes, o el nombre por el que rigen en esta plataforma, chaincodes, presentan la ventaja de poder ser escritos en Golang o JavaScript, sin la necesidad de aprender un nuevo lenguaje de programación como sucede en Ethereum con Solidity.

Componentes básicos de Hyperledger Fabric [9]:

Activos: Permite el intercambio de cualquier cosa que tenga valor y estado.

Chaincodes: Ejecución de órdenes a través de scripts, permitiendo eliminar la confianza a través del protocolo de consenso.

Libro mayor: Guarda el histórico de transacciones de forma inmutable para cada canal, lo que permite una auditoría eficiente y resolución de disputas. Un canal es una minired donde se puede realizar transacciones privadas entre 2 o más nodos de la red.

Privacidad: Se consigue mediante la aplicación de canales y uso de colecciones de datos privados.

Consenso: El consenso de una red blockchain es una parte fundamental pues es lo que permite eliminar la confianza de la red. En este caso particular, al ser una red permisionada, los agentes que la forman son más confiables pues demuestran credenciales para ello. El protocolo aún así es tolerante a fallas bizantinas (Kafka,Raft) y descentralizado (Raft). [13]

Nodos: Son los distintos componentes de la red, que representan a cada una de las organizaciones. Posteriormente se desarrollará más sobre los distintos tipos de nodos.

Servicio de pedidos: Proporciona el consenso y la solicitud de transacción.

SDK: Permite la interacción entre una aplicación cliente y la red blockchain. Permite al cliente consultar el estado de la blockchain y conectarse a la misma.

Autoridad de certificación: Sirve para identificar a cada nodo de la red. Sirven para realizar firmas criptográficas y determina qué acciones puede hacer cada nodo.

Roles dentro de la red

Tenemos tres tipos de nodos en la red:

El comitting peer: Mantiene el libro mayor actualizado, confirma transacciones y puede contener chaincodes.

El endorsing peer: Recibe las propuestas de la aplicación cliente y decide si llevarla a cabo o denegarla. Necesita contener el contrato inteligente para saber que reglas aplicar a la hora de decidir si ejecutar la solicitud o no. Firma la propuesta en caso de ser aceptada.

El ordering node: Aprueba la inclusión de bloques de transacciones en el libro mayor y se comunica constantemente con todos los nodos pares para que la información sea consistente. No contiene contratos inteligentes ni libro mayor.

¿Cómo se ineractúa con una red Hyperledger Fabric?

Por lo general, una aplicación cliente debe conocer cuál es el end-point o punto de contacto con la red (estos son los nodos endosadores). Va a tener que conocer una URL, en Hyperledger Fabric siempre será una URL gRPC, utilizando el protoclo de Golang. Es similar al que usa Google, por lo que está más que testado. Cabe destacar en este punto que el core de la arquitectura de Hyperledger Fabric está desarrollada en Golang. El chaincode contiene las reglas para poder actualizar la información en el Ledger. La aplicación usa un SDK provisto por Hyperledger Fabric para poder enviar una propuesta de transacciones al contrato inteligente. A continuación, se ejecuta el protocolo de ordenamiento y de consenso, y una vez confirmada en la red se notifica a la aplicación

de la actualización. El protocolo de ordenamiento es vital a la hora de llevar a cabo transacciones, pues se estarán realizando muchas transacciones al mismo tiempo y es importante saber el orden en el cuál fueron solicitadas.

Más específicamente, se llevan a cabo los siguientes pasos:

1. **El cliente realiza una transacción** (ya sea una propuesta de transacción o una consulta).
   El cliente conecta con los nodos endosadores o confirmadores, enviando una petición para llamar a una función del contrato inteligente para leer/escribir datos. La propuesta es empaquetada por un SDK, que se asegura que la aplicación cliente tiene las credenciales para producir una firma única y envía la propuesta empaquetada (búfer de protocolo sobre gRPC) a los endorser peers. Se puede observar gráficamente lo que ocurre en este paso en la Fig. 19.

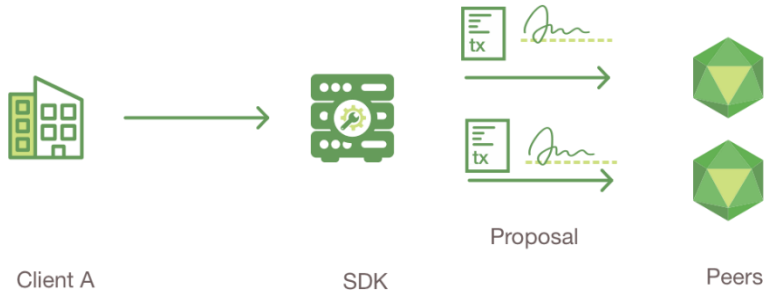

Fig. 19: Paso 1 [33].

2. **Los endorser peer verifican la firma y ejecutan la transacción.**

   Los endorser peer comprueban, según el chaincode, que la transación ha sido ejecutada de forma correcta y que la aplicación cliente tiene permisos para realizar dicha acción (a través de un MSP). Esto está definido en el chaincode (quién puede escribir en qué canales), al igual que las políticas de endorsment, las cuales describen el número de endorser peers deben estar de acuerdo. Todavía no se actualiza el Ledger, sino que se envía al SDK la propuesta de respuesta firmada por los endorser peers. Se puede observar gráficamente lo que ocurre en este paso en la Fig. 20.

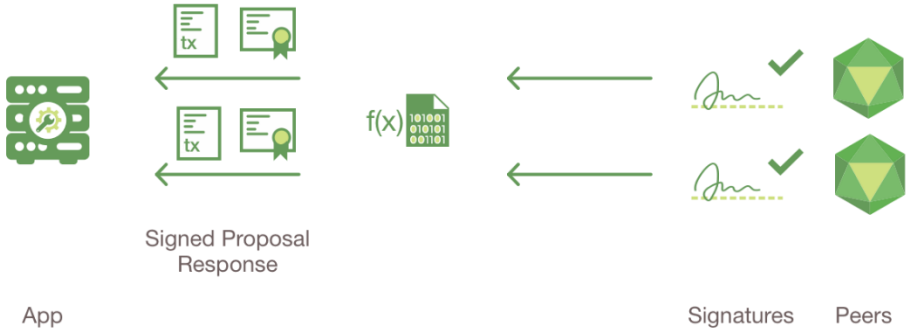

Fig. 20: Paso 2 [33].

3. **Las propuestas de respuesta son inspeccionadas**

La aplicación comprueba que la propuesta de respuesta está debidamente firmada y es correcta. Si se trata de una consulta, no sería necesario enviar la petición al servicio de pedidos, en ca.mbio, si se quiere escribir en el libro mayor, sí se devolvería la petición de respuesta al servicio de pedidos tras haber comprobado que las políticas de endorsamiento son correctas. Por cómo está formada la arquitectura se podría enviar la propuesta de respuesta al servicio de pedidos sin chequearla o incluso enviar una petición no endosada. Los nodos más tarde confirmarán que se cumplen las políticas de endoso. Es importante tener en cuenta, que cuando construimos una red blockchain se debe cambiar la forma de pensar de la programación normal. El desarrollador se encuentra en un entorno donde puede no haber confianza, de modo que toda la lógica debe estar presente que cualquier participante puede actuar malintencionadamente, como sería el caso de enviar una petición no endosada. Esta idea se vuelve particularmente importante a la hora de la programación de contratos inteligentes, se visualizará mejor con el siguiente ejemplo. Un script en un ambiente centralizado y confiado puede recibir una petición del cliente y en muchos casos ni siquiera comprobará que el formato es correcto, simplemente ejecutaría los procesos. Esto es porque se confía en que el cliente enviará los datos correctamente. En un ambiente sin confianza, es de vital importancia realizar ese tipo de comprobaciones, pues el cliente podría enviar información errónea y habría errores en el proceso. Se puede observar gráficamente lo que ocurre en este paso en la Fig. 21.

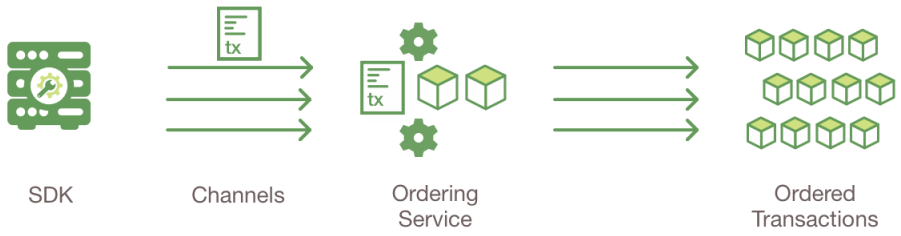

Fig. 21: Paso 3 [33].

4. **El cliente ensambla la confirmación en una transacción**

La aplicación enviaría entonces la propuesta de respuesta y de transacción al servicio de pedidos encapsulada en un “mensaje de transacción”. El servicio de pedidos comprueba las firmas de los pares endosantes, y la identificación del canal, ordena las peticiones cronológicamente según los canales y crea un bloque para cada canal. Se puede observar gráficamente lo que ocurre en este paso en la Fig. 22.

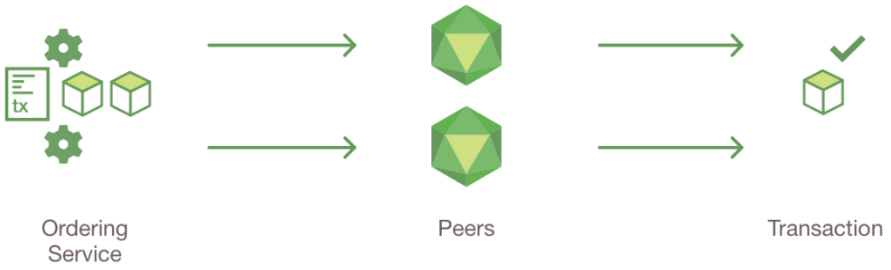

Fig. 22: Paso 4 [33]

5. **La transacción es validada y comprometida**

   Por último, se envían los bloques a cada nodo, éstos comprueban las política y finalmente devuelven una respuesta: válida o no válida. Posteriormente se envía una notificación al cliente con el resultado y, en caso de ser válida, se actualiza el ledger [14]. Se puede observar gráficamente lo que ocurre en este paso en la Fig. 23.

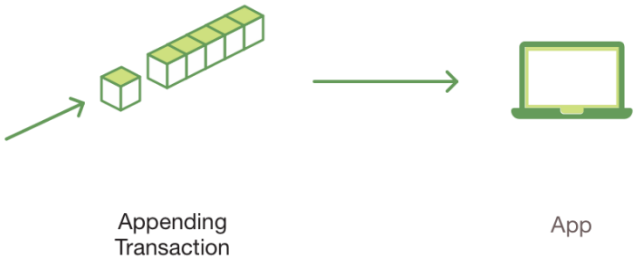

Fig. 23: Paso 5 [33]

Características:

-Red permisionada

-Modular

-Baja latencia de confirmación

-Varias posibilidades para privatizar el dato: canales o colecciones de datos privadas.

-Posibilidad de escribir los chaincodes en varios idiomas

-Modelo de respaldo flexible para lograr el consenso entre las organizaciones requeridas

-Datos consultables (usando una base de datos tipo clave valor con archivos JSON)

2.4 Dockers

El logo de Docker se muestra en Fig. 24.

Fig. 24. Logo de Docker [34].

Docker es una plataforma que trata de del uso de containers para automatizar el despliegue de aplicaciones. Todos los elementos de una aplicación (motor, conjunto de librerías, kernel..) pueden estar agrupados en un mismo contenedor, pudiéndose ejecutar de forma independiente del sistema operativo [15]. Esto permite ejecutar la aplicación donde quieras, eliminando preocupaciones sobre los elementos para iniciar una aplicación, versiones, etc. Esto te libra de tener que descargar y almacenar una gran cantidad de librerías [16].

Capítulos 3. Descripción del modelo desarrollado.

En este capítulo se profundizará acerca de cómo levantar una red blockchain basada en Hyperledger y cómo interactuar con los chaincodes para poder operar la red.

Se explicará posteriormente la arquitectura del funcionamiento de la aplicación desarrollada.

3.1 Objetivos y especificación

Los objetivos de este trabajo son:

-Ahondar acerca de la tecnología blockchain. En el contexto actual se palpa una gran ilusión acerca de esta tecnología, reflejada en los mercados. No únicamente con Bitcoin sino con muchos otros proyectos cuya motivación no es conseguir una moneda transaccionable sino formar parte de alguna forma de lo que se denomina la web3, la nueva revolución, cuyo eje principal es blockchain.

- Comprobar su aplicabilidad para la trazabilidad de activos a través de la cadena de suministro, proporcionando ventajas tanto al consumidor final como a la propia empresa, mejorando sus procesos.

-Desarrollar una red blockchain basada en Hyperledger con la que sea posible transaccionar un activo y poder observar su trazabilidad, conociendo en todo momento quién es el poseedor actual del mismo. Conectarse a esta red a través de un SDK de node proporcionado por Hyperledger. La aplicación cliente interactuará con el SDK a través de una api desarrollada en ExpressJS.

3.2 Datos

A continuación, se mostrará como se diseñan los procesos y su estructura de datos relacionada en una red de Hyperledger. En nuestra red hay cuatro organizaciones que usan la red MyChannel para crear (addAsset), consultar (QueryAsset) o transferir (TransferAsset) activos. En la Fig. 25 se muestra gráficamente quién compone esta red.

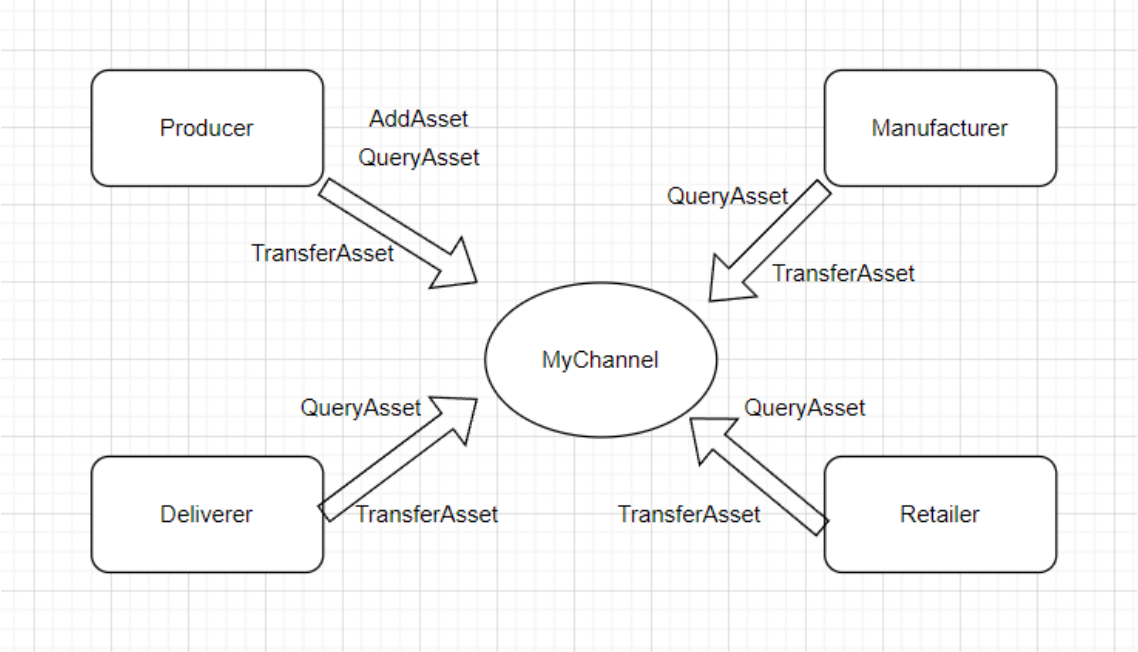

Fig. 25 Actores principales de la red MyChannel [17].

3.2.1 Ciclo de vida

Hay dos conceptos importantes cuando hablamos de este tipo de redes: el estado y las transacciones. Comprender correctamente estos dos conceptos es de vital importancia. Se puede representar el ciclo de vida de un activo con el siguiente diagrama de estado-transacciones en la Fig. 26:

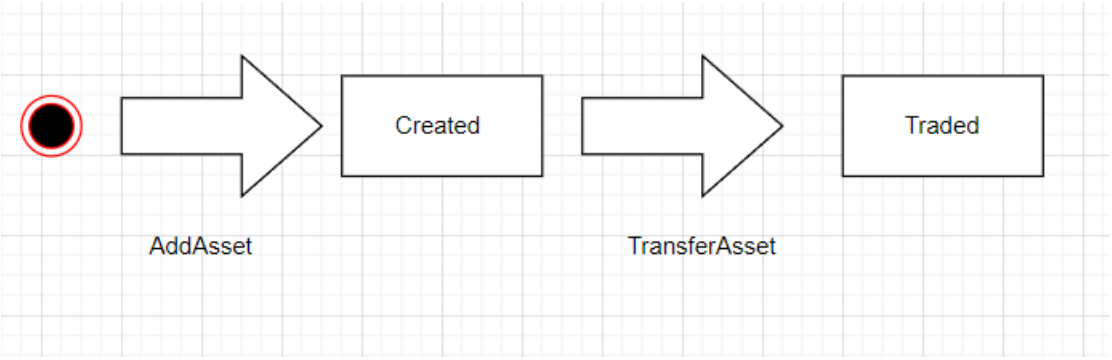

Fig. 26. Diagrama de estado-transacciones de un activo [17].

Este diagrama muestra como un activo puede pasar por diferentes estados: creado y transferido. Los diferentes estados por los que pasa un activo se quedan guardados en el libro mayor.

3.2.2 Estado de libro mayor

Un activo se puede ver en un determinado momento como se muestra en la Fig. 27.

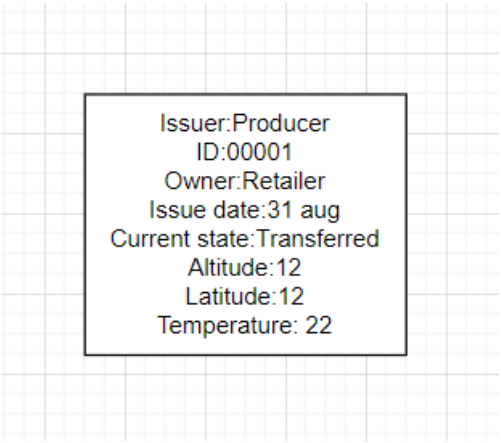

Fig. 27. Cómo se ve un activo con unos atributos en una red de Hyperledger [17].

Un activo se define con un conjunto de propiedades, con un valor. Normalmente, una combinación de valores para estas propiedades hace que cada activo tenga una clave única.

La propiedad más importante de estos activos es la de current state, que nos dice si el activo está creado o transferido. El estado actual de un activo se implementa en una base de datos de tipo clave-valor que permite hacer consultas avanzadas, el valor de las propiedades se suele guardar en un formato de tipo JSON.

A continuación, en la Fig. 28, se muestra un ejemplo de cómo un activo va cambiando el valor de su estado.

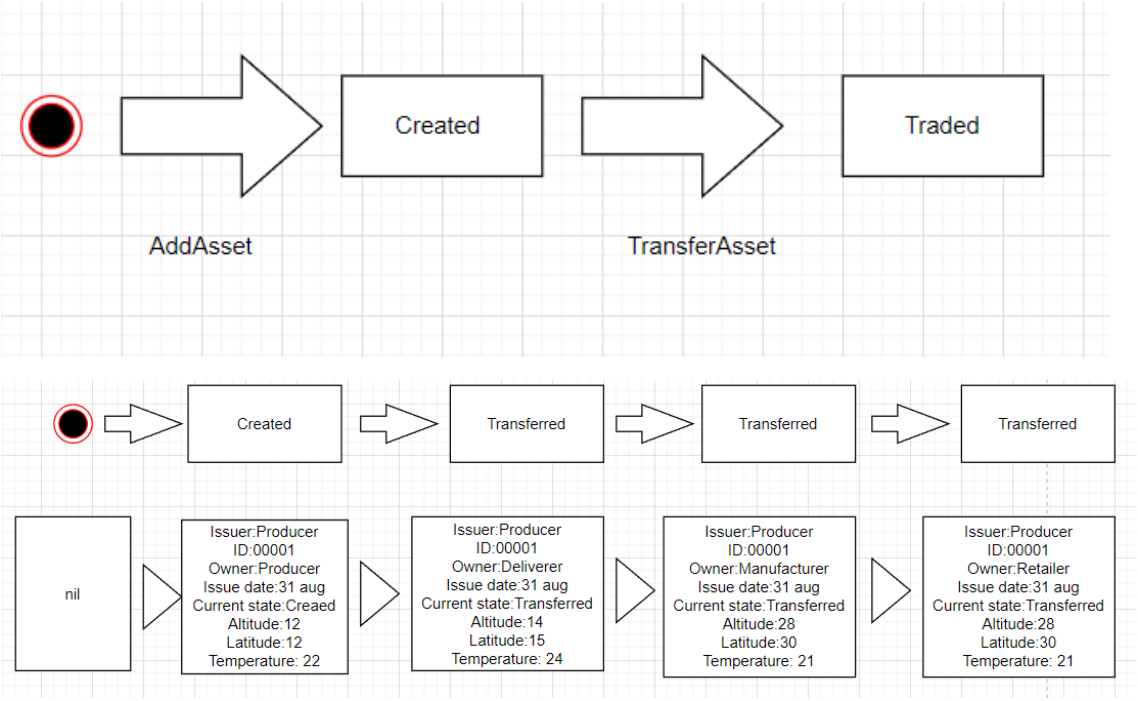

Fig. 28. Diferentes estados por los que pasa el activo 00001 [17].

Al principio, el activo papel no tiene un valor para cada una de las propiedades (nil), posteriormente se crea el papel con la transacción addAsset y luego va actualizando su resultado con transferAsset. Fabric, además del valor, permite actualizar la estructura de un activo, esto es, añadir o eliminar propiedades [17].

3.2.3 Múltiples estados

Los estados de un libro mayor deben ser vistos más como un vector que como un escalar. Como hemos visto, el valor de las propiedades de los activos va variando a través de las transacciones que se van cometiendo. Es una base de datos de tipo clave-valor donde cada propiedad tiene un valor que se va archivando normalmente en un fichero tipo JSON. Esta base de datos permite consultas avanzadas, como se ha mencionado anteriormente, y para ello se agrupan los activos en una lista lógica. En el caso representado, las transacciones se guardan en un contenedor que se va actualizando cada vez que realiza una transacción de papeles. En el ejemplo de la documentación oficial lo representan de la forma en la que se muestra en la Fig. 29.

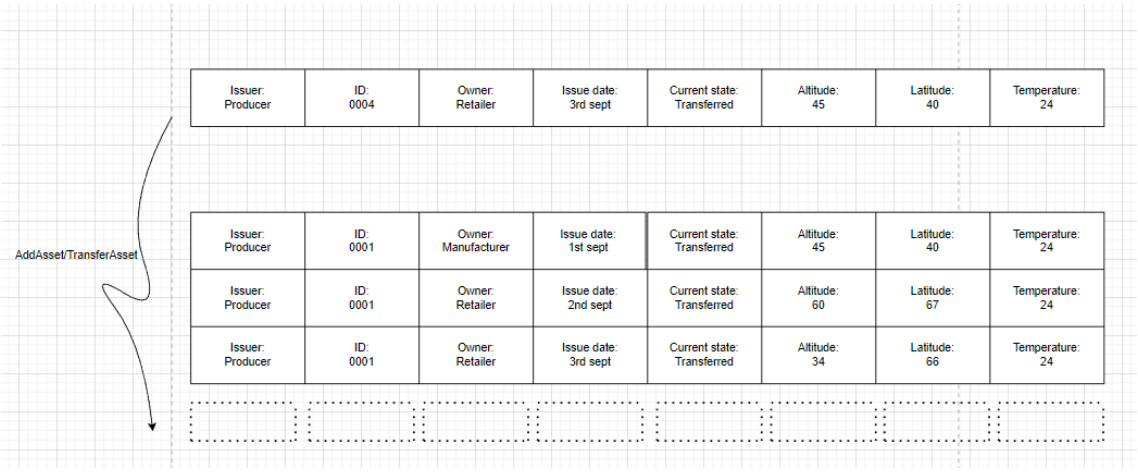

Fig. 29. Vista de una lista lógica que permite consultas avanzadas en la blockchain [17].

Cuando la el Producer realiza la orden addAsset, o cualquiera de los participantes transfieren el asset a otra persona, se añaden dichas operaciones al ledger.

Sin embargo, aunque esté muy bien pensar en una única lista como una lista lógica, es mejor verlo como un único estado para cada activo, de forma que con una clave puedas consultar su estado siendo cada estado única y de fácil consulta.

3.3 Algoritmo

El proceso que realizaremos en Hyperledger consistirá en iniciar un libro mayor con activos predeterminados, consultar esos activos, crear nuevos y consultar un activo en concreto usando su identificador. Para explicarlo, utilizaremos el ejemplo de la documentación oficial [17].

La aplicación cliente interactúa con la red blockchain a través del uso de un SDK proporcionado por Fabric. Para ello, la aplicación debe invocar al Smart contract que le permita interactuar con la red. Esto es lo que ocurre cuando una aplicación cliente quiere crear un activo, mostrado en la Fig. 30.

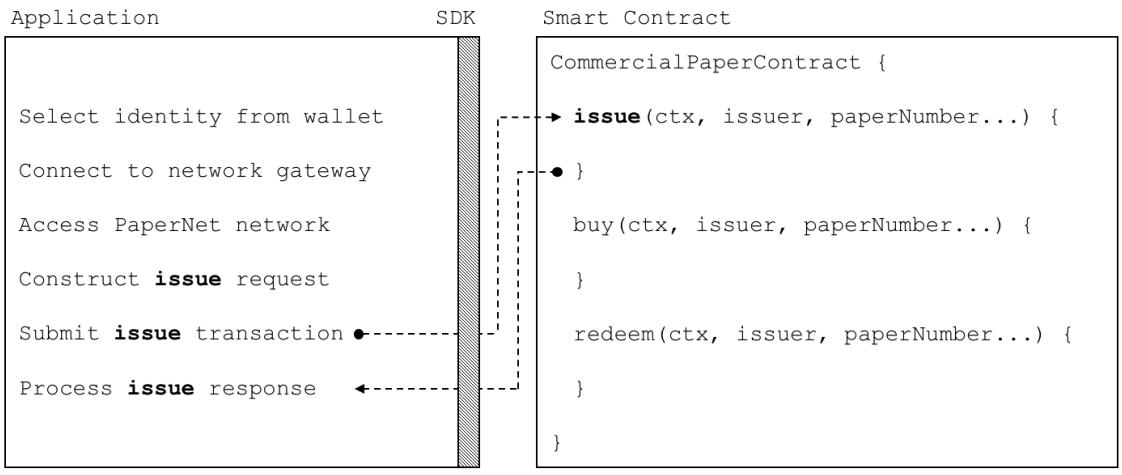

Fig. 30. Conexión cliente-red [17].

Una aplicación tiene que realizar seis pasos para realizar una transacción:

Seleccionar una identidad de un wallet.

Conectarse a un Gateway.

Acceder a la red que se requiera.

Construir una petición de transacción para el chaincode.

Presentar la solicitud a la red.

Procesar una respuesta.

3.3.1 Selección de identidad de una wallet.

Con la siguiente línea de código, se declaran las dos clases de Fabric:

```
const { Wallets, Gateway } = require('fabric-network');
```

La clase "Wallet" se usa como se muestra en esta línea:

```
const wallet = await
Wallets.newFileSystemWallet('../identity/user/producer/wallet');
```

"wallet" crea una carpeta local, que contiene la identidad del usuario Producer. Contiene un conjunto de identidades (certificados x.509) con las que se puede unir a una red de Fabric. Dependiendo del contenido que hay en el certificado x.509, el usuario podrá acceder a la red o no, o tener un rol u otro. Es importante resaltar que las wallets no contienen dinero o tokens, sino identidades.

3.3.2 Conectarse al Gateway.

La segunda clase clave de Fabric es "Gateway". Esta clase se encarga de identificar peers que den acceso a una determinada red. Con esta línea nos conectamos al Gateway:

```
await gateway.connect(connectionProfile, connectionOptions);
```

Tiene dos parametros:

-connectionProfile, dice donde se encuentra la carpeta que identifica el conjunto de peers que dan acceso a la red.

-connectionOptions, es un conjunto de opciones con las que el cliente puede interactuar con la red.

El cliente se abstrae de cómo sea la topología de la red gracias al gateway, incluso si ésta cambia. El gateway se encarga de enviar la transacción a los nodos apropiados, que tienen el rol de endorsers.

connectionProfile es de tipo YAML, lo convertimos a un objeto JSON con la siguiente línea:

```
let connectionProfile =
yaml.safeLoad(file.readFileSync('./gateway/connectionProfile.yaml', 'utf8'));
```

Esto contiene mucha información, pero nos centramos en lo relativo al canal:

```
channels:
  papernet:
    peers:
      peer1.producer.com:
        endorsingPeer: true
        eventSource: true

      peer2.manufacturer.com:
        endorsingPeer: true
        eventSource: true
```

En canales, identificamos la red firstStep y dos peers, uno perteneciente a Producer y otro perteneciente a Manufacturer, ambos teniendo el rol de endorsers.

En el objeto connectionOptions:

```
let connectionOptions = {
    identity: userName,
    wallet: wallet,
    discovery: { enabled:true, asLocalhost: true }
};
```

Se observa que se especifica la identidad y el wallet que se deberían usar para conectarse a la red. Hay otras opciones adicionales como:

```
let connectionOptions = {
  identity: userName,
  wallet: wallet,
  eventHandlerOptions: {
    commitTimeout: 100,
    strategy: EventStrategies.MSPID_SCOPE_ANYFORTX
  },
}
```

Que indica al SDK que espere cien segundos para ver si se ha llevado a cabo una transacción.

`strategy: EventStrategies.MSPID_SCOPE_ANYFORTX` indica que se avise a la aplicación cuando uno de los nodos que endorsan, confirmen la transacción. Se podría haber usado la estrategia `strategy: EventStrategies.NETWORK_SCOPE_ALLFORTX` para notificar a la aplicación solo cuando todos los nodos hubieran confirmado la transacción.

3.3.3 Acceder a la red deseada

Los nodos definidos en el gateway connectionProfile.yaml tienen acceso a la red mychannel. Pero estos nodos podrían unirse a otros canales, asique el gateway le daría a la aplicación acceso a varios canales. Así es cómo la aplicación selecciona un canal determinado:

```
const network = await gateway.getNetwork('mychannel');
```

Si se quisiera usar otra red al mismo tiempo, por ejemplo la red fisrtStep, se podría hacer con

```
const network2 = await gateway.getNetwork('firstStep');
```

Así, el producer puede pertenecer al canal mychannel y al canal fisrtStep, proporcionando privacidad entre el Producer y el Manufacturer. Esto es importante ya que es posible que la información compartida entre estos dos agentes sea confidencial y potencialmente peligrosa si es conocida por el retailer, quién puede pensar que compra a un precio mucho superior que al precio que el Producer vende al Manufacturer. Por ello, se crea ese canal de privacidad pero todos los participantes pertenecen al canal general donde pueden ver cómo se va moviendo el activo.

3.3.4 Construir una petición de transacción para el Smart contract.

Para realizar una transacción, se llama al SDK con:

```
const issueResponse = await contract.submitTransaction('addAsset', 'Producer', '0001', 'Producer','2022-08-31', 'created', '50','45', '32');
```

Los argumentos pasados a esta función coinciden con los de petición de transacción. Son los que se le pasarán a la función issue() del Smart contract, que se muestra aquí:

```
async issue(ctx, issuer, ID,owner, issueDate, currentState, altitude, latitude,temperature )
```

Sin embargo, con esta función no nos comunicamos directamente con el Smart contract, lo que pasa realmente es que el SDK usa connectionOptions y connectionProfile para enviar una petición de transacción a los nodos endorsers, que dan permiso si muestra los credenciales apropiados. Sin embargo el cliente no se tiene que preocupar de nada de esto, lo hace el SDK por detrás. La API submitTransacction incluye un proceso por el que el cliente se mantiene escuchando a las transacciones realizadas para ver si la petición ha sido validada y realizada.

3.3.5 Procesar la respuesta de la red

El Smart contract de la red devuelve un valor tal que así:

```
return asset.toBuffer();
```

El activo creado necesita devolverse a la aplicación cliente en forma de buffer. La aplicación cliente tiene un método `Asset.fromBuffer()` que permite usar el activo creado de una forma natural:

```
console.log(`${asset.issuer} asset : ${asset.assetNumber} successfully issued for value ${asset.faceValue}`);
```

Igual que antes, puede parecer que el Smart contract se comunica con la aplicación cliente directamente, pero no es así. El SDK se encarga de todo el protocolo de consenso dependiendo de las opciones de estrategia que se hubiesen elegido.

3.3.6 Representación de la arquitectura de la red

En resumen, el usuario se conectaría con su wallet aportando sus claves criptográficas a la red a través de una API desarrollada en Express que facilita la interacción con el SDK de nodeJS de Fabric y con el que puede acceder a la red para interactuar con ella.

Esta es una representación gráfica del proceso:

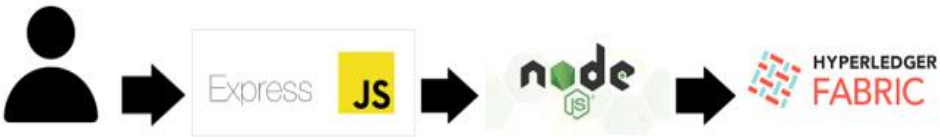

Fig. 2: Arquitectura de la aplicación

4. Análisis de resultados

Se usará una aplicación ofrecida en la documentación oficial de la página de Hyperledger Fabric [18] para representar las operaciones que se han de llevar a cabo en una cadena de suministros digitalizada.

Estas operaciones son las mismas que realizarían los participantes de nuestra red y cuyos contratos inteligentes son muy similares, pues es la creación y traspaso de activos, además de la consulta de los mismos.

4.1 Creación del canal

En primer lugar, se apaga cualquier posible red levantada en el entorno local de pruebas para no tener problemas en el entorno mediante el comando:

```
./network.sh down
```

Esto es lo que se muestra por pantalla:

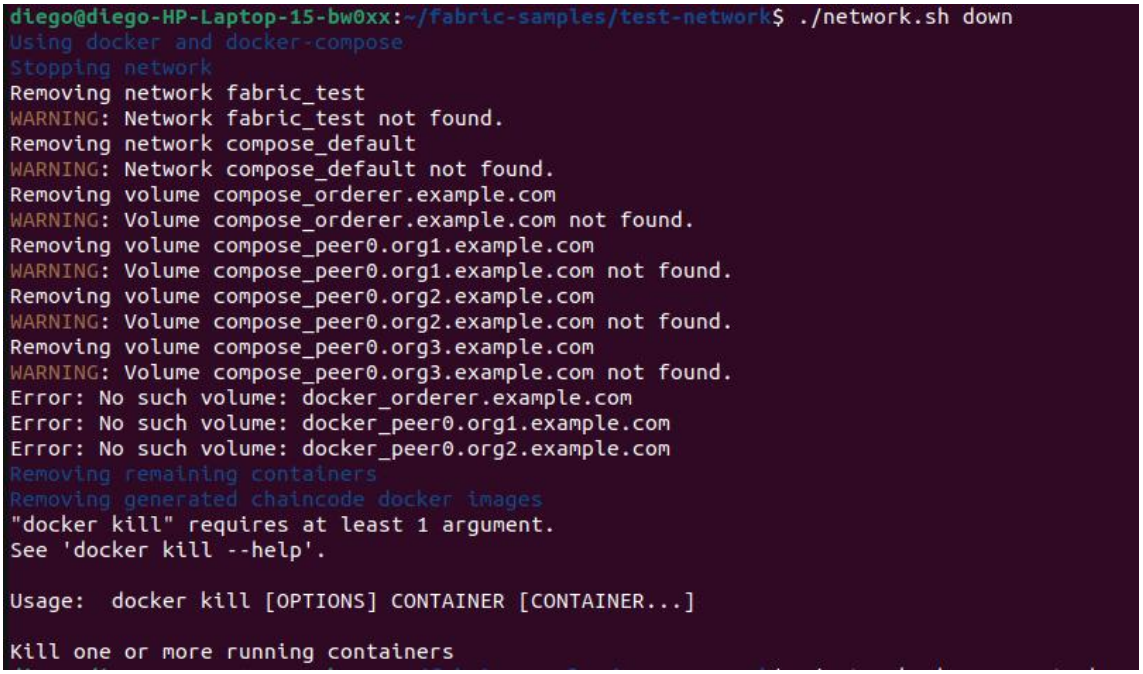

Fig. 31. Captura a la respuesta de apagar la red.

Una vez estando la red en funcionamiento, se añadirá a la misma dos peers de cada participante, un servicio de ordenamiento y certificados de autoridad (CA).Estos, junto con los Smart contracts, quedan almacenados en contenedores Dockers independientes. Los dos peers hacen referencia a dos nodos pertenecientes a cada uno de los participantes en la cadena de suministros del aceite de oliva (se usan solo dos por simplificar, pero puede haber tantos nodos o participantes como se deseen). En nuestro caso, el primer participante es el agricultor, con el nombre de Producer (de nuevo, en este caso hay un único agricultor, pero se pueden añadir tantos como se deseen, lo cuál es muy beneficioso para una comunidad de agricultores) el segundo participantes es el transportista, con el nombre de deliverer. El tercero, una cooperativa donde se machaca y prepara la oliva para su venta y el cuarto un gran supermercado al que la cooperativa vende el aceite para que se comercializado. Hay más intermediarios en el camino, como pueden ser los mismos sensores de localización geográfica o temperatura que indican el estado de las olivas a lo largo del año, pero a efectos técnicos para el desarrollo de la red se considerarán solo dos. La propiedad que tienen en esta red relacionada con la lo'calización y temperatura es un parámetro ofrecido por el participante sin la verificación de un sensor conectado a la red. Escalarlo a más peers es sencillo modificando parámetros de configuración, pero lleva asociado un gasto extra de computación. Para añadir a los agentes a la red y otorgarles sus certificados, se corre el siguiente comando:

```
./network.sh up createChannel -c mychannel -ca
```

Nótese que en el propio comando se elige la opción -ca, lo cual quiere decir que se está usando un procedimiento de identificación en la red mediante certificados de autoridad. Otra opción para levantar la red podría ser generando material criptográfico usando la herramienta cryptogen.

A continuación, se muestran fragmentos (no todo) de lo que se devuelve por pantalla. Se muestra para un único participantes y para el otro es similar.

```
diego@diego-HP-Laptop-15-bw0xx:~/fabric-samples/test-network$ ./network.sh up createChannel -c mychannel -ca
Using docker and docker-compose
Creating channel 'mychannel'.
If network is not up, starting nodes with CLI timeout of '5' tries and CLI delay of '3' seconds and using database 'leveldb with crypto from '
Certificate Authorities'
Bringing up network
LOCAL_VERSION=2.4.4
DOCKER_IMAGE_VERSION=2.4.4
CA_LOCAL_VERSION=1.5.4
CA_DOCKER_IMAGE_VERSION=1.5.4
Generating certificates using Fabric CA
Creating network "fabric_test" with the default driver
Creating ca_org2    ... done
Creating ca_org1    ... done
Creating ca_orderer ... done
Creating Org1 Identities
Enrolling the CA admin
+ fabric-ca-client enroll -u https://admin:adminpw@localhost:7054 --caname ca-org1 --tls.certfiles /home/diego/fabric-samples/test-network/org
anizations/fabric-ca/org1/ca-cert.pem
2022/08/03 23:32:04 [INFO] Created a default configuration file at /home/diego/fabric-samples/test-network/organizations/peerOrganizations/org
1.example.com/fabric-ca-client-config.yaml
2022/08/03 23:32:04 [INFO] TLS Enabled
2022/08/03 23:32:04 [INFO] generating key: &{A:ecdsa S:256}
2022/08/03 23:32:04 [INFO] encoded CSR
2022/08/03 23:32:04 [INFO] Stored client certificate at /home/diego/fabric-samples/test-network/organizations/peerOrganizations/org1.example.c
om/msp/signcerts/cert.pem
2022/08/03 23:32:04 [INFO] Stored root CA certificate at /home/diego/fabric-samples/test-network/organizations/peerOrganizations/org1.example.
com/msp/cacerts/localhost-7054-ca-org1.pem
2022/08/03 23:32:04 [INFO] Stored Issuer public key at /home/diego/fabric-samples/test-network/organizations/peerOrganizations/org1.example.co
m/msp/IssuerPublicKey
2022/08/03 23:32:04 [INFO] Stored Issuer revocation public key at /home/diego/fabric-samples/test-network/organizations/peerOrganizations/org1
.example.com/msp/IssuerRevocationPublicKey
```

Fig. 32. Captura de respuesta por pantalla al crear un canal: generación de certificados de autoridad y creación de la primera organización con su admin correspondiente.

```
Registering peer0
+ fabric-ca-client register --caname ca-org1 --id.name peer0 --id.secret peer0pw --id.type peer --tls.certfiles /home/diego/fabric-samples/tes
t-network/organizations/fabric-ca/org1/ca-cert.pem
2022/08/03 23:32:04 [INFO] Configuration file location: /home/diego/fabric-samples/test-network/organizations/peerOrganizations/org1.example.c
om/fabric-ca-client-config.yaml
2022/08/03 23:32:04 [INFO] TLS Enabled
2022/08/03 23:32:04 [INFO] TLS Enabled
Password: peer0pw
Registering user
+ fabric-ca-client register --caname ca-org1 --id.name user1 --id.secret user1pw --id.type client --tls.certfiles /home/diego/fabric-samples/t
est-network/organizations/fabric-ca/org1/ca-cert.pem
2022/08/03 23:32:04 [INFO] Configuration file location: /home/diego/fabric-samples/test-network/organizations/peerOrganizations/org1.example.c
om/fabric-ca-client-config.yaml
2022/08/03 23:32:04 [INFO] TLS Enabled
2022/08/03 23:32:04 [INFO] TLS Enabled
Password: user1pw
Registering the org admin
+ fabric-ca-client register --caname ca-org1 --id.name org1admin --id.secret org1adminpw --id.type admin --tls.certfiles /home/diego/fabric-sa
mples/test-network/organizations/fabric-ca/org1/ca-cert.pem
2022/08/03 23:32:04 [INFO] Configuration file location: /home/diego/fabric-samples/test-network/organizations/peerOrganizations/org1.example.c
om/fabric-ca-client-config.yaml
2022/08/03 23:32:04 [INFO] TLS Enabled
2022/08/03 23:32:04 [INFO] TLS Enabled
Password: org1adminpw
```

Fig. 33. Captura de respuesta por pantalla al crear un canal: unión de los peers de la primera organización y el administrador en el canal.

```
Generating the peer0 msp
+ fabric-ca-client enroll -u https://peer0:peer0pw@localhost:7054 --caname ca-org1 -M /home/diego/fabric-samples/test-network/organizations/
erOrganizations/org1.example.com/peers/peer0.org1.example.com/msp --csr.hosts peer0.org1.example.com --tls.certfiles /home/diego/fabric-samp
s/test-network/organizations/fabric-ca/org1/ca-cert.pem
2022/08/03 23:32:04 [INFO] TLS Enabled
2022/08/03 23:32:04 [INFO] generating key: &{A:ecdsa S:256}
2022/08/03 23:32:04 [INFO] encoded CSR
2022/08/03 23:32:05 [INFO] Stored client certificate at /home/diego/fabric-samples/test-network/organizations/peerOrganizations/org1.example
om/peers/peer0.org1.example.com/msp/signcerts/cert.pem
2022/08/03 23:32:05 [INFO] Stored root CA certificate at /home/diego/fabric-samples/test-network/organizations/peerOrganizations/org1.exampl
com/peers/peer0.org1.example.com/msp/cacerts/localhost-7054-ca-org1.pem
2022/08/03 23:32:05 [INFO] Stored Issuer public key at /home/diego/fabric-samples/test-network/organizations/peerOrganizations/org1.example.
m/peers/peer0.org1.example.com/msp/IssuerPublicKey
2022/08/03 23:32:05 [INFO] Stored Issuer revocation public key at /home/diego/fabric-samples/test-network/organizations/peerOrganizations/or
.example.com/peers/peer0.org1.example.com/msp/IssuerRevocationPublicKey
Generating the peer0-tls certificates
+ fabric-ca-client enroll -u https://peer0:peer0pw@localhost:7054 --caname ca-org1 -M /home/diego/fabric-samples/test-network/organizations/
erOrganizations/org1.example.com/peers/peer0.org1.example.com/tls --enrollment.profile tls --csr.hosts peer0.org1.example.com --csr.hosts lo
lhost --tls.certfiles /home/diego/fabric-samples/test-network/organizations/fabric-ca/org1/ca-cert.pem
2022/08/03 23:32:05 [INFO] TLS Enabled
2022/08/03 23:32:05 [INFO] generating key: &{A:ecdsa S:256}
2022/08/03 23:32:05 [INFO] encoded CSR
2022/08/03 23:32:05 [INFO] Stored client certificate at /home/diego/fabric-samples/test-network/organizations/peerOrganizations/org1.example
om/peers/peer0.org1.example.com/tls/signcerts/cert.pem
2022/08/03 23:32:05 [INFO] Stored TLS root CA certificate at /home/diego/fabric-samples/test-network/organizations/peerOrganizations/org1.ex
ple.com/peers/peer0.org1.example.com/tls/tlscacerts/tls-localhost-7054-ca-org1.pem
2022/08/03 23:32:05 [INFO] Stored Issuer public key at /home/diego/fabric-samples/test-network/organizations/peerOrganizations/org1.example.
m/peers/peer0.org1.example.com/tls/IssuerPublicKey
2022/08/03 23:32:05 [INFO] Stored Issuer revocation public key at /home/diego/fabric-samples/test-network/organizations/peerOrganizations/or
.example.com/peers/peer0.org1.example.com/tls/IssuerRevocationPublicKey
```

Fig. 34. Captura de respuesta por pantalla al crear un canal: generación del MSP (membership provider) y certificados tls para el peer0.

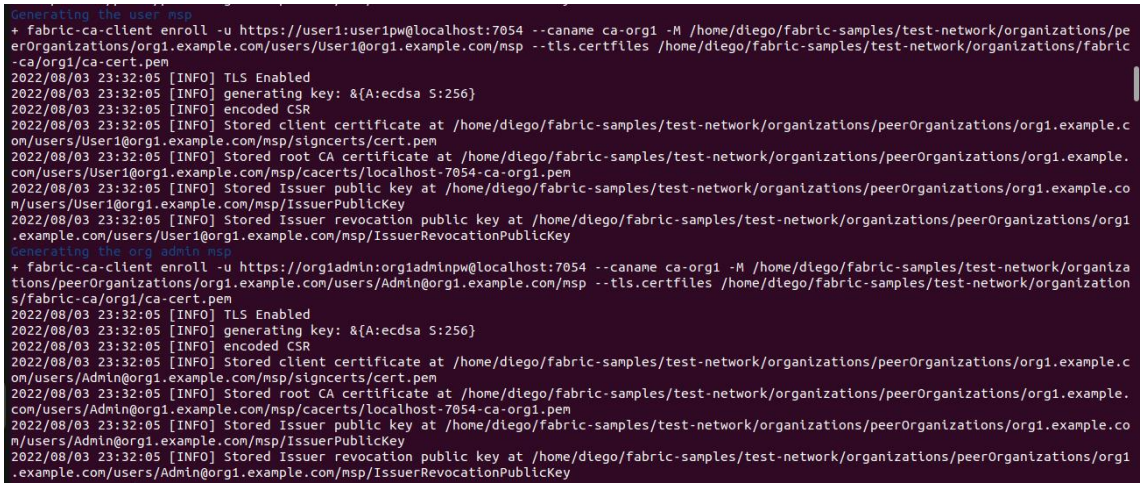

Fig. 35. Captura de respuesta por pantalla al crear un canal: generación del MSP (membership provider) y certificados tls para el peer user.

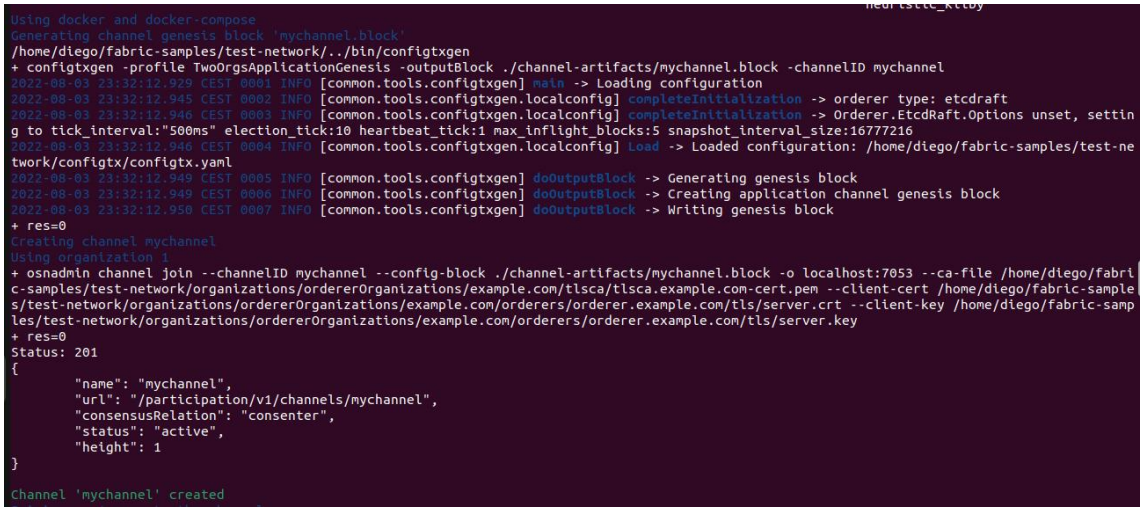

Fig. 36.Captura de respuesta por pantalla al crear un canal: generación del bloque génesis.

Esta captura está repleta de información sobre lo que está sucediendo. En primer lugar, informa de que va a crear el bloque génesis, el primer bloque del libro mayor. El bloque génesis en una red de Hyperledger Fabric no es un bloque al uso, sino un bloque de configuración. Ahí es donde se configuran cosas importantes que permanecerán mientras la red viva, una muy importante es el tipo de consenso que tiene que darse y cómo funcionará el ordenamiento. EL tipo de consenso elegido es tolerante a fallas bizantinas, que es de los más simples. También, si se desease, se puede usar un nuevo protocolo llamado RAFT que asegura en un mayor grado la descentralizaciñon. Este protocolo está disponible en la última versión de Fabric exclusivamente. Si un nuevo nodo desea unirse a la red, lo primero que hará será leer este primer bloque y aplicar esas condiciones de configuración. También se configura el “gossip”. Esto es, cuando un nodo se cae por cualquier razón, la red sigue funcionando. Cuando este nodo se reestablezca habrán ocurrido una serie de operaciones en el ledger que él mismo no ha podido registrar. Para ello, se conecta al nodo que se configure para hacer una copia del ledger. La configuración actal, al tener cada participante dos nodos, es que si se cae uno

de ellos, se conecte al otro nodo del mismo participante para hacer la copia. En caso de que caigan los dos nodos de la misma organización, se conectará a los nodos más cercanos en la cadena de suministros. Si se desease, en la configuración también se puede poner que si se caen los dos nodos de uno de los participantes, se cancelasen todas las operaciones hasta que se reestableciese.

Fig. 37. Captura de respuesta por pantalla al crear un canal: Generación de la primera propuesta de transacción.

Aquí se confirma la correcta conexión entre el nodo endorsador y el servicio de ordenamiento, además de indicar que la propuesta de la primera transacción (el bloque génesis) se lleva a cabo de forma correcta.

Fig. 38. Captura de respuesta por pantalla al crear un canal: Estado inicial del libro mayor.

Se muestra cómo se encuentra el libro mayor antes de realizar la primera transacción. Como es lógico antes de la generación del primer bloque de configuración, el libro mayor está vacío, con ningún bloque registrado ni tramitado.

Fig. 39. Captura de respuesta por pantalla al crear un canal: Configuración del anchor peer para la organización 1.

Otra de las cosas que se configura en el bloque génesis, es qué nodo va a ser el anchor peer de cada organización, es decir, el nodo con el que se tiene que comunicar alguien que quiera realizar algún tipo de operación con la organización. Todas esta configuraciones se encuentran en un archivo JSON, que el programa tiene que descodificar para entender.

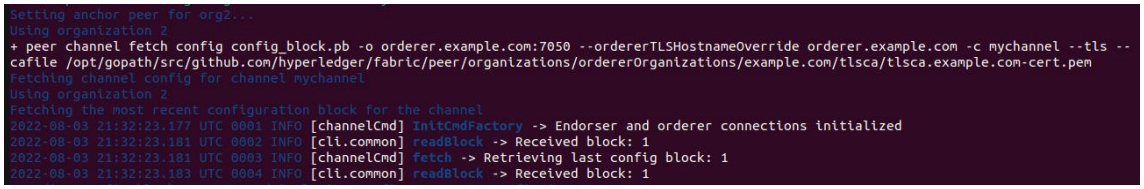

Fig. 40. Captura de respuesta por pantalla al crear un canal: Tras finalizar la primera transacción, se vuelve a mostrar el estado del libro mayor.

Tras realizar correctamente la primera transacción (el bloque génesis), se vuelve a mostrar el estado del libro mayor. Ahora, se puede observar que ha registrado un bloque.

Todas estas mismas respuestas se dan de forma similar pero para la segunda organización y para el nodo servicio de ordenamiento.

Por último, se instala el Smart contract en el canal y en cada uno de los peers de las organizaciones que se requieran con el comando:

```
./network.sh deployCC -ccn basic -ccp ../asset-transfer-basic/chaincode-javascript/ -ccl javascript
```

Como respuesta a dicho comando, aparece la siguiente respuesta por pantalla:

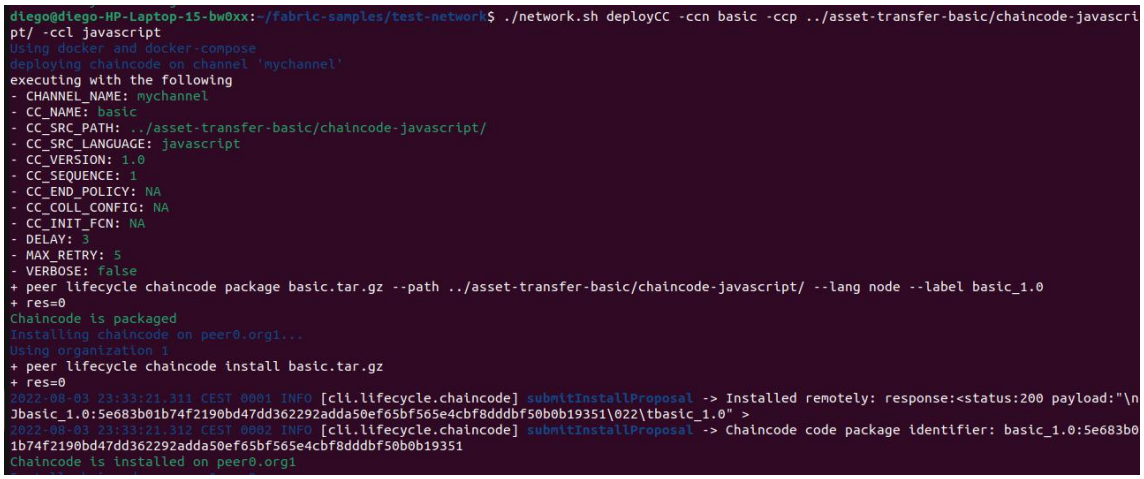

Fig. 41. Captura de respuesta por pantalla al instalar el Smart contract: se empaquetae instala en el peer0 de la primera organización.

Se observa cómo el Smart contract es empaquetado e instalado en el peer0 del primer participante. No se instala en el resto de nodos de esa organización porque el peer0 es el único nodo endorsador. Se puede instalar en nodos no endorsadores modificando la configuración pero no es útil. También se puede elegir otro nodo distinto al peer0 para ser el endorser e incluso puede haber varios.

Se realiza lo mismo en el resto de participantes.

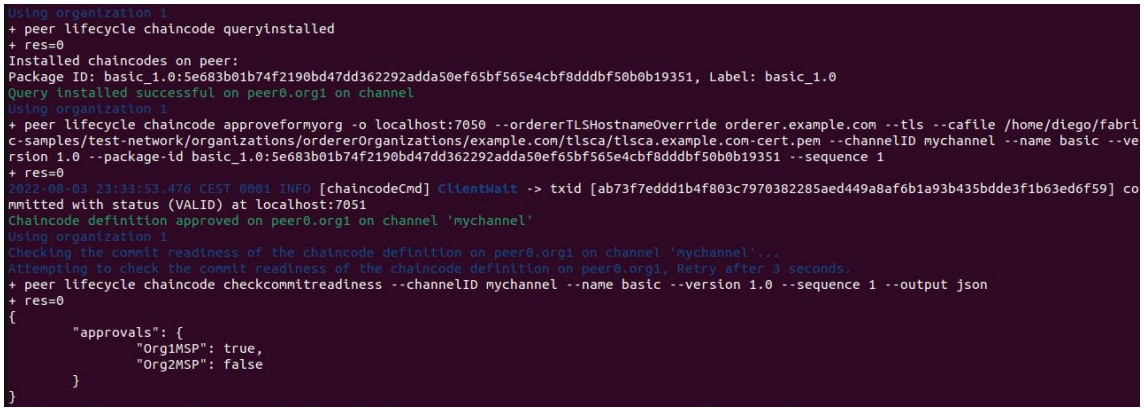

Fig. 42. Captura de respuesta por pantalla al instalar el Smart contract: aprobado por la primera organización.

El primer participante se conecta con sus credenciales y aprueba la instauración del Smart contract en el canal, modificando el parámetro de su membership provider a true. Dependiendo de la política de consenso configurada, podría valer con que una única organización apruebe la instauración de un contrato inteligente para que este se instale en el canal. De hecho, esta era la configuración predeterminada en las primeras versiones de Hyperledger Fabric. Sin embargo, esto no parece lo más democrático y tanto en este programa como en la configuración predeterminada de las últimas versiones de Fabric, se necesita del consenso de todos los nodos para instalar un contrato inteligente en un canal.

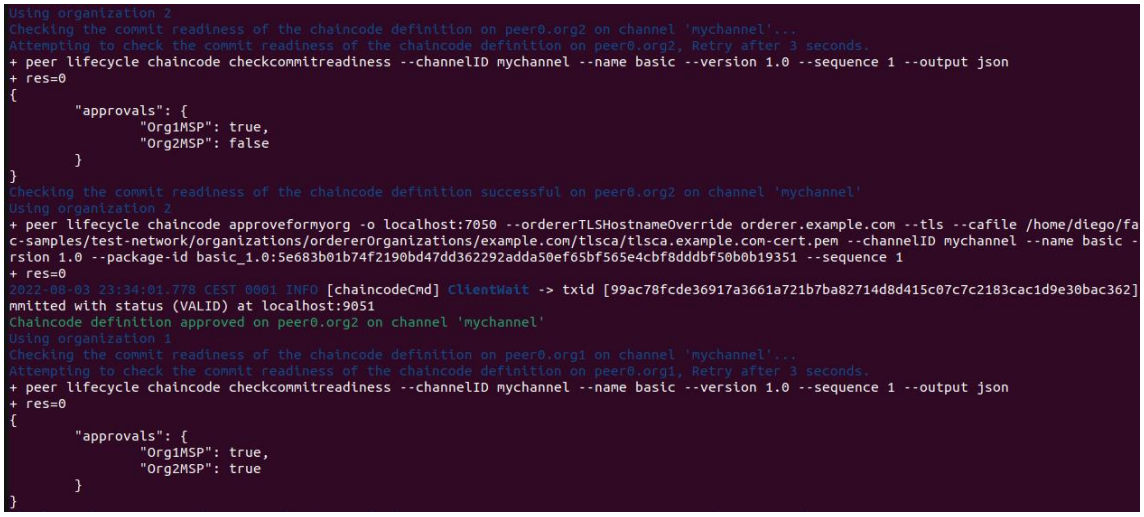

Fig. 43. Captura de respuesta por pantalla al instalar el Smart contract: aprobación de la segunda organización.

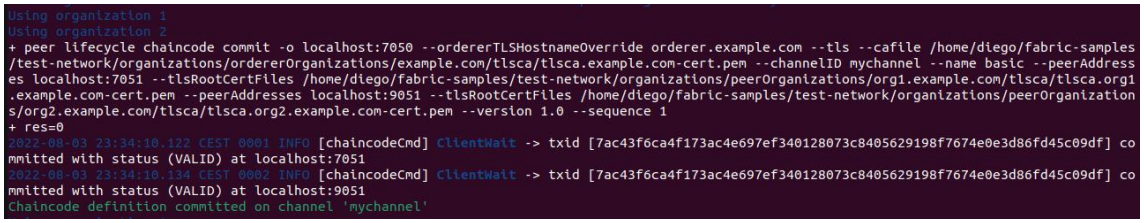

Fig. 44. Captura de respuesta por pantalla al instalar el Smart contract: confirmación de instalación.

Se muestra por pantalla el ID de la transacción (para instalar el Smart contract se ha llevado a cabo una transacción) y la confirmación de que el contrato se ha instalado en el canal.

Tras haber observado las respuestas por pantalla, se analizará más detenidamente las partes más importantes del código.

En primer lugar, obtiene el perfil de conexión, se asegura de que el perfil exista y se dice dónde va a crear la carpeta wallet para guardar las identidades y certificados. Posteriormente, se ejecuta “enrollAdmin()” y se generan los credenciales del administrador del sistema a partir de los certificados de autoridad. Esa parte del código es la siguiente:

```
async function main() {
  try {
    // build an in memory object with the network configuration (also known as a
connection profile)
    const ccp = buildCCP();

    // build an instance of the fabric ca services client based on
    // the information in the network configuration
    const caClient = buildCAClient(FabricCAServices, ccp);

    // setup the wallet to hold the credentials of the application user
    const wallet = await buildWallet(Wallets, walletPath);

    // in a real application this would be done on an administrative flow, and
only once
    await enrollAdmin(caClient, wallet);
```

Lo que se hace posteriormente es generar las credenciales de la primera organización (como hemos dicho en nuestro caso, al agricultor). Esta función se ejecutará cada vez que queramos añadir nuevos actores a la cadena de suministros, y en efecto, se volverá a ejecutar para unir al supermercado a la misma:

```
// in a real application this would be done only when a new user was required to
be added
// and would be part of an administrative flow
await registerAndEnrollUser(caClient, wallet, mspOrg1, org1UserId,
'org1.department1');
```

Cuando ya se tienen las credenciales adecuadas, se podrá hacer uso de las funciones del contrato inteligente con el nombre del canal y del contrato que se desee usar. Con las siguientes líneas de código, se permite al agricultor o al supermercado (o, en definitiva, a cualquier componente de la red) a conectarse con el SDK de Fabric para interactuar con el contrato. Para ello se usa Gateway, y es necesario conocer el nombre del contrato y del canal, además obviamente de tener las credenciales necesarias para ejecutar dichas funciones.

Posteriormente, la aplicación realiza una transacción para llenar el libro mayor con información a través del Smart contract "InitLedger". Se llama al contrato:

Esta función llena el libro mayor con información. Construye varios activos con varias propiedades como puede ser su identificador o el dueño. En nuestro caso, estos activos serían el aceite de oliva, y las propiedades son tales como un identificador, un tamaño (medido en kg), un dueño…y realmente se pueden poner tantas como se crean necesarias o de valor para el consumidor. Se pueden incluir campos como temperatura media, condiciones de humedad diferentes parámetros de calidad. Esta es la información que contiene la función InitLedger:

4.2 Uso de la aplicación

Una vez levantada la red, se procede a realizar distintas funciones de uso que se darían en una cadena de suministros, como puede ser: consultar el conjunto de activos, crear un nuevo activo, consultar el estado de un activo en concreto a través de un identificador, comprobar su existencia, actualizarlo y, por supuesto, transferirlo.

4.2.1 Consulta masiva de activos

Todos los nodos de una red de Hyperledger Fabric tienen guardados una copia del ledger. Alguien autorizado puede realizar operaciones de lectura del mismo. Las consultas más recurrentes son del estado actual del ledger, por ejemplo de quién es el dueño actual de un activo y no el historial de poseedores del mismo. Este tipo de consulta, denominadas consultas del "world state", son representadas en una base de datos de tipo clave-valor. Es una base de datos que bien configurada, da pie a consultas potentes. Por ejemplo, se pueden ver cuántos kilos de aceituna de un determinado tipo (aceite verde, por ejemplo) hay en la red.

Para realizar una consulta masiva de activos, se llama a la función "GetAllAssets" de la siguiente forma:

```
// Let's try a query type operation (function).
// This will be sent to just one peer and the results will be shown.
console.log('\n--> Evaluate Transaction: GetAllAssets, function returns all the
current assets on the ledger');
let result = await contract.evaluateTransaction('GetAllAssets');
console.log(`*** Result: ${prettyJSONString(result.toString())}`);
```

Va iterando a través del ledger, buscando activos con las premisas de entrada, las convierte en JSON y las muestra por pantalla.

Al llamar a la función mediante la terminal, muestra lo siguiente:

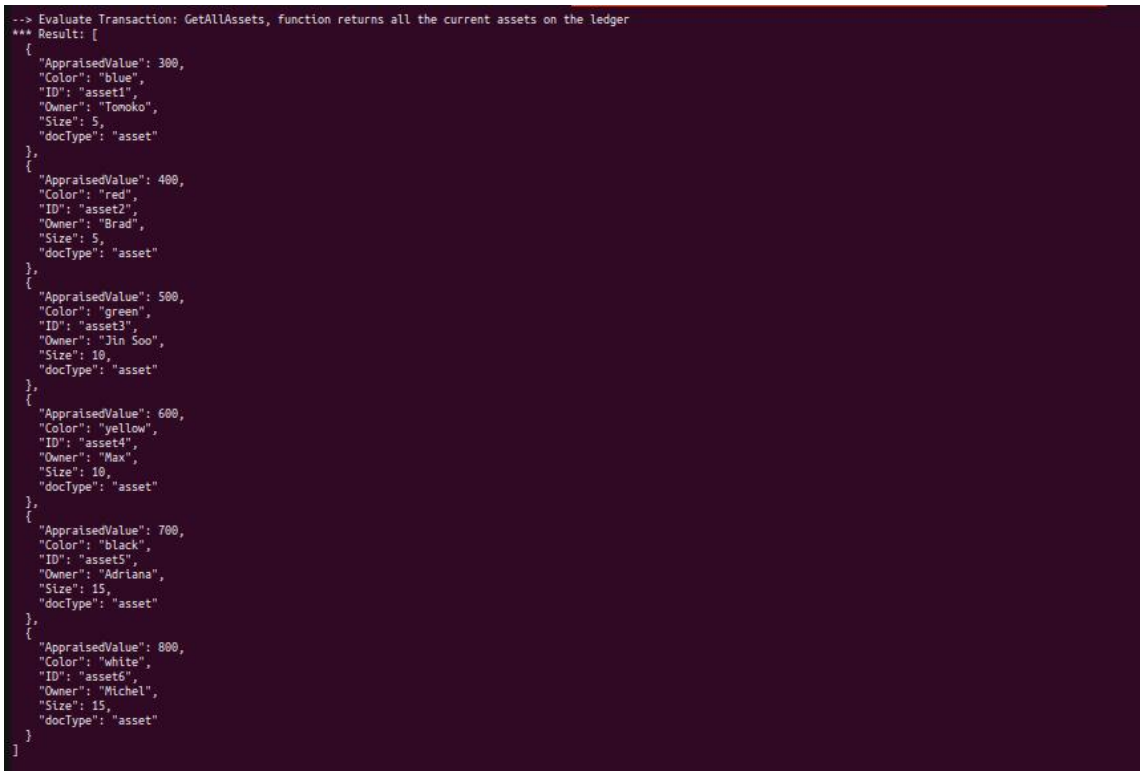

Fig. 45. Captura de respuesta por pantalla al llamar a la función GetAllAssets.

Esto son todos los activos que hay registrados en el Ledger, los registrados anteriormente mediante la función InitLedger.

4.2.2 Consulta de un activo a través de su identificador

Si no se desea llevar a cabo una consulta de todos los activos, se puede consultar un activo en concreto a través de su identificador. Para ello, se llama a la función ReadAsset:

```
console.log('\n--> Evaluate Transaction: ReadAsset, function returns an asset
with a given assetID');
result = await contract.evaluateTransaction('ReadAsset', 'asset13');
console.log(`*** Result: ${prettyJSONString(result.toString())}`);
```

Esta función simplemente va iterando el ledger hasta dar con el identificador dado como parámetro de entrada.

Al llamar a la función por pantalla, buscando el asset 1 que se inscribió con la función InitLedger, se devuelve esto:

```
--> Evaluate Transaction: ReadAsset, function returns "asset1" attributes
*** Result: {
  "AppraisedValue": "350",
  "Color": "blue",
  "ID": "asset1",
  "Owner": "Tomoko",
  "Size": "5"
}
```

Fig. 46. Captura de respuesta por pantalla al llamar a la función ReadAsset.

4.2.3 Creación de un activo

Otra de las funcionalidades de la aplicación es la de añadir un nuevo activo a la red. Por ejemplo, cuando el agricultor realiza la recogida de su aceituna. Para ello, se llama a la función "CreateAsset" así:

```
// Now let's try to submit a transaction.
// This will be sent to both peers and if both peers endorse the transaction, the
endorsed proposal will be sent
// to the orderer to be committed by each of the peer's to the channel ledger.
console.log('\n--> Submit Transaction: CreateAsset, creates new asset with ID,
color, owner, size, and appraisedValue arguments');
await contract.submitTransaction('CreateAsset', 'asset13', 'yellow', '5', 'Tom',
'1300');
console.log('*** Result: committed');
```

Básicamente, con los parámetros de entrada (que deben coincidir en tipo y cantidad con los de las propiedades de un activo, además del mismo orden), crea el activo y lo inscribe en el ledger.

Al llamar a la función mediante la terminal, se muestra lo siguiente:

```
--> Submit Transaction: CreateAsset, creates new asset with ID, color, owner, size, and appraisedValue arguments
*** Result: committed
*** Result: {
  "ID": "asset13",
  "Color": "yellow",
  "Size": "5",
  "Owner": "Tom",
  "AppraisedValue": "1300"
}
```

Fig. 47. Captura de respuesta por pantalla al llamar a la función CreateAsset.

Justo después de crear un activo, se llama automáticamente a la función ReadAsset para ver si se ha registrado correctamente:

```
--> Evaluate Transaction: ReadAsset, function returns an asset with a given assetID
*** Result: {
  "AppraisedValue": "1300",
  "Color": "yellow",
  "ID": "asset13",
  "Owner": "Tom",
  "Size": "5"
}
```

Fig. 48. Captura de respuesta por pantalla al llamar a la función ReadAsset de forma automática tras crear un activo nuevo.

4.2.4 Comprobar si un activo existe

Otras de las funciones es la de comprobar si un activo existe en la red. Para ello, toma como parámetro de entrada el identificador y devuelve un booleano con true o false dependiendo de si existe en la red o no.

Se llama a la función de la siguiente forma:

```
console.log('\n--> Evaluate Transaction: AssetExists, function returns "true" if
an asset with given assetID exist');
result = await contract.evaluateTransaction('AssetExists', 'asset1');
console.log(`*** Result: ${prettyJSONString(result.toString())}`);
```

Al llamarla por terminal, con el identificador asset1, creado con la función InitLedger, devuelve un true:

```
--> Evaluate Transaction: AssetExists, function returns "true" if an asset with given assetID exist
*** Result: true
```

Fig. 49. Captura de respuesta por pantalla al llamar a la función AssetExist.

4.2.5 Actualizar un activo

También se pueden actualizar los campos de un activo mediante la función UpdateAsset. En este caso, cambiamos el campo appraisedValue del asset1.

Llamamos a la función así:

```
console.log('\n--> Submit Transaction: UpdateAsset asset1, change the
appraisedValue to 350');
await contract.submitTransaction('UpdateAsset', 'asset1', 'blue', '5', 'Tomoko',
'350');
console.log('*** Result: committed');
```

Esta función itera el ledger para encontrar el activo. Si no existe, devuelve un error y si existe, sobrescribe los campos del activo con los nuevos dados como parámetros de entrada.

Al llamar a la función por terminal, devuelve esto:

```
--> Submit Transaction: UpdateAsset asset1, change the appraisedValue to 350
*** Result: committed

--> Evaluate Transaction: ReadAsset, function returns "asset1" attributes
*** Result: {
  "AppraisedValue": "350",
  "Color": "blue",
  "ID": "asset1",
  "Owner": "Tomoko",
  "Size": "5"
}
```

Fig. 50. Captura de respuesta por pantalla al llamar a la función UpdateAsset.

Para poder ver si se ha actualizado correctamente, se vuelve a llamar a la función ReadAsset.

Si intentamos actualizar un activo que no existe, nos devolverá este error por pantalla:

```
--> Submit Transaction: UpdateAsset asset70, asset70 does not exist and should return an error
2022-08-03T21:35:39.793Z - error: [Transaction]: Error: No valid responses from any peers. Errors:
    peer=peer0.org2.example.com:9051, status=500, message=error in simulation: transaction returned with failure: Error: The asset asset70 doe
s not exist
    peer=peer0.org1.example.com:7051, status=500, message=error in simulation: transaction returned with failure: Error: The asset asset70 doe
s not exist
*** Successfully caught the error:
    Error: No valid responses from any peers. Errors:
    peer=peer0.org2.example.com:9051, status=500, message=error in simulation: transaction returned with failure: Error: The asset asset70 doe
s not exist
    peer=peer0.org1.example.com:7051, status=500, message=error in simulation: transaction returned with failure: Error: The asset asset70 doe
s not exist
```

Fig. 51. Captura de respuesta por pantalla al llamar a la función UpdateAsset para un activo que no existe.

4.2.6 Transferir un activo

La función probablemente más importante para el caso de uso requerido en este trabajo, es la de transferir activos. A través de la cadena de suministros, se va transfiriendo el activo por distintos agentes hasta llegar al consumidor final.

Se llama a la función de la siguiente forma:

```
console.log('\n--> Submit Transaction: TransferAsset asset1, transfer to new
owner of Tom');
await contract.submitTransaction('TransferAsset', 'asset1', 'Tom');
console.log('*** Result: committed');
```

Esta función lee el activo con la función ReadAsset dado un identificador, lo pasa a tipo JSON y modifica el campo del owner. Luego lo sube al ledger.

Al llamar a la función por pantalla, dando los parámetros requeridos, devuelve esto:

```
--> Submit Transaction: TransferAsset asset1, transfer to new owner of Tom
*** Result: committed

--> Evaluate Transaction: ReadAsset, function returns "asset1" attributes
*** Result: {
  "AppraisedValue": "350",
  "Color": "blue",
  "ID": "asset1",
  "Owner": "Tom",
  "Size": "5"
}
```

Fig. 52. Captura de respuesta por pantalla al llamar a la función TransferAsset.

Para poder ver si se ha transferido correctamente, se vuelve a llamar a la función ReadAsset.

Capítulo 5: Conclusiones y futuros desarrollos

Tras la realización del proyecto, se puede apreciar un correcto funcionamiento del sistema en general, habiendo conseguido el objetivo principal de representar una cadena de suministros en la blockchain, con los beneficios de trazabilidad, seguridad y automatización que se comentan en el capítulo 2.

Como futuros proyectos, sería interesante conectar esta aplicación con una interfaz gráfica más visual a fin de poder observar qué es lo que ocurre de una forma más clara.

Por otro lado, también sería de interés desarrollar la aplicación en distintos nodos para conseguir una descentralización real, así como aumentar los nodos por organización con el mismo fin. Todo ha sido desarrollado en un ambiente local por un motivo claro de facilidades de desarrollo, pero si esta aplicación se desarrollará en un entorno de producción sería necesario instalarla en distintos nodos.

Otra acción para llevar a cabo en esta aplicación es la de aumentar el número de agentes en la cadena de suministros. Por simplicidad, la cadena de suministros ha sido representada únicamente por dos agentes, sin embargo, en un entorno más real habría más intermediarios que son necesarios tener en cuenta. Precisamente, un elemento de mejora es añadir sensores de temperatura y humedad a la red con el fin de monitorear el estado de la tierra donde crecen las olivas así cómo el entorno donde son transportadas, toda esta información sería verificada por la blockchain para asegurar su veracidad.

Capítulo 6: ODS

Los principales Objetivos de Desarrollo que se atacarán gracias a este proyecto serán los siguientes:

El ODS número 3.- Garantizar una vida saludable y promover el bienestar para todos para todas las edades. Esto es debido a que gracias a la implementación de este proyecto, nos aseguraremos de los procesos por los que ha ido sucediendo el producto final. En ocasiones, se utilizan abonos u otros productos de mantenimiento que pueden llegar a ser perjudiciales ara nuestra salud.

El número 8.- Fomentar el crecimiento económico sostenido, inclusivo y sostenible, el empleo pleno y productivo, y el trabajo decente para todos. En los últimos años, las condiciones de trabajo de nuestros agricultores han ido cayendo cada vez en condiciones más precarias. Éstos, obligados por la ley española, llevan a cabo distintos procesos para asegurar la seguridad del producto. Unos procedimientos de un alto valor económico que no se realiza en otros países y que, por tanto, consiguen un precio de su producto mucho más competitivo que el producido en España. Además, este aceite se acaba mezclando con el producido en España y vendiéndolo en nuestros supermercados sin muchas veces especificar el origen del mismo. Esto a largo plazo, conlleva la desaparación de dicha industria en nuestro país, afectando gravemente a nuestros ciudadanos y a la economía general del país.

El número 13.- Tomar medidas urgentes para combatir el cambio climático y sus efectos. Como se ha comentado anteriormente, en algunos de los países productores de aceite se usan unos productos de mantenimiento y producción que se han prohibido en muchos países por su daño al mediambiente. No tiene sentido que si se pohiben en España, obligando a nuestros agricultores a realizar un desembolso mayor en productos más comprometidos con el medioambiente, se comercialice el aceite de países extranjeros que no tienen este compromiso con el medioambiente. Con este sistema, podremos ver qué país es el productor del aceite y todos los procesos con los que han sido tratada la tierra, pudiendo identificar si se han usado productos maliciosos o no.

15.- Proteger, restaurar y promover la utilización sostenible de los ecosistemas terrestres, gestionar de manera sostenible los bosques, combatir la desertificación y detener y revertir la degradación de la tierra, y frenar la pérdida de diversidad biológica. Los ya comentados abonos y pesticidas usados, no solo son problemáticos para la

atmósfera y contribuyen al cambio climático, sino que también son perjudiciales para la tierra donde son depositados.

Capítulo 7 : Bibliografía

[1] "Emerge Stronger At A Time Of Uncertainty: Blockchain For Supply Chain",2020. [Online]. Available: https://www.ibm.com/downloads/cas/JX9KDGPJ

[2] "LA BLOCKCHAIN: FUNDAMENTOS, APLICACIONES Y RELACIÓN CON OTRAS TECNOLOGÍAS DISRUPTIVAS",2020. [Online]. Available: https://www.mincotur.gob.es/Publicaciones/Publicacionesperiodicas/EconomiaIndustrial/RevistaEconomiaIndustrial/405/DOLADER,%20BEL%20Y%20MU%C3%91OZ.pdf

[3] "Hyperledger: La Blockchain Empresarial", 2019. [Online]. Available: https://101blockchains.com/es/hyperledger-blockchain-guia/

[4] "Smart contracts: Casos reales de uso", 2021. [Online]. Available: https://www.wdiarium.com/es/smart-contracts-casos-reales-de-uso

[5] "Hyperledger Sawtooth", 2018. [Online]. Available: https://sawtooth.hyperledger.org/

[6] "Hyperledger Iroha Tutorial – Getting Started Guide", 2018. [Online]. Available: https://www.srcmake.com/home/iroha-tutorial

[7] "Hyperledger Indy", 2018. [Online]. Available: https://hyperledger-indy.readthedocs.io/en/latest/

[8] "Hyperledger Burrow – Even faster and easier to use!", 2018. [Online]. Available: https://www.hyperledger.org/blog/2018/08/29/hyperledger-burrow-even-faster-and-easier-to-use

[9] "Hyperledger Fabric Getting Started Tutorial + Installation Guide", 2018. [Online]. Available: https://www.srcmake.com/home/fabric

[10] "Qué es Hyperledger?", 2018. [Online]. Available: https://blogs.imf-formacion.com/blog/tecnologia/que-es-hyperledger-201805/

[11] "Welcome to Hyperledger Cello", 2018. [Online]. Available: https://hyperledger.github.io/cello/

[12] "Hyperledger Composer Introduction and Playground Tutorial (with Demo Code)", 2018. [Online]. Available: https://www.srcmake.com/home/composer

[13] "How Hyperledger Fabric works", 2020. [Online]. Available:

https://www.coding-bootcamps.com/blog/review-of-hyperledger-fabric-architecture-and-components.html

[14] "Flujo de transacciones", 2017. [Online]. Available:

https://fabric-ericjl.readthedocs.io/es/release/txflow.html

[15] "¿Por qué usar Docker? Cuandpo la infraestructura se vuelve código", 2020. [Online]. Available: https://www.toolboxtve.com/es/por-que-usar-docker-cuando-la-infraestructura-se-vuelve-codigo/

[16] "Loggins for Rails apps in Docker", 2015. [Online]. Available:

https://manas.tech/blog/2015/12/15/logging-for-rails-apps-in-docker/

[17] "Application", 2020. [Online]. Available:

https://hyperledger-fabric.readthedocs.io/en/release-2.2/developapps/application.html#basic-flow

[18] "Writing your first application", 2020. [Online]. Available:

https://hyperledger-fabric.readthedocs.io/en/release-2.2/write_first_app.html

[19] Dolores, C.(2020). La blockchain: fundamentos, aplicaciones y relación con otras tecnologías disruptivas. [Online]. Available:

https://www.mincotur.gob.es/Publicaciones/Publicacionesperiodicas/EconomiaIndustrial/RevistaEconomiaIndustrial/405/DOLADER,%20BEL%20Y%20MU%C3%91OZ.pdf

[20] "Derriba las barreras para usar blockchain en tu negocio", s.f. [Online]. Available:

https://icommunity.io/

[21] "Whiterpaper", 2019. [Online]. Available: https://wiki.hyperledger.org/display/PSSIG/Whitepaper

[22] "Hyperledger Sawtooth Security Audit", 2018. [Online]. Available:

https://www.hyperledger.org/blog/2018/05/22/hyperledger-sawtooth-security-audit

[23] "Hyperledger Iroha Security Audit Results", 2019. [Online]. Available:

https://www.hyperledger.org/blog/2019/02/04/hyperledger-iroha-security-audit-results

[24] Smith, C. (2019). Solving the challenges of Digital Identity with Hyperledger Indy. [Online]. Available:

https://gadgetgang.com/solving-the-challenges-of-digital-identity-with-hyperledger-indy/

[25] "Hyperledger Burrow – Even faster and easier to use", 2018. [Online]. Available:

https://www.hyperledger.org/blog/2018/08/29/hyperledger-burrow-even-faster-and-easier-to-use

[26] "Hyperledger Fabric incorpora Ethereum a su Plataforma", s.f. [Online]. Available:

https://andreshevia.com/2018/11/04/hyperledger-fabric-incorpora-ethereum-a-su-plataforma/

[27] "Hyperledger_Quilt_logo_color", s.f. [Online]. Available:

https://cn.hyperledger.org/projects-2-2/attachment/hyperledger_quilt_logo_color-2

[28] "Hyperledger Explorer", 2018. [Online]. Available:

https://blockchain.intellectsoft.net/blog/hyperledger-framework-ecosystem-variations-of-the-open-source-blockchain/hyperledger_explorer/

[29] "Welcome to Hyperledger Cello", s.f. [Online]. Available:

https://hyperledger.github.io/cello/

[30] "Hyperledger Composer Introduction and Playground Tutorial (with Demo code)", 2018. [Online]. Available:

https://www.srcmake.com/home/composer)

[31] "Hyperledger Composer Introduction and Playground Tutorial (with Demo code)", 2018. [Online]. Available:

https://www.srcmake.com/home/composer)

[32] Rodríguez, N. (2019). La blockchain empresarial. [Online]. Available:

https://101blockchains.com/es/hyperledger-blockchain-guia/)

[33] "Flujo de transacciones", 2017. [Online]. Available:

https://fabric-ericjl.readthedocs.io/es/release/txflow.html)

[34] Palladino, S. (2015). Logging for rail apps in Docker. [Online]. Available:

https://manas.tech/blog/2015/12/15/logging-for-rails-apps-in-docker/)