FACULTAD DE INGENIERÍA INFORMÁTICA
DOBLE GRADO EN ADMINISTRACIÓN Y DIRECCIÓN DE EMPRESAS E INGENIERÍA INFORMÁTICA

TRABAJO DE FIN DE GRADO

Título: Gestión documental mediante el uso de blockchain

Autor: *Patricia Llamas Roque*
Directores: *María Inmaculada Pardines Lence*
José Luis Risco Martín

Curso Académico 2021-2022
Convocatoria de Junio

Índice General

Índice de tablas

Índice de figuras

Resumen

Hoy en día las empresas trabajan con infinidad de documentos a la vez, lo que puede suponer un gran problema por la posible aparición de documentos duplicados, la pérdida o incluso la modificación malintencionada de estos. Este trabajo pretende solucionar este problema mediante el uso de la tecnología blockchain.

El objetivo de este trabajo se centra en demostrar la utilidad de la tecnología blockchain en el sector de la medicina. El desarrollo del trabajo se centra en la creación de una red de blockchain a través del framework Hyperledger Fabric que unido a una aplicación web se encargue de gestionar historiales médicos de pacientes. Se quiere asegurar la integridad y la seguridad de todos los datos contenidos en dicha aplicación.

Blockchain es una tecnología muy utilizada en los últimos tiempos, pero en la realidad sus aplicaciones en gestión documental son escasas. Se quiere explotar ese hueco en el mercado diseñando una red que permita almacenar registros de historiales médicos mediante el almacenamiento de un hash que haga referencia a toda la información contenida en dichos historiales y cualquier acción realizada sobre los mismos en una red de blockchain.

La creación de la red se realiza de tal manera que sea posible su unión no solo con la aplicación previamente mencionada si no con cualquier producto futuro relacionado con la gestión documental.

Palabras clave: Blockchain, red permisionada, Hyperledger Fabric, chaincode, Google Cloud Platform, ledger, estado mundial, gestión de historiales de pacientes.

Abstract

Nowadays companies must deal with tons of documents at the same time. This situation can lead to problems due to the possible appearance of duplicated, lost or even maliciously modified documents. This work pretends to solve the problem that has just been mentioned using blockchain technology.

This work focuses on proving the utility of blockchain in the medicine sector. The development of the project focuses on the creation of a blockchain network using Hyperledger Fabric framework. This implementation joins with a patient record management web page. The goal is to guarantee security and immutability of all data contained in the application.

Blockchain is a "cutting-edge" technology but its applications in document management are limited. This work wants to exploit this market leak by designing a network that allows to store registers of patient's records using hashes of the information contained in them and any other action referred to the patient's records in a blockchain network.

This project has been formulated in a way that enables the blockchain network to join not only with the previously mentioned application but also with any feature related with document management.

Keywords: Blockchain, permissioned blockchain, Hyperledger Fabric, chaincode, Google Cloud Platform, ledger, world state, patient's record management.

Capítulo 1. Introducción

1.1. Motivación

Hoy en día los grandes flujos de documentos suponen una enorme preocupación para la mayoría de los negocios. Esta cantidad ingente de documentos puede llevar a las empresas a enfrentarse a serios problemas entre los que se encuentra la información duplicada, la pérdida y la modificación maliciosa de documentos que pueden acarrear serias consecuencias, las cuales van ligadas a una pérdida de productividad por parte de la compañía. Son muchas las empresas que se están centrando en buscar alternativas eficientes para automatizar los procesos de trabajo, facilitar y hacer más segura la gestión documental. En este punto es donde entra en juego el blockchain.

Las redes de blockchain están formadas por un conjunto de bloques unidos mediante criptografía que permiten que la información que se encuentra almacenada en ellos sea inmutable y segura. Debido a sus características es ampliamente utilizada en el ámbito financiero, así como en logística y distribución.

En gestión documental el blockchain permite validar y verificar todos los documentos existentes en una empresa. Gracias a sus propiedades se puede comprobar entre otras cosas la persona que ha creado un documento, el momento, el lugar, su última modificación y además permite comprobar si este ha intentado ser manipulado. Además, al existir redes de blockchain privadas los documentos solamente serán accesibles a las personas autorizadas para ello, proporcionando así toda la privacidad y transparencia necesarias en una empresa.

El blockchain ha sido la tecnología escogida para este proyecto ya que permite realizar un seguimiento eficaz de todos los cambios de un documento y además sirve para verificar la autenticidad de un documento, consiguiendo así garantizar algunos de los objetivos más importantes de la seguridad informática:

- **Confidencialidad:** En seguridad informática la confidencialidad es el principio que asegura que la información solo es accesible a aquellas personas que tienen permiso para ello. El uso de redes privadas de blockchain permite cumplir este objetivo ya que garantiza el acceso a los datos únicamente a las partes autorizadas.
- **Integridad:** Este concepto hace referencia a la certeza de que la información que se encuentra almacenada en determinados dispositivos no ha sido manipulada por terceras partes de manera maliciosa. El blockchain asegura la inmutabilidad de los datos gracias a la criptografía que une los bloques de su estructura.
- **Disponibilidad:** La información debe estar disponible para las personas autorizadas en los momentos en los que sea necesario. En blockchain cada nodo participante en la red contiene una copia del registro distribuido. Por ello, en el caso de haber una avería o un problema en una de las partes, la información no se perdería y sería accesible desde el resto de ellas.

1.2. Objetivos

El objetivo del presente trabajo se centra en crear una red de blockchain para poder así demostrar la utilidad de esta tecnología en relación con la integridad y la seguridad de la información en el área de la gestión documental.

Este objetivo principal se divide en cuatro objetivos secundarios:

- Tomar decisiones sobre la estructura de la red y el software y hardware necesario en base a la investigación de la temática de trabajo seleccionada y el material disponible en fuentes primarias y secundarias.
- Aprender sobre el funcionamiento y la tecnología blockchain debido a su enorme auge en los últimos tiempos y a las enormes posibilidades que ofrece.
- Aplicar y profundizar en los conocimientos aprendidos y las tecnologías utilizadas durante los cinco años de carrera, siendo todo esto de gran ayuda durante el desarrollo total del trabajo.
- Unión de la red de blockchain con un caso de uso real para demostrar la utilidad de esta tecnología en el ámbito empresarial.

1.3. Plan de trabajo

Este trabajo se ha comenzado a desarrollar a finales del mes de junio de 2021. Durante los dos primeros meses del proyecto el trabajo realizado se ha ido revisando por los jefes de la empresa Evenbytes, para la que se realiza la implementación de la red de blockchain. Posteriormente, la organización del trabajo se basa en el trabajo personal del alumno con revisiones periódicas mensuales por parte de ambos tutores, para guiar y comprobar los avances realizados.

Para conseguir desarrollar el trabajo de una forma eficiente se ha decidido agrupar el proyecto en una serie de etapas, estableciendo objetivos a cumplir en cada una de estas.

- *Primera etapa – Investigación*

El objetivo de esta etapa es realizar una investigación sobre conceptos teóricos relacionados con el blockchain. Esta etapa es crucial para entender el marco teórico básico que debe servir de base a la hora de tomar decisiones sobre la estructura de la red.

- *Segunda etapa – Análisis*

En ese tiempo se ha de investigar sobre aplicaciones de blockchain en diferentes ámbitos, poniendo el foco en la gestión documental. Además de eso se quiere realizar una concienzuda búsqueda sobre las distintas aplicaciones disponibles para la implementación de la red. Como hito principal de esta fase se ha de destacar la toma de decisiones en base a los conceptos previos estudiados para saber si son necesarios para el proyecto, como por ejemplo el uso de smart contracts, especificaciones sobre el hardware, etc.

- *Tercera etapa – Estructura de la red*

Esta etapa se centra en detallar en profundidad la estructura de la red fijando el número de nodos, canales y organizaciones entre otros elementos.

- *Cuarta etapa – Conocimiento del framework y creación de la red*

La cuarta parte del trabajo se centra en el conocimiento del framework a utilizar, Hyperledger Fabric. Para ello es necesario leer un extenso tutorial sobre el funcionamiento de esta donde se detallan numerosos conceptos teóricos y se explica la manera de levantar la red de blockchain.

- *Quinta etapa – Implementación*

El siguiente paso se basa en la creación de los scripts de las transacciones y el chaincode. Para ello se dedicará una serie de días a la familiarización de los lenguajes que utiliza Hyperledger (Golang y Node js), desconocidos hasta el momento. Esta fase también comprende el desarrollo de la API que se utiliza para realizar llamadas a la red de blockchain. El último paso de esta etapa comprende la unión de la red con una aplicación que gestiona historiales clínicos de pacientes.

- *Sexta etapa – Pruebas*

En esta parte de la memoria se realizan pruebas de los scripts de transacciones mediante línea de comandos y desde la aplicación de gestión de pacientes.

- *Séptima etapa – Documentación*

Esta parte se debe de realizar periódicamente durante el desarrollo de todo el proyecto para conseguir profundizar y sintetizar los conocimientos aprendidos y las decisiones tomadas.

1.4. Organización de la memoria

Se procede a detallar la disposición de este documento para facilitar la comprensión del trabajo realizado. Este escrito está dividido en seis capítulos.

En este primer capítulo se especifican la motivación y objetivos que marcan este trabajo de fin de grado, así como el plan de trabajo seguido para asegurar una correcta división del tiempo y de los recursos disponibles.

El capítulo dos comprende el estado del arte de este trabajo. En esta parte se debe de realizar una investigación documental de todos los escritos y aplicaciones de esta tecnología en los diferentes sectores. El objetivo es conocer los proyectos que se han realizado y dilucidar en qué aspectos el trabajo a realizar se diferencia de los que ya se han llevado a cabo. Asimismo, se enumeran las plataformas existentes, tanto a nivel de software como de hardware, para la creación y manejo de la red de blockchain.

En el tercer capítulo se pormenorizan los conceptos básicos alrededor del término blockchain, comenzando por la enumeración de los diferentes tipos de redes distribuidas (entre los que encontramos el blockchain). Posteriormente se explican los distintos tipos de redes de blockchain, los elementos que forman una red y el funcionamiento de estas.

El capítulo cuatro explica de manera más exhaustiva la plataforma escogida, el hardware que se precisa para la implementación de la red, así como las herramientas auxiliares necesarias. El último

apartado de este capítulo abarca la explicación de los conceptos básicos del framework escogido y la especificación del prototipo a desarrollar.

El quinto capítulo contiene los cambios realizados en el código de la red tanto en el chaincode, como en los scripts de transacciones. Además, se incluye una explicación detallada de la generación del entorno y de la creación de la Interfaz de Programación de Aplicaciones (API). Por último, se detallan las funcionalidades de la aplicación web creada, así como pruebas realizadas desde la línea de comandos y desde la aplicación que se acaba de mencionar.

A continuación, en el capítulo seis, se exponen las conclusiones a las que se ha llegado tras el proyecto y se enuncian posibles acciones futuras y maneras en las que se puede ampliar el proyecto.

Por último, se podrán encontrar apéndices donde se amplía información relevante del proyecto, para que el lector pueda profundizar en algunos aspectos de secciones previas donde no se ha entrado en detalle.

Capítulo 2. Estado del arte

Antes de comenzar el desarrollo de este trabajo, se ha realizado una investigación para conocer los proyectos existentes en el ámbito de la gestión documental que emplean la tecnología blockchain. El objetivo de este estudio es conocer las similitudes y diferencias de este trabajo respecto a los proyectos existentes para así conocer dónde se sitúa éste en el mercado. Otro de los motivos de esta búsqueda es el de utilizar la información hallada como guía y punto de partida a la hora del desarrollo final.

Este capítulo se agrupa principalmente en dos secciones. La primera de ellas detalla una serie de ejemplos de aplicaciones reales de la tecnología blockchain en el sector de la gestión documental. La segunda sección enumera las diferentes plataformas existentes para la creación de la red tanto a nivel de software como de hardware.

2.1. Aplicaciones de la tecnología blockchain

El blockchain es una tecnología muy utilizada en el sector financiero y en logística por su inmutabilidad y seguridad. A continuación, se procede a enumerar una serie de casos de uso de este tipo de tecnología en gestión documental, por su aplicación en la red que se pretende implementar.

- *JOISTO* [19]

En enero de 2020, un pequeño equipo de desarrolladores creó una plataforma mediante Hyperledger Fabric que permite guardar documentos de manera inmutable mediante una red de blockchain.

El archivo de blockchain de Joisto usa almacenamiento fuera de la cadena, lo que significa que los propios archivos o metadatos no están almacenados en la misma. En la cadena de bloques se almacenan comprobaciones criptográficas de los documentos además de registros de eventos que han ocurrido en los documentos.

Los documentos almacenados en Joisto se guardan en dos lugares distintos. El propio documento se guarda cifrado en los archivos de la compañía. Por otro lado, todas las modificaciones realizadas a esos documentos se conservan cifradas en la inmutable cadena de bloques. Se podrían almacenar los documentos de la misma manera de forma tradicional pero este sistema otorga una memoria eterna e intocable donde se almacena toda esta información.

Cada vez que se realiza un cambio en el documento, el evento es comprobado por Joisto y por un verificador institucional lo que permite asegurar todos los archivos contra cualquier actividad maliciosa.

Cuando una nueva compañía se une, se crea un nuevo canal. En ese momento los miembros del canal acuerdan una serie de políticas como reglas para añadir bloques, detalles sobre identificación y

modificaciones en la red. De esta forma la información solo será accesible para aquellas personas con autoridad para ello.

Esta empresa lleva a cabo una gestión documental muy parecida a la que se va a implementar. Almacena los hashes de la información del documento y no el propio documento en sí (almacenamiento fuera de la cadena). Se desea implementar la red de la misma forma dado que la idea no es guardar información de forma permanente si no asegurarse de que esta no es modificada. Se guardará información por un lado en una base de datos local y posteriormente en la cadena se guardará un hash y un identificador sobre esos documentos a almacenar. El almacenamiento de toda la información de manera inmutable podría traer problemas por guardar información sensible de usuarios. Además, el almacenamiento fuera de la cadena permite reducir el tamaño de la información almacenada.

El uso de canales es un elemento que destacar de este proyecto, idea que también se implementará en la futura red. La mayor ventaja de las redes permisionadas es que la información es privada y accesible únicamente por los integrantes de la red. Sin embargo, esta característica no es suficiente, además de eso hace falta un sistema, en este caso los denominados canales, que permita garantizar que se pueden hacer distinciones entre unos miembros de la red y otros, otorgando permisos desiguales a cada uno de ellos.

- *Gobierno de Estonia* [40]

Estonia lleva gestando desde el año 2000 un proyecto denominado e-Estonia, para construir una sociedad digital. En este año, Estonia declaró el acceso a internet como un derecho humano básico. Este país es considerado por muchos como la sociedad digital más avanzada del mundo. Un 99% de sus servicios de gobierno se realizan de manera online.

En el año 2008 introdujo la tecnología blockchain como elemento para continuar y apoyar su transformación digital. Su introducción se debió a la necesidad de mitigar la posibilidad de manipulaciones de datos debido al ataque cibernético nacional sufrido en 2007. Este país fue la primera nación en el mundo en implementar tecnología blockchain en sus sistemas de producción. La tecnología escogida por el gobierno estonio es el blockchain KSI (Keyless Signature Infrastructure), donde los datos nunca abandonan el sistema si no que se envía un hash de estos a la cadena de blockchain. Al no almacenar los documentos, se pueden guardar hasta petabytes de datos. Todas las bases de datos de este país se apoyan bajo la tecnología blockchain lo que les permite asegurar la integridad de todos los registros almacenados en ellas.

Este país almacena todos los datos de los ciudadanos en Oracle y almacena registros de estos en la cadena de bloques. Esto se debe a las razones comentadas previamente. En ninguna circunstancia se debe almacenar información personal de los ciudadanos, por ello utilizan el mismo sistema que Joisto y almacenan hashes de esa información.

- *Izertis – Gobierno de Bizkaia* [20]

La empresa Izertis, a través de Hyperledger, ha creado una plataforma gratuita de blockchain para el gobierno de Bizkaia que permite intercambiar información de los ciudadanos, las empresas y el gobierno de una forma eficiente y segura que garantiza la confidencialidad y privacidad de la

información compartida. La idea de este proyecto se centraba en crear un nexo eficiente entre las empresas, los ciudadanos y el gobierno mediante una herramienta que permitiera aumentar la confidencialidad y la privacidad de la información enviada.

Izertis optó por utilizar diferentes frameworks y herramientas de Hyperledger entre las que se encuentra Hyperledger Caliper, Hyperledger Explorer e Hyperledger Fabric, además empleó Docker y Kubernetes. Esta tecnología le permitió diseñar una red con una arquitectura independiente y modular, con mecanismos de consenso para garantizar la confianza y con la posibilidad de controlar y gestionar a los participantes de la red. En relación con las tecnologías utilizadas en nuestro trabajo se utilizará, al igual que en este proyecto, Hyperledger Fabric y Docker.

- *Izertis – Endesa* [34]

La pobreza energética es un problema que afecta al 11% de los hogares en España. Izertis ha trabajado en un proyecto junto a Endesa que permite a ayuntamientos y comunidades autónomas en España detectar usuarios vulnerables en riesgo de exclusión social en tiempo real. Este proyecto, CONFIA, utiliza la tecnología IBM Blockchain, permitiendo agilizar de manera notable los anteriores trámites necesarios para la protección energética de colectivos vulnerables.

Esta iniciativa de Endesa, pionera en Europa, permite detectar a consumidores que se encuentran en riesgo de exclusión. Previamente Endesa necesitaba contactar primero con las Comunidades Autónomas y estas posteriormente con los ayuntamientos que son los encargados de gestionar esas ayudas. Este tedioso proceso normalmente se traducía en problemas en la concesión de las ayudas y retrasos en los pagos. Con el nuevo proyecto se permite detectar en tiempo real si un usuario cumple los requisitos adecuados para beneficiarse del Bono Social mediante la verificación de los datos de los ciudadanos.

El trabajo se ha apoyado en la tecnología IBM Blockchain, de IBMCloud, que posibilita el almacenamiento de información en una base de datos distribuida respetando en todo momento la ley de protección de datos. La implementación de la red se ha sustentado en el uso de Smart Contracts para sacar el máximo partido a las características del blockchain.

- *Izertis – Caribbean Examinations Council (CXC)* [35]

CXC es un organismo fundado en 1972 que se encarga de otorgar certificados educativos en 16 países y territorios caribeños. Desde siempre los registros académicos se han gestionado en formato físico. Este hecho hace depender a los estudiantes de instituciones que certifiquen los logros académicos alcanzados, viéndose envueltos en largos y costosos procesos de papeleo.

En este país, se puede dar la casuística de que las entidades que gestionan esos procesos quiebren, pierdan la información o se destruya debido a desastres naturales. Por estos motivos, es crucial que se implemente una solución que asegure el intercambio fluido de certificados y la inmutabilidad de estos. Por ello, Izertis ha desarrollado una plataforma escalable y segura que permite la emisión, presentación, verificación y gestión de credenciales.

- *Telefónica Tech – Servicio de certificación documental* [36]

Telefónica Tech ha anunciado el lanzamiento de un servicio de certificación documental que utiliza la tecnología blockchain y permite a las empresas autenticar cualquier tipo de contenido digital de manera fácil y rápida. Cada certificación contiene la firma digital del usuario, una marca de tiempo para asegurar el momento en el que se emite la certificación y la huella del contenido a certificar. Esta plataforma se puede aplicar a numerosos casos de uso de diferentes sectores como la formación, los seguros, el sector administrativo y legal, a contenidos multimedia, etc.

TrustOS es una plataforma que permite a las empresas conectar sus procesos de negocio con una red de blockchain. Una autoridad certificadora emite y firma un certificado a cada usuario que desea formar parte de la red para asegurar así la identidad de los participantes en la red y para evitar suplantaciones. Esta red incorpora además unos módulos de seguridad de hardware (HSM) que gestionan las claves criptográficas de los usuarios de manera segura.

Está formada por cuatro módulos independientes entre sí, Track, Token, Settle y Trust, pero que pueden interactuar entre ellos para solucionar cualquier problema. La lógica de cada uno de ellos se compone principalmente de un smart contract y de una API que permite interactuar con el propio módulo de manera sencilla. Esta estructura permite añadir una nueva API por cada nueva funcionalidad que se añada a la plataforma.

El primero de ellos, Track, permite llevar a cabo la trazabilidad del ciclo de vida de un activo. Token permite otorgar valores transferibles a objetos físicos. Settle permite crear acuerdos mediante contratos inteligentes entre las partes implicadas. Por último, Trust es una certificadora instantánea de activos.

- *Filecoin* [37]

Filecoin es una criptomoneda de código abierto y una plataforma de pagos destinada a ser una plataforma para el almacenamiento de datos utilizando la tecnología blockchain. Este sistema de archivos ha sido creado por Protocol Labs encima del sistema de archivos InterPlanetary (IPFS), permitiendo a los usuarios alquilar espacio libre de disco duro.

Filecoin es un sistema de ficheros descentralizado que permite un almacenamiento local y eficiente de toda la información, permitiendo así a cada usuario estar cerca de la información que tiene almacenada en la red. Cuando un cliente quiere almacenar datos en el sistema de archivos observa los diferentes proveedores de espacio y los precios de cada uno. Los proveedores de espacio compiten por ganar el contrato de almacenamiento y posteriormente el cliente escoge el ganador. En ese momento el cliente envía la información que desea almacenar. La red comprueba que los datos se están guardando de forma segura mediante pruebas criptográficas que son almacenadas en la red en nuevos bloques. Solo los bloques que son correctos se aceptan y de esta forma los proveedores de almacenamiento ganan recompensas.

Cuando un cliente quiere acceder a un archivo busca al proveedor que puede tenerlo almacenado, y escoge, de entre todos los que contienen el documento, el más barato y rápido. Paga al proveedor elegido y obtiene los documentos. Todas las transacciones de la red utilizan la criptomoneda FIL.

2.2. Ecosistema Distributed Ledger Technology (DLT)

2.2.1. Hardware

Cuando hablamos del hardware necesario en una red distribuida nos referimos a nodos que actúen como sistemas de almacenamiento, servidores u ordenadores. A continuación, mencionamos las dos soluciones más utilizadas:

- *Amazon Managed Blockchain* [21]

Amazon Managed Blockchain es un servicio totalmente administrado que facilita la adhesión a redes públicas o la creación y administración de redes privadas escalables mediante el uso de los populares marcos de código abierto Hyperledger Fabric y Ethereum.

El servicio Managed Blockchain de AWS facilita la administración y el mantenimiento de la red de blockchain, administra certificados y permite invitar nuevos miembros a la red. Además, elimina la necesidad de proporcionar el hardware, configurar el software, los componentes de la red y se encarga de la seguridad.

Cuenta con una API que permite a los participantes de la red agregar y eliminar miembros. Monitoriza la red y cambia por otros los nodos con un rendimiento deficiente. Este servicio permite escalar la red de blockchain a medida que esta crezca. Por otro lado, proporciona una serie de combinaciones de CPU y memoria dando flexibilidad a los usuarios para escoger los recursos adecuados a su carga de trabajo. Además, protege los certificados de la red con la tecnología AWS Key Management Service, eliminando la necesidad de establecer almacenamiento seguro de claves.

El servicio de solicitud de Hyperledger Fabric no guarda el historial completo de las transacciones, haciendo difícil el seguimiento de estas y su recuperación si fuera necesario. Amazon Managed Blockchain mejora este servicio porque cuenta con un registro de cambios inmutables, asegurando que todos los datos se guardan de forma duradera.

- *Google Cloud Platform*

En 2018, Google se asoció con Digital Asset y BlockApps para ofrecer servicios de blockchain en Google Cloud Platform (GCP). GCP integra la posibilidad de crear redes de bloques con Hyperledger Fabric y con Ethereum. Esta integración permite a los usuarios de Google crear aplicaciones descentralizadas (dApps) para sus negocios sin necesidad de ocuparse y configurar la plataforma subyacente [38].

A comienzos del año 2022 Yolande Piazza, vicepresidenta de servicios financieros de Google Cloud, anunció que Google había creado un equipo centrado en el desarrollo de la tecnología blockchain. La idea de la compañía es garantizar a los clientes que creen proyectos de blockchain una infraestructura escalable y segura para el respaldo de sus redes.

El equipo se encargará de llevar a cabo una serie de iniciativas para apoyar a compañías con sus ecosistemas blockchain, entre las que se encuentran, entre otras, la posibilidad de alojamiento de nodos para facilitar el despliegue de redes, validación de nodos, alojamiento en BigQuery de

transacciones de blockchain para Bitcoin, Ethereum, Dash y otras criptomonedas o ayuda para la integración de las cadenas de bloques con los demás sistemas de GCP [43].

2.2.2. *Software*[1]

Las redes distribuidas no están limitadas a un lenguaje de programación concreto y pueden trabajar con cualquier tipo de herramienta. Encontramos numerosos ejemplos de plataformas para blockchain entre las que destacan las siguientes:

- *Ethereum* [26]

Ethereum es posiblemente el blockchain más conocido después de bitcoin. Es una red no permisionada que soporta la creación de redes públicas en un entorno de prueba. Ethereum fue la primera plataforma que permitió la creación de smart contracts. En estos momentos, la red se encuentra bastante saturada y la velocidad por transacciones no es muy alta, además el coste por transacción es bastante elevado.

Para poder conectarse a Ethereum hace falta un wallet, que te permite la conexión a una red pública y la administración de cuentas y contratos. Este monedero te permite conectarte por un lado a Mainnet, red que se utiliza para proyectos de producción y para funcionar necesita la inversión de dinero, y por otro, a la red pública Testnet, utilizada para pruebas. Es posible hacer pruebas en redes privadas.

- *Quorum* [22]

Quorum es una plataforma de código abierto con permisos que modifica mínimamente el núcleo de Ethereum, por lo que puede incorporar la mayoría de las actualizaciones de esta. Al no utilizar el mecanismo de consenso Proof of Work, que se explicará posteriormente, permite procesar multitud de transacciones al segundo.

- *Cardano* [23]

Cardano es una plataforma de código abierto que ofrece una alta escalabilidad y seguridad gracias a su estructura diseñada por capas. Además, permite la creación de aplicaciones descentralizadas (dApps) y contratos inteligentes. Esta plataforma utiliza como mecanismo de consenso Proof of Stake, eliminando el consumo de energía necesario en el mecanismo de Proof of Work.

- *Ripple* [25]

Ripple es una plataforma de código abierto que permite transacciones rápidas y baratas. Ripple se utiliza para temas financieros, entre los que se encuentran el cambio de divisas con baja comisión, la realización de transferencias internacionales rápidas. Además, se emplea como ecosistema de pago.

- *VMWare Blockchain* [24]

VMWare Blockchain es una plataforma que permite la creación de redes privadas y con permisos. Su implementación puede hacerse tanto en entornos locales como en la nube o mediante servicios en la nube. Está diseñada para ser compatible con múltiples lenguajes de smart contracts. Cada usuario

[1] Para obtener más información sobre las plataformas blockchain, consultar las referencias bibliográficas desde la [22] a la [28].

únicamente tiene acceso a los datos relevantes para él, accediendo solo a las transacciones a las que tiene derecho.

- *Hyperledger* (Reyes, 2019) [27,28]

Hyperledger es una tecnología de código abierto creada con el objetivo de promover proyectos utilizando la tecnología blockchain privada orientada al mundo empresarial. Esta tecnología incluye una serie de blockchains con diferentes consensos posibles, sistemas de identidad y almacenamiento.

Hyperledger ofrece los siguientes frameworks:

- *Hyperledger Fabric:* Es el más conocido. Permite crear canales entre organizaciones y el despliegue de smart contracts en múltiples lenguajes. Permite escoger el consenso de la red y diferentes bases de datos, entre otras características.
- *Hyperledger Burrow:* Está basada en la red de Ethereum y permite el despliegue de smart contracts en Solidity.
- *Hyperledger Indy:* Se centra en el desarrollo de la identidad digital.
- *Hyperledger Iroha:* Se centra en el desarrollo de la identidad digital. Permite el despliegue de smart contracts en Java.
- *Hyperledger Sawtooth:* Se basa en Ethereum y permite el despliegue de smart contracts en Solidity. Con una sola máquina permite el procesamiento de miles de transacciones por segundo.
- *Hyperledger Grid:* Pensado para el control de cadenas de suministro.

Por otro lado, se encuentran una serie de herramientas:

- *Hyperledger Calliper:* Es una herramienta de benchmarking.
- *Hyperledger Cello:* Reduce el esfuerzo de la creación de nodos.
- *Hyperledger Explorer:* Proporciona una interfaz visual donde ver los datos de las transacciones y los bloques.
- *Hyperledger Quilt:* Ofrece interoperabilidad entre registros distribuidos de distintas redes mediante un protocolo denominado Interledger.
- *Hyperledger Ursa:* Es una librería criptográfica que evita repetición de desarrollos y aumenta la seguridad.
- *Hyperledger Composer:* Permite la creación de una red privada sin necesidad de conocer los detalles a bajo nivel. Es muy utilizada para desarrollo y pruebas de concepto, pero no es recomendable para su uso en proyectos de producción.

La tabla 1 muestra una comparativa entre Ethereum e Hyperledger Fabric. Como se puede observar, Ethereum soporta redes no permisionadas, utiliza solamente el algoritmo de consenso PoW, para implementar smart contracts ha de utilizarse el lenguaje Solidity y además es necesario disponer de Ethers para realizar transacciones. Por el contrario, Hyperledger es un framework que soporta redes permisionadas, utiliza numerosos mecanismos de consenso, soporta muchos lenguajes de programación a la hora de la creación de smart contracts y no precisa de la utilización de tokens. Por estas razones, consideramos que Hyperledger es más adecuado para el ámbito empresarial y para este proyecto.

Tabla 1: Comparativa Ethereum vs Hyperledger Fabric.

	Ethereum	**Hyperledger Fabric**
Uso	Propósito general	Transacciones B2B
Tipo de red	No permisionada	Permisionada
Algoritmos de consenso	PoW	PBFT u otros
Smart contracts	Solidity	Java, Node js, Go
Token	Ether	No es necesario

Capítulo 3. Marco teórico

Este capítulo se centra en explicar de manera teórica todos los conceptos que rodean a la tecnología blockchain para conseguir un mejor entendimiento de la implementación de la red y de la estructura de esta. El análisis realizado en los capítulos dos y tres ayudará a la elección de una tecnología para la implementación de la red. Podemos dividir esta parte del trabajo en cuatro subcategorías:

- Tecnología de registro distribuido: en esta sección se profundiza sobre los diferentes tipos de Distributed Ledger Technologies (DLTs) para situar en un contexto adecuado el tema principal de este trabajo, el blockchain.
- Tipos de redes blockchain: se explicarán los diferentes tipos de redes de blockchain, concepto muy importante para entender el tipo de red que es necesario implementar en el trabajo.
- Elementos de una red: posteriormente se enumeran los elementos que forman parte en una red de blockchain, así como otras nociones esenciales en el ámbito en el que nos movemos.
- Funcionamiento: finalmente se explica y se ilustra con un ejemplo sencillo el desempeño de esta tecnología.

A continuación, se procede a entrar en detalle sobre cada una de estas categorías.

3.1. ¿Qué es un registro distribuido?

Distributed ledger technology, también conocido como libro de contabilidad distribuido, es una tecnología que permite diseñar sistemas electrónicos e informáticos como una base de datos que contiene datos descentralizados y sincronizados, distribuidos geográficamente. Esta tecnología no requiere de una autoridad central para transmitir los registros a cada miembro.

Los sistemas DLT pueden ser públicos o privados, dependiendo de sus características. Blockchain es un tipo de DLT que se caracteriza por su seguridad y su inmutabilidad, y es altamente conocida por ser el primer tipo de DLT funcional, de ahí, que mucha gente utilice el término DLT y blockchain indistintamente, aunque como se ha comentado anteriormente, blockchain solamente es una de las subcategorías de los registros distribuidos.

Además de blockchain, como se puede observar en la Figura 1, existen numerosos tipos de DLT, los cuales se detallan a continuación [4]:

- ***Blockchain***

El blockchain es un tipo de DLT que se caracteriza por guardar sus transacciones en la forma de una cadena de bloques. Una red de blockchain está formada por un conjunto de datos inmutables y es segura gracias a la criptografía que une los bloques. La red está formada por una serie de datos: por un lado, la transacción y por otro, el bloque, que es un conjunto de datos protegidos con criptografía que contienen información codificada del bloque anterior.

Un bloque se agrega mediante varias etapas a la cadena de blockchain: en primer lugar, es necesario que se realice una transacción. Una vez la transacción se ha realizado, ha de ser verificada. Después de su verificación toda la información almacenada en la transacción empezará a formar parte del bloque y a este se le adjudica un ID único y un hash. Todo este proceso se comentará con más detalle posteriormente.

Entre los ejemplos más conocidos de blockchain se encuentran la mayor parte de las criptomonedas, como por ejemplo Bitcoin, Ethereum o Litecoin.

- ***HashGraph***

Es una red distribuida que gestiona múltiples transacciones con la misma marca de tiempo. En este tipo de DLT ningún nodo puede cambiar la información ni las transacciones. Una de las ventajas de esta red es el uso de unidades pequeñas de almacenamiento, porque no necesita almacenar información en la cadena para siempre.

- ***Directed Acyclic Graph (DAG)***

Otra de las opciones es el DAG, el cual tiene la posibilidad de ofrecer nano-transacciones. Además, la escalabilidad de la red mejora a medida que esta crece. Se diferencia del resto de métodos por los mecanismos de consenso que utiliza. Todos los nodos en la red validan y pueden iniciar transacciones. Sin embargo, para confirmar su propia transacción tienen que verificar al menos dos de las transacciones anteriores. Este tipo de red sería una buena opción para compañías que tienen que tratar con volúmenes masivos de datos.

- ***Holochain***

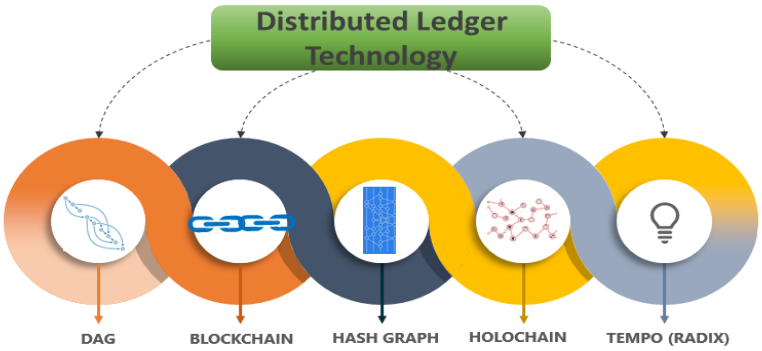
Este es uno de los sistemas más nuevos de DLT. Es una de las alternativas más avanzadas para desarrolladores que quieren encontrar nuevos enfoques para la creación de aplicaciones descentralizadas. La diferencia de este nuevo sistema radica principalmente en que su estructura se basa en un agente central. Evita mecanismos de consenso otorgando a cada agente su propio sistema de “forking”. En informática este concepto hace referencia a la creación de una copia de sí mismo por parte de un programa, que actúa como un "proceso hijo" del proceso originario, ahora llamado "padre" [42].

Holochain es una gran alternativa para aquellos casos de uso en los que el sistema necesite gran escalabilidad e integridad.

- ***Tempo (Radix)***

Tempo es una alternativa relativamente nueva que añade la posibilidad de añadir marcas de tiempo, así como otras funcionalidades. Uno de sus principales beneficios es que no será necesario ningún hardware añadido para la creación de aplicaciones descentralizadas, monedas o tokens.

Figura 1: Tipos de DLTs [40]



3.2. Tipos de redes de blockchain

Las cadenas de bloques se pueden diseñar de diversas maneras:

Figura 2: Permissionless ledger [3]

Permissionless blockchain

Estas redes son normalmente conocidas como redes públicas. En este tipo de cadenas de bloques, los usuarios no necesitan permisos para participar en la red. Cada nodo en la red tiene una copia total y actualizada de la cadena. Cada elemento nuevo que se quiere añadir a la red se comunica a todos los nodos. Estos validan conjuntamente el cambio a través de un algoritmo previamente consensuado. Si se acepta, el nuevo contenido se añade a todas las copias de la cadena para asegurar la consistencia de contenido en toda la red.

Los participantes de la red pueden unirse y salir según les convenga, sin necesidad de contar con una autoridad que regule dichas acciones. Lo único necesario es unirse a la red y añadir la transacción a la cadena.

Entre las ventajas de este tipo de redes se destaca su seguridad. Cualquier persona puede entrar en la red para comprobar la consistencia de los datos y las transacciones. Otra ventaja es la privacidad que aporta a los usuarios, los cuales no tienen que proporcionar ninguna documentación personal para unirse, permitiendo a los usuarios mantenerse en el anonimato. Además, como son abiertas, cualquiera puede usarlas sin necesidad de software adicional [6].

Como principales desventajas destaca su lentitud, dado que solamente se pueden validar una serie de transacciones por segundo. Aunque la mayoría de las redes son seguras contra hackeos, no se puede prevenir la entrada a la red de los cibercriminales. Una de las amenazas más importantes para estas redes es el ataque del 51% (ataque de blockchain -más comúnmente bitcoin- en el que un atacante se hace con más de la mitad de los hashes de la red).

Blockchains populares como Bitcoin, Ethereum, Litecoin, Dash y Monero entran en esta categoría.

Figura 3: Permissioned ledger [3]

Permissioned blockchain

También conocidas como consortium blockchain o federated blockchain. Son redes privadas donde los nodos del sistema necesitan permiso de una entidad central para acceder a la red y hacer cambios al ledger.

Cuentan con un dueño o administrador que preselecciona los miembros de la red, que la controla y establece una serie de reglas de comportamiento, permitiendo a un número determinado de usuarios la validación, mejorando así la eficiencia. Este tipo de redes son más adecuadas para el ámbito empresarial.

Como ventajas de este tipo de redes se puede destacar sus controles de acceso, una mayor eficiencia en comparación con las redes públicas, mejor escalabilidad y personalización. Por el contrario, como desventajas hay que tener en cuenta que la seguridad depende completamente de la integridad de sus miembros, es menos anónima y más vulnerable a ataques y hackeos [6].

Como ejemplos de estas redes destaca Hyperledger Fabric o Corda.

Hybrid

Una forma menos habitual de redes distribuidas es la híbrida, en la que se combinan los dos tipos mencionados anteriormente ofreciendo los beneficios de ambos.

Podemos observar una comparativa más detallada de los dos primeros tipos de redes en la Tabla 2 que se encuentra a continuación.

Tabla 2: Comparación permissioned y permissionless ledgers[2]

	Permissionless ledgers	**Permissioned ledgers**
Validaciones	Todo el mundo	Usuarios permitidos

[2] La tabla se ha elaborado con información encontrada en el artículo [6] de la bibliografía.

Nivel de descentralización	Alto	Limitado
Mecanismos de consenso	Normalmente PoW, PoS	PBTF, Paxos, Raft
Gobernada	Por la comunidad de usuarios	Miembros de la organización creadora
Seguridad	Alta. El alto número de validadores elimina puntos únicos de fallos	Depende de los controles de acceso a la red
Transparencia	Alta	Preferencias de la organización
Velocidad y escalabilidad	Baja. Muchos validadores tienen que llegar a un consenso	Alta. Pocos validadores permiten una mayor eficiencia y velocidad
Customización	Baja	Alta. La organización decide según sus necesidades
Privacidad de usuarios	Alta. No hay necesidad de confirmar sus identidades	Baja. Es necesaria la identificación
Mejor para	El público en general	Empresas, gobiernos, sector financiero

3.3. Elementos

A continuación, se procede a explicar los diferentes elementos que forman parte de una red de blockchain.

Transacciones [7]

La transacción está formada por un conjunto de datos que tienen una firma digital. Se componen de tres partes: el encabezado, las entradas y las salidas. El encabezado contiene el hash de la transacción, la versión de software con la que debe validarse el bloque, una fecha o altura del bloque y el número de entradas y salidas.

Bloque [7]

Un bloque en blockchain está formado en su mayoría por transacciones que han efectuado los usuarios y por una cabecera. Esta, está formada por seis elementos: la versión, el hash del bloque anterior, la raíz de merkle, una marca de tiempo, un objetivo y un nonce.

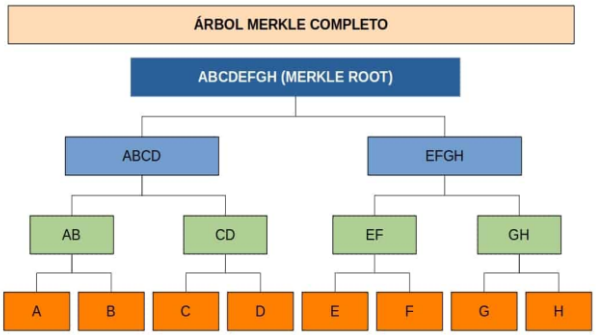
- *Versión:* Indica el nivel de desarrollo del software en el momento del minado del bloque.

- *Hash:* Es un algoritmo matemático que transforma un conjunto de datos en una secuencia de caracteres con una longitud fija que es independiente del tamaño de los datos de entrada. Para que un algoritmo de hash sea seguro debe ser resistente a colisiones.

- *Raíz de merkle:* Los hashes se ordenan en una estructura en forma de árbol que se denomina árbol de merkle o árbol de hash. La raíz de merkle representa la raíz del árbol a la que están unidos

todos los hashes y contiene un resumen de todos los valores hoja [8]. Se puede observar la estructura de un árbol de merkle en la Figura 4, incluida a continuación.

Figura 4: Ejemplo de un árbol de merkle [8]



- *Timestamp (marca de tiempo):* Indica el momento en el que fue minado un bloque.

- *Nonce:* Es un número aleatorio que es modificado por los mineros para averiguar el hash válido de cada bloque.

- *Objetivo (Target):* Es un número formado por una serie de bits (los mismos que el hash) que comienza por un determinado número de ceros y cuyo valor depende de la dificultad establecida por el sistema para el minado. Los mineros deben de encontrar un número igual o menor a ese objetivo para poder minar y crear un nuevo bloque. Por ello, cuanto más bajo es el objetivo, más difícil es la generación del bloque.

Este número se utiliza juntamente con el nonce. Si con un nonce determinado no se consigue generar un número más bajo o igual que el objetivo, entonces se aumenta el nonce y se comienza de nuevo.

Smart contracts [9]

El concepto de contrato inteligente fue ideado por el criptógrafo Nicholas Szabo, a quien se le ocurrió la idea de crear unos protocolos informáticos que, sustituyendo el papeleo legal, permitieran el comercio electrónico entre dos partes desconocidas. Hoy en día, este concepto hace referencia a un contrato que se escribe como un programa informático y se ejecuta por sí mismo sin la necesidad de involucrar a terceras partes. No solamente se trata de almacenar documentación, sino también de que los programas informáticos analicen y ejecuten partes de su lógica interna tomando la información como 'input' y procesándola según una serie de reglas.

Numerosas plataformas entre las que se encuentran Ethereum, Hyperledger o Corda, permiten la creación de smart contracts.

El funcionamiento de los contratos inteligentes se basa en sencillas sentencias "if/when … then…" escritas en código y ejecutadas en una red de blockchain. Una serie de ordenadores ejecutan las acciones incluidas en el smart contract cuando se han cumplido y verificado una serie de acciones. Los

participantes necesitan determinar cómo se representan las transacciones y los datos en la red de bloques, aceptar las reglas "if/when … then…", explorar todas las excepciones posibles y definir un marco de referencia que resuelva las posibles disputas.

En la Figura 5, que se muestra a continuación, se ilustra el funcionamiento de los contratos inteligentes.

Figura 5: Funcionamiento de los contratos inteligentes [9]

Los contratos inteligentes cuentan con numerosas ventajas entre las que destacan [11]:

- Rapidez, eficiencia y exactitud: Una vez el evento desencadenante ocurre, el contrato se ejecuta inmediatamente.
- Confianza y transparencia: No se necesita un intermediario y se utilizan datos cifrados.
- Seguridad: El uso de datos cifrados hace difícil la tarea de hackearlos. Además, como están conectados al siguiente y al anterior, sería necesario alterar la cadena entera para cambiar un dato concreto.
- Ahorro: Al no hacer uso de intermediarios, se ahorran costes y se evitan retrasos.

Cifrado

La criptografía es un proceso que permite que los datos solo sean visibles y procesables por las personas autorizadas a ello, siendo este su principal objetivo. Se basa en la transformación de la información para modificarla y hacerla incomprensible, es decir, en la transformación de texto en claro a texto cifrado.

Nodos [45]

En informática, un nodo es un punto de conexión donde se puede intercambiar información de todo tipo. En lo relativo al blockchain, si hablamos de nodos nos estamos refiriendo a una red de ordenadores en la que se ejecuta la cadena de bloques. Estos nodos están conectados gracias a una red peer-to-peer (P2P), funcionando todos de la misma manera. Existen numerosos tipos de nodos dependiendo de su objetivo y su funcionalidad.

Initial Coin Offering (ICO)

Las ICOs hacen referencia a una nueva forma de financiación empresarial en la que una empresa ofrece sus tokens a cambio de otras criptomonedas conocidas, como bitcoins o ethers. Estos tokens

pueden representar productos o servicios de la nueva empresa, o bien capital o deuda del nuevo proyecto de blockchain.

Token

Un token es "una unidad de valor que una organización crea para gobernar su modelo de negocio y dar más poder a sus usuarios para interactuar con sus productos, al tiempo que facilita la distribución y reparto de beneficios entre todos sus accionistas" (William Mougayar; 'The business blockchain', el nuevo término de la economía digital). "Un token servirá para aquello que una persona u organización decida" (Cristina Carrascosa).[3]

Dentro de la criptografía podemos encontrar tres tipos de token [13]:

- Security token: Vinculado a valores financieros intercambiables.
- Utility token: Permite a los usuarios acceso a los recursos de una empresa.
- Equity token: Representa el valor de un activo o empresa y se asocia con los security tokens.

El valor de los tokens fluctúa dependiendo de la oferta y la demanda del mercado.

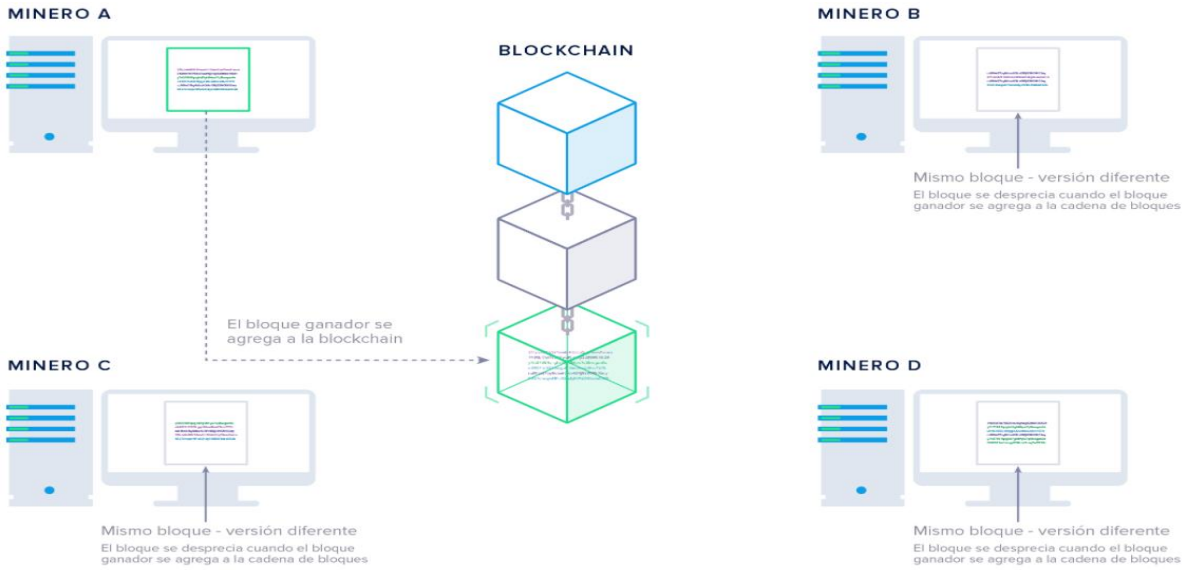
Minería [14]

Este término surgió de la mano del bitcoin. El proceso de minería consiste en la unión de nuevos bloques a la cadena mediante la resolución de un problema matemático basado en un mecanismo de consenso.

El minero es un tipo de nodo implicado en el proceso de minería que se encarga de validar las transacciones y crear nuevos bloques con ellas. Los nuevos bloques se minan tras un periodo de tiempo determinado. A partir del momento en el que un nuevo bloque minado pasa a formar parte de la cadena de bloques, comienza un nuevo intervalo de tiempo para que otro minero solucione el algoritmo escogido por la red de bloques. Por cada intervalo de tiempo, solamente un minero logrará solucionar el problema por lo que solamente se creará un bloque. Esta explicación se encuentra ilustrada en la Figura 6 que se encuentra a continuación.

[3] Ambas citas se han obtenido de la referencia [12] de la bibliografía.

Figura 6: Explicación minería [14]



Mecanismos de consenso

Es un protocolo que determina la forma en la que los nodos de la red distribuida validan las transacciones. También se utiliza para saber qué nodos deben realizar la acción de validación. Se procede a explicar algunos de los más importantes:

- *Proof of Work (PoW)* [15]

Es el primer algoritmo introducido. El principio de esta tecnología es resolver una serie de problemas matemáticos. La resolución de estos problemas requiere mucha energía. Este tipo de método tiene limitaciones ya que, si la red crece mucho, al necesitar mucha energía, se hace más difícil para los nodos la validación. Bitcoin utiliza este tipo de metodología.

- *Proof of Stake (PoS)* [15]

Para que un nodo pueda participar en la red, tiene que invertir. Cualquier persona con suficientes monedas en la red podrá validar una transacción. Esta alternativa es energéticamente eficiente y resuelve muchos problemas generados por PoW.

Aunque por lo comentado previamente parece mucho más beneficioso que PoW, tiene una importante desventaja y es que este sistema no permite la completa descentralización. Inevitablemente los individuos con el mayor número de monedas acabarán controlando la red.

- *Proof of Elapsed Time (PoET)* [15]

PoET es un algoritmo de consenso creado por Intel Corporation, que previene la utilización alta de recursos y de energía usando un sistema que proporciona a todos los nodos las mismas posibilidades de ganar. El algoritmo genera un periodo de tiempo de espera de manera aleatoria para cada nodo. Estos nodos deben dormirse durante ese periodo de tiempo. El nodo con el menor tiempo de espera se despertará primero y podrá minar un nuevo bloque a la cadena. El algoritmo debe asegurar que el

tiempo de espera es efectivamente aleatorio y que el ganador ha cumplido con el tiempo de espera establecido.

- *Practical Byzantine Fault Tolerance (PBFT)* [15]

PBFT asume que podría haber fallos en la red y que algunos nodos podrían funcionar mal. Se selecciona un nodo primario y el resto funciona como plan de respaldo. Destaca por su facilidad de implementación, pero no es recomendable su utilización en redes con grupos pequeños de nodos.

- *Proof of Authority (PoA)* (Bit2MeAcademy, s.f.) [17]

Proof-of-Authority es una nueva familia de algoritmos de consenso que ofrece alto rendimiento y tolerancia a fallos. El derecho a crear nuevos bloques se reserva a aquellos nodos que tienen autoridad para ello. Para ganar esa autoridad, los nodos deben pasar una autenticación preliminar. El validador debe revelar su identidad de manera voluntaria, consiguiendo así establecer las responsabilidades de cada uno de los actos que ocurren en la red. De esta manera se asegura que los validadores cuidarán del buen funcionamiento de las operaciones de la red.

Esta familia de algoritmos, a diferencia de PoW, no requiere que los nodos gasten recursos resolviendo algoritmos matemáticos complejos. El intervalo de tiempo en el que se generan nuevos nodos es fijo y la velocidad de las transacciones es alta. Es sencillo defender ataques de denegación de servicio y es muy complicado que se produzca un ataque del 51%, dado que en este mecanismo los atacantes necesitan hacerse con el 51% de los nodos, tarea que es mucho más difícil que obtener el 51% de la energía computacional que es lo que se requeriría para realizar este tipo de ataques en el mecanismo PoW. Por su centralización, es perfecto para redes que necesitan alta escalabilidad y seguridad.

La Tabla 3 muestra a modo de resumen una comparativa entre las distintas características de los algoritmos de consenso explicados previamente.

Tabla 3: Comparativa algoritmos de consenso[4]

	PoW	**PoS**	**PoET**	**BFT y otros**	**PoA**
Tipo de red	Permissionless	Ambos	Ambos	Permissioned	Ambos
Velocidad de transacciones	Baja	Alta	Media	Alta	Alta
Token	Sí	Sí	No	No	No
Coste	Sí, energía	Sí, inversión monetaria	No	No	No
Consumo energía	Alto	Más bajo que PoW	Medio	Medio	Bajo

[4] La tabla se ha elaborado con información del informe elaborado por Vishal Sharma y Niranjan Lal, "A novel comparison of consensus algorithms" que se encuentra en la referencia bibliográfica [18] y de los artículos [15], [16] y [17] sobre mecanismos de consenso.

Ataque 51%	Sí	No	No	No	No
Críticas	Mucha energía	Descentralización parcial	Grupos de nodos pequeños	Dependencia tecnología Intel	Centralización, Identidad conocida
Ejemplos	Bitcoin	Ethereum	Hyperledger Sawtooth	Hyperledger Fabric	Hyperledger

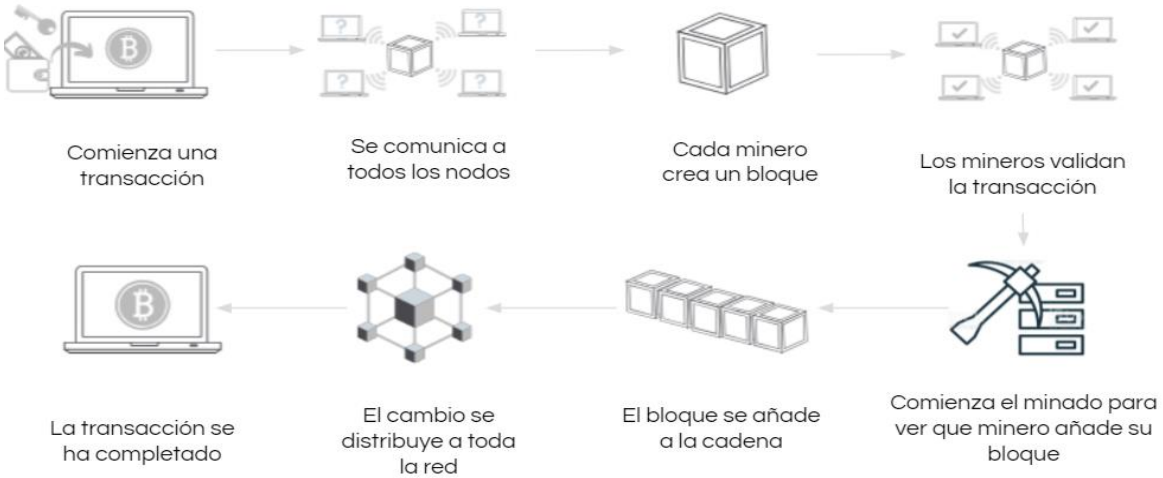
3.4. Funcionamiento

Como se ha comentado previamente, blockchain es una cadena de bloques que una vez añadidos a la red no se pueden modificar. Una vez comienza una transacción, la información contenida en ella se comunica a todos los usuarios de la red y se agrupa en un bloque.

Este bloque contiene una serie de elementos, entre ellos el hash del bloque anterior. Esta función resumen se utiliza para conseguir unir la cadena de bloques mediante un proceso denominado minería. Los mineros que tienen permisos para validar bloques compiten unos con otros para encontrar el hash válido que satisfaga el mecanismo de consenso utilizado en la red.

Tras la victoria de uno de los mineros, se comunica la operación a la totalidad de la red. El nuevo bloque se añade a la cadena de bloques y cada nodo pasará a conservar una copia de esta con la nueva operación ya incluida. Una vez se haya unido a la cadena, la transacción se valida y es oficialmente finalizada. Lo que se acaba de explicar se puede observar en la Figura 7, incluida a continuación.

Figura 7: Funcionamiento red de blockchain



Como ejemplo práctico, se va a explicar una transacción en la que Pepe quiere enviar a Juan 100€. Lo primero que hace Pepe es avisar a todo el mundo de la operación que quiere realizar. Al enviar este mensaje, el resto de los usuarios de la red comprueban si Pepe tiene en su cuenta bancaria esos 100€ que pretende enviar. Si dispone de dicha cantidad, el resto de los usuarios, en este caso llamémosles banqueros, anotan esta operación en su libreta. Esta operación se ha completado, pero todavía no forma parte de la cadena de forma definitiva.

Va pasando el tiempo y nuevos usuarios realizan operaciones, que los banqueros añaden a su libreta. Esta libreta tiene una capacidad limitada. Cuando esta ha llegado a su máxima capacidad, llega el momento de validar las operaciones; ahí es donde entra en juego el concepto de minado.

Los mineros, en este caso los banqueros mencionados previamente, compiten por validar las transacciones realizadas. El banquero que consiga realizarlo de la manera en la que está definido en la red (mecanismo de consenso) recibirá una recompensa. Después, comunicará al resto su victoria y la operación se añadirá de forma permanente a la red.

Capítulo 4. Desarrollo

Tras finalizar el marco teórico se está en posición de proceder al desarrollo del proyecto. Esta sección del trabajo se dividirá en tres bloques principales: aplicaciones de la tecnología DLT, arquitectura, y, por último, especificación del sistema.

- Aplicaciones de la tecnología: en esta sección se detalla la aplicación principal que tendrá la implantación de esta novedosa tecnología en una aplicación sobre gestión de pacientes.
- Arquitectura: se detalla en mayor medida la plataforma escogida, las características hardware necesarias para la implementación de la red, así como las herramientas auxiliares necesarias para realizar el proyecto.
- Especificación del sistema: en este apartado se explican en profundidad conceptos básicos del framework escogido además de entrar en detalles en la especificación del prototipo a desarrollar.

4.1. Utilidad del blockchain en el ámbito sanitario

El objetivo principal de este trabajo es demostrar la utilidad de la tecnología escogida, el blockchain, en un ámbito de la vida real. En este caso se ha escogido el sector sanitario desarrollando una aplicación que gestiona historiales de pacientes. Se considera un ámbito en el que la tecnología blockchain podría ser de gran utilidad debido a la cantidad de agentes implicados en la modificación y visualización de historiales de pacientes y además debido a la sensibilidad de los datos que se manejan.

Los historiales médicos de los pacientes se encuentran distribuidos a lo largo de todos los hospitales públicos del país. Debido a la cantidad de médicos implicados en la gestión de esta información podrían ocurrir pérdidas de información entre los cambios hechos por unos médicos u otros. Además, debido al tipo de información que se está tratando, si se llegase a sufrir un ataque o una modificación maliciosa de la información almacenada se pondría en peligro la vida de personas.

La idea es crear una página web conectada a una base de datos MySQL y a su vez a la red de blockchain creada. Cada vez que se cree, modifique o elimine un historial de un paciente se enviará a la red un identificador del historial del paciente y un hash que haga referencia a toda la información incluida en su historial. De esta forma se puede comprobar en todo momento que la información contenida en la cadena de bloques es la misma almacenada en la base de datos de historiales de pacientes. Por otro lado, sería posible comprobar la versión del historial, así como la persona que lo ha modificado, la fecha y la hora.

La totalidad de la información del historial del paciente se almacena en la base de datos, pero no en el blockchain ya que es información sensible sobre ciudadanos que se almacenaría de manera inmutable en la cadena, atentando contra la Ley Orgánica de Protección de Datos Personales y garantía de los derechos digitales. A continuación, se listan los casos de uso más importantes de la implementación.

Descripción Caso de Uso 1

Caso de uso 1	Crear
Identificador	CU_CREAR_HISTORIAL
Objetivo en contexto	Un usuario realiza una petición para almacenar el hash y el identificador de un historial recientemente creado en la cadena.
Actor principal	Usuario final (Personal sanitario)
Actores secundarios	Peers y orderers
Precondiciones	El usuario debe tener una identidad en el wallet y una cuenta de usuario en la aplicación web y el identificador del documento no debe existir.
Postcondiciones	○ Éxito: Se crea un historial en la base de datos asociada a la página web. Además, se genera una transacción asociada a ese movimiento que se añade a la cadena de bloques y a la base de datos de la cadena. ○ Fallo: Lanza un mensaje de error y no se realiza ninguna acción en la cadena ni de manera local.

Flujo principal

1. Se crea el historial de un paciente a través de una página web y se realiza la transacción en la base de datos asociada a la misma.
2. Se envía una petición con el identificador del historial recientemente creado y un hash sobre el contenido de un historial médico a la cadena de bloques a través de una API.
3. Se comprueba la identidad del usuario y su MSP y que el identificador del documento no exista previamente.
4. Los orderers una vez reciben la petición de la transacción, introducen esa transacción en un bloque y ejecutan el mecanismo de consenso aprobado por la red.
5. Cuando un nodo ha conseguido generar el hash válido se comunica al resto de nodos para que realicen la comprobación.
6. Los nodos peer reciben el bloque, lo validan y lo introducen en el ledger. Una vez se ha introducido en la cadena se comunica el cambio a todos los usuarios.
7. Se muestra el id de la transacción ejecutada.

Flujos secundarios

3a. Si las comprobaciones fallan, el usuario recibe un mensaje de error.

Descripción Caso de Uso 2

Caso de uso 2	Actualizar
Identificador	CU_ACTUALIZAR_HISTORIAL
Objetivo en contexto	Un usuario realiza una petición para almacenar el nuevo hash asociado al historial de un paciente tras la realización de cambios en este.
Actor principal	Usuario final (Personal sanitario)
Actores secundarios	Peers y orderers
Precondiciones	El usuario debe tener una identidad en el wallet y una cuenta de usuario en la aplicación web y el identificador del documento debe existir.
Postcondiciones	○ Éxito: Se edita un historial médico en la base de datos asociada a la página web. Se genera una transacción asociada a ese movimiento que se añade a la cadena de bloques, además se modifica el valor del documento del estado mundial. ○ Fallo: Lanza un mensaje de error y no se realiza ninguna acción en la cadena ni de manera local.

Flujo principal

1. Se edita el historial de un paciente a través de una página web y se realiza la transacción en la base de datos asociada a la misma.
2. Se envía una petición a la cadena de blockchain proporcionando un identificador del historial modificado y un hash con el nuevo contenido del historial médico a través de una API.
3. Se comprueba la identidad del usuario y su MSP y que el identificador del documento exista previamente.
4. Los orderers una vez reciben la petición de la transacción, introducen esa transacción en un bloque y ejecutan el mecanismo de consenso aprobado por la red.
5. Cuando un nodo ha conseguido generar el hash válido se comunica al resto de nodos para que realicen la comprobación.
6. Los nodos peer reciben el bloque, lo validan y lo introducen en el ledger. Una vez se ha introducido en la cadena se comunica el cambio a todos los usuarios.
7. Se muestra el id de la transacción ejecutada.

Flujos secundarios

3a. Si las comprobaciones fallan, el usuario recibe un mensaje de error.

Descripción Caso de Uso 3

Caso de uso 3	Borrar
Identificador	CU_BORRAR_HISTORIAL
Objetivo en contexto	Un usuario realiza una petición para borrar un historial médico.
Actor principal	Usuario final (Personal sanitario)
Actores secundarios	Peers y orderers
Precondiciones	El usuario debe tener una identidad en el wallet y una cuenta de usuario en la aplicación web y el identificador del historial debe existir.
Postcondiciones	○ Éxito: Se elimina el historial médico de la base de datos asociada a la página web. Se crea una transacción asociada a ese movimiento que se añade a la cadena de bloques. Además, se elimina el documento del estado mundial. ○ Fallo: Lanza un mensaje de error y no se realiza ninguna acción en la cadena ni en el estado mundial.

Flujo principal

1. Un médico elimina a uno de sus pacientes eliminando la información asociada al mismo de la base de datos médica.
2. Se proporciona el identificador del historial médico a borrar de la base de datos de la cadena.
3. Se comprueba la identidad del usuario y su MSP y que el identificador del documento exista previamente.
4. Los orderers una vez reciben la petición de la transacción, introducen esa transacción en un bloque y ejecutan el mecanismo de consenso aprobado por la red.
5. Cuando un nodo ha conseguido generar el hash válido se comunica al resto de nodos para que realicen la comprobación.
6. Los nodos peer reciben el bloque, lo validan y lo introducen en el ledger. Una vez se ha introducido en la cadena se comunica el cambio a todos los usuarios.
7. Se muestra el id de la transacción ejecutada.

Flujos secundarios

3a. Si las comprobaciones fallan, el usuario recibe un mensaje de error.

Descripción Caso de Uso 4

Caso de uso 4	Visualizar
Identificador	CU_VISUALIZAR_HISTORIAL
Objetivo en contexto	Se muestra información asociada a un historial médico. Se distinguen dos casos, la aplicación web y la cadena de bloques.
Actor principal	Usuario final (Personal sanitario)
Actores secundarios	CouchDB y MySQL
Precondiciones	El usuario debe tener una identidad en el wallet y una cuenta de usuario en la aplicación web y en caso de proporcionar un identificador, debe existir.
Postcondiciones	○ Éxito: Se realiza una consulta a la base de datos y se muestran los datos consultados. Si se realiza en la aplicación web se muestra información detallada sobre el paciente. Si por el contrario se realiza en el blockchain se mostrará solamente el identificador y el hash asociado al historial médico. ○ Fallo: Lanza un mensaje de error y no se realiza ninguna acción.

Flujo principal

1. Realizar consulta a la base de datos MySQL.
    a. Consulta de un historial médico concreto, realizando una búsqueda del número de la seguridad social de un paciente.
    b. Consulta de todos los historiales médicos asociados a un doctor.
2. Realizar consulta a la base de datos de la red de blockchain (CouchDB).
    a. Consulta de un historial médico concreto, proporcionando el identificador del historial que se quiere ver.
    b. Consulta de todos los historiales médicos asociados a un doctor.
    c. Consulta de la historia de todas las transacciones asociadas a un historial concreto (fecha de realización, usuario, id de la transacción y datos del historial), proporcionando su identificador.
3. Se comprueba que el identificador del documento exista previamente, en el caso de que se haya proporcionado uno.
4. La base de datos devuelve los datos consultados.

Flujos secundarios

3a. Si las comprobaciones fallan, el usuario recibe un mensaje de error.

Descripción Caso de Uso 5

Caso de uso 5	Comprobar hash
Identificador	CU_COMPROBAR_HASH
Objetivo en contexto	Al mostrar el historial de un paciente en la página web se comprueba en tiempo real que su hash coincide con el hash asociado a este guardado en la cadena de bloques.
Actor principal	Usuario final (Personal sanitario)
Actores secundarios	CouchDB y MySQL
Precondiciones	El usuario debe tener una identidad en el wallet y cuenta en la aplicación web y el historial médico debe existir.
Postcondiciones	○ Éxito: Se muestra en la aplicación web que el historial médico almacenado es el mismo que el de la red de blockchain a través de un mensaje. ○ Fallo: Se muestra un mensaje de error en el que se indica al médico del peligro que puede suponer que no coincida la información de la base de datos con la almacenada en la cadena.

Flujo principal

1. Una vez se accede en la aplicación web al historial de un usuario se realiza una consulta a la base de datos del blockchain utilizando el identificador del historial médico proporcionado.
2. Se comprueba si el hash proporcionado por parámetros es el mismo que se encuentra en la cadena.
3. Se muestra un mensaje en la página web.

4.2. Arquitectura

4.2.1. Hyperledger Fabric

La plataforma escogida para implementar la red de blockchain es Hyperledger Fabric. Hyperledger Fabric es un framework modular de código abierto diseñado por IBM que sirve como base para la creación de productos, soluciones y aplicaciones basadas en redes de bloques usando componentes plug-and-play diseñados para ser usados por negocios privados [41].

En esta plataforma la red debe ser permisionada y los participantes deben ser identificables. Está diseñada para aquellas entidades que quieren reservar el acceso a la información solamente a usuarios autorizados otorgándolas un mayor control y mayor seguridad sobre los datos. Entre sus características destaca su arquitectura altamente modular. Soporta Smart Contracts en numerosos lenguajes de programación como pueden ser Java, Nodejs o Go. Utiliza diferentes mecanismos de consenso y no utiliza criptomonedas, aunque es posible su implementación. Al no utilizar criptomonedas se reduce el riesgo relativo a ataques dado que no se asocia un valor monetario a cada transacción.

Se basa en una arquitectura para las transacciones llamada "execute-order-validate". *Execute* ejecuta una transacción y comprueba que es correcta, posteriormente la añade. *Order* ordena las transacciones por medio de un protocolo de consenso y por último *Validate* valida las transacciones contra una política antes de incluirlas en la cadena. Esta separación ofrece numerosas ventajas entre las que destaca la necesidad de bajos niveles de confianza y verificación, una mejora de la escalabilidad de la red y un rendimiento superior [28].

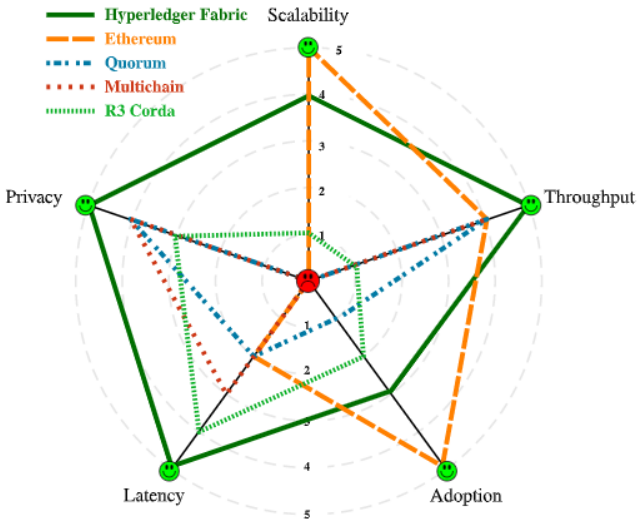
La privacidad es la razón por la cual las empresas deciden utilizar blockchains privados en primera instancia. Como novedosa característica que la diferencia del resto de plataformas incluye la creación de canales dentro de la red. Estos canales tienen una lista de identidades autorizadas. De esta forma, se puede crear una red de blockchain en la que los usuarios que estén autorizados a unirse no tengan permiso para unirse a algún canal. Por otro lado, se pueden especificar permisos distintos para los canales en los que un usuario de la red es miembro. De esta forma se consiguen aislar los datos de los usuarios que no deben conocerlos, aumentando la privacidad.

Según un estudio realizado en 2021 por el instituto coreano de comunicaciones y ciencias de la información[5], Hyperledger es la plataforma más prometedora y útil en comparación con otros frameworks para la creación de redes de blockchain.

Esto se puede observar en la Figura 8. Esta figura compara los frameworks Hyperledger Fabric, Ethereum, Quorum, Multichain y Corda en las áreas de privacidad, latencia, uso de las plataformas, escalabilidad y rendimiento. Hyperledger destaca en privacidad y latencia, con una media de los resultados obtenidos de 4.3. El resto de frameworks obtienen medias de 3.3, 2.2, 2 y 2.2 respectivamente.

[5] Datos obtenidos del paper titulado "Permissioned blockchain frameworks in the industry: A comparison" escrito por Julien Polge, Jérémy Robert e Yves Le Traon. Más información en la referencia bibliográfica 33.

Figura 8: Comparación frameworks [33]



4.2.2. Hardware

Para la implementación de la red de blockchain se pretendía usar una máquina virtual que ofrece Google Cloud con la instalación de Hyperledger Fabric y Composer incluida. Se ha decidido no utilizar dicha solución debido a que la versión de Fabric proporcionada es muy antigua y además Hyperledger Composer está obsoleto desde agosto de 2019. La intención de usar dicha herramienta era la reducción del tiempo de desarrollo, además de facilitar la implementación del código y la red, pero se llegó a la conclusión de que las desventajas eran mucho mayores que las ventajas proporcionadas.

Al no disponer de ninguna máquina virtual, se ha creado en Compute Engine una máquina virtual con características muy similares a la máquina proporcionada por Google Cloud con ambos frameworks ya instalados. Esta máquina cuenta con 2 CPUs virtuales, 8GB de memoria, un disco persistente de 20 GB y sistema operativo Ubuntu 20.04 LTS.

4.2.3. Herramientas auxiliares[6]

El primer paso para poder utilizar Hyperledger Fabric es instalar en la máquina virtual las herramientas auxiliares que requiere este framework para funcionar.

- Primero es necesario instalar la versión más reciente de Git.
- Como segundo paso, es necesario instalar la versión más nueva de cURL.
- El tercer paso será instalar la herramienta Docker, utilizada para desplegar aplicaciones dentro de contenedores.
- El cuarto paso consiste en instalar Node.js, lenguaje de programación que se utilizará para el despliegue del Chaincode y de los scripts que contienen las transacciones.
- Por último, es necesario instalar la versión 2.7 de Python.

[6] En la bibliografía se incluye una guía más detallada del proceso de instalación de todas las herramientas necesarias. [29] y [30].

Por otro lado, se han utilizado una serie de aplicaciones y herramientas adicionales para apoyar y facilitar el desarrollo del trabajo.

VSCode: Es necesario el uso de un editor de texto para la parte de desarrollo de la prueba de concepto.

Postman: Se ha instalado la herramienta Postman que se utilizará para realizar peticiones a una API que se comunique con la red de blockchain.

GitHub: Se cuenta con el uso de esta plataforma para alojar el proyecto y para tener almacenado todo el código.

El proyecto parte del código proporcionado por Hyperledger Fabric en la página de GitHub https://github.com/hyperledger/. Para facilitar la implementación de la prueba de concepto, el primer paso es clonar los ficheros de ejemplo que se encuentran en la página https://github.com/hyperledger/fabric-samples.

Para la prueba de concepto, dentro de los ejemplos que proporciona la carpeta fabric-samples, se ha decidido partir del código de "auction-simple" por su detallada explicación. Se trata de una red donde interactúan compradores y vendedores en una subasta. A continuación, se procederá a detallar el esquema que sigue la red, acompañándolo de conceptos básicos de la aplicación de Hyperledger Fabric, necesarios para la posterior implementación.

4.3. Especificación

En esta sección se detallan los conceptos clave de Hyperledger Fabric acompañados de una figura que representa la definición de la red a implementar. Los diferentes componentes de la red son los siguientes [31]:

- Ledger (L)
- Autoridades certificadoras (CA)
- Organización (O)
- Nodo peer (P)
- Servicio de ordenación (OS)
- Smart contract (S)
- Aplicación (A)
- Canal (C)
- Configuración de red (NC)

Se puede encontrar una representación gráfica de la red donde se ilustran la mayoría de los elementos nombrados previamente en la Figura 9, que se muestra a continuación.

Elementos de la red

A continuación, se procede a enumerar todos los elementos que forman la red de blockchain creada, entre los que se encuentran:

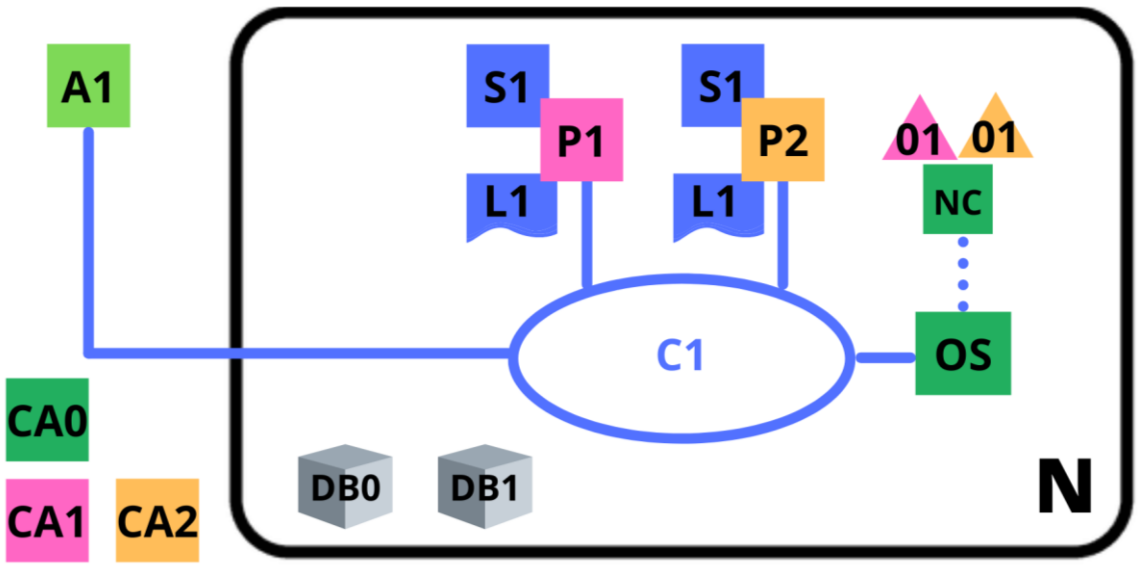
- Canal al que se unirán las organizaciones, denominado mychannel.
- Un chaincode donde se implementa la lógica de las transacciones, denominado auction.
- Dos organizaciones, cada una con un nodo peer y una autoridad certificadora.
  - Org1
    - peer0.org1.example.com
    - ca_org1
  - Org2
    - peer0.org2.example.com
    - ca_org2
- Un servicio de ordenación con un solo nodo Raft y una CA.
  - orderer.example.com
  - ca_orderer
- Dos bases de datos couchDB.
  - couchdb0
  - couchdb1

Ledger (L) [31]

En Fabric un ledger está formado por dos partes relacionadas entre sí. Primero por un *estado mundial* que es una base de datos que contiene los valores actuales de un conjunto de estados del ledger. Esta base de datos hace más sencillo acceder a un valor concreto de la red, en vez de calcularlo recorriendo todo el log de transacciones. Estos estados se expresan como pares clave-valor.

Por otro lado, el ledger está formado por *la cadena de bloques*, es decir, un log de transacciones que graba todos los cambios que han ocurrido en el estado mundial. Las transacciones se guardan en el interior de bloques que se introducen en la cadena permitiendo entender el histórico de todos los cambios ocurridos en el estado mundial. A diferencia del otro componente, la cadena de bloques es inmutable. Como se puede observar en la Figura 9 cada peer guarda una copia del ledger.

Figura 9: Red de blockchain para PoC [44]

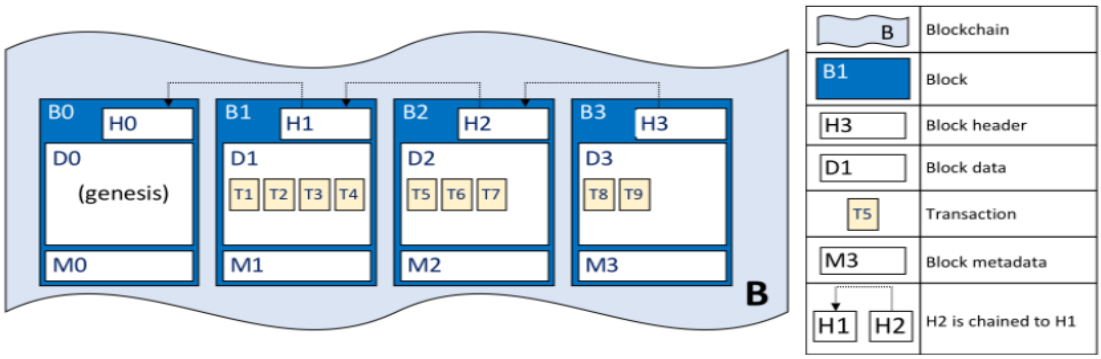


El *estado mundial* es implementado, como se ha comentado previamente, como una base de datos. Como opciones de bases de datos se encuentran LevelDB y CouchDB. La primera es la base de datos por defecto y es apropiada cuando los estados del ledger son simples pares clave-valor. CouchDB es apropiado cuando los estados del ledger son estructuras JSON porque esta base de datos soporta consultas más complejas. Para la implementación se utilizará esta segunda opción.

La *cadena* se basa en una serie de bloques unidos entre sí, como si se tratara de una lista enlazada. El primero de los bloques se denomina génesis y hace referencia al comienzo del ledger. Este bloque no contiene ni usuarios ni transacciones. Todos estos bloques contienen una cabecera, unos datos y unos metadatos.

La Figura 10, incluida a continuación, muestra la estructura y elementos de una cadena de bloques.

Figura 10: Representación de la cadena de bloques en Hyperledger Fabric [31]



En la ***cabecera*** del bloque se encuentran por un lado el número del bloque que es un entero que comienza en 0 haciendo referencia al bloque génesis, y que se incrementa en una unidad cuando un nuevo bloque se añade a la red. Incluye a su vez el hash del bloque actual y del previo. La sección de ***datos*** contiene una lista de todas las transacciones realizadas. Por último, la sección de ***metadatos*** contiene el certificado y la firma del creador del bloque que es usada para verificar el bloque por los nodos de la red.

Autoridad certificadora (CA) [46]

Una autoridad certificadora se encarga de emitir certificados a diferentes actores. Estos certificados son firmados digitalmente por ella y vinculan al actor con su clave pública. Por ello, si una persona confía en una CA, puede confiar que una persona está unida a la clave pública incluida en el certificado.

Fabric otorga un componente que permite crear CAs en la red de blockchain. Este componente, denominado Fabric CA, es un proveedor CA raíz privado que permite manejar identidades digitales en forma de certificados. Estas entidades juegan un papel clave en la red porque dispensan los certificados X.509. Las CAs se pueden usar también para firmar transacciones. El mapeo de los certificados a las organizaciones se consigue por una estructura denominada Proveedor de Servicios de Membresía (MSP), el cual define una organización creando un MSP que está atado a un certificado CA raíz para identificar los componentes e identidades creados por la CA raíz.

En la implementación se utilizará una CA por cada organización, CA1 y CA2, pertenecientes a la Org1 y Org2, respectivamente, y una tercera que pertenece al orderer CA0.

Organización (O) [47]

Son los miembros que se unen a la red de blockchain. En esta red hay dos organizaciones. La organización Org1 es la iniciadora de la red y se comunica con Org2 a través del canal C1.

Peer (P) [47]

Este tipo de nodos son un elemento fundamental de la red ya que almacenan los ledgers y los contratos inteligentes. Estos nodos guardan la red y validan las transacciones antes de incluirlas en el ledger, además, se encargan de ejecutar los smart contracts que se usan para manejar la lógica de la red. Un nodo se vincula a una organización.

En la Figura 9 previa existen dos nodos P1 y P2. Ambos nodos se han unido al canal C1, por el que se comunican, y están vinculados a la Org1 y Org2 respectivamente.

Servicio de ordenación (OS) [48]

El servicio de ordenación está formado por nodos llamados *orderers* que se encargan de escoger el orden de las transacciones después de recibir la petición por parte de los clientes y posteriormente las añaden a un bloque. Más adelante, estos bloques se distribuyen a los nodos peer que se encargan de añadirlos al ledger. Hyperledger Fabric ofrece tres servicios de ordenación diferentes: Raft, Kafka y Solo.

- Raft es un servicio de ordenación que se basa en un mecanismo de consenso tolerante a fallos (CFT). Raft sigue un modelo de 'líder y seguidor', donde el nodo líder es elegido y sus decisiones se replican en todos los seguidores. Este es el servicio recomendado dado que es más fácil de implementar y los otros dos se encuentran obsoletos para esta versión de Fabric.
- Kafka es similar al servicio de Raft dado que es un servicio CFT y usa el modelo 'líder y seguidor'. Además de ser más difícil de implementar, este servicio no está diseñado para ser utilizado en redes amplias.
- Solo es un servicio cuya implementación está recomendada únicamente para pruebas y consiste en un único nodo orderer. Está obsoleto y es posible que se elimine en próximas actualizaciones.

En esta prueba de concepto se utiliza simplemente un nodo Raft, pero para entornos de producción es necesario implementar como mínimo tres o más nodos de este tipo.

Chaincode y smart contracts (S) [49]

Como se ha definido previamente, los contratos inteligentes son unos documentos que contienen una serie de reglas que definen la lógica del modelo de negocio. En Fabric, estos contratos inteligentes se denominan Chaincodes. Cuando se despliega un chaincode en una red, todos los contratos inteligentes a los que hace referencia se convierten en disponibles para toda la red. Todos los contratos inteligentes tienen una política de ordenación asociada, que determina qué organizaciones en la red deben firmar una transacción generada por un contrato inteligente para convertirla en válida.

Para la prueba de concepto se van a implementar las siguientes funcionalidades que servirán para realizar acciones con un documento y para comprobar su estado:

- **enrollAdmin.js**
- **registerEnrollUser.js**
- **createDocument.js**
- **updateDocument.js**
- **deleteDocument.js**
- **queryDocument.js**
- **queryAllDocuments.js**
- **queryDocumentHistory.js**
- **hashCheck.js**

Aplicación (A)

La aplicación se encarga de interactuar con la red de blockchain emitiendo transacciones al ledger o realizando consultas. En este caso solamente existe una aplicación A1 a través de la cual se van a realizar transacciones en el canal C1.

Para poder realizar este proceso es necesario seguir los siguientes pasos:

1. Seleccionar una identidad del wallet.
2. Conectarse a una gateway.
3. Acceder a la red deseada.
4. Realizar una petición de transacción a un smart contract, en este caso a auction.
5. Emitir la transacción a la red.
6. Procesar la respuesta.

Estos pasos se explicarán posteriormente con más detalle con relación al código implementado.

Canal (C) [47]

Hyperledger Fabric se diferencia del resto de plataformas de blockchain porque permite la creación de canales. Estos canales permiten aislar la información entre los miembros que pertenecen a dicho canal. De esta forma los peers P1 y P2 tienen acceso a la información que se comparte en el canal C1, pero si se uniera una tercera organización a la red los nodos pertenecientes a la misma no tendrían acceso a dicha información hasta que se les permitiera unirse al canal.

Identidad [46]

Todos los elementos que forman parte de una red (peers, orderers, aplicaciones clientes, administradores, etc.), contienen una identidad digital encapsulada en un certificado digital X.509.

Estas identidades son realmente importantes porque determinan los permisos exactos que tienen los actores de la red sobre recursos e información. Para que sea verificable una identidad debe haber sido emitida por una autoridad de confianza.

Infraestructura de clave pública (PKI) [46]

Una PKI está formada por cuatro elementos: certificados digitales, claves públicas y privadas, entidades certificadoras y listas de revocación de certificados. La red de blockchain de Hyperledger Fabric utiliza una infraestructura de clave pública para asegurar una comunicación segura entre los miembros participantes en la red y además para asegurarse de que los mensajes publicados en la cadena de bloques estén adecuadamente autenticados. Se procederá a detallar en pocas líneas en qué consisten los cuatro básicos de la infraestructura de clave pública.

- *Certificado digital:* Un certificado digital es un documento que contiene una serie de atributos relacionados con el dueño del certificado, entre ellos, su clave pública. El certificado más común es el emitido por el estándar X.509.
- *Autenticación, claves públicas y privadas*: El concepto de autenticación y de integridad son conceptos importantes en comunicaciones seguras. Autenticación implica que las partes que intercambian un mensaje están seguras de la identidad que creó un determinado mensaje. Integridad significa que un mensaje no se ha modificado durante su transmisión. Mecanismos tradicionales de autenticación utilizan las firmas digitales para, como su nombre indica, firmar los mensajes. Para poder generar las firmas digitales, cada parte envuelta en la comunicación debe tener dos elementos, una clave pública y una privada. Estas claves se pueden utilizar para verificar la integridad y autenticidad de los mensajes envueltos en una comunicación.
- *Autoridades certificadoras (CAs)*: Explicado previamente.
- *Listas de revocación de certificados (CRL):* Una CRL es simplemente una lista de referencias a certificados que una CA ha revocado, y que, por tanto, ya no son válidos. Cuando una tercera parte quiere verificar una identidad, primero comprueba la CRL de la CA para asegurarse de que el certificado no ha sido revocado.

Proveedor de servicios de membresía (MSP) [50]

En las redes permisionadas, los participantes necesitan una forma de probar su identidad al resto de miembros de la red para poder realizar transacciones. Como se ha comentado previamente las CAs emiten identidades generando una clave pública y otra privada que forman una clave-valor que se puede usar para probar identidades. Como la clave privada no se puede compartir se necesita un mecanismo que permita probar esa identidad, ahí es donde el MSP entra en juego.

La implementación de un MSP consiste en un conjunto de carpetas que se añaden a la configuración de la red y son usadas para definir a la organización de puertas para adentro (para decidir quién es el administrador) y de puertas para afuera (permitiendo a otras organizaciones validar las entidades que tienen autoridad para hacer lo que quieren hacer). Un MSP contiene una lista de todas las identidades permitidas.

Políticas [47]

Son un conjunto de reglas que definen cómo se toman las decisiones en una red y cómo se llega a outcomes específicos. Las políticas son una de las características que hacen a Hyperledger Fabric distinto de otras blockchains como Bitcoin o Ethereum. En esas redes las transacciones pueden ser validadas por cualquier nodo en la red. Al ser Fabric una red permisionada, se necesita del uso de estas políticas para decidir qué organizaciones pueden acceder y actualizar la red. Las políticas contienen una lista de las organizaciones que tienen acceso a un recurso determinado. Además, especifican cuántas organizaciones hacen falta para confirmar una propuesta de actualización de un activo.

Transacciones

Una transacción captura los cambios en el estado mundial. Están formadas por una serie de campos, aunque los más importantes son por un lado la cabecera, que contiene metadatos esenciales sobre la transacción como podría ser el nombre del chaincode y su versión. Por otro lado, contiene la firma digital, usada para comprobar que los detalles de la transacción no se han manipulado. Contiene

además una propuesta, una respuesta y aprobaciones (endorsements) que hacen referencia a una lista de respuestas de cada organización.

Wallet

El wallet es una cartera que contiene un conjunto de identidades. Una aplicación selecciona una de esas identidades cuando se conecta a un canal. Cuando se permite a un nuevo usuario acceder al sistema se le añade en esta cartera un documento de texto que verifica su identidad.

Capítulo 5. Implementación

En esta parte de la memoria se precisan los detalles del código para facilitar su entendimiento. Se explican por secciones las modificaciones del chaincode y de los scripts de las transacciones. Se incluye una explicación detallada de la generación del entorno y de los pasos seguidos para la creación de la API que se utiliza para realizar llamadas a la red. Por otro lado, se explica la estructura de la aplicación web y de la base de datos utilizada. Se incluye un apartado de pruebas de ejecución para asegurar el correcto funcionamiento del sistema.

Objetivos

La tarea se divide en tres objetivos principales:

- Primero es necesario alterar el chaincode que se encuentra en la carpeta "fabric-samples/auction-simple/chaincode-go/smart-contract/auction.go" de forma que pasen a guardar documentos en vez de subastas. Además de eso, es necesario añadir todas las funciones necesarias para crear y mostrar los documentos mencionados.
- El segundo objetivo se basa en añadir los ficheros necesarios para ejecutar transacciones en el ledger a la carpeta "fabric-samples/auction-simple/application-javascript". Estos ficheros están escritos en Node js y hacen llamadas al chaincode implementado en el lenguaje de programación Golang.
- Creación de una API que permita la interacción desde cualquier SaaS con la red creada.

5.1. Creación del entorno

El primer paso de la prueba de concepto consiste en navegar hasta la carpeta "fabric-samples/test-network" y ejecutar el script network.sh que levanta una red de Fabric usando imágenes de Docker.

Primero, es necesario ejecutar el siguiente comando para eliminar cualquier contenedor Docker o material criptográfico de anteriores ejecuciones:

./network.sh down

Después de haber eliminado los rastros de ejecuciones anteriores es necesario levantar la red. Por simplicidad se ejecutará un comando más complejo que ejecuta varios pasos al mismo tiempo, que se detallan a continuación.

./network.sh up createChannel -ca -s couchdb

El primer paso para levantar la red es generar el material criptográfico tanto de la organización 1, como de la organización 2, como del servicio de ordenación, que define a las organizaciones de la red. Por defecto, la creación de dicho material se realiza mediante la herramienta cryptogen, por razones de efectividad se decide utilizar la Fabric CA para la creación de este material, para ello es necesario añadir el flag -ca, como se ve en el comando previo. Este script utiliza Docker Compose para levantar

tres CAs, una por cada organización y otra para el ordering service. Esto se puede observar en las Figuras 11, 12 y 13, incluidas a continuación.

Figura 11: Generación del entorno (I)

```
Stopping network
Stopping cli                      ... done
Stopping peer0.org1.example.com ... done
Stopping peer0.org2.example.com ... done
Stopping couchdb0                 ... done
Stopping couchdb1                 ... done
Stopping orderer.example.com      ... done
Stopping ca_orderer               ... done
Stopping ca_org2                  ... done
Stopping ca_org1                  ... done
Removing cli                      ... done
Removing peer0.org1.example.com ... done
Removing peer0.org2.example.com ... done
```

[…]

Figura 12: Generación del entorno (II)

```
Generating certificates using Fabric CA
Creating network "fabric_test" with the default driver
Creating ca_orderer ... done
Creating ca_org2    ... done
Creating ca_org1    ... done
Creating Org1 Identities
Enrolling the CA admin
+ fabric-ca-client enroll -u https://admin:adminpw@localhost:7054 --caname ca-org1 --
samples/test-network/organizations/fabric-ca/org1/tls-cert.pem
2021/11/21 11:47:56 [INFO] Created a default configuration file at /home/patricia_lla
ns/peerOrganizations/org1.example.com/fabric-ca-client-config.yaml
2021/11/21 11:47:56 [INFO] TLS Enabled
2021/11/21 11:47:56 [INFO] generating key: &{A:ecdsa S:256}
2021/11/21 11:47:56 [INFO] encoded CSR
2021/11/21 11:47:56 [INFO] Stored client certificate at /home/patricia_llamas/go/src/
nizations/org1.example.com/msp/signcerts/cert.pem
2021/11/21 11:47:56 [INFO] Stored root CA certificate at /home/patricia_llamas/go/src
anizations/org1.example.com/msp/cacerts/localhost-7054-ca-org1.pem
2021/11/21 11:47:56 [INFO] Stored Issuer public key at /home/patricia_llamas/go/src/g
izations/org1.example.com/msp/IssuerPublicKey
2021/11/21 11:47:56 [INFO] Stored Issuer revocation public key at /home/patricia_llam
s/peerOrganizations/org1.example.com/msp/IssuerRevocationPublicKey
Registering peer0
+ fabric-ca-client register --caname ca-org1 --id.name peer0 --id.secret peer0pw --id
07489/fabric-samples/test-network/organizations/fabric-ca/org1/tls-cert.pem
2021/11/21 11:47:56 [INFO] Configuration file location: /home/patricia_llamas/go/src/
nizations/org1.example.com/fabric-ca-client-config.yaml
2021/11/21 11:47:56 [INFO] TLS Enabled
2021/11/21 11:47:56 [INFO] TLS Enabled
Password: peer0pw
```

[…]

Figura 13: Generación del entorno (III)

```
Generating the peer0 msp
+ fabric-ca-client enroll -u https://peer0:peer0pw@localhost:7054 --caname ca-org1 -M /
network/organizations/peerOrganizations/org1.example.com/peers/peer0.org1.example.com/m
_llamas/go/src/github.com/74507489/fabric-samples/test-network/organizations/fabric-ca/
2021/11/21 11:47:57 [INFO] TLS Enabled
2021/11/21 11:47:57 [INFO] generating key: &{A:ecdsa S:256}
2021/11/21 11:47:57 [INFO] encoded CSR
2021/11/21 11:47:57 [INFO] Stored client certificate at /home/patricia_llamas/go/src/gi
nizations/org1.example.com/peers/peer0.org1.example.com/msp/signcerts/cert.pem
2021/11/21 11:47:57 [INFO] Stored root CA certificate at /home/patricia_llamas/go/src/g
anizations/org1.example.com/peers/peer0.org1.example.com/msp/cacerts/localhost-7054-ca-
2021/11/21 11:47:57 [INFO] Stored Issuer public key at /home/patricia_llamas/go/src/git
izations/org1.example.com/peers/peer0.org1.example.com/msp/IssuerPublicKey
2021/11/21 11:47:57 [INFO] Stored Issuer revocation public key at /home/patricia_llamas
s/peerOrganizations/org1.example.com/peers/peer0.org1.example.com/msp/IssuerRevocationP
Generating the peer0-tls certificates
+ fabric-ca-client enroll -u https://peer0:peer0pw@localhost:7054 --caname ca-org1 -M /
network/organizations/peerOrganizations/org1.example.com/peers/peer0.org1.example.com/t
.hosts localhost --tls.certfiles /home/patricia_llamas/go/src/github.com/74507489/fabri
2021/11/21 11:47:57 [INFO] TLS Enabled
2021/11/21 11:47:57 [INFO] generating key: &{A:ecdsa S:256}
2021/11/21 11:47:57 [INFO] encoded CSR
2021/11/21 11:47:57 [INFO] Stored client certificate at /home/patricia_llamas/go/src/gi
nizations/org1.example.com/peers/peer0.org1.example.com/tls/signcerts/cert.pem
2021/11/21 11:47:57 [INFO] Stored TLS root CA certificate at /home/patricia_llamas/go/s
rOrganizations/org1.example.com/peers/peer0.org1.example.com/tls/tlscacerts/tls-localho
2021/11/21 11:47:57 [INFO] Stored Issuer public key at /home/patricia_llamas/go/src/git
```

[…]

Posteriormente, se generan los contenedores con Docker para los elementos de la red, como muestra la Figura 14.

Figura 14: Generación del entorno (IV)

```
CONTAINER ID   IMAGE                               COMMAND                  CREATED          STATUS
                                                                                            NAMES
e9ad86fcf5db   hyperledger/fabric-tools:latest     "/bin/bash"              1 second ago     Up Less t
                                                                                            cli
c2fcde6c2c7a   hyperledger/fabric-peer:latest      "peer node start"        3 seconds ago    Up 1 seco
tcp, 7051/tcp, 0.0.0.0:19051->19051/tcp, :::19051->19051/tcp                                peer0.
0ab9ea2c2452   hyperledger/fabric-peer:latest      "peer node start"        3 seconds ago    Up 1 seco
tcp, 0.0.0.0:17051->17051/tcp, :::17051->17051/tcp                                          peer0.
a4d1beae775d   couchdb:3.1.1                       "tini -- /docker-ent…"   5 seconds ago    Up 3 seco
/tcp, :::5984->5984/tcp                                                                     couchd
a0bed36485ef   hyperledger/fabric-orderer:latest   "orderer"                5 seconds ago    Up 2 seco
tcp, 0.0.0.0:7053->7053/tcp, :::7053->7053/tcp, 0.0.0.0:17050->17050/tcp, :::17050->17050/tcp   ordere
bd6a7854faa2   couchdb:3.1.1                       "tini -- /docker-ent…"   5 seconds ago    Up 3 seco
/tcp, :::7984->5984/tcp                                                                     couchd
9d57beb17a83   hyperledger/fabric-ca:latest        "sh -c 'fabric-ca-se…"   11 seconds ago   Up 9 seco
tcp, 7054/tcp, 0.0.0.0:18054->18054/tcp, :::18054->18054/tcp                                ca_org
667e5ad524f8   hyperledger/fabric-ca:latest        "sh -c 'fabric-ca-se…"   11 seconds ago   Up 9 seco
tcp, 0.0.0.0:17054->17054/tcp, :::17054->17054/tcp                                          ca_org
8fdd574c5bac   hyperledger/fabric-ca:latest        "sh -c 'fabric-ca-se…"   11 seconds ago   Up 9 seco
tcp, 7054/tcp, 0.0.0.0:19054->19054/tcp, :::19054->19054/tcp                                ca_ord
```

Después, el script utiliza la herramienta configtxgen para crear el bloque génesis como muestra la Figura 15. Este primer bloque de la cadena necesita una creación especial al no contener el hash del bloque anterior.

Figura 15: Generación del entorno (V)

```
Generating channel genesis block 'mychannel.block'
/home/patricia_llamas/go/src/github.com/74507489/fabric-samples/test-network/../b
+ configtxgen -profile TwoOrgsApplicationGenesis -outputBlock ./channel-artifacts
2021-11-21 11:48:05.317 UTC [common.tools.configtxgen] main -> INFO 001 Loading c
2021-11-21 11:48:05.328 UTC [common.tools.configtxgen.localconfig] completeInitia
2021-11-21 11:48:05.328 UTC [common.tools.configtxgen.localconfig] completeInitia
```

[…]

Una vez se ha creado el material criptográfico y el bloque génesis se pueden levantar los peers y el servicio de ordenación.

Una vez que los nodos se encuentran corriendo en la máquina, es necesaria la creación de un canal para permitir comunicaciones privadas entre miembros específicos de la red. createChannel permite la creación de un canal, con nombre mychannel, que permite comunicaciones entre la organización 1 y la organización 2. La Figura 16 muestra la creación exitosa de este canal.

Se podría especificar el nombre de otro canal con el flag -c seguido del nombre escogido. Se escogen los anchor peers, que permiten a la Org1 y Org2 tener permisos de MSP y decidir quién entra en el canal.

Figura 16: Generación del entorno (VI)

```
Creating channel mychannel
Using organization 1
+ osnadmin channel join --channelID mychannel --config-block ./channel-artifacts/
ithub.com/74507489/fabric-samples/test-network/organizations/ordererOrganizations
-cert.pem --client-cert /home/patricia_llamas/go/src/github.com/74507489/fabric-s
s/orderer.example.com/tls/server.crt --client-key /home/patricia_llamas/go/src/gi
ations/example.com/orderers/orderer.example.com/tls/server.key
+ res=0
Status: 201
{
        "name": "mychannel",
        "url": "/participation/v1/channels/mychannel",
        "consensusRelation": "consenter",
        "status": "active",
        "height": 1
}

Channel 'mychannel' created
```

El comando docker ps -a permite observar todos los nodos y organizaciones creadas y los contenedores asociados a los mismos.

El siguiente paso es desplegar el chaincode en el canal, para que sea aceptado por todos los usuarios de este. Para ello hay que ejecutar el siguiente comando que permite a cualquier miembro del canal crear documentos.

./network.sh deployCC -ccn auction -ccp ../auction-simple/chaincode-go/ -ccl go -ccep "OR('Org1MSP.peer','Org2MSP.peer')"

Este último paso completa la generación del entorno de la prueba de concepto.

5.2. Implementación del chaincode

El siguiente paso es introducir una serie de modificaciones en el fichero “auction.go” que se encuentra en la carpeta “auction-simple/chaincode-go/smart-contract”.

El chaincode comienza con la definición de la estructura de los historiales de pacientes proporcionados por la página web. Se ha optado por un contenido muy sencillo de los documentos por temas de espacio y eficiencia, además debido a la información almacenada en los historiales de los pacientes se decide proporcionar únicamente el hash por la imposibilidad de guardar esos datos de manera permanente. Esta estructura contiene simplemente un id y un hash que hace referencia a la información de un historial médico, como se muestra en la Figura 17.

Figura 17: Estructura de un historial de un paciente

```
type Document struct {
        Id string `json:"Id"`
        Hash string `json:"Hash"`
}
```

Además de eso, el chaincode contiene numerosas funciones que son llamadas desde los scripts de transacciones.

- **DocumentExists**
  - Información: Se comprueba si existe el id del historial médico que se ha introducido como parámetro.
  - Parámetros de entrada: El identificador del historial a comprobar.
  - Parámetros de salida: En caso de error devuelve un valor booleano a falso y un mensaje de error.

  La Figura 18 muestra el código que comprueba si existe o no un historial médico con un determinado historial en la cadena de bloques.

Figura 18: Función DocumentExists

```
exists, err := s.DocumentExists(ctx, documentID)
if err != nil {
return "", err
}
if exists {
return "", fmt.Errorf("the document %s already exists", documentID)
}
```

- **CreateDocument**
  - Información: Esta función crea un registro que contiene información relativa a un historial médico y emite la transacción al estado mundial. Para ello, comprueba si el historial ya existía previamente, si ya existe ese identificador devuelve un mensaje de error, y si no ha habido ningún problema crea el registro.
  - Parámetros de entrada: Se introducen todos los datos necesarios para la creación del registro del historial médico, es decir un string equivalente a un identificador y el hash del contenido del historial del paciente.
  - Parámetros de salida: Devuelve el identificador de la transacción o un mensaje de error en caso de ocurrir algún problema.

    En la Figura 19 se ilustra el código de la función que se utiliza para crear un historial médico de un paciente, utilizando un identificador y un hash.

Figura 19: Función CreateDocument

```
document := Document{
    Id: documentID,
    Hash:     data,
}

ejemploJSON, err := json.Marshal(document)
if err != nil {
    return "", err
}

txID := ctx.GetStub().GetTxID()

// put document into state
err = ctx.GetStub().PutState(documentID,ejemploJSON)
if err != nil {
    return "", fmt.Errorf("failed to put document in public data: %v", err)
}

return txID, nil
```

- **UpdateDocument**
  - Información: Si se modifica el contenido de un historial médico en la aplicación web se cambia en la red de bloques el registro asociado al mismo, dado que al cambiar la información el hash ya no es el mismo. Para que la operación se realice con éxito comprueba que el documento a actualizar existe. Si no hay fallo, se modifican los datos y se emite la transacción.
  - Parámetros de entrada: El identificador del documento a actualizar y el nuevo contenido.
  - Parámetros de salida: Si no hay ningún problema devuelve el identificador de la transacción, si no, un mensaje de error.

La Figura 20 muestra parte del código de la función que se encarga de actualizar el hash de un historial médico de un paciente en la cadena tras realizar cambios en el mismo.

Figura 20: Función UpdateDocument

```
document, err := s.QueryDocumentByKey(ctx,documentID)
document.Hash = newHash

ejemploJSON, err := json.Marshal(document)
if err != nil {
    return "", err
}
```

- **DeleteDocument**
  - Información: Si se elimina el historial de un paciente en la página web, se llama a esta función que elimina el registro asociado a ese historial de la base de datos de la cadena.
  - Parámetros de entrada: El identificador del documento a eliminar.
  - Parámetros de salida: Devuelve un mensaje de error si no se ha conseguido eliminar el documento y si funciona correctamente devuelve el id de la transacción.

La Figura 21, mostrada a continuación, muestra cómo se elimina un historial de un paciente del estado mundial.

Figura 21: Función DeleteDocument

```
// delete document from state
err = ctx.GetStub().DelState(documentID)
if err != nil {
    return "", fmt.Errorf("failed to delete document of public data: %v", err)
}
```

- **QueryAllDocuments**
  - Información: Obtiene del estado mundial todos los registros de los historiales.
  - Parámetros de entrada: Ninguno.
  - Parámetros de salida: Un array con todos los registros existentes en la cadena o un mensaje de error en el caso de que no consiga devolverlos.

La Figura 22 muestra el código de la función que se encarga de devolver todos los historiales médicos de pacientes que se encuentren almacenados en la cadena de bloques.

Figura 22: Función QueryAllDocuments

```
var documents []*Document

for resultsIterator.HasNext() {
    queryResponse, err := resultsIterator.Next()
    if err != nil {
      return nil, err
    }

    var document Document
    err = json.Unmarshal(queryResponse.Value, &document)
    if err != nil {
      return nil, err
    }
    documents = append(documents, &document)
}
```

- **QueryDocumentByKey**
  - Información: Devuelve el historial médico que contiene el identificador proporcionado por parámetro, si existe o si no hay ningún error.
  - Parámetros de entrada: El identificador del historial del paciente a consultar.
  - Parámetros de salida: Devuelve un mensaje de error si no ha conseguido obtener el historial o si este no existe. En caso de encontrar dicho historial, lo devuelve.

- **QueryDocumentHistory**
  - Información: Devuelve la información de todas las transacciones que se han realizado sobre un registro de un historial concreto. Para ello hace uso de la API GetHistoryForKey.
  - Parámetros de entrada: El identificador del historial.
  - Parámetros de salida: Un array que para cada transacción realizada sobre un historial devuelve el id de la transacción, la marca de tiempo, los datos del historial y un booleano que indica si el historial del paciente ha sido eliminado.

La Figura 23, que se incluye a continuación, muestra el código que se utiliza para devolver información de todas las versiones de un historial médico.

Figura 23: Función QueryDocumentHistory

```
var document Document
if len(response.Value) > 0{
    err = json.Unmarshal(response.Value, &document)
    if err != nil {
        return nil, err
    }
} else {
    document = Document {
        Id: documentID,
    }
}

timestamp, err := ptypes.Timestamp(response.Timestamp)
if err != nil {
    return nil, err
}

result := QueryResult{
    TxId:      response.TxId,
    Timestamp: timestamp,
    Record:    &document,
    IsDelete: response.IsDelete,
}
results = append(results, result)
```

- **HashCheck**
  - Información: Comprueba que el hash de un historial médico coincide con el almacenado en la cadena de bloques.
  - Parámetros de entrada: El identificador del historial a comprobar y el hash.
  - Parámetros de salida: Un booleano que indica si el hash es el mismo o no.

La Figura 24 muestra la comprobación del hash de un historial médico guardado en la cadena con el hash que se proporciona como parámetro.

Figura 24: Función HashCheck

```
if document.Hash != hash {
    return false, fmt.Errorf("The hash %s does not match the one kept on the ledger", hash)
}
```

5.3. Implementación de los scripts de las transacciones

Se procede a dar una explicación más detallada de los scripts desde los que se llama al chaincode para realizar las transacciones deseadas. En esta parte del trabajo se han creado seis ficheros en Node js y se han reutilizado dos de los que otorga el código de ejemplo.

- **enrollAdmin**
  - Información: Se crea un administrador para la organización proporcionada por parámetro, para ello se pide al MSP los permisos necesarios. Se introduce en el wallet un fichero que contiene la clave privada y el certificado público del administrador.
  - Parámetro de entrada: Organización de la que se quiere crear el administrador.

- **registerEnrollUser**
  - Información: Se crea un usuario para la organización proporcionada por parámetro, para ello se pide al MSP los permisos necesarios, en este caso de usuario. Se introduce en el wallet un fichero que contiene la clave privada y el certificado público del usuario.
  - Parámetro de entrada: Organización de la que se quiere crear el usuario y el nombre del usuario a crear.

- **createDocument**
  - Información: Con los datos de entrada que componen un registro de un historial (hash e identificador) se realiza la petición de introducir un nuevo documento a la cadena. Posteriormente se llama a la función de queryDocument, que muestra el documento que se acaba de crear.
  - Parámetro de entrada: Se necesita proporcionar la organización y el usuario que crea el documento y los datos que componen el mismo, es decir, un id y el contenido.

    Se incluye una imagen del código de la llamada a esta función en la Figura 25, incluida a continuación. Todas ellas funcionan de manera muy similar.

Figura 25: Función createDocument

```
let statefulTxn = contract.createTransaction('CreateDocument');
let txID = statefulTxn.getTransactionId();

console.log('\n--> Submit Transaction: Create a new document');
await statefulTxn.submit(documentId, data);
console.log('*** Result: committed');
console.log('*** Result ***SAVE THIS VALUE*** TransactionID: ' + txID.toString());
```

- **updateDocument**
  - Información: Con los datos de entrada que componen un registro de un historial (hash e identificador) se realiza la petición de modificar el hash asociado a un historial que ya se encuentra en la cadena. Posteriormente se llama a la función de queryDocument, que muestra el documento que se acaba de editar.
  - Parámetro de entrada: Se necesita proporcionar la organización y el usuario que edita el documento. Después es necesario proporcionar el id del historial que se ha editado y su nuevo contenido.

- **deleteDocument**
  - Información: Se realiza la petición de eliminar un registro de un historial que ya se encuentra en la cadena.

- Parámetro de entrada: Se necesita proporcionar la organización y el usuario que elimina el historial además del id del historial que se desea borrar.

- **queryDocument**
  - Información: Se muestra la información del documento que se introduce por parámetro.
  - Parámetro de entrada: Se necesita proporcionar la organización y el usuario que quiere ver el documento. Después es necesario proporcionar el id del documento a mostrar.

- **queryAllDocuments**
  - Información: Se muestra la información del documento que se introduce por parámetro.
  - Parámetro de entrada: Se necesita proporcionar la organización y el usuario que quiere ver el documento. Después es necesario proporcionar el identificador de la transacción del documento a mostrar.

- **queryDocumentHistory**
  - Información: Se muestra la información de todas las transacciones realizadas sobre un documento.
  - Parámetro de entrada: Se necesita proporcionar la organización y el usuario que quiere realizar la consulta. Después es necesario proporcionar el identificador del documento sobre el que se quiere mostrar el historial de transacciones.

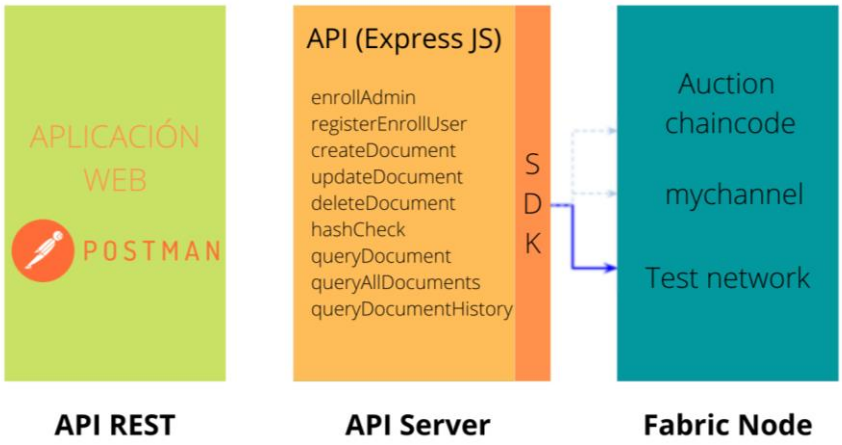
- **hashChecks**
  - Información: Se comprueba que el hash de un historial que se encuentra en la base de datos de la aplicación es el mismo que se encuentra en esos momentos en la cadena.
  - Parámetro de entrada: Se necesita proporcionar la organización y el usuario que quiere comprobar la integridad de los historiales. Por otro lado, es necesario enviar el hash del documento almacenado en la base de datos de la aplicación para compararlo con el de la cadena.

5.4. Creación API

Aunque Hyperledger Fabric proporciona un Software Development Kit, se ha decidido crear una API REST para poder interactuar de manera sencilla con la red.

Para ello se utiliza el entorno de trabajo Express.js para poder interactuar con los scripts creados en Node.js.

Figura 26: Esquema funcionamiento API



La Figura 26, incluida previamente, muestra la unión de la API con la red de blockchain y con cualquier servicio desde el que se quieran hacer peticiones. En este caso se utilizará la aplicación Postman y una aplicación web.

El primer paso consiste en crear en VSCode una carpeta con un fichero denominado app.js donde se definen las peticiones que se van a realizar. Es necesario ejecutar desde la ruta donde se encuentra la carpeta el comando npm init, que crea un archivo package.json con las especificaciones de la API.

Posteriormente, es necesario interactuar con la máquina virtual que se ha creado en Compute Engine. Una vez se ha iniciado la máquina es necesario descargar el código que se ha subido a GitHub e instalar Express. Después, en la carpeta de la API es necesario instalar las dependencias de la API.

La Figura 27 ilustra cómo se han instalado de manera correcta las dependencias de la API, mediante el comando *npm install.*

Figura 27: Instalación dependencias API

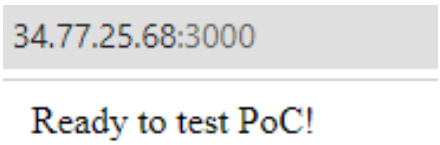
```
audited 58 packages in 1.046s

2 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
```

Ahora se puede escuchar por el puerto 3000, como ilustra la Figura 28, mediante *npm run start*.

Figura 28: API escuchando por el puerto 3000



Además de los pasos previos, es necesario agregar en GCloud una regla de firewall que permita a la máquina escuchar por dicho puerto, para ello es necesario ejecutar el siguiente comando:

gcloud compute firewall-rules create default-allow-http-3000 --allow tcp:3000 --source-ranges 0.0.0.0/0 --target-tags http-server --description "Allow port 3000 access to http-server"

Una vez se encuentra la API funcionando, se comienzan a realizar las modificaciones pertinentes en el fichero app.js para poder realizar peticiones a la red. Como ejemplo a las modificaciones realizadas se muestra la llamada a la función de crear un documento, ilustrado en la Figura 29 que se encuentra a continuación. El resto se realizan de manera bastante similar.

Figura 29. Función de la API createDocument

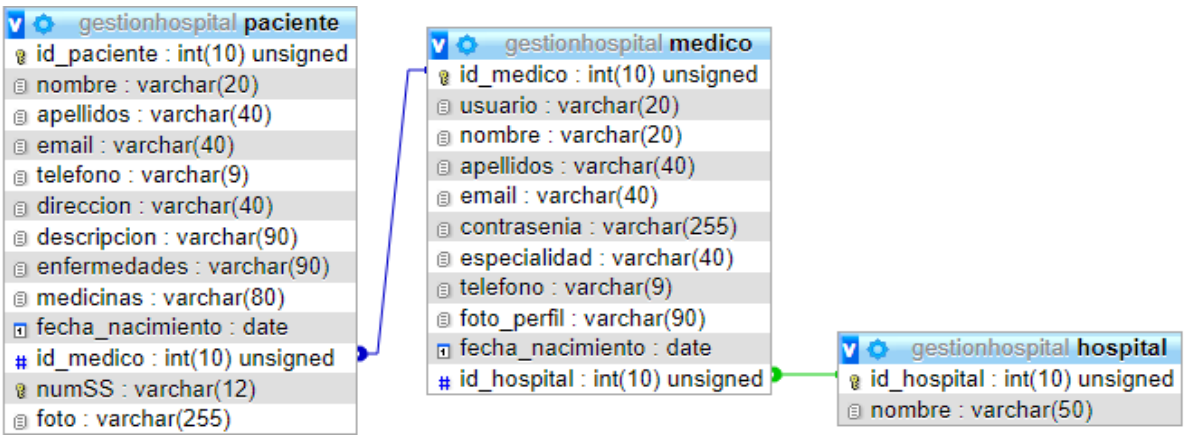
```
app.post('/create/:orgId/:clientId/:documentId/:hash', async (req, res) => {
    const org_Id = req.params.orgId;
    const client_Id = req.params.clientId;
    const document_Id = req.params.documentId;
    const hash = req.params.hash;
    let result;
    try {
        result = await create.createDocument(org_Id, client_Id, document_Id, hash);
    } catch (err) {
        console.log(`Error in creating document ${document_Id}: ${err}`);
        res.status(500).send(err);
    }

    res.status(200).send(result);
})
```

5.5. Creación de la página web de gestión de pacientes

Como último paso se ha creado una página web que gestiona historiales de pacientes para aplicar a un caso real la utilidad de las características del blockchain. Esta página web está conectada a una base de datos con la estructura mostrada en la Figura 30.

Figura 30: Esquema de la base de datos de la aplicación web



A continuación, se procede a detallar la funcionalidad de la página web:

- *Creación de un perfil asociado a un doctor.* Cada doctor perteneciente a un hospital que utilice la aplicación de gestión de historiales debe registrarse en la página web. Cada vez que se realice está acción se creará una identidad asociada a cada doctor en el blockchain. Se puede acceder al perfil de cada doctor en el que se muestra el hospital al que pertenece, su especialidad y otra información personal relevante.
- *Creación de un historial de un paciente.* Un doctor puede crear un historial médico por cada paciente que acuda a consulta. El doctor debe rellenar un formulario con la información personal de cada paciente, los problemas médicos que tiene el paciente y las medicinas que le recete. Cada vez que se realice esta acción se creará un registro asociado a cada historial médico de un paciente en el que se realizará una petición a la cadena de bloques enviando el identificador del paciente y un hash que hace referencia a toda la información contenida en el historial médico del paciente.
- *Visualización de todos los pacientes de un médico.* Se muestra una lista con todos los pacientes que tiene un médico. Se puede acceder a la información detallada de un historial de un paciente desde la lista pulsando en la fotografía del paciente. También se puede acceder a un paciente determinado desde la barra de búsqueda proporcionando el número de la seguridad social del paciente.
- *Visualización del historial de un paciente. Se* accede a los historiales de las dos formas comentadas previamente. En la página se observa la información personal y médica del paciente. Se muestra a su vez el número de versión del historial médico actual y un mensaje que informa al médico de si la información que se le muestra coincide con la almacenada en el blockchain. Así se evita que el doctor realice ediciones sobre una versión que no es la final y se consigue que preste especial interés por si se ha modificado el historial de manera maliciosa. Para dicha comprobación se envía un hash que hace referencia a la información del historial del paciente contenida en la base de datos y se comprueba con el hash del registro almacenado en el blockchain.
- *Edición del historial de un paciente.* Se pueden modificar los datos asociados al historial médico. Por cada modificación se calcula un nuevo hash que se envía a la red de blockchain actualizando así el registro del historial.
- *Eliminación de un paciente.* Se le permite al doctor eliminar el historial de un paciente. Una vez se realiza esta acción de la base de datos local se realiza la consiguiente eliminación del registro asociado a dicho paciente del estado mundial de la cadena de bloques.

Se realizan peticiones a la red de blockchain en el momento en el que se registra un doctor, cuando se crea un historial médico de un paciente, para editar y eliminar un historial médico, para saber la versión de un historial y para comprobar que el hash del historial médico del paciente almacenado en la base de datos unida a la página web es el mismo que el almacenado en la red de bloques. Se llama a la cadena de bloques de la forma que se detalla a continuación. Se incluye en la Figura 31 el código de una de las peticiones comentadas dado que todas siguen la misma estructura.

Figura 31: Petición desde la aplicación web a la cadena de bloques

```
$hash=self::hashPaciente($paciente);
$url="http://34.77.25.68:3000/update/org1/1/$paciente->id_paciente/$hash";

$opciones= array(
    "http" => array(
        "header" => "Content-Type: application/x-www-form-urlencoded\r\n".
        "User-Agent:MyAgent/1.0\r\n",
        "method" => "POST",
    ),
);

$contexto = stream_context_create($opciones);

$result= file_get_contents($url,false,$contexto);

if ($result==false){
    echo "Error en la peticion";
    exit;
}
```

5.6. Pruebas

A continuación, se procede a realizar una serie de pruebas sobre la red por un lado desde línea de comandos y desde la aplicación de gestión de historiales de pacientes.

Para poder interactuar con la red desde línea de comandos es necesario cambiar al directorio auction-simple/application-javascript e instalar las dependencias de la aplicación mediante *npm install*.

Por otro lado, para realizar peticiones desde la API es necesario situarse en el directorio auction-simple/api y ejecutar el comando *npm run start* para escuchar por el puerto 3000.

- **Unión del administrador de cada organización a la red**

Para poder realizar transacciones es necesario registrar y unir identidades a la aplicación. Para poder interactuar con la aplicación es necesario introducir administradores de la organización con la que se va a trabajar. Aunque haya posibilidad de unir dos organizaciones solamente se harán peticiones desde la Org1. Para ello es necesario ejecutar el comando *node enrollAdmin.js org1*.

Cada organización en la aplicación de gestión de pacientes hará referencia a un hospital, en este caso el Hospital de la Princesa de Madrid. Cada vez que se una un nuevo hospital a la aplicación de gestión de pacientes será necesario incluirlo como nueva organización en la red de blockchain. Para ello se utiliza el script enrollAdmin.js. Esto se puede observar en la Figura 32 incluida a continuación:

Figura 32: Unión de una organización a la red de blockchain

```
--> Enrolling the Org1 CA admin
Loaded the network configuration located at /home/patricia_llamas/go/src/gith
ub.com/74507489/fabric-samples/test-network/organizations/peerOrganizations/o
rg1.example.com/connection-org1.json
Built a CA Client named ca-org1
Built a file system wallet at /home/patricia_llamas/go/src/github.com/7450748
9/fabric-samples/auction-simple/application-javascript/wallet/org1
Successfully enrolled admin user and imported it into the wallet
```

Si se accede al directorio /wallet se puede observar que se han creado los ficheros que contienen las credenciales de esos administradores, pudiendo observar en el fichero “admin.id” el certificado del usuario y la clave privada. Esto se puede observar en la Figura 33.

Figura 33: Certificado de usuario y clave privada de la organización 1


```
{"credentials":{"certificate":"-----BEGIN CERTIFICATE-----\nMIIB8zCCAZmgAwIBA
gIUKEts4aVOaVJZaSfsoONloP4RTX8wCgYIKoZIzj0EAwIw\ncDELMAkGA1UEBhMCVVMxFzAVBgNV
BAgTDk5vcnRoIENhcm9saW5hMQ8wDQYDVQQH\nEwZEdXJoYW0xGTAXBgNVBAoTEG9yZzEuZXhhbXB
sZS5jb20xHDAaBgNVBAMTE2Nh\nLm9yZzEuZXhhbXBsZS5jb20wHhcNMjEwODI3MDgwMDAwWhcNMj
IwODI3MDgwOTAw\nWjAhMQ8wDQYDVQQLEwZjbGllbnQxDjAMBgNVBAMTBWFkbWluMFkwEwYHKoZIz
j0C\nAQYIKoZIzj0DAQcDQgAEVrmAohM5BkUIaW1jS7MOH1r+fLWxPkbnEbI6lRSiFFsb\nWUfzKN
rsaTNUS8P7zhMXBjl1tFQ9rpek8VfQPc8ADKNgMF4wDgYDVR0PAQH/BAQD\nAgeAMAwGA1UdEwEB/
wQCMAAwHQYDVR0OBBYEFJTDuUb82qdleUk6h9yql7TWoz2M\nMB8GA1UdIwQYMBaAFJPQHQdqo6yu
ooaF21ymInJgqR6vMAoGCCqGSM49BAMCA0gA\nMEUCIQCNv8dSqEZsxziWMK124n5AZaMHllvTQHC
shVLGFjPVcwIgJYBs7pYt3K/Z\nzNYJPA23h/pBI95REkr1LiuhPmfMccM=\n-----END CERTIFI
CATE-----\n","privateKey":"-----BEGIN PRIVATE KEY-----\r\nMIGHAgEAMBMGByqGSM4
9AgEGCCqGSM49AwEHBG0wawIBAQQgMA6pxlWIUkAnD02m\r\nJq1WQAX0HmVNzE8XT33cFgGMOx6h
RANCAARWuYCiEzkGRQhpbWNLsw4fWv58tbE+\r\nRucRsjqVFKIUWxtZR/Mo2uxpM1RLw/vOExcGO
XW0VD2ul6TxV9A9zwAM\r\n-----END PRIVATE KEY-----\r\n"},"mspId":"Org1MSP","typ
e":"X.509","version":1}
```

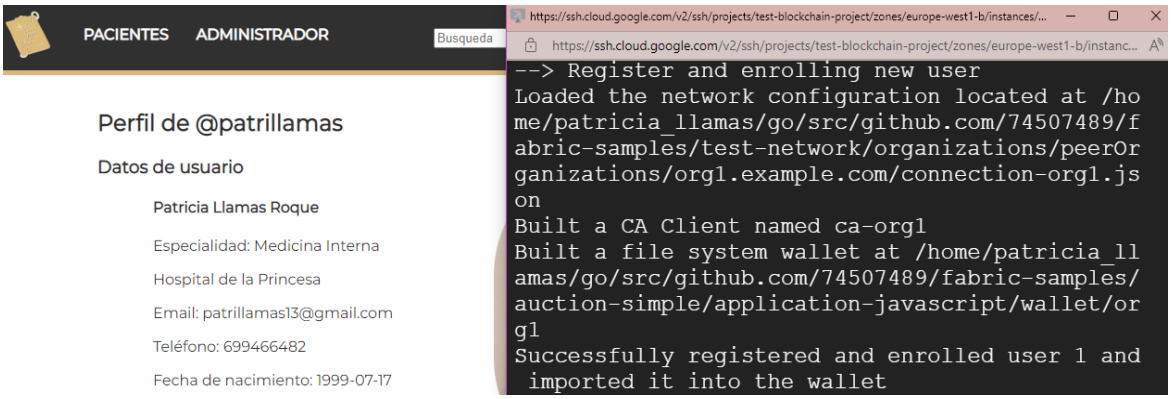

- **Creación de usuarios**

Gracias al script registerEnrrollUser.js se pueden introducir usuarios para cada organización. Para ello se debe utilizar el comando *node registerEnrollUser.js org1 client1.* En la aplicación web por cada doctor que se registre se realizará una llamada a esta función creando una transacción en la red de blockchain, como se puede observar en las Figuras 34 y 35.

Figura 34: Unión de un doctor a la red de blockchain

```
--> Register and enrolling new user
Loaded the network configuration located at /home/patricia_llamas/go/src/gith
ub.com/74507489/fabric-samples/test-network/organizations/peerOrganizations/o
rg1.example.com/connection-org1.json
Built a CA Client named ca-org1
Built a file system wallet at /home/patricia_llamas/go/src/github.com/7450748
9/fabric-samples/auction-simple/application-javascript/wallet/org1
Successfully registered and enrolled user client1 and imported it into the wa
llet
```

Figura 35: Unión de un doctor desde la aplicación web



Si se accede de nuevo al wallet se pueden observar las credenciales de cada cliente en la carpeta de la organización a la que pertenecen. En este caso encontramos la identidad del cliente uno en la carpeta la carpeta /wallet/org1.

Ahora se procede a realizar una serie de acciones para interactuar con la cadena y comprobar su funcionamiento.

- **Creación de un registro de un historial médico**

Ahora se procede a crear una serie de registros para realizar acciones posteriormente con ellos. Cada vez que un médico da de alta a un paciente se llamará a esta función a la que se le enviará el identificador del paciente y el hash que se obtiene de la totalidad de datos del historial médico. La Figura 36 ilustra lo ocurrido.

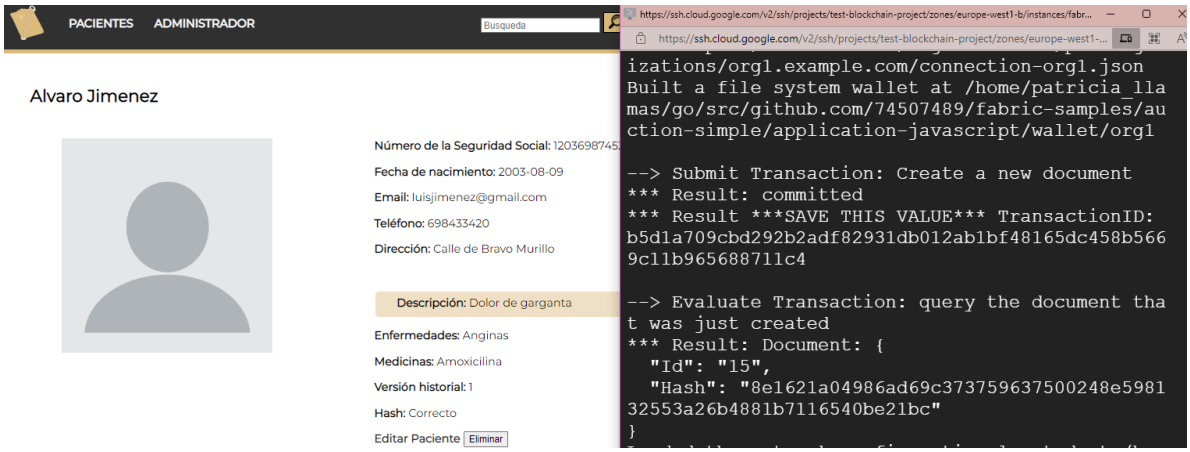
Figura 36: Creación de un historial médico en la red de blockchain

```
--> Submit Transaction: Create a new document
*** Result: committed
*** Result ***SAVE THIS VALUE*** TransactionID: 3a749aa24a74204509d77d00f9c27c4406

--> Evaluate Transaction: query the document that was just created
*** Result: Document: {
  "Id": "doc1ID",
  "Hash": "fa3cccc718c7598ddc909a8e2dc7cb8e2d0179164a965b293f6b681b9f78f2bb"
}
```

Se observa cómo además de crear el historial y emitir la transacción, devuelve el ID de la misma y llama al script queryDocument.js que realiza una consulta al estado mundial para comprobar que el documento se ha creado correctamente. La Figura 37 muestra el proceso, tanto en la interfaz web como el mantenimiento de la red de blockchain, que se realiza siempre en segundo plano.

Figura 37: Creación de un historial médico desde la aplicación web



- **Actualización del hash de un historial**

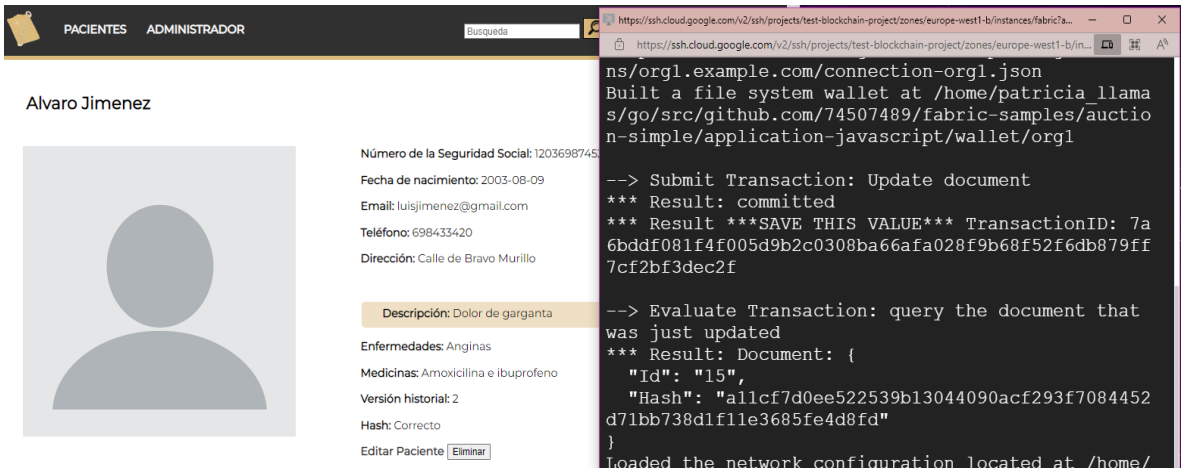
Se envía a la cadena el nuevo hash de un historial después de haber hecho cambios en él. El resultado se muestra en la Figura 38:

Figura 38: Actualización del hash de un historial médico en la red de blockchain

```
--> Evaluate Transaction: query the document that was just updated
*** Result: Document: {
  "Id": "doc1ID",
  "Hash": "bd4526534df7b33772c2f1ee26d97c39ff11379c8848e4e19"
}
```

Si se modifica el historial de un paciente en la aplicación se envía el nuevo hash a la cadena de bloques y se emite una nueva transacción, tal como ilustra la Figura 39:

Figura 39: Actualización del hash de un historial médico desde la aplicación web



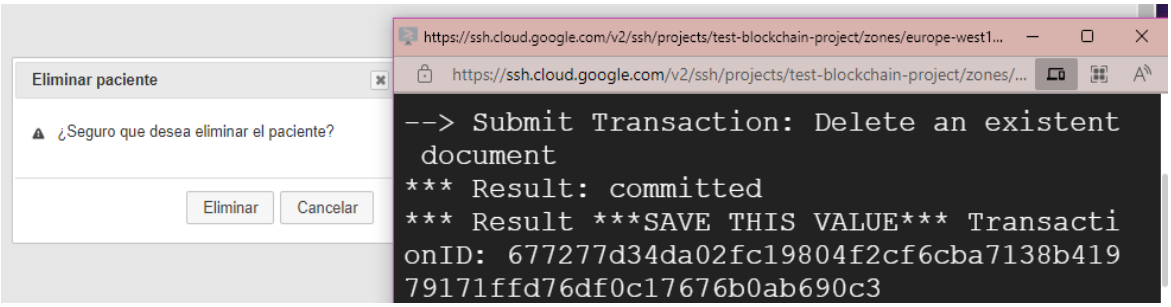
- **Eliminación del registro de un historial médico**

Al eliminar un historial de un paciente se debe hacer la consiguiente eliminación del registro asociado al mismo en la cadena, para ello es necesario proporcionar el ID del documento que se quiere eliminar. Las Figuras 40 y 41 ilustran el progreso de dicha acción:

Figura 40: Eliminación de un historial médico de la red de blockchain

```
--> Submit Transaction: Delete an existent document
*** Result: committed
```

Figura 41: Eliminación de un historial médico desde la aplicación web



- **Comprobar que el hash de un historial coincide con el proporcionado**

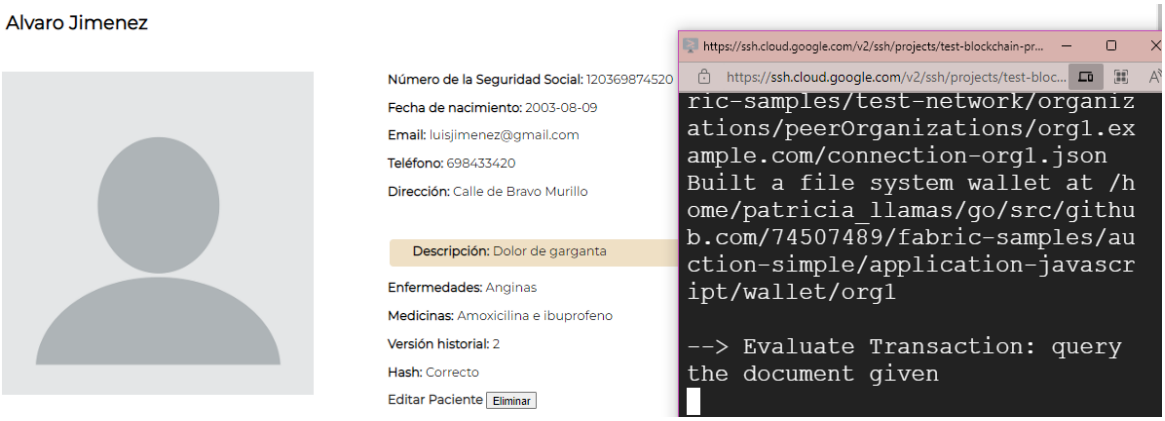
Se procede a comprobar si el contenido de un registro en la cadena de bloques coincide con el que se proporciona a la función por parámetros. En este caso se comprueba que el historial médico de un paciente almacenado en la base de datos coincide en todo momento con el registro correspondiente en la red de blockchain asegurando así la integridad de este. Esto se muestra en las Figuras 42 y 43, en caso de éxito, y 44 y 45, en caso de fallo.

  - Éxito

Figura 42: Hash correcto de un historial médico desde la red de blockchain

```
true
```

Figura 43: Hash correcto de un historial médico desde la aplicación web



- Fallo

Figura 44: Hash incorrecto de un historial médico desde la red de blockchain

```
--> Evaluate Transaction: query the document given
******** FAILED to submit document: Error: The hash bd4626534df7b33772c2f1
ee26d97c39ff11379c8848e4e19 does not match the one kept on the ledger
```

Figura 45: Hash incorrecto de un historial médico desde la aplicación web

En la página web en el momento que se accede al historial de un paciente se muestra en todo momento si los datos almacenados en el blockchain coinciden con los almacenados en la base de datos de la aplicación. En el caso de que no sea así se alerta al doctor de que esto puede suponer un peligro para el paciente para que lo tenga en cuenta a la hora de realizar las modificaciones pertinentes en el historial.

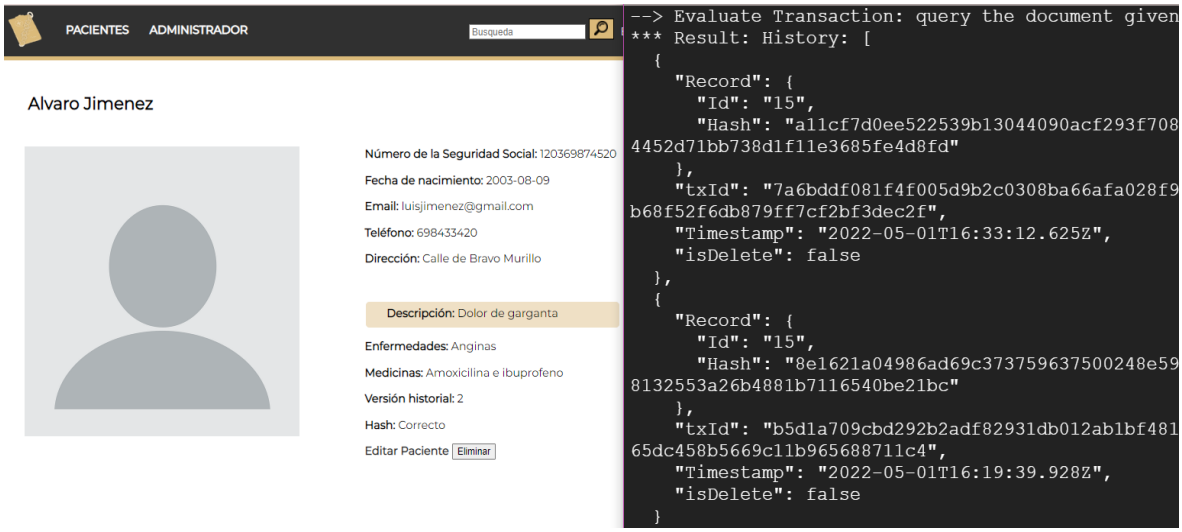
- **Mostrar el historial de transacciones asociadas a un historial**

Se muestran todos los movimientos realizados sobre un historial concreto ordenado desde la transacción más nueva a la más antigua. Para su correcto funcionamiento es necesario proporcionar el identificador del historial a comprobar. En las Figuras 46 y 47 que se muestran a continuación se pueden observar todos los cambios realizados al documento “doc1ID”.

Figura 46: Número de versiones de un historial médico desde la red de blockchain


```
{
  "Record": {
    "Id": "doc1ID",
    "Hash": "fa3cccc718c7598ddc909a8e2dc7cb8e2d0179164a965b293f6b681b9f78f2bb"
  },
  "txId": "3a749aa24a74204509d77d00f9c27c4406bc4f40fc99855696015f6b3a54d85a",
  "Timestamp": "2021-11-23T15:58:01.758Z",
  "isDelete": false
},
{
  "Record": {
    "Id": "doc1ID",
    "Hash": ""
  },
  "txId": "c77446050c274c005be75813cf45db1556f1af1c47f3f0614e38ab9fd1368d0a",
  "Timestamp": "2021-11-23T15:56:00.696Z",
  "isDelete": true
},
```


Figura 47: Número de versiones del historial médico desde la aplicación web



En la página web gracias a esta función se le muestra al doctor la versión actual del historial del paciente, como se puede ver en la captura adjunta. Se han realizado cambios dos veces al historial médico de ese paciente, como se puede ver en la parte derecha de la foto. Eso se refleja en la aplicación web de la manera descrita.

- **Mostrar el contenido de un historial proporcionando su ID**

Simplemente es necesario proporcionar el ID de un historial y se realiza una consulta que devuelve los datos de este. El resultado se muestra en la Figura 48, incluida a continuación.

Figura 48: Elementos de un historial médico desde la cadena de bloques

```
--> Evaluate Transaction: query the document given
*** Result: Document: {
  "Id": "doc1ID",
  "Hash": "bd4526534df7b33772c2f1ee26d97c39ff11379c8848e4e19"
}
```

- **Mostrar el contenido de todos los registros almacenados en la cadena**

Desde la cadena de bloques se puede listar el contenido de todos los historiales médicos de pacientes que se encuentran almacenados en la misma. Esto se puede observar en la Figura 49.

Figura 49: Mostrar el contenido de todos los historiales médicos en la cadena de bloques

```
--> Evaluate Transaction: query all the documents
*** Result: Documents: [
  {
    "Id": "doc1ID",
    "Hash": "bd4526534df7b33772c2f1ee26d97c39ff11379c8848e4e19"
  },
  {
    "Id": "doc2ID",
    "Hash": "dnbpomc718c7598ddc909a8e2dc7cb8e2d0179164a965b293f
  }
```

- **Mostrar el contenido de un historial que no existe**

En caso de que se intente mostrar un historial médico que no exista la cadena de bloques devuelve, como se puede observar en la Figura 50, un mensaje de error.

Figura 50: Mostrar el contenido de un historial médico que no existe

```
--> Evaluate Transaction: query the document given
******** FAILED to submit document: Error: document does not exist
```

- **Creación de un historial con un identificador existente**

En caso de intentar crear un historial médico proporcionando un historial médico existente la red devuelve un mensaje de error, como se muestra en la Figura 51:

Figura 51: Creación de un historial médico con un identificador existente

```
******** FAILED to submit document: Error: No valid responses from any peers. Errors:
    peer=peer0.org1.example.com:7051, status=500, message=the document doc2ID already exists
Error in creating document doc2ID: Error: No valid responses from any peers. Errors:
    peer=peer0.org1.example.com:7051, status=500, message=the document doc2ID already exists
```

Capítulo 6. Conclusiones y trabajo futuro

6.1. Conclusiones

En el ámbito personal siempre me he considerado una persona muy curiosa, que disfruta aprendiendo e investigando sobre lo desconocido. El blockchain es una tecnología que brinda enormes avances en inimaginables ámbitos de la vida como son las finanzas, el sector energético, la agricultura, el comercio o la sanidad como se ha podido ver con este trabajo. Por estas razones y por muchas otras considero que es un proyecto y un tema que merece la pena y en el que he estado muy incentivada para trabajar.

A pesar de todo, por tratarse de una tecnología totalmente nueva para mí, el desarrollo del proyecto ha supuesto un duro esfuerzo. Después de la realización de éste puedo llegar a la conclusión de que es un elemento diferenciador para las compañías por sus características y por sus enormes aplicaciones en la mayoría de los sectores a pesar de la dificultad que conlleva su desarrollo.

Volviendo a los orígenes el proyecto, que han podido quedar en segundo plano después de numerosas páginas llenas de conceptos teóricos sobre la tecnología utilizada, hay que recordar el objetivo principal del proyecto: “crear una red de blockchain para poder así demostrar la utilidad de esta tecnología en relación con la integridad y la seguridad de la información en el área de gestión documental”. Después de casi un año de trabajo se puede llegar a la conclusión de que ese objetivo principal se ha cumplido.

Este proyecto está diseñado de tal forma que es compatible con numerosos proyectos relacionados con la gestión documental, y el sector de la medicina es idóneo para demostrar la utilidad de la tecnología blockchain debido a la cantidad de agentes implicados y la importancia de la información gestionada.

El blockchain es una tecnología que, aunque haya surgido hace numerosos años, sigue en auge debido a sus cualidades. Como se ha podido ver en este proyecto, es posible asegurar la integridad y la seguridad de los datos almacenados en la red, así como llevar un historial de versiones de estos pudiendo saber qué persona ha realizado cambios en el documento, la fecha y otros datos relevantes sobre la información almacenada tanto en redes públicas como en redes privadas. Además, en el ámbito empresarial, estas últimas cobran especial importancia por la posible limitación de la entrada de usuarios a la red y a los diferentes elementos de ésta mediante el uso de canales.

Las aplicaciones del blockchain en gestión documental son limitadas hasta el momento, pero seguro que se verá un gran incremento del uso de esta tecnología en futuros años.

Competencias adquiridas y reforzadas

Este trabajo ha permitido obtener y fortalecer conocimientos existentes sobre multitud de ramas de ingeniería informática, entre las que se puede destacar:

- Conocimiento de dos nuevos lenguajes de programación: Node JavaScript, que es un lenguaje de programación orientado a objetos y Go, un lenguaje concurrente inspirado en la sintaxis de C.
- Utilización de lenguajes de programación utilizados en la asignatura de páginas web como son PHP, HTML y CSS, así como de bases de datos SQL.
- Puesta en marcha de la metodología usada en Ingeniería del Software como son los casos de uso.
- Trabajo en áreas de redes relacionadas con la seguridad mediante el protocolo TLS y otros conceptos relacionados con criptografía, claves y certificados digitales estudiados en las asignaturas de Redes y Seguridad 1 y 2.
- Dominio de conceptos teóricos relacionados con redes blockchain y con frameworks para la implementación de estas.

6.2. Trabajo futuro

En este trabajo se ha realizado una prueba de concepto que permite comprobar la utilidad del blockchain en los diferentes sectores para asegurar la integridad y seguridad de documentos. A continuación, se presenta un listado con posibles mejoras del trabajo.

- **Modificación de la estructura de la red**
  Como se ha comentado en previas ocasiones se ha partido de una estructura de red básica que no es válida para entornos de desarrollo ni de producción. En un futuro sería necesario modificar los elementos que componen la red añadiendo al menos tres orderers, que es el número mínimo de este tipo de nodos que se recomienda. Sería interesante considerar añadir al menos un nodo validador más a cada organización.

- **Modificación de la aplicación para gestionar ficheros**
  En estos momentos los historiales de los pacientes están formados por una serie de variables que se almacenan en la base de datos. Como trabajo futuro sería necesario gestionar documentos reales que se almacenen en alguna aplicación de gestión documental. Los médicos cada vez que necesiten modificar un historial de un paciente se descargarán el fichero a su ordenador y realizarán los pertinentes cambios y luego lo volverán a subir a la aplicación web. Antes de la descarga se deberá comprobar que el hash almacenado en la base de datos es el mismo que se encuentra en el blockchain, asegurando así que el documento descargado se trata de la última versión disponible. Esto permitiría que varios doctores pudieran realizar cambios en el documento asegurándose de que no se sobrescriben los datos unos a otros. Además, al ir guardando todos los ficheros en una aplicación de gestión documental sería posible moverse por las distintas versiones de estos.

- **Aplicación a otro tipo de casos de uso**
  La estructura de los documentos almacenados en la cadena de bloques es muy general estando formada solamente por un hash y un identificador del documento del que se quiera almacenar un registro. Debido a estas características tan abstractas, esta misma solución se podría aplicar a innumerables casos de uso.

- **Utilización de Hyperledger Explorer**
  Hyperledger Explorer es una herramienta de Hyperledger que permite ver, invocar, desplegar y consultar bloques, transacciones y los datos asociados a las mismas. Permite a su vez, consultar información de la red, como el nombre, estado, nodos, información sobre chaincodes y otra información relevante sobre el ledger. Sería interesante su uso para conocer en mayor profundidad la red implementada [51].

Apéndice 1. Fabric Contract Api

Fabric Contract API, es una API de alto nivel que permite implementar contratos inteligentes. Trabajar desde la API otorga a los desarrolladores un punto de entrada desde el que escribir la lógica del negocio. Al usar esta API se abre la posibilidad de llamar al Chanicode Stub que tiene funciones para llamar al ledger.

Se van a explicar en más detalle las APIs que se han usado del Stub, las cuales se pueden dividir en numerosas categorías. Por simplicidad se van a detallar únicamente las usadas [52].

- **APIs del estado mundial.**
    - Getstate(): Esta función devuelve el valor actual de la variable "key". Utilización en:
        - DocumentExists() para comprobar si un documento con una clave concreta se ha creado previamente o no.
        - QueryDocumentByKey() para devolver un documento concreto por su clave.
        - HashCheck() para obtener el documento que se pasa por parámetro y comprobar su hash.
    - PutState(): Introduce un elemento mediante una clave y un valor. Utilización en:
        - Createdocument(), introduce un documento utilizando como clave su id y como valor el hash.
        - UpdateDocument(), modifica un documento utilizando como clave su id y como valor el nuevo hash de sus datos.
    - DelState(): Elimina una clave del estado mundial. Utilizado en:
        - DeleteDocument(), elimina un documento.
    - GetStateByRange(): Devuelve un iterador sobre las claves que se encuentran en el rango "startKey" y "endKey" proporcionado. Utilizado en:
        - QueryAllDocuments(), introduciendo un rango vacío devuelve todas las claves que se encuentran en el estado mundial.
- **APIs de transacciones.**
    - GetTxID(): devuelve el identificador de una transacción. Utilizado en:
        - CreateDocument(), UpdateDocument() y DeleteDocument() para devolver el id de la transacción realizada.
- **APIs de claves:**
    - GetHistoryForKey(): devuelve una historia de los valores de una clave a lo largo del tiempo. Para cada clave, devuelve su valor, el identificador de transacción asociado y la marca de tiempo. Utilizado en:
        - QueryDocumentHistory(), para una clave en concreto devuelve su valor, el identificador de la transacción, su marca de tiempo y si ese documento se ha borrado o no.

Apéndice 2. Introduction

2.1. Motivation

Large flows of documents are a huge concern for most businesses today. This huge amount of documents can lead companies to face serious problems including duplicate information, loss, and malicious modification of documents, etc. Many companies are focusing on finding alternatives to automatize work processes and to make document management more secure. This is where the blockchain comes into play.

Blockchain networks are made up of a set of blocks linked by cryptography that allow the information stored in them to be immutable and secure. Due to its characteristics, it is widely used in the financial field, as well as in logistics and distribution.

In document management, blockchain allows validating and verifying all existing documents in a company. Thanks to its properties, it is possible to check, among other things, the person who created a document, the time, the place, its last modification, and it also allows you to check if it has been manipulated. In addition, as there are private blockchain networks, documents will only be accessible to authorized persons, thus providing all the privacy and transparency necessary in a company.

Blockchain has been the technology chosen for this project since it allows effective monitoring of all changes to a document and serves to verify the authenticity of a document, thus ensuring some of the most important objectives of computer security:

- **Confidentiality:** In computer security, confidentiality is the principle that ensures information is only available for those who have permission for it. Private blockchain networks allow information to be obtainable only for authorized parts.
- **Integrity:** This concept explains that information stored in certain devices has not been manipulated by third parties in a malicious way. Blockchain assures integrity of data because of the cryptography that joins the blocks of its structure.
- **Availability:** Information must remain available for all authorized people in the moments it is necessary. In blockchain, every node in the network contains a record of the ledger. Due to that, in case of a problem in one of the parts, the information would not be lost, and it would be accessible for the rest of the network.

2.2. Goals

The goal of this project is to create a blockchain network to prove the utility of this technology in relation to the integrity and security of information around document management.
This main objective is divided into four secondary objectives:

- Make decisions about the structure of the network and the software and hardware needed based on the investigation of blockchain and the material available in primary and secondary sources.
- Learn about the behaviour of blockchain technology due to the enormous possibilities it offers.
- Apply and deepen the knowledge learnt and the technologies used during the five years of the degree.
- Union of the blockchain network with a real-life use case to prove the usefulness of the technology in a business domain.

2.3. Work plan

This work has started to develop at the end of June 2021. During the first two months of work the project has been supervised by the bosses of the company Evenbytes, for whom this blockchain implementation is developed. Afterwards, the work plan is based on personal work periodically revised by both tutors to guide and check the advances made.

To efficiently develop the project, it has been divided in a series of steps, stablishing goals to accomplish in each of them.

Para conseguir desarrollar el trabajo de una forma eficiente se ha decidido agrupar el proyecto en una serie de etapas, estableciendo objetivos a cumplir en cada una de estas.

- *First phase – Investigation*

The objective of this phase is to investigate the theoretical aspects regarding blockchain. This phase is crucial to understand the theoretical framework that needs to be the base for decisions about the structure of the network.

- *Second phase – Analysis*

This time is dedicated to investigating use cases of blockchain in different life aspects, focusing on document management. It is necessary to look for possible applications for the development of the network. Using the information gathered in phase one important decisions need to be made relating the use of smart contracts, hardware specifications, etc.

- *Third phase – Network structure*

This phase focusses on detailing the structure of the network deciding the number of nodes, channels, and organizations among other elements.

- *Fourth phase – Knowledge of the framework and creation of the network*

The fourth phase focuses on the knowledge of the used framework, Hyperledger fabric. For this matter, an extense tutorial needs to be read. This tutorial explains how to deploy the network.

- *Fifth phase – Implementation*

The next step is to create the transaction scripts and the chaincode. For this a few days must be dedicated to familiarizing with the two programming languages used by Hyperledger, Golang and

Node js. This phase also includes the development of the API used to call the blockchain network. The last step of this phase is joining the network with a patients' management website.

- *Sixth phase – Tests*

In this phase of the document the blockchain network is tried through command line and through the patients' management website.

- *Seventh phase – Documentation*

This part is periodically performed throughout the whole Project to deepen and synthesize the knowledge gained and the decisions taken.

2.4. Organization of the document

The organization of the document is going to be detailed to facilitate the comprehension of the work done. This document is divided in six chapters.

In the first chapter the motivation and objectives of the final degree Project are specified, and the working plan and organization of the document to assure a correct division of time and efficient use of resources.

Chapter two includes the state of art of this work. In this part a document investigation of papers and applications of blockchain technology is carried out. The goal is to know what projects have been developed to know which aspects are similar and which are not from the project developed. Also, existing software and hardware platforms are enumerated for the creation of the network.

The third chapter includes basic concepts relating blockchain such as the different types of blockchain, the elements of the network and the functioning of the blockchain network.

The fourth chapter explains in a more exhaustive way the chosen platform, the hardware needed to implement the network and the auxiliary tools used. The last section explains the basic concepts of the chosen framework and the specification of the implemented prototype.

The fifth chapter contains the code of the chaincode and the transactions. It includes an explanation of the deployment of the network and of the application programming interface (API). Lastly, it details the functionality of the web application created and the tests done with the command line and through the web application.

Consecutively, in chapter six, the conclusions of the project are exposed as well as possible ways to improve it.

Last of all, it includes appendixes to amplify relevant information of the project so that the reader can dive into some aspects of it.

Apéndice 3. Conclusions and future work

3.1. Conclusions

I have always considered myself as a curious person, who enjoys learning and investigating about the unknown. Blockchain is a technology that offers enormous opportunities in many aspects of life such as finance, agriculture, commerce, or health care as this project has shown. For this and many other reasons I think it is a worth working project and I have been very motivated to work in it.

Even thought, the development of the project has been very hard because it is an unknown technology for me. After the realization of the project, I can assure it is a differentiator for companies due to its characteristics and for its numerous applications in real life.

Going back to the origins of the project, the main goal must be remembered: "creating a blockchain network to prove the utility of this technology regarding the integrity and security of information in document management". After almost one year it is possible to conclude that the goal has been accomplished.

This Project has been designed in a way that it is compatible with many projects relating document management. Healthcare is optimal to prove the utility of blockchain due to the quantity of agents implied and due to the importance of the information managed.

Blockchain is a technology that, even thought it was created a long time ago, it is still emerging due to its characteristics. As it has been seen in this project, it is possible to assure the integrity and security of information stored in a blockchain network as well as been able to have a version check of who has created, updated, or deleted a document, both in public and private blockchain networks. Furthermore, in a business environment, private blockchains are extremely important because they offer the opportunity to limit the entrance of users to the network and to the different channels on it.

Blockchain applications in document management are rare but in the next few years they are going to increase exponentially.

Acquired and strengthened skills

This work has allowed me to obtain and reinforce so existent knowledge in many areas related to computer science, such as:

- Knowledge of two new programming languages: Node js, an object-oriented programming language, and Golan, a concurrent programming language inspired in the syntax of C.
- Use of programming languages used in the subject of web applications such as PHP, HTML y CSS, and SQL databases.
- Use of the methodology learnt in software engineering, such as use cases amongst other elements.
- Work in network areas related to security and other concepts related top cryptography, keys, and digital certificates studied in the subjects of networks and security 1 and 2.

- Domain of theoretical concepts related to blockchain networks and frameworks for the implementation of them.

3.2. Future work

In this work, a proof of concept has been carried out that allows verifying the usefulness of the blockchain in the different sectors to ensure the integrity and security of documents. Below is a list of possible work improvements.

■ Modification of the network structure

As has been mentioned on previous occasions, we have started from a basic network structure that is not valid for development or production environments. In the future, it would be necessary to modify the elements that make up the network by adding at least three orderers, which is the minimum number of this type of nodes that is recommended. It would be interesting to consider adding at least one more validator node to each organization.

■ Modification of the application to manage files

Currently, patient records are made up of a series of variables that are stored in the database. As future work, it would be necessary to manage real documents that are stored in some document management application. Every time they need to modify a patient's history, doctors will download the file to their computer and make the pertinent changes and then upload it back to the web application. Before downloading the documents, it is necessary to the hash stored in the database is the same as the hash found in the blockchain, thus ensuring that the downloaded document is the latest version available. This would allow multiple doctors to make changes to the document while making sure they don't overwrite each other's data. In addition, by saving all the files in a document management application, it would be possible to move through the different versions of these.

■ Application to other use cases

The structure of the documents stored in the blockchain is very general, being formed only by a hash and an identifier of the document of which a record is to be stored. Due to these very abstract features, this same solution could be applied to countless use cases.

■ Use of Hyperledger Explorer

Hyperledger Explorer is a Hyperledger tool that allows you to view, invoke, deploy, and query blocks, transactions, and the data associated with them. In turn, it allows querying network information, such as the name, status, nodes, information about chaincodes and other relevant information about the ledger. It would be interesting to use it to learn more about the implemented network [51].

Bibliografía

[1] BBVA. (2018). *What is the difference between DLT and blockchain?* https://www.bbva.com/en/difference-dlt-blockchain/

[2] Maldonado, J. (2020). *DLT vs Blockchain: ¿Cuáles son sus diferencias?* COINTELEGRAPH. https://es.cointelegraph.com/explained/dlt-vs-blockchain-what-are-the-differences

[3] World Bank Group. (2017). *Distributed Ledger Technology (DLT) and Blockchain.* https://documents1.worldbank.org/curated/en/177911513714062215/pdf/122140-WP-PUBLIC-Distributed-Ledger-Technology-and-Blockchain-Fintech-Notes.pdf

[4] Rodriguez, N. (2019). *Tecnología de Registro Distribuido: Donde la Revolución Tecnológica Comienza.* 101 Blockchains. https://101blockchains.com/es/tecnologia-de- registro-distribuido-dlt/#4

[5] 101 Blockchains (2020) .*What is DLT (Distributed Ledger Technology)?* 101 Blockchains. https://101blockchains.com/what-is-dlt/

[6] Vitaris, B. (2021). *Permissioned vs. Permissionless Blockchains Explained.* Permission.Io. https://permission.io/blog/permissioned-vs-permissionless-blockchain/

[7] Isabel Pérez (2021). *Blockchain: bloques, transacciones, firmas digitales | CriptoNoticias.* CriptoNoticias - Noticias de Bitcoin, Ethereum y criptomonedas. https://www.criptonoticias.com/criptopedia/blockchain-bloques-transacciones-firmas-digitales-hashes/#Bloque_completo

[8] Bit2Me Academy (2020). *¿Qué es un Árbol Merkle?* Bit2Me Academy. https://academy.bit2me.com/que-es-un-arbol-merkle/

[9] Fernández Espinosa, L. (2019). *Qué son los «smart contracts» o contratos inteligentes.* BBVA. https://www.bbva.com/es/smart-contracts-los-contratos-basados- blockchain-no-necesitan-abogados/

[10] IG*. ¿Qué es Ethereum y cómo funciona?* . IG. https://www.ig.com/es/ethereum-trading/ que-es-ether-y-como-funciona#information-banner-dismiss

[11] IBM. *What are smart contracts on blockchain? | IBM.* IBM. https://www.ibm.com/topics/smart-contracts

[12] BBVA (2021) *Qué es un «token» y para qué sirve.* BBVA NOTICIAS. https://www.bbva.com/es/que-es-un-token-y-para-que-sirve/

[13] Bit2Me Academy (2021). *¿Qué es un token?* (bit2me.com)

[14] Horizen Academy. *Los elementos de una cadena de bloques.* Horizen Academy. https://academy.horizen.io/es/technology/beginner/the-elements-of-a-blockchain/

[15] Anwar, H. (2021). *Consensus Algorithms: The Root of Blockchain Technology.* 101 Blockchains. https://101blockchains.com/consensus-algorithms-blockchain/#1

[16] Apla. *Proof-of-Authority consensus — Apla Blockchain Platform Guide documentation.* Apla. https://apla.readthedocs.io/en/latest/concepts/consensus.html

[17] Bit2Me Academy (2020) *¿Qué es PoA (Proof of Authority - Prueba de Autoridad)?* (2020). Bit2Me Academy. https://academy.bit2me.com/que-es-proof-of-authority-poa/

[18] Sharma, V., & Lal, N. (2020). *A novel comparison of consensus algorithms in blockchain.* https://www.mililink.com/upload/article/1765852599aams_vol_201_nov_2020_a1_p1-13_vishal_sharma_and_niranjan_lal.pdf

[19] Joisto. (2021). *Proof of Originality | JOISTO.* https://joisto.com/

[20] Izertis. https://www.izertis.com/es/-/casos-de-exito/bizkaia-lantik
[21] Amazon Web Services. *Amazon Managed Blockchain.* Amazon Web Services, Inc. https://aws.amazon.com/es/managed-blockchain/
[22] Quorum. *ConsenSys.* https://consensys.net/quorum/
[23] Cardano. https://cardano.org/
[24] VMWare Blockchain | ES. https://www.vmware.com/es/products/blockchain.html
[25] Ripple (2021). *Global Payment Solutions - Instant Processing.* https://ripple.com/
[26] Ethereum. https://ethereum.org/es/
[27] Hyperledger (2021) *Hyperledger – Open Source Blockchain Technologies.* https://www.hyperledger.org/
[28] Reyes, V. M. (2019). *Introducción a Hyperledger Fabric en Español Medium.* Introducción a Hyperledger Fabric en Español | by Victor Muñoz Reyes | Babel — go2chain | Medium
[29] Hyperledger Fabric. *Prerequisites - Hyperledger-fabricdocs main documentation.* Hyperledger Fabric. https://hyperledger-fabric.readthedocs.io/en/latest/prereqs.html
[30] Hyperledger Fabric. *Install Fabric and Fabric Samples - Hyperledger-fabricdocs main documentation.* Hyperledger Fabric. https://hyperledger-fabric.readthedocs.io/en/latest/install.html
[31] Hyperledger Fabric. *Ledger - Hyperledger-fabricdocs main documentation.* Hyperledger Fabric. https://hyperledger-fabric.readthedocs.io/en/latest/ledger/ledger.html
[32] Google. *Cloud Endpoints for OpenAPI | Cloud Endpoints with OpenAPI.* Google Cloud. https://cloud.google.com/endpoints/docs/openapi
[33] Polge, J., Robert, J., & le Traon, Y. (2021). *Permissioned blockchain frameworks in the industry: A comparison.* https://reader.elsevier.com/reader/sd/pii/S2405959520301909?token=E520066AF24F72ABFFC758A422D127723C3FE7ADF0D9331A1601457C0EFCA18245B753E0B2380D916837EBC3BE348B7D&originRegion=eu-west-1&originCreation=20211019171504
[34] Izertis (2021). *Endesa encarga a Izertis la explotación y mantenimiento del proyecto CONFÍA para la protección energética de colectivos vulnerables*
[35] Izertis (2021). *Success stories - CXC*
[36] Telefónica (2021). *Telefónica Tech lanza un servicio de certificación documental basado en blockchain*
[37] Filecoin. *A decentralized storage network for humanity's most important information | Filecoin*
[38] Gómez, Javier (2018). *Criptomonedas*. Google Cloud anuncia nuevas asociaciones e integración de Hyperledger Fabric y Ethereum (criptonoticias.com)
[39] Nitin Gaur | Venkatraman Ramakrishna | L. . (2018). *Oracle Blockchain Services Quick Start Guide.* https://subscription.packtpub.com/book/data/9781789804164/1/ch01lvl1sec04/dlt-and-blockchain
[40] E-estonia. *Home page - e-Estonia*
[41] IBM. *¿Qué es Hyperledger Fabric? | IBM*
[42] Wikipedia, la enciclopedia libre. *Bifurcación (sistema operativo)*. Wikipedia. https://es.wikipedia.org/wiki/Bifurcaci%C3%B3n_(sistema_operativo)
[43] Yolande Piazza (2022). *Google Cloud launches new dedicated Digital Assets Team* – Google Cloud. https://cloud.google.com/blog/topics/financial-services/google-cloud-launches-dedicated-digital-asset-team

[44] Hyperledger Fabric. *Blockchain network.* https://hyperledger-fabric.readthedocs.io/es/latest/network/network.html
[45] Bit2Me Academy. *¿Qué es un nodo?* (bit2me.com)
[46] Hyperledger Fabric. *Identity — hyperledger-fabricdocs main documentation*
[47] Hyperledger fabric. *Glossary — hyperledger-fabricdocs main documentation*
[48] Hyperledger Fabric. *The Ordering Service — hyperledger-fabricdocs main documentation*
[49] Hyperledger Fabric. *Smart Contracts and Chaincode — hyperledger-fabricdocs main documentation*
[50] Hyperledger Fabric. *Membership Service Provider (MSP) — hyperledger-fabricdocs main documentation*
[51] Hyperledger Foundation. *Hyperledger Explorer – Hyperledger Foundation*
[52] Github*. Hyperledger Fabric SDK for node.js Class: ChaincodeStub*