UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf

Escola Tècnica Superior d'Enginyeria Informàtica

Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

Diseño e implementación de un sistema de trazabilidad de lotes basado en tecnología blockchain/DLT para la trasparencia en las donaciones de juguetes

TRABAJO FIN DE GRADO

Grado en Ingeniería Informática

Autor: María Inmaculada Romero Ramal

Tutores: Lenin Guillermo Lemus Zúñiga y Begoña Saiz Mauleón

Curso 2020-2021

Resum

Aquest treball té com a objectiu principal el disseny i implementació d'un sistema de traçabilitat per a lots de joguets donats utilitzant la tecnologia *Blockchain*/DLT. Tot això amb la intenció d'augmentar en gran mesura la transparència, seguretat, confiança i seguiment d'aquest tipus de donacions, evitant d'aquesta manera accions malicioses.

Un objectiu secundari és el disseny i implementació d'una pàgina web que faciliti la gestió del sistema de traçabilitat, on els usuaris podran fer un seguiment de les seues donacions amb una interfície intuïtiva i senzilla.

Amb tot això es divideix el pla de treball en quatre fases. La fase 0 es correspon amb la recerca bibliogràfica, anàlisi de el problema, del context tecnològic, l'elecció de la tecnologia utilitzada i el disseny inicial de la solució. La fase 1 s'encarrega de la codificació i preparació de tots els components que es relacionen amb la part de l'aplicació que s'executa de la banda de servidor (*backend*). Pel que fa a la fase 2, aquesta té com a objectiu la implementació de la part de l'aplicació que s'executa de la banda de el client (*frontend*), la qual mostra una interfície d'usuari i es comunica amb el *backend*. Dit *frontend* s'ha implementat com una aplicació web anomenada: "Donated Toys". Finalment, la fase 3 es correspon amb la realització de proves unitàries, mesuraments de temps, conclusions i possibles ampliacions que es podrien realitzar en un futur.

Paraules clau: Traçabilitat, lots, *Blockchain*, joguets, DLT, Hyperledger

Resumen

Este trabajo tiene como objetivo principal el diseño e implementación de un sistema de trazabilidad de lotes de juguetes donados utilizando *software* basado en tecnología *Blockchain*/DLT. Todo esto con la intención de aumentar en gran medida la transparencia, seguridad, confianza y seguimiento de este tipo de donaciones, evitando de esta forma acciones maliciosas.

Un objetivo secundario es el diseño e implementación de una aplicación web que facilite el manejo del sistema de trazabilidad, en donde los usuarios podrán rastrear sus donaciones con una interfaz intuitiva y sencilla.

Con todo esto se divide el plan de trabajo en cuatro fases. La fase 0 se corresponde con la búsqueda bibliográfica, análisis del problema, del contexto tecnológico, la elección del *software* utilizado y el diseño inicial de la solución. La fase 1 se encarga de la codificación y preparación de todos los componentes que se relacionan con la parte de la aplicación que se ejecuta del lado del servidor (*backend*). En cuanto a la fase 2, ésta tiene como objetivo la implementación de la parte de la aplicación que se ejecuta del lado del cliente (*frontend*), la cual muestra una interfaz de usuario y se comunica con el *backend*. Dicho *frontend* se ha implementado como una aplicación web denominada: "Donated Toys". Por último, la fase 3 se corresponde con la realización de pruebas unitarias, mediciones de tiempos, conclusiones y posibles ampliaciones que se podrían realizar en un futuro.

Palabras clave: Trazabilidad, lotes, *Blockchain*, juguetes, DLT, Hyperledger

Abstract

The main objective of this work is the design and implementation of a traceability system for lots of donated toys using software based on *Blockchain*/DLT technology. All this with the intention of greatly increasing the transparency, security, trust, and monitoring of this type of donations, thus avoiding malicious actions.

A secondary objective is the design and implementation of a web application that facilitates the management of the traceability system, where users can track their donations with a very intuitive and simple interface.

Considering all of this, the work plan is divided into four phases. Phase 0 corresponds to the bibliographic search, analysis of the problem, the technological context, the choice of the software used and the initial design of the solution. Phase 1 is responsible for the coding and preparation of all the components that are related to the part of the application that runs on the server side (backend). As for phase 2, this has as its objective the implementation of the part of the application that runs on the client side (frontend), which shows a user interface and communicates with the backend. This frontend has been implemented as a web application called: "Donated Toys". Finally, phase 3 corresponds to the performance of unit tests, time measurements, conclusions and possible extensions that could be carried out in the future.

Key words: Traceability, batches, *Blockchain*, toys, DLT, Hyperledger

Índice general

Índice de figuras

Índice de tablas

CAPÍTULO 1
Introducción

Desde el curso 2017-2018 he colaborado con el grupo de investigación Tecnologías de la Información y la Comunicación contra el Cambio Climático (ICTvsCC) de la Universidad Politécnica de Valencia (UPV). El grupo en cuestión propuso un proyecto relacionado con el uso de la tecnología *Blockchain* para que fuese realizado por un alumno o alumna que obtuviera una beca de colaboración del Ministerio de Educación y Formación Profesional de España, con el objetivo de desarrollar herramientas para su uso en la Oficina de Cooperación al Desarrollo de la UPV. Es en el contexto de este proyecto y bajo el amparo de dicha beca de colaboración, en el que se ha desarrollado el presente TFG.

De hecho, actualmente está bastante extendido el desarrollo de aplicaciones basadas en *Blockchain* unidas al uso de la trazabilidad para poder conocer la procedencia o destino de un artículo. No obstante, en el ámbito de cooperación al desarrollo este tema está poco implementado siendo prácticamente inexistente en las donaciones de juguetes, y a pesar de los escándalos mediáticos de robo de productos donados, por parte de altos cargos de las instituciones que realizaban la distribución de los donativos, no hay un claro avance por cortar el problema de raíz.

Por tanto, este trabajo se centrará en la realización de un sistema de trazabilidad que empleando *Blockchain* otorgará transparencia y seguridad como solución al problema específico del robo de juguetes donados.

Finalmente, cabe mencionar que el código de la aplicación realizada puede encontrarse en el siguiente enlace: `https://github.com/AiramAire/RedDonaciones.git`.

1.1 Motivación

La utilización de la tecnología *Blockchain* y el uso de Contratos Inteligentes (*Smart Contracts*) en un sistema trazable, tanto en información como en productos, es una faceta que considero que se debería explotar más debido a las ventajas que ofrece. Sí que es verdad que se emplea en algunos ámbitos, pero a nivel de cooperación al desarrollo no hay muchas aplicaciones y se encuentran aún menos en la trazabilidad de donaciones de juguetes, pese a que se han dado casos bastante mediáticos de apropiaciones indebidas y falsificación de datos, como es el caso de Goodwill cuyos altos cargos estaban implicados en la desaparición de artículos donados.

Entre las ventajas que ofrece el uso de la tecnología *Blockchain* y la trazabilidad que la acompaña está una gran seguridad, para el usuario que realiza la donación y para la información que se guarda pues los datos en una *Blockchain* no se pueden modificar. También aumenta la confianza de los usuarios o clientes de la institución encargada de recoger y/o distribuir las donaciones debido a que aporta transparencia a dichas institu-

ciones y, además, permite detectar mucho antes casos de corrupción o malversación de bienes al poder rastrear los datos (información sobre los artículos donados) almacenados, [1] [2].

Por todo esto considero que el progreso en aplicaciones con Contratos Inteligentes basados en tecnología *Blockchain* y trazabilidad de los artículos, pese a ser un campo que todavía no está del todo desarrollado, es de gran ayuda en muchos sectores de la sociedad y, sobre todo, en ámbitos de cooperación al desarrollo y ayuda humanitaria.

1.2 Objetivos

Este trabajo tiene como objetivo principal el diseño e implementación de un sistema de trazabilidad de lotes de juguetes donados utilizando *software* basado en tecnología *Blockchain*/DLT.

Un objetivo secundario es el diseño e implementación de una aplicación web que facilite el manejo del sistema de trazabilidad, en donde los usuarios podrán rastrear sus donaciones con una interfaz intuitiva y sencilla.

En otras palabras, se diseñará e implementará una aplicación de tipo cliente/servidor, en donde la implementación de la parte de la aplicación que se ejecuta del lado del cliente (*frontend*) representará la aplicación web y es con la cual interactuarán los usuarios para ver el estado de sus donaciones. Mientras que, el código necesario para que la información se almacene en la *Blockchain* y que se realicen las consultas para conocer el estado de las donaciones, se implementará como una aplicación que se ejecutará del lado del servidor (*backend*).

1.3 Impacto esperado

Se espera que este trabajo tenga un claro impacto en la mejora de las modalidades de consumo y producción sostenibles, es decir, que se pueda dar una nueva utilidad o "segunda vida" a los juguetes en manos de los niños más necesitados.

De esta forma se alinearía con el objetivo número 1 del Desarrollo Sostenible que aprobó la ONU en 2015: "Poner fin a la pobreza en todas sus formas en todo el mundo". Pues los juguetes permitirían a los niños divertirse, olvidando su precaria situación e historias vividas hasta el momento y que, por desgracia, les han quitado un parte de su infancia para obligarlos a exponerse a una vida injusta y cruel.

1.4 Metodología

La siguiente lista muestra los pasos que se van a seguir para alcanzar los objetivos de este trabajo:

1. Definir la funcionalidad del sistema.
2. Buscar, analizar y comparar los marcos de trabajo (*frameworks*) que faciliten el desarrollo de aplicaciones basadas en *Blockchain*/DLT existentes.
3. Seleccionar el marco de trabajo que conforme a lo visto en el apartado anterior resulte más interesante para la implementación del sistema.
4. Proponer la arquitectura del sistema.

5. Desarrollar e implementar la aplicación.
6. Validar el sistema.
7. Estimar el coste del sistema.
8. Incluir las conclusiones acerca de lo conseguido en el tiempo de desarrollo y los posibles trabajos futuros.

1.5 Estructura de la memoria

A continuación, se enumeran y comentan cada uno de los capítulos que componen la memoria. También se explicará de forma concisa el contenido de cada capítulo.

- **Capítulo 1: Introducción:** en este capítulo se ha realizado una introducción al trabajo y se ha hablado de la motivación, de los objetivos, del impacto esperado y de la metodología a seguir.
- **Capítulo 2: Contexto tecnológico**: se proseguirá en el segundo capítulo con el contexto tecnológico que envuelve este trabajo.
- **Capítulo 3: Análisis del problema:** a continuación, se dedicará el tercer capítulo a analizar de forma detallada el problema, donde se comentará y se analizará todo lo referente a la seguridad, la eficiencia, el marco legal y ético, y también se identificarán varias soluciones al problema, se propondrá una a desarrollar, se establecerá un plan de trabajo y se estimará un presupuesto.
- **Capítulo 4: Diseño de la solución:** en el cuarto capítulo se expondrá el diseño de la solución propuesta en el capítulo anterior, hablando también acerca de la arquitectura del sistema, sus detalles y las tecnologías que serán utilizadas.
- **Capítulo 5: Desarrollo de la solución propuesta:** a caballo con el capítulo anterior, se proseguirá en el quinto capítulo con la descripción detallada del desarrollo de la solución propuesta.
- **Capítulo 6: Validación y pruebas:** para finalizar el ciclo de vida del proyecto, se describirán las pruebas y validaciones llevadas a cabo, tanto funcionales como mediciones de tiempos.
- **Capítulo 7: Conclusiones:** en el octavo capítulo se comentarán las conclusiones que se pueden extraer de todo el proyecto.
- **Capítulo 8: Trabajos futuros:** por último, en el noveno capítulo se hablará de todas las posibles mejoras y ampliaciones que se podrían aplicar en trabajos futuros.

CAPÍTULO 2
Contexto Tecnológico

En este capítulo se contextualizará el entorno tecnológico para la creación de aplicaciones *Blockchain*, posteriormente se comentarán algunas de los usos de esta tecnología y se sacarán conclusiones de todo lo recogido en este apartado.

2.1 *Blockchain* y Distributed Ledger Technology (DLT)

Los términos Distributed Ledger Technology (DLT) y *Blockchain* suelen confundirse y emplearse para lo mismo. Esto es así porque ambos tienen un mismo concepto de base: son libros de registro digitalizados y descentralizados. Sin embargo, existen algunas características diferenciadoras, [3].

Una DLT puede interpretarse como una base de datos no centralizada, es decir, que no existe una autoridad central que verifique ni modifique en ninguna medida la información contenida y que, además, al disponer de un registro distribuido, se caracteriza por la transparencia de su contenido, por lo que evita en gran medida la tergiversación, manipulación y fraude de información.

Por otro lado, una *Blockchain* es una DLT cuyos bloques que conforman la base de datos están unidos entre sí mediante un mecanismo criptográfico llamado hash[1] para formar la cadena.

Las DLT/*Blockchain* poseen una naturaleza distribuida y descentralizada que les permite mantener múltiples copias del libro mayor (registro de las transacciones) a partir de datos consensuados y sin la necesidad de participe un tercero. Por tanto, proporcionan altos niveles de eficiencia, transparencia y confianza, [4].

2.1.1. *Blockchain*

Una *Blockchain*, o cadena de bloques, es un registro consensuado, distribuido en varios nodos de una red descentralizada y resistente a la manipulación de datos. En cada bloque que forma la cadena se almacenan: [5]

- Una cantidad de registros o transacciones validadas.
- Información relacionada con ese bloque.
- Su vinculación con el bloque anterior (mediante el hash), por lo que tiene una posición concreta e inamovible dentro de la cadena.

[1] Codifica datos para formar una cadena de caracteres única sin importar la cantidad de datos introducidos inicialmente. Se emplea para asegurar la autenticidad de datos.

Al crearse nuevos registros (transacciones), estos se verifican y validan por todos los nodos de la red para más tarde añadirse a un bloque nuevo que se enlaza a la cadena mediante el hash del bloque anterior (ver figura 2.1). La cadena completa se guarda en cada nodo que conforma la *Blockchain*.

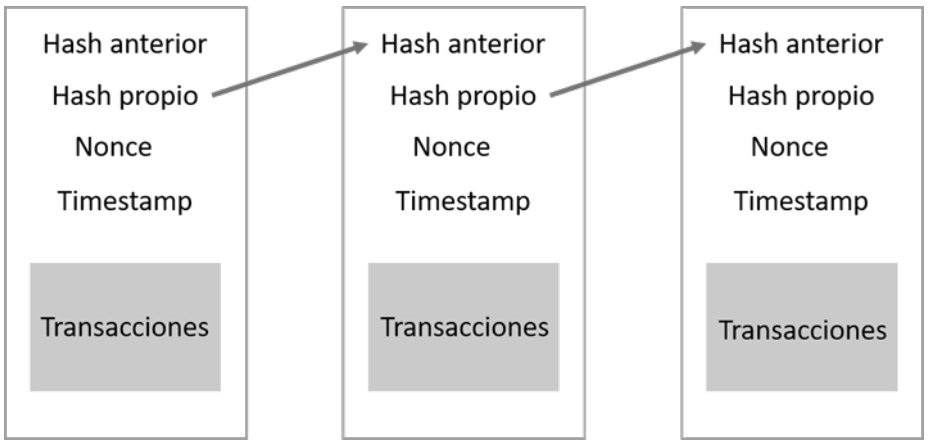

Figura 2.1: Cadena de bloques unidos por el hash del bloque anterior

La *Blockchain* mantiene un esquema distribuido (figura 2.2), el cual es una de sus principales ventajas puesto que hace que hackear dicha red sea casi imposible. Por ejemplo, uno de los posibles ataques se podría realizar es conocido como el ataque del 51 % y consiste en que los atacantes tendrían que hacerse con el control de al menos el 51 % de los nodos, lo que a nivel práctico supone controlar una enorme cantidad de poder de cómputo[2] y, aun así, podría no ser suficiente para destruir la red. Este ataque tan solo puede suceder cuando dos mineros están calculando el hash de un bloque al mismo tiempo y obtienen los mismos resultados, por tanto, la *Blockchain* se divide y se generan dos cadenas diferentes, ambas consideradas correctas, [6].

Figura 2.2: Esquemas de tipos de redes

A cada participante de la *Blockchain* se le conoce como nodo, los cuales se conectan en una red peer-to-peer (P2P) mediante la que se comunican entre sí usando un mismo lenguaje o protocolo.

Pero, además de ser una base de datos casi imposible de atacar, el uso de las funciones matemáticas denominadas hashes criptográficos es otra de sus principales virtudes, ya que permiten que todos los registros se almacenen de forma segura e impiden la modificación de la información una vez es introducida dentro de la cadena.

[2]Según lo calculado por el portal Crypto51, en el caso de Bitcoin sería equivalente a un gasto de 5000,000 dólares por hora.

Por tanto, la tecnología *Blockchain* emplea la criptografía para asegurar que los registros estén cifrados y solamente los poseedores de una clave privada puedan acceder a ellos (salvo en las *Blockchain* públicas las cuales están abiertas a cualquier usuario sin restricciones), no para modificarlos, sino para demostrar que son propietarios de los datos que ahí se han registrado. Todo ello sin que haya intermediarios en el proceso para certificar la autenticidad de dicha clave, pues la tecnología *Blockchain* realiza la tarea de certificación por sí misma.

2.1.1.1. Tipos de *Blockchain*

Entre los tipos de cadenas de bloques más conocidas se encuentran las cadenas: públicas, privadas e híbridas, [7] [8].

Blockchain pública:

Este fue el primer tipo de *Blockchain* y se caracteriza por tener abierto al público sus datos, *software* y desarrollo, por lo que cualquier persona puede desarrollarlos, revisarlos o mejorarlos. Entre las características de este tipo de *Blockchain* están:

- Permiten que cualquier persona pueda formar parte sin restricciones.
- El funcionamiento de la red es totalmente transparente y abierto. Los datos almacenados están disponibles en todo momento sin restricciones.
- No existen entidades centralizadas que regulen su funcionamiento.

Ejemplos de este tipo de cadenas de bloques son: Bitcoin, Ethereum y Monero.

Blockchain privada o permisionada:

Posteriormente, con la evolución de la tecnología *Blockchain* y su expansión, muchos actores del sector empresarial se interesaron por esta. Por lo que se comenzaron a desarrollar soluciones *Blockchain* privadas.

Esta clase de *Blockchain*, a diferencia de la pública, depende de una unidad central que permite dar o revocar el acceso a los usuarios, así como controlar sus funciones y permisos dentro de la misma. Entre las características de este tipo de *Blockchain* están:

- La unidad central de control es la única que puede dar o revocar el permiso para que un usuario acceda a la red.
- El acceso al libro de transacciones es privado.
- Normalmente no cuenta con criptomonedas ni se minan bloques.
- La red está controlada por una sola empresa o entidad.

Algunos de los desarrollos de *Blockchain* privadas más importantes del mundo criptográfico son: Corda y Quorum.

Blockchain híbrida:

Esta clase de *Blockchain* se trata de una mezcla entre las *Blockchain* vistas previamente. En las híbridas la participación en la red es privada, por lo que para acceder a ella existe una o varias unidades centrales que dan o revocan los permisos.

Por otro lado, el libro de transacciones es privado, por lo que solo los usuarios de la red que poseen una clave de acceso pueden consultarlo.

El objetivo de este tipo de *Blockchain* es mantener la privacidad sin que la red esté controlada por una sola organización, sino por un grupo de compañías o entidades.

Entre las características de este tipo de *Blockchain* están:

- La unidad central de control es la única que puede dar o revocar el permiso para que un usuario acceda a la red.
- El acceso al libro de transacciones es privado.
- No cuenta con criptomonedas ni se minan bloques.
- Al ser parcialmente descentralizada, esta red aporta un alto nivel de seguridad y transparencia.
- La red está controlada por una varias empresas o entidades.

Uno de los desarrollos de *Blockchain* híbrida más importantes es: Hyperledger.

	Libro de transacciones	**Participantes**	**Seguridad**	**Velocidad de transacción**
***Blockchain* pública**	Público	Sin necesidad de un permiso y anónimos	Mecanismos de consenso descentralizados	Lenta
***Blockchain* privada**	Privado	Con permisos e identidades conocidas	Algoritmos de consenso propios	Muy ágil y rápida
***Blockchain* híbrida**	Privado	Con permisos e identidades conocidas	Algoritmos de consenso propios	Muy ágil y rápida

Tabla 2.1: Tipos de *Blockchain*

2.1.1.2. *Smart Contracts*

Un Contrato Inteligente (*Smart Contracts* en inglés) es un conjunto de reglas o programa informático que facilita, asegura, hace cumplir y ejecuta acuerdos contractuales entre dos o más partes.

Dicho programa se encuentra en un sistema no controlado por ninguna de las partes y ejecuta un contrato automático entre ellas. Con lo cual se emplea para realizar acciones automáticas sin necesidad de que un tercero intervenga.

Los Contratos Inteligentes funcionan de la siguiente manera: cuando se dispara una condición preprogramada pasan a ejecutar la cláusula contractual relacionada.

Tienen como objetivo brindar seguridad, reducir costos de transacción asociados a la contratación de un auditor y automatizar y agilizar la respuesta ante el cumplimiento o incumplimiento de una de sus cláusulas. [9]

2.2 Principales marcos de trabajo (*frameworks*)

Para facilitar la creación de *Blockchain* y posterior despliegue de aplicaciones en la misma, existen diversos marcos de trabajo que automatizan y ponen a disposición del desarrollador una serie de recursos que disminuyen en gran medida el tiempo necesario para llevar a cabo un proyecto. Los cuatro más conocidos son:

2.2.1. Hyperledger

Hyperledger es una plataforma *Blockchain* de código libre gestionada por la fundación The Linux Foundation y creada con el objetivo de obtener una solución *Blockchain* para el sector empresarial reduciendo costes y la complejidad a la hora de hacer negocios.

El proyecto fue lanzado en 2016 y hoy en día consiste en nueve proyectos, cinco de ellos son plataformas *Blockchain*: [10]

- **Hyperledger Burrow:** llamada previamente Monax, es una *Blockchain* privada basada en Ethereum, la cual permite desarrollar Contratos Inteligentes en Solidity.
- **Hyperledger Fabric:** está dirigida al sector empresarial. Se trata de una plataforma multidisciplinar que permite desarrollar Contratos Inteligentes (llamados en este caso chaincodes) en su mayoría en el lenguaje de programación de Google (Golang), aunque también en JavaScript y TypeScript. Esta *Blockchain* posee un diseño muy flexible puesto que permite decidir qué algoritmo de consenso utilizar en la red (*pluggable consensus*).
- **Hyperledger Indy:** en una plataforma cuya meta es proponer una solución de identidad digital soberana. Pues creen que los datos personales deben ser administrados por su dueño, por lo que apuesta por el "Zero Knowledge Protocol" que tiene como objetivo revelar la cantidad de información justa en procesos en los que usa la identidad digital.
- **Hyperledger Iroha:** es una plataforma *Blockchain* simple y modularizada que permite el desarrollo de Contratos Inteligentes en Java y que también apuesta por la identidad digital soberana. Esta plataforma fue desarrollada en Japón por Soramitsu, Hitachi y NTT Data and Colu.
- **Hyperledger Sawtooth:** este proyecto procede en gran medida de Intel. Es una *Blockchain* privada de ámbito empresarial que incorpora la capacidad de despliegue de Contratos Inteligentes (desarrollados en Solidity).

Cabe decir que todos los marcos de trabajo de Hyperledger incluyen: un registro distribuido inmutable, un algoritmo de consenso entre los nodos para decidir el estado del registro, privacidad de transacciones y acceso permisionado y Contratos Inteligentes para la lógica de negocio.

Por otro lado, es importante destacar que Hyperledger es una red *Blockchain* híbrida. De esta forma se reduce el riesgo de que se una cualquier tipo de usuario (bien o

malintencionado), dejando unirse a la red solo a participantes conocidos que deben ser aprobados por el resto de los participantes y, si descubren que uno de ellos está actuando malintencionadamente siempre pueden denegarle el acceso consensuándolo entre ellos.

Asimismo, "Hyperledger está pensado para redes *Blockchain* con un reducido número de participantes conocidos lo que le permite usar algoritmos de consenso mucho más eficientes que los algoritmos en redes públicas alcanzando un rendimiento mucho mayor que los alcanzados en Bitcoin o Ethereum", [11].

Por último, también implementa la privacidad de transacciones a través de canales, subredes dentro de la propia red que se pueden privatizar para que solo unos pocos tengan acceso a ellos.

2.2.2. Ethereum

Desde su lanzamiento en 2015 Ethereum es la *Blockchain* programable líder mundial.

Ethereum tiene una criptomoneda propia llamada Ether (ETH), es decir, dinero digital que puede enviarse a cualquier persona instantáneamente. La oferta de esta liquidez digital no está controlada por ningún gobierno ni compañía, sino que es descentralizada.

Además, Ethereum es programable, por lo que se puede utilizar para crear nuevas aplicaciones para controlar activos digitales, mejorar la trazabilidad de productos, etc.

Estas aplicaciones descentralizadas (o "Dapps") son seguras puesto que una vez que se han guardado en Ethereum siempre funcionan tal como fueron programadas y al ser descentralizadas, ninguna entidad o persona las controle ni intervenga.

"El propósito inicial del proyecto Ethereum es el de descentralizar la web mediante la introducción de cuatro componentes como parte de la hoja de ruta de su Web 3.0: publicación de contenido estático, mensajes dinámicos, transacciones confiables y una interfaz de usuario integrada y funcional. Estos componentes están diseñados para reemplazar algunos aspectos de la experiencia Web de una manera completamente descentralizada y anónima", [12].

Las principales diferencias con Hyperledger son las siguientes:

- Es una plataforma B2C (Business to Consumer) que permite la creación de aplicaciones generalizadas orientadas a que una empresa ofrezca sus productos directamente a los consumidores, mientras que Hyperledger es B2B (Business to Business) y está más enfocada a conectar empresas o instituciones entre sí.
- Es totalmente transparente, cualquier persona puede ver las transacciones de otros. Hyperledger las mantiene confidenciales, solo visibles para los participantes en las mismas.
- Permite la creación de una red *Blockchain* de cualquier clase.
- Utiliza algoritmos de consenso mediante el minado de bloques (Proof of Work o PoW), aunque están en proceso de cambiarlo por un algoritmo menos contaminante a nivel energético (Proof of Stake o PoS).
- Los contratos están escritos en lenguaje de programación Solidity.
- Tiene una criptomoneda propia: Ether.

Cabe decir que existe un *framework* muy conocido que tiene su base en Ethereum y que permite la realización de transacciones confidenciales (transparentes solo para los

participantes en la transacción) en una red autorizada y enfocada al sector financiero. Dicho *framework* recibe el nombre de Quorum.

2.2.3. Multichain

MultiChain es una plataforma que posibilita el diseño, implementación y operación con registros distribuidos de forma rápida y sencilla.

Esta herramienta permite crear *Blockchain* privadas, pudiendo decidir quién puede conectarse a la plataforma para enviar y recibir transacciones; además de activos y bloques.

Asimismo, también permite decidir si hacer pública la *Blockchain* inicialmente privada. Además, la plataforma permite crear criptomonedas propias para realizar intercambios y mantener el registro de transacciones en la red, [13].

Las diferencias más claras con las opciones anteriores son:

- No permite redactar Contratos Inteligentes, a diferencia de las dos anteriores.
- Trabaja con Bitcoins.
- Se emplea para la creación de *Blockchain*, pero no para aplicaciones descentralizadas, por lo que no dispone de ayuda para la creación de estas.

2.2.4. R3 Corda

Corda es un proyecto *Blockchain* de código abierto, orientado al sector empresarial desde sus inicios. Fue desarrollado por R3, una empresa de *software* de *Blockchain* empresarial, en colaboración con más de 200 instituciones financieras y bancos internacionales. [14]

Sin embargo, Corda no se considera una cadena de bloques per se, sino una DLT que combina las características de la cadena de bloques con las DLT[3].

Entre sus características están: no tiene criptomoneda propia, solo comparte datos con los participantes requeridos y ofrece aplicaciones interoperables para el sector financiero y comercial (CorDapps).

En esta *Blockchain*, las transacciones son privadas solo para las partes incluidas en la transacción. En las *Blockchain* públicas (véase Ethereum) sus transacciones transparentes para todos los participantes de la red y deben asegurarse de que estas sean válidas. En Corda, la confianza se consigue gracias a la autoridad de certificación de la red.

Por lo que, al eliminar la necesidad de consenso entre todas las partes de la red, pues es necesario únicamente entre los participantes de una transacción, se reduce el tiempo para completar una acción.

Con todo esto Corda es un libro mayor global utilizado por las empresas para mantener un libro mayor compartido de transacciones. Por tanto, la privacidad y la identidad son los dos principios de esta *Blockchain*.

[3]Base de datos descentralizada, mientras que, en *Blockchain*, los datos se almacenan en bloques que están vinculados entre sí mediante un mecanismo criptográfico llamado hash. Para más información leer apartado 2.1. de la memoria.

En resumen, este proyecto *Blockchain* comparte características de las vistas anteriormente:

- Al igual que Hyperledger no posee una criptomoneda propia y mantiene las transacciones confidenciales (solo los participantes pueden verlas).
- Permite realizar aplicaciones descentralizadas y crear Contratos Inteligentes (en el lenguaje de programación Kotlin) como Ethereum y Hyperledger.

Sin embargo, está especializada en el sector financiero.

2.3 Aplicaciones prácticas

El sector económico y financiero ha sido el primero en el cual la tecnología de la cadena de bloques reflejó sus numerosas ventajas, siendo la criptomoneda Bitcoin una de las aplicaciones más notables. Esto es así debido a que la seguridad, rapidez y descentralización que ofrece esta tecnología supera con creces a cualquier forma de transferir dinero en la actualidad.

Sin embargo, el término de "Tecnología *Blockchain*" más allá de las aplicaciones financieras y se puede ver su utilización en otras áreas industriales y sociales.

Algunos ejemplos de proyectos son: [15]

- **R3Cev:** en donde los 40 bancos más grandes del mundo se han unido para buscar soluciones a sus procesos financieros mediante esta herramienta.
- **Storj:** iniciativa de almacenamiento basada en *Blockchain* y disponible para cualquier usuario de la red.
- **PoE:** Proof of Existence es un servicio de certificación online de todo tipo de documentos.

Blockchain también tiene aplicación en el Internet de las cosas y en proyectos de trazabilidad de productos. En cuanto a este último encontramos diversos proyectos como son:

- **PorkChain:** una aplicación que usa Ethereum por debajo para certificar datos referentes al sector agroalimentario, es decir, funciona como servicio de verificación de la información. El objetivo final de Porkchain es reemplazar los documentos físicos actuales por documentos digitales certificados por la red, [16].
- **Veritas:** proyecto creado con el objetivo de desarrollar un sistema de información de registro de trazabilidad confiable para las empresas de envase alimentario en la Comunidad Valenciana, que permita obtener la trazabilidad completa de fabricación de los envases de alimentos en situaciones de alerta sanitaria, [17].
- **Walmart:** proyecto que emplea el *software* de Hyperledger Fabric para establecer sistemas de trazabilidad de alimentos, mejorando la seguridad alimentaria al ser capaces de rastrear la procedencia o destino de un determinado alimento, [18].
- **Building Blocks:** el Programa Mundial de Alimentos de la ONU ha creado este proyecto con el cual pretende utilizar la red de Ethereum para hacer mucho más rápidas, económicas y seguras las transferencias basadas en efectivo que la organización provee a los más necesitados. Así como establecer un registro de hacia dónde se dirige la ayuda alimentaria y atenuar el riesgo por fraude de identidad o una mala gestión de los datos, [19].

2.4 Conclusiones

En la época actual el desarrollo de *Blockchain*, su aplicabilidad y herramientas complementarias como lo son los Contratos Inteligentes, todavía no han llegado a su máximo potencial.

Sin embargo, mientras se perfilan y realizan avances en esta tecnología algunos proyectos han decidido tomar la iniciativa contribuyendo a la mejora económica, logística y humanitaria de la sociedad.

La elección de qué marco de trabajo usar para el desarrollo de estas redes es altamente importante y para ello se deben de tener claras las características que se quiere que la aplicación a desarrollar posea en un futuro. Tales distinciones pueden basarse en índices de privacidad, transparencia, seguridad o minimización de la contaminación energética.

Por ejemplo, Ethereum (como cadena de bloques pública) contamina más energéticamente y es más lento debido al algoritmo de consenso que utiliza. No obstante, es completamente transparente y esto puede provocar inseguridad en los usuarios debido a la posible existencia de un conflicto con su idea de privacidad o, al contrario, mayor confianza.

Por otro lado, ni Hyperledger ni Corda disponen de una criptomoneda propia. Por lo que en numerosas ocasiones se utiliza Ethereum y así se incentiva al uso de las Dapps mediante la posibilidad de ganar Ether tras realizar una determinada acción en la aplicación. Cabe decir que Hyperledger también permite la creación de criptomonedas, pero Ethereum hace su creación y utilización más sencilla.

CAPÍTULO 3
Análisis del problema

En este capítulo se va a analizar el problema que se ha planteado en apartados anteriores junto con algunos de los aspectos más importantes del sistema, como son la seguridad o el marco legal y ético, y también se va a proponer una solución, un plan de trabajo acorde con ella y se estimará el presupuesto resultante de dicha solución escogida.

Previo al comienzo de este capítulo cabe decir que, este trabajo se centra en las donaciones que se realizan a países en vías de desarrollo, porque existen problemas de falsificación de datos pues algunos artículos "desaparecen" de los registros o nunca llegan a su destino. Esto genera una falta de confianza y credibilidad por parte de la población general hacia este tipo de acciones. Asimismo, existe una alta escasez de páginas web que permitan realizar donaciones y rastrear la localización de estas, por lo que las personas que las realizan nunca saben si estas han ido a parar a manos de niños verdaderamente necesitados.

3.1 Análisis de seguridad

En cuanto a la seguridad en este tipo de campañas, organizaciones, asociaciones y demás, es escasa y en pequeñas asociaciones nula. Los problemas que suceden se pueden clasificar en:

- El donante no tiene, en muchos casos, forma de saber si lo que ha donado ha llegado a las manos de quien lo necesitaba o no.
- Los registros son manipulados con suma facilidad.
- Los altos cargos desvían grandes cantidades de donaciones sin que estas consten en ningún lugar.

Algunos ejemplos de donaciones que se convierten en estafas son el caso de Salvation Army, Goodwill y Anesvad (esta última en España). En el primer caso, en 2017 el exdirector ejecutivo del centro, David Rennie, llevó a cabo un plan para vender, en vez de donar a gente necesitada, grandes cantidades de artículos donados, sobre todo juguetes y alimentos, [20]. Goodwill no fue muy diferente, también altos cargos ejecutivos se vieron envueltos en la malversación de fondos, donaciones que "desaparecían", que "no llegaban al centro", muchas de las cuales se las quedaban para su uso y disfrute personal, [21]. En última instancia, el presidente de Anesvad, José Luís Gamarra, fue acusado de la apropiación indebida de fondos de dicha ONG, la cual gestionaba 35 millones de euros anualmente, [22]. Todos estos casos son ejemplos de cómo altos cargos deciden apropiarse o desviar los objetos o donaciones que confiadamente aportan otras personas, creando

inseguridad en ellas, porque una vez que estas regularidades son descubiertas, queda la duda de si volverán a ocurrir, no con las mismas personas, pero sí con el mismo resultado. No obstante, la falsificación y malversación de fondos y bienes no se da solo en las altas esferas de las asociaciones que claman no tener ánimo de lucro, sino que, en pequeñas asociaciones, recogidas de juguetes en parroquias, etc., también hay casos, pues las personas que separan los juguetes deciden recoger lo que consideran más interesante y en mejor estado para dárselo a sus familias.

Por todo esto, es necesario un sistema que permita al donante realizar un seguimiento de sus donaciones con un registro claro e imposible de modificar, como el que se obtendría de la realización de una aplicación descentralizada basada en tecnología *Blockchain*. De esta forma resultaría más difícil que alguien pretendiese malversar la información o apropiarse de los bienes, al mismo tiempo que facilitaría la localización y descubrimiento de este tipo de acciones junto con sus perpetradores.

Por lo tanto, los diversos problemas encontrados en el ámbito de la seguridad en este tipo de procesos se pueden subsanar con el uso de *Blockchain*, aportando, por ende, una gran seguridad y, para evitar filtraciones de información se utiliza una red híbrida donde el acceso a los datos almacenados en la *Blockchain* para cada usuario estará condicionado por su tipo (los donantes solo pueden ver la parte que les concierne mientras que los administradores son capaces de obtener más información del sistema).

3.2 Análisis del marco legal y ético

En este apartado se analizarán las implicaciones legales y éticas que afectan al proyecto.

3.2.1. Análisis de la protección de datos

Los datos utilizados en este proyecto no son reales, sin embargo, si se lleva a cabo con datos reales hay algunos aspectos que se deben tener en cuenta.

Como normalmente las donaciones son anónimas, no se guarda información sensible, tan solo se introduce en el sistema el tipo y cantidad del producto, nada que identifique a su donante.

Sin embargo, si se realiza un sistema trazable como es el objetivo de este proyecto, dichos objetos donados deberían de tener un código de identificación único asociado a un donante concreto. De esta forma, en lo referente al apartado de la protección de datos, se almacenaría como información sensible el nombre del donante, el cual no sería revelado a los usuarios normales (donantes), tan solo se utilizaría para que, si un donante quiere realizar el seguimiento de sus donaciones, no tenga por qué buscar artículo por artículo, sino que se le muestren todos los artículos asociados con él (donaciones) y, al mismo tiempo, pueda ver solo los suyos y no los de otra persona.

Así se protegería la única información sensible almacenada de los usuarios, la cual solo podría ser vista por los administradores o trabajadores que lo requiriesen. Al mismo tiempo que permitiría realizar un seguimiento más cómodo y seguro para el usuario.

3.2.2. Otros aspectos legales

En cuanto a los aspectos legales, cabe decir que todas las librerías y fuentes de código que se han utilizado en el proyecto son de código libre.

3.2.3. Ética

A nivel ético no se ha realizado ninguna acción ilícita para favorecer a ningún usuario o institución, puesto que este es uno de los problemas observados en la misión del proyecto (falsificaciones y apropiaciones indebidas de bienes por parte de altos cargos con el fin de lucrarse). Al contrario, con tal de preservar la privacidad de los datos de todos los usuarios y aumentar por ello su confianza y seguridad, estos únicamente pueden ver la parte que les corresponde, de tal forma que la estructura de la *Blockchain* es invisible para ellos, pudiendo ser transparente para aquellos trabajadores que necesitasen acceder a cierta información más sensible. Asimismo, se incorpora un apartado de trazabilidad para evitar acciones indebidas por parte de la gente envuelta en todo el proceso de las donaciones.

3.3 Identificación y análisis de soluciones posibles

Existen diversas soluciones a los problemas encontrados, algunas de las cuales son:

- El almacenamiento de la información en una base de datos privada. La cual almacenaría la información de los lotes de objetos donados, modificando su localización (tanto la del lote si se ha movido un objeto concreto como la del lugar si se ha enviado a otro sitio) cuando se necesitase, y la de los usuarios del sistema, para relacionar donaciones con donante y permitir que el usuario pueda realizar un seguimiento de sus artículos.
- Una segunda solución pasa por utilizar una cadena de bloques o *Blockchain*, en donde la información no puede ser alterada, lo que se escribe en ella no puede ser borrado ni modificado. En este caso se utilizaría un mayor espacio para almacenar los cambios, pues al trasladar un lote o contenedor, este deberá de quedar nuevamente registrado en la cadena añadiendo un bloque más, ya que el nuevo cambio, según la filosofía de *Blockchain*, debe de registrarse como un nuevo bloque, evitando así operaciones de modificación de la información existente.
    - Dentro de esta opción existen muchas *Blockchain* disponibles, cada una dispone de unos beneficios y desventajas que ya se han comentado en el capítulo 2 y que se volverán a recordar en el siguiente apartado.

Siguiendo los requisitos que se han marcado en el apartado de seguridad y que se han destacado como parte del problema a solucionar, llegamos a la siguiente clasificación:

	Base de datos	***Blockchain***
Seguridad (modificación y falsificación de la información)	Resulta más fácil falsificar los datos al poderse modificar libremente	Al no poderse modificar ni borrar los datos, la información almacenada será veraz y fiable
Almacenamiento de información	Al modificarse los datos introducidos la cantidad de información almacenada en menor	Al crearse un nuevo bloque cuando se "modifica" un dato, se aumenta la cantidad de datos a almacenar

Velocidad	Es más rápida a la hora de realizar las operaciones	La velocidad de las operaciones con la cadena es más lenta si la *Blockchain* es pública, si no dependerá del algoritmo escogido
Automatización (sin necesidad de terceros)	No hay	Al disponer de un Contrato Inteligente que regula la información almacenada en la *Blockchain*, se automatizan ciertos procesos sin necesidad de terceros
Gran gasto y contaminación energética	No hay	Dependiendo del tipo de algoritmo y de *Blockchain* puede ser poco contaminante o, suponer un problema en grandes cantidades de información

Tabla 3.1: Comparativa entre posibles soluciones

3.4 Solución propuesta

La solución que se propone es pues el empleo de una base de datos no relacional (MongoDB) para almacenar los usuarios del sistema junto con las contraseñas encriptadas. Mientras que la parte más costosa y significativa se almacenaría en una *Blockchain* para evitar falsificaciones de información, que es, al fin y al cabo, una de las razones clave por las que se ha realizado este estudio, junto con las facilidades que esto supone para desarrollar un sistema de trazabilidad seguro y confiable.

Asimismo, la *Blockchain* utilizada será Hyperledger, concretamente Fabric, pues no es necesaria la creación de una criptomoneda (en el caso que nos atañe sería inservible), consume mucha menos energía que las que emplean el algoritmo de PoW[1] pues no todos sus nodos necesitan participar en el sistema de consenso, consta de un Contrato Inteligente llamado *chaincode* para automatizar e interactuar con la *Blockchain*, es híbrida (tan solo la gente con acceso puede ver la red) y más rápida y segura al realizar operaciones que otras *Blockchain* muy conocidas (véase Ethereum).

[1]Proof of Work: algoritmo de consenso mediante el minado de bloques

3.5 Plan de Trabajo

Para llevar a cabo este trabajo, se ha decidido agrupar las tareas en un esquema de desglose del trabajo (EDT).

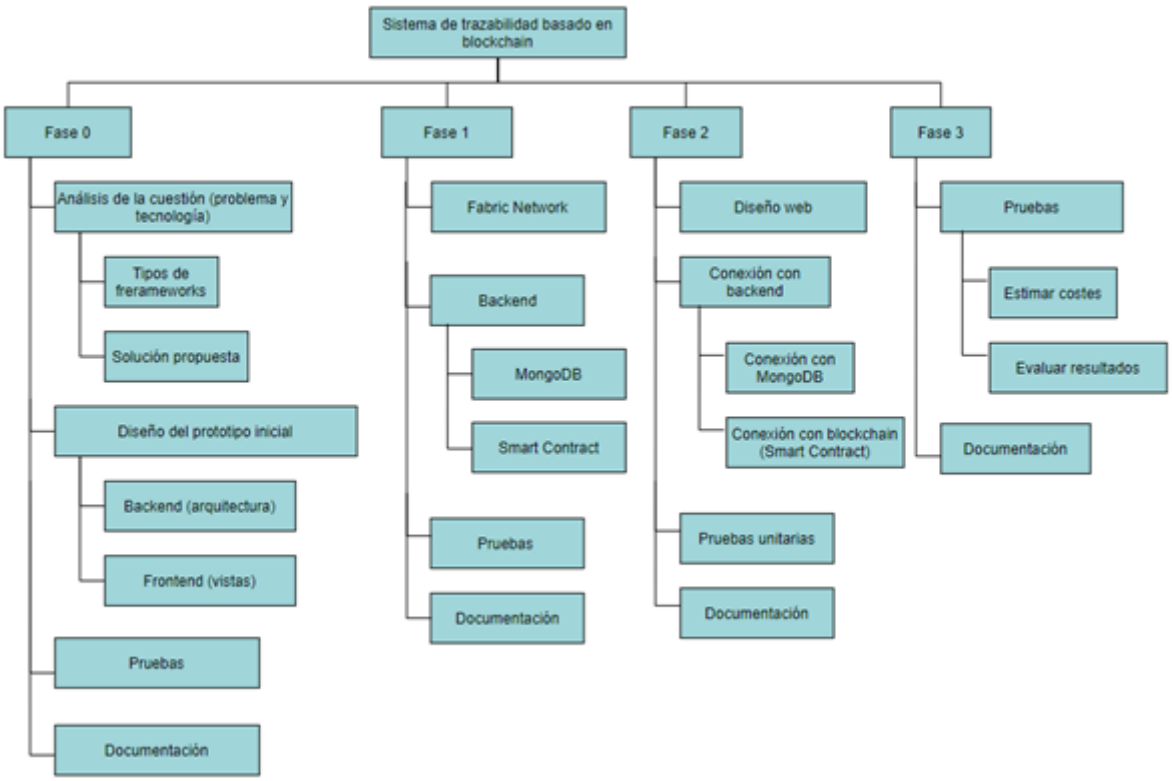

Figura 3.1: Esquema de Desglose del Trabajo (EDT)

En el esquema de la figura 3.1 se representan distintas fases para la implementación del proyecto, siendo la fase 0 la correspondiente al análisis previo del problema y contexto tecnológico, y la realización de los prototipos. La 1 se relaciona con la construcción del *backend*, conexiones con MongoDB, el *chaincode* (código y conexión con este) y la *Blockchain*. En la 2 se implementa el *frontend* y se enlaza este con el *backend*. Y, por último, en la 3 se realizan pruebas para estimar costes de tiempo y evaluar los resultados obtenidos con la propuesta de este trabajo, al mismo tiempo que se comentan posibles ampliaciones de cara al futuro.

A continuación, veremos cada una de las fases más en detalle.

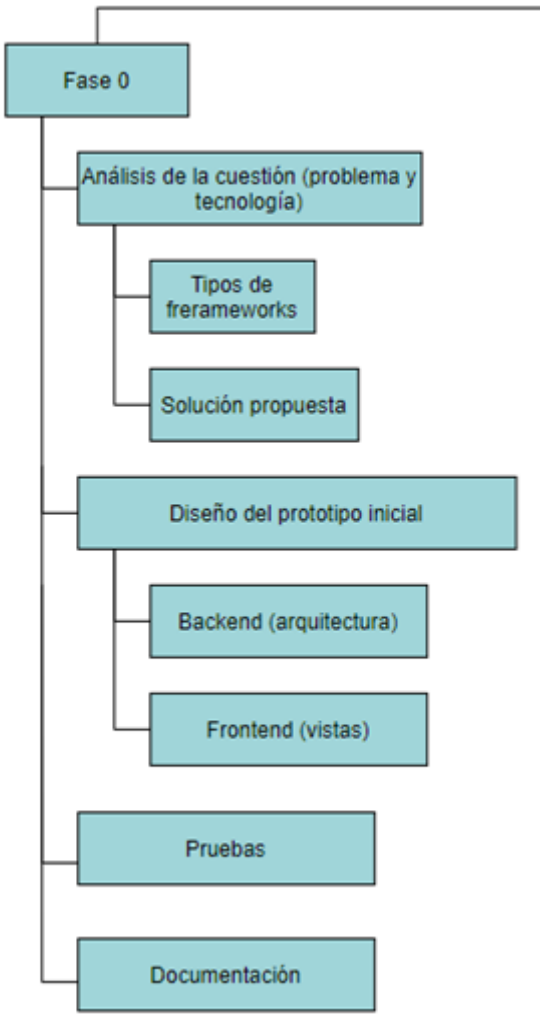

Figura 3.2: Primera rama de la EDT

En la fase 0 (figura 3.2), encontramos las siguientes tareas:

0.1 **Análisis de la cuestión (problema y tecnología):** esta tarea se subdivide en dos subtareas que tienen como objetivo analizar la problemática y el contexto tecnológico en el que se va a basar la solución propuesta. Se estima que tendrá una duración de 40h.

0.1.1 **Tipos de *frameworks*:** en esta subtarea analiza los conceptos de DLT y *Blockchain* junto con sus diferencias. Seguidamente, destaca los principales *frameworks* de *Blockchain* que existen y sus características, para finalmente concluir argumentado cuál se va a escoger y por qué. Se estima que tendrá una duración de 25h.

0.1.2 **Solución propuesta:** esta subtarea compara soluciones para el estado de la cuestión (problemática en el sector de las donaciones), así como, selecciona la mejor solución para el objetivo a conseguir. Se estima que tendrá una duración de 15h.

0.2 **Diseño del prototipo inicial:** esta tarea tiene como meta el diseño de la arquitectura del sistema y las vistas de la página web. Se estima que tendrá una duración de 12h.

0.2.1 ***Backend* (arquitectura):** en esta subtarea se realizará el diseño arquitectónico de la aplicación. Se estima que tendrá una duración de 4h.

0.2.2 ***Frontend* (vistas):** en esta subtarea se realizará un prototipo inicial de la interfaz de la página web. Se estima que tendrá una duración de 8h.

0.3 **Pruebas:** se realizarán pruebas sobre el prototipo para comprobar si es intuitivo y sencillo para los usuarios. Se estima que tendrá una duración de 4h.

0.4 **Documentación:** finalmente se recogerán los pasos, análisis y conclusiones acordadas a lo largo de esa fase. Se estima que tendrá una duración de 15h.

En suma, esta fase tendrá una duración total estimada de 71 horas.

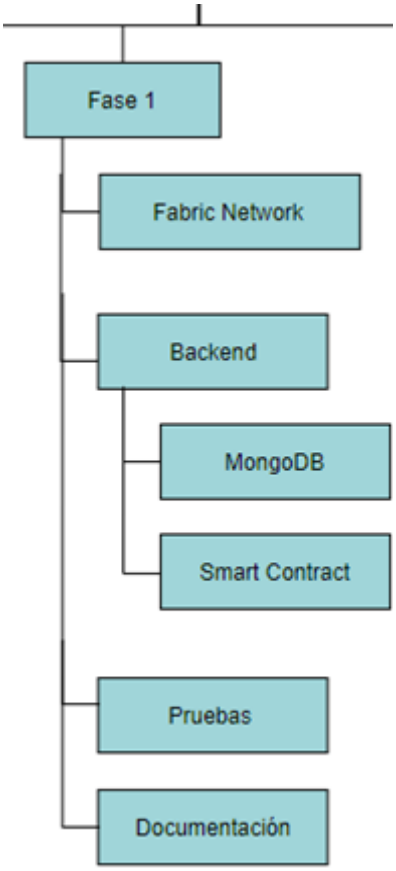

Figura 3.3: Segunda rama de la EDT

En la fase 1 (figura 3.3), encontramos las siguientes tareas:

1.1 **Fabric Network:** en esta tarea se realizará todo lo necesario para crear la red híbrida que sustente la *Blockchain*, los canales, etc. Se estima que tendrá una duración de 25h.

1.2 ***Backend*:** esta tarea se divide en dos subtareas que tienen como objetivo la implementación de los controladores y servicios para MongoDB y el *chaincode* que se comunicará con la *Blockchain*. Se estima que tendrá una duración de 35h.

1.2.1 **MongoDB:** esta subtarea implica la creación del controlador y servicios necesarios para registrar y comprobar la identidad de los usuarios que acceden a la página web. Se estima que tendrá una duración de 5h.

1.2.2 ***Smart Contract*:** esta subtarea tiene como objetivos la creación del controlador que accede al *chaincode* (o Contrato Inteligente), así como el código de este. Se estima que tendrá una duración de 30h.

1.3 **Pruebas:** en esta tarea se realizarán las pruebas necesarias para comprobar el correcto funcionamiento del *backend*, su comunicación con MongoDB, con el *chaincode* y con la *Blockchain*. Se estima que tendrá una duración de 10h.

1.4 **Documentación:** finalmente se documentarán los pasos, métodos y demás información necesaria y utilizada a lo largo de esta fase. Se estima que tendrá una duración de 15h.

Con todo esto esta fase durará aproximadamente 85h.

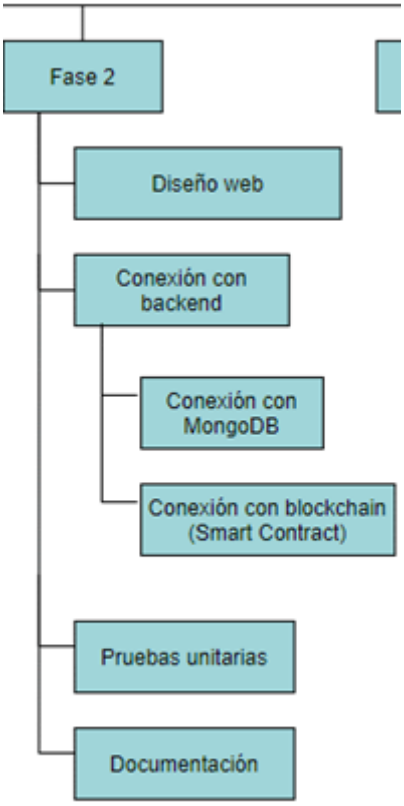

Figura 3.4: Tercera rama de la EDT

En la fase 2 (figura 3.4), encontramos las siguientes tareas:

2.1 **Diseño web:** en esta tarea se realizará el diseño de la interfaz partiendo del prototipo de la fase 0 y de los cambios que requieran las vistas para adaptarse a los datos recibidos del *backend*. Se estima que tendrá una duración de 25h.

2.2 **Conexión con *backend*:** en esta tarea se conectarán el *frontend* y *backend*. Se estima que tendrá una duración de 15h.

2.2.1 **Conexión con MongoDB:** esta subtarea se encarga de conectar el *frontend* con la parte del *backend* que lleva el servicio de MongoDB. Se estima que tendrá una duración de 5h.

2.2.2 **Conexión con *Blockchain*:** esta subtarea conecta el *frontend* con los servicios que acceden al *chaincode* y posteriormente a la *Blockchain*. Se estima que tendrá una duración de 10h.

2.3 **Pruebas unitarias:** se realizarán pruebas del funcionamiento. Se estima que tendrá una duración de 10h.

2.4 **Documentación:** se documentarán las modificaciones y aportes que se hayan realizado a la interfaz, así como a otras partes de la aplicación. Se estima que tendrá una duración de 10h.

En resumen, esta fase durará unas 60 horas.

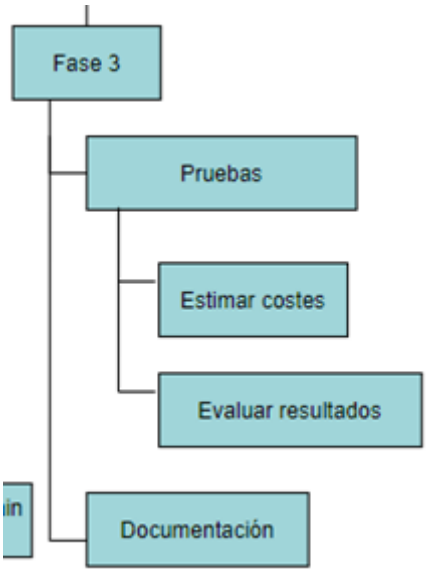

Figura 3.5: Cuarta rama de la EDT

En la fase 3 (figura 3.5), encontramos las siguientes tareas:

3.1 **Pruebas:** esta tarea se divide en dos subtareas que tienen como objetivo la evaluación de tiempos y resultados, así como del funcionamiento correcto de toda la aplicación en conjunto. Se estima que tendrá una duración de 14h.

3.1.1 **Estimar costes:** se estimarán costes de tiempo tras realizar las pruebas y ver que el sistema funciona como debe. Se estima que tendrá una duración de 6h.

3.1.2 **Evaluar resultados:** se evaluarán los resultados obtenidos, teniendo en cuenta el tipo de aplicación que es, los objetivos marcados para mejorar con la propuesta, la tecnología escogida, etc. Se estima que tendrá una duración de 8h.

3.2 **Documentación:** se documentarán los resultados de la tarea anterior junto con las conclusiones a las que se ha llegado y posibles mejoras. Se estima que tendrá una duración de 10h.

Esta fase tendrá una duración estimada de 24 horas.

En conclusión, el plan de trabajo (EDT) tiene una duración inicial aproximada de 240 horas, pero tras realizar el trabajo la duración total real ha sido de 283 horas pues el desarrollo de la *Blockchain* y del Contrato Inteligente, junto con la búsqueda bibliográfica

sobre el funcionamiento del marco de trabajo de Hyperledger Fabric han incrementado el tiempo previsto inicialmente para cada una de las tareas relacionadas. El resto de horas que faltan hasta llegar a las 360 que deberían de haber, se han utilizado en un profundo estudio previo al TFG sobre la tecnología *Blockchain* y otros aspectos relacionados con ella (criptomonedas, algoritmos de minado y consenso, árboles de Merkle, etc.), en la realización de la memoria del proyecto, en las reuniones con los tutores, en la preparación de la defensa y en exposiciones de prueba cronometradas para ajustar bien el tiempo y contenido.

3.6 Presupuesto

Pese a que este sistema ha sido desarrollado de manera individual haciendo uso de los materiales de los que se disponía, se procede a realizar una estimación del presupuesto asociado al desarrollo y mantenimiento del proyecto, tanto en recursos humanos como materiales.

Comenzando por los recursos humanos y teniendo en cuenta los datos extraídos de la web indeed.es[2], el proyecto debería de contar con un equipo compuesto por los siguientes integrantes:

- **Un investigador:** encargado de la búsqueda bibliográfica y de probar y analizar las distintas soluciones y tecnologías propuestas. Se estima un gasto de 11,01€ la hora.
- **Un jefe de proyecto:** encargado de organizar y coordinar el resto del equipo. También realiza tareas de seguimiento sobre el desarrollo del sistema. Se estima un gasto de 19,56€ la hora.
- **Un analista:** encargado de validar y garantizar la calidad del código, así como coordinar y organizar el trabajo de los desarrolladores. Se estima un gasto de 10,22€ la hora.
- **Un jefe de equipo:** encargado de coordinar los esfuerzos de los desarrolladores del equipo de desarrollo. Se estima un gasto de 14,60€ la hora.
- **Dos desarrolladores:** encargados de la realización del código, uno del *frontend* y otro del *backend*. Se estima un gasto de 3,28€ la hora (por desarrollador).
- **Un administrador de sistemas:** encargado del correcto funcionamiento de la infraestructura, servidores y equipos. Se estima un gasto de 13,52€ la hora.

Por lo que respecta a los materiales, serían necesarios los siguientes:

- **Siete ordenadores:** los ordenadores personales son necesarios para que todos los integrantes del equipo puedan estar en comunicación, realizar tareas de desarrollo, estructuración y/o organización. Se utilizarán ordenadores básicos, para lo cual se ha escogido el modelo DELL XPS 15 [3] con un coste de 1.158,22€ por unidad (con IVA[4]), suponiendo un coste total de 8.107,54€. Pero asumiendo que la vida útil de cada uno de ellos es de 4 años y que un año consta con 250 días laborales, el coste

[2] `https://es.indeed.com/career/salaries`

[3] `https://www.amazon.es/Dell-i7-8750H-generaci%C3%B3n-procesadores-1080Pixeles/dp/B07D4ZFHW9/ref=sr_1_3__mk_es_ES=%C3%85M%C3%85%C5%BD%C3%95%C3\*%91&dchild=1&keywords=dell+xps+15&linkCode=sl2&linkId=158dc4fac90a1a38dbb18232248bce48&qid=\*1619537684&sr=8-3`

[4] Impuesto de Valor Añadido

de amortiguación durante las 283 horas del proyecto es de 163,89€ por ordenador, sumando un total de 1.147,22€.

- **Servidor:** necesario para poder mantener activo el *backend* y que este responda a la máxima cantidad de peticiones realizadas por los usuarios a través de la página web. Así como para mantener activa la *Blockchain*. El coste del servidor sería de 2.002,26€ con IVA tomando como modelo el servidor de torre PowerEdge T640[5].
- **Dominio de la web:** registrar el dominio donatedtoys.es costaría una media de 10€[6] por año, lo que se corresponde con 12,10€ tras añadirle el IVA.
- **Infraestructura**: gastos relativos a la red que interconectará los equipos. Al tratarse de un proyecto realizado en un entorno universitario se prevé que estos gastos estarán cubiertos por la universidad.

Teniendo en cuenta que el número de horas totales utilizadas para el desarrollo del proyecto es de 283 y que cada hora el gasto es de 75,47€ brutos, los gastos de los recursos humanos ascienden a 21,359.01€. Mientras que los gastos de los recursos materiales son de 3.161,58€. Obteniendo en total un presupuesto necesario de 24.520,59€ que más un 15 % de costes indirectos asciende a 28.198,68€.

Tipo	Recurso	Coste
Recursos humanos	Investigador	3.115,83
	Jefe de proyecto	5.535,48
	Analista	2,892.26
	Jefe de equipo	4.131,80
	Desarrollador x 2	1.856,48
	Administrador de sistemas	3.826,16
Recursos materiales	Ordenadores personales x 7	1.147,22
	Servidor	2.002,26
	Dominio de la página web	12,10
Coste total		24.520,59
Costes indirectos		3.678,08
Total		**28.198,68**

Tabla 3.2: Tabla del presupuesto

[5] `https://www.dell.com/es-es/work/shop/povw/poweredge-t640#features_section`
[6] `https://www.ionos.es/domaincheckresult`

CAPÍTULO 4
Diseño de la solución

A continuación, se va a explicar el diseño seleccionado para el sistema, empezando por su arquitectura, prosiguiendo con los detalles de esta y finalizando con la tecnología utilizada.

4.1 Arquitectura del sistema

En este proyecto se va a desarrollar una aplicación en base a una arquitectura cliente-servidor (figura 4.1). En donde el cliente o *frontend* se encargará de la página web y el servidor o *backend* del acceso a datos.

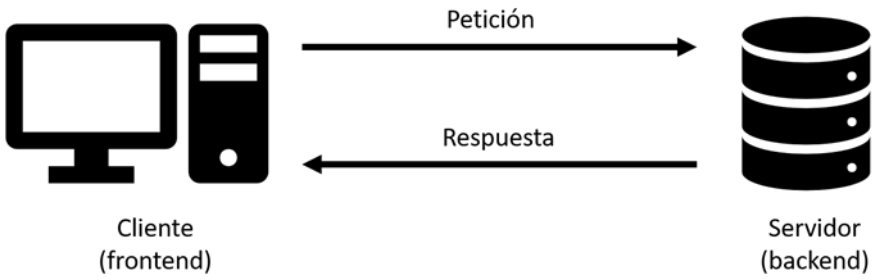

Figura 4.1: Arquitectura cliente/servidor

Con todo esto se ha seleccionado la siguiente arquitectura para el desarrollo del sistema (figura 4.2), en donde se pueden observar 3 módulos que se corresponden con 3 de las fases explicadas anteriormente en el apartado 3.5 referente a la EDT[1] (obviando la fase 0) y dicha relación se detallará en el apartado 4.2.

[1]Estructura de Desglose del Trabajo

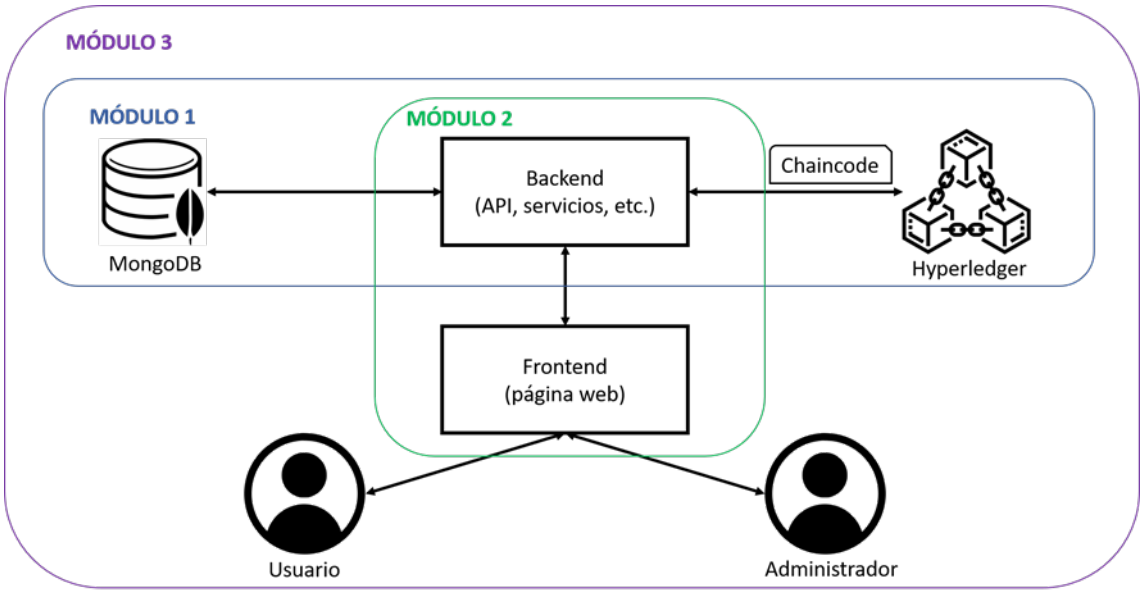

Figura 4.2: Arquitectura del sistema propuesto

Cabe decir que la *Blockchain* de Hyperledger Fabric debería de estar, en un caso ideal, junto con el *backend* en un servidor lo suficientemente potente como para así permitir interactuar a gran cantidad de usuarios paralelamente y en tiempo real con el sitio web (y, en su defecto, con la *Blockchain*).

4.2 Diseño Detallado

Como ya se ha mencionado, los módulos destacados en la figura 4.2 del apartado anterior se corresponden con las fases del Plan de Trabajo o EDT.

4.2.1. Módulo 0

Este módulo se corresponde con la fase 0, que, al tratarse de la búsqueda de información, selección de *software* y prototipado, no se ha incluido en la arquitectura de la figura 4.2.

4.2.2. Módulo 1

Este módulo se corresponde con la primera fase y engloba tres apartados:

- La creación de la red en la *Blockchain* de Hyperledger Fabric (junto con las organizaciones, canales, etc.).
- La codificación y conexión con el *chaincode* o Contrato Inteligente (controlador y servicios).
- La conexión con la base de datos de MongoDB (controlador y servicios).

La tarea de este módulo es, por tanto, dar forma a una red híbrida basada en el *software* de Hyperledger y en donde se va a instaurar la cadena de bloques, codificar el *chaincode* para acceder a dicha cadena y realizar las acciones pertinentes (como añadir una

nueva donación, obtener las donaciones realizadas por un usuario, etc.) y conectar con MongoDB para almacenar los usuarios de la página web (usuarios donantes y administradores que pueden ser recepcionistas en centros de recogida o distribuidores en centros de distribución). Todo ello con el fin de ofrecer los servicios que se van a necesitar y/o mostrar en el *frontend* (seguimiento de las donaciones, trazabilidad de un lote, creación de lotes, registro de nuevos usuarios e ingreso en la página web).

Para ello se ha definido unos datos de entrada ficticios tanto en MongoDB como en el *chaincode*, para así poder inicializar la *Blockchain* y trabajar con un ejemplo.

4.2.3. Módulo 2

En este módulo, relacionado con la segunda fase, el objetivo es la creación de la página web a partir del prototipo de la fase 0, así como la conexión con los respectivos controladores del *backend* (MongoDB y *chaincode*), para finalmente acabar con la realización de pruebas funcionales.

4.2.4. Módulo 3

Este último módulo o fase 3 se encarga de realizar pruebas unitarias, estimar tiempos y analizar los resultados obtenidos de dichas pruebas, además de proponer posibles mejoras que se puedan desarrollar en un futuro.

4.2.5. Casos de uso

Con todo lo recogido hasta ahora vemos que el sistema dispone de dos usuarios: los usuarios donantes conocidos comúnmente como usuarios y los administradores (recepcionistas de los centros de recogida y distribuidores de los centros de distribución). Cada uno se relaciona con el sistema de la siguiente forma (figura 4.3):

- **Caso de uso 1: Registrar usuario.** Este caso de uso se encarga de registrar al usuario en MongoDB. Pertenece a los módulos 1 (al conectar con MongoDB) y 2 (al diseñar la página desde donde interactuará el usuario). Afecta a todos los usuarios: donantes y administradores.
- **Caso de uso 2: Iniciar sesión.** Este caso de uso se encarga de dar permiso para acceder al contenido de la web a un usuario ya registrado en la base de datos. Pertenece a los módulos 1 (al conectar con MongoDB) y 2 (al diseñar la página desde donde interactuará el usuario). Afecta a todos los usuarios: donantes y administradores.
- **Caso de uso 3: Consultar donaciones propias.** Este caso de uso se basa en la obtención de todas las donaciones realizadas por el usuario que ha iniciado sesión. Pertenece a los módulos 1 (al conectar con MongoDB) y 2 (al diseñar la página desde donde interactuará el usuario). Está disponible solo para los usuarios donantes.
- **Caso de uso 4: Obtener información de donde donar.** Este caso de uso muestra información de los lugares en donde se pueden realizar donaciones (centros de recogida de donaciones). Pertenece al módulo 2 (al diseñar la página desde donde interactuará el usuario). Está disponible solo para los usuarios donantes.
- **Caso de uso 5: Crear/juntar/mover lotes.** Este caso de uso permite crear los lotes con las donaciones, juntar varios lotes en uno nuevo y mover el contenido a otro, todo ello de forma total o parcial indicando el destino en donde va a permanecer

o ser enviado, así como el tipo de donaciones que incluye (juguetes, peluches y/o juegos). Pertenece a los módulos 1 (al conectar con el *chaincode*) y 2 (al diseñar la página desde donde interactuará el usuario). Está disponible solo para los administradores.

- **Caso de uso 6: Consultar los lotes activos o donaciones.** Este caso de uso permite consultar los lotes que no están vacíos en la *Blockchain*, así como las donaciones realizadas (mostrando información no sensible de cada uno). Pertenece a los módulos 1 (al conectar con el *chaincode*) y 2 (al diseñar la página desde donde interactuará el usuario). Está disponible solo para los administradores.
- **Caso de uso 7: Obtener la traza de un lote.** En este caso de uso se muestran los lotes anteriores y posteriores a uno dado, con el fin de poder observar la traza en caso de que algún lote o contenido de este se pierda (ya sea parcial o totalmente), para así evitar la apropiación indebida de los artículos donados. Pertenece a los módulos 1 (al conectar con el *chaincode*) y 2 (al diseñar la página desde donde interactuará el usuario). Está disponible solo para los administradores.

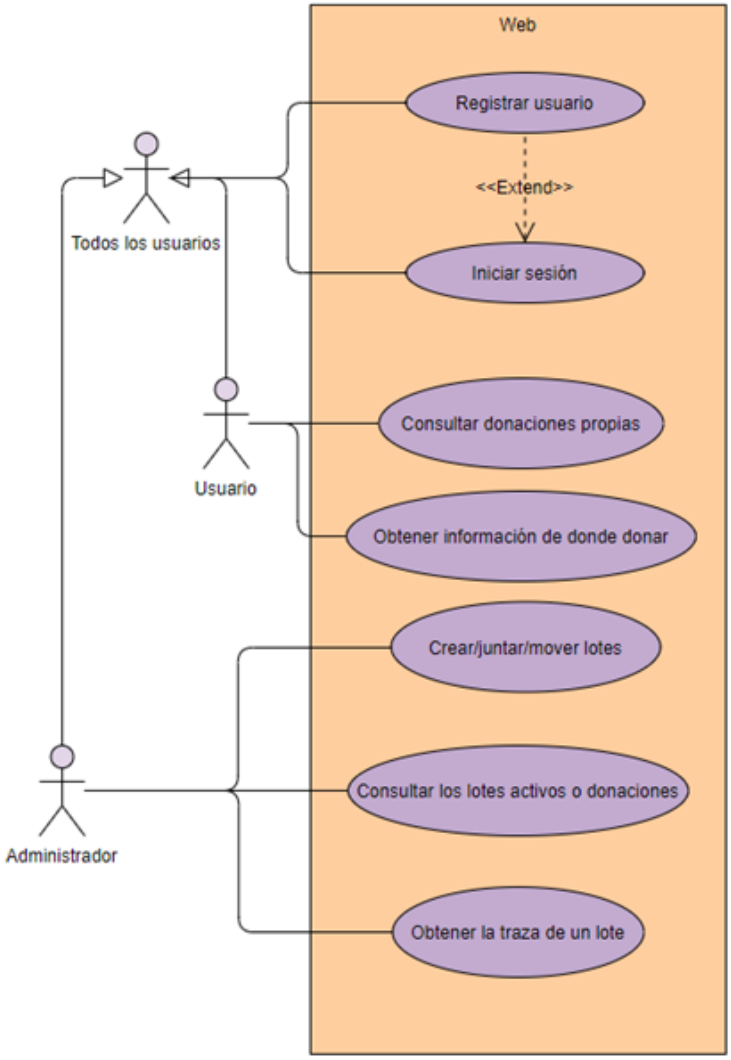

Figura 4.3: Diagrama de casos de uso del sistema

4.3 Tecnología Utilizada

En este apartado se describen las tecnologías que serán empleadas para desarrollar el sistema.

- **Hyperledger Fabric**

  Hyperledger Fabric es la más conocida, utilizada y extendida de las implementaciones disponibles de Hyperledger. Esto es así porque Fabric permite la creación de redes privadas e híbridas, permitiendo de esta manera dar permisos y privilegios a los distintos agentes de la red para que puedan o no acceder a cierta información. Su estructura es la siguiente: [23]

  - **Organizaciones:** representan agentes externos como empresas, asociaciones o instituciones.
  - **Nodos:** están relacionados con las organizaciones, y se encargan de firmar y transmitir las transacciones realizadas por los usuarios. Uno de estos nodos recibe el nombre de Orderer y tiene como función ordenar y transmitir las transacciones al resto de nodos.
  - **Clientes:** representan a los usuarios de cada organización al interactuar con la *Blockchain*, enviando transacciones a los nodos con los que dichas organizaciones están relacionadas.
  - **MSP:** Membership Service Provider define las reglas a través de las cuales se validan y autentican nuevos usuarios. Maneja distintas ids (MSPid) para comprobar si un cliente dispone de privilegios para realizar una acción.
  - **Canal:** conecta las distintas organizaciones a la red. Se pueden establecer canales privados entre organizaciones que forman parte de otro canal común para intercambiar o transmitir datos sensibles.
  - ***Chaincode*:** representa el código que interactuará con la *Blockchain* como si de un Contrato Inteligente se tratase. Agrupa, por ende, las acciones que se pueden llevar a cabo en la red.

  Por otro lado, en lo referente al empleo de esta tecnología cabe destacar que se utilizarán las librerías de Hyperledger junto con el código de ejemplo para la creación de la red de *Blockchain* híbrida y el *chaincode* para interactuar con ella.

- **Visual Studio Code**

  Visual Studio Code es un editor de código desarrollado por Microsoft para Windows, Linux y macOS. Destaca por ser muy versátil y reconocer gran cantidad de lenguajes de programación, además de permitir descargar complementos y librerías específicos para los mismos. También incluye soporte para la depuración, control integrado de Git, resaltado de sintaxis, finalización inteligente de código, fragmentos y refactorización de código.

  Este *software* se utilizará para la edición de código implementado en los módulos 2 y 3.

- **Lenguajes de programación**

  - **JavaScript:** se trata de un lenguaje de programación interpretado y orientado a objetos, el cual se utiliza principalmente para desarrollar el código propio de los sitios web, permitiendo la creación de páginas web dinámicas. Actualmente todos los navegadores modernos interpretan el código JavaScript integrado en las páginas web.

Este lenguaje de programación será utilizado para la realización del código del *chaincode*, así como de archivos varios de la red híbrida.

- **TypeScript:** en este caso es un lenguaje de programación libre y de código abierto desarrollado y mantenido por Microsoft. Se trata, concretamente de un superconjunto de JavaScript, que añade tipos estáticos y objetos basados en clases. Es utilizado para desarrollar aplicaciones JavaScript tanto para el lado del cliente (web) como para el servidor. Además, soporta ficheros de definición con información sobre los tipos de librerías JavaScript, permitiendo a otros programas usar los valores definidos en los ficheros como si fueran entidades TypeScript. Asimismo, existen cabeceras para librerías populares como jQuery, MongoDB y D3.js, y los módulos básicos de Node.js.

  Este lenguaje de programación será utilizado para la realización del código del *backend* (salvo el *chaincode* y la *Blockchain*) y *frontend* (salvo los archivos .html y .css).

- **HTML y CSS:** en cuanto al HTML, es el lenguaje de marcado que emplea para estructurar y dar significado a la página web. Por otro lado, CSS se trata de un lenguaje de reglas de estilo, utilizado para aplicar un estilo al contenido HTML.

  Ambos lenguajes de programación serán utilizados para la realización del código del *frontend* junto con el lenguaje comentando previamente.

- **Go o Golang:** el último lenguaje de programación se trata de un lenguaje concurrente y compilado inspirado en la sintaxis de C, orientado a objetos, dinámico como Python y con el rendimiento de C o C++. Ha sido desarrollado por Google.

  Este lenguaje de programación será necesario para el correcto funcionamiento de la red, ya que el *chaincode* tiene como lenguaje original Golang.

- **WSL de Ubuntu en Windows**

  Windows Subsystem for Linux (WSL) es una capa de compatibilidad desarrollada por Microsoft para ejecutar versiones de Ubuntu nativamente en Windows 10. Este subsistema proporciona una interfaz que simula un *kernel* de Linux, el cual puede ejecutar aplicaciones de espacio de usuario GNU. Dicho entorno puede contener una terminal, junto con ejecutables de línea de comandos GNU/Linux nativos y lenguajes de programación (Ruby, Python, etc).

  Esta herramienta se empleará para hacer uso de Ubuntu 20.04 LTS y así implementar y ejecutar todo el código en este sistema operativo, pues a la hora de realizar la red híbrida resulta más sencillo que emplear Windows.

- **MongoDB**

  MongoDB es un sistema de base de datos NoSQL, no relacional, orientado a documentos y de código abierto. Este sistema permite guardar los datos en tablas, tal y como se hace en las bases de datos relacionales, empleando estructuras de datos BSON (una especificación similar a JSON) con un esquema dinámico, facilitando y haciendo más rápida la integración de datos.

  Este sistema de base de datos se empleará para almacenar los usuarios de la página web y así como probar su autenticidad al iniciar sesión en la misma.

- **Docker**

  Docker es un proyecto de código abierto que automatiza el despliegue de aplicaciones dentro de contenedores de *software*. Se caracteriza por proporcionar una capa

adicional de abstracción y automatización de virtualización de aplicaciones en múltiples sistemas operativos. Además, emplea características de aislamiento de recursos del *kernel* Linux, para permitir que "contenedores" independientes se ejecuten dentro de una sola instancia, consiguiendo con esto evitar una sobrecarga al iniciar y mantener máquinas virtuales. Cabe destacar que se puede ejecutar en cualquier servidor Linux, dotando de flexibilidad y portabilidad a las aplicaciones.

Este *software* será utilizado para cargar los certificados e imágenes de Hyperledger para hacer funcionar la red de Fabric y el *chaincode*.

- **NPM**

  Node Package Manager ayuda a gestionar los paquetes de Node, un entorno de ejecución de JavaScript para compartir herramientas, instalar varios módulos y administrar dependencias.

  Esta herramienta será utilizada para cargar librerías varias que hagan funcionar el código del *backend* y *frontend*.

- **Angular**

  Angular es un marco de trabajo para aplicaciones web desarrollado en TypeScript, de código abierto, mantenido por Google y que se emplea para crear y mantener aplicaciones web de una sola página.

  Este marco de trabajo será utilizado para realizar el *frontend* de la aplicación.

- **NestJS**

  NestJS es un marco de trabajo de Node.js para la creación de aplicaciones confiables, eficientes y escalables del lado del *backend* o servidor.

  Este marco de trabajo será empleado para realizar el *backend* de la aplicación.

CAPÍTULO 5
Desarrollo de la solución propuesta

En este capítulo se describirán los detalles de implementación del sistema. Para realizar una correcta estructuración del capítulo, este se dividirá en cuatro secciones, una por cada fase (o módulo) del proyecto.

5.1 Módulo 0

En esta fase se ha realizado una búsqueda y recopilación de información previa a la elección del tipo de *Blockchain* (capítulo 2 y apartados 3.3 y 3.4 del capítulo 3), un análisis de la situación y posible solución (capítulo 3), un diseño de la arquitectura para el sistema a desarrollar (capítulo 4) y, finalmente, el prototipado del *frontend*, dichos prototipos utilizados en la fase inicial se corresponden con las figuras que se van a mostrar a continuación.

Figura 5.1: Pantalla de inicio (prototipo)

Figura 5.2: Inicio de sesión (prototipo)

Figura 5.3: Registro de usuario (prototipo)

Figura 5.4: Registro de administrador (prototipo)

Figura 5.5: Vista de usuario (prototipo)

Figura 5.6: Vista de usuario: donar (prototipo)

Figura 5.7: Vista de usuario: seguimiento (prototipo)

Figura 5.8: Vista de administrador (prototipo)

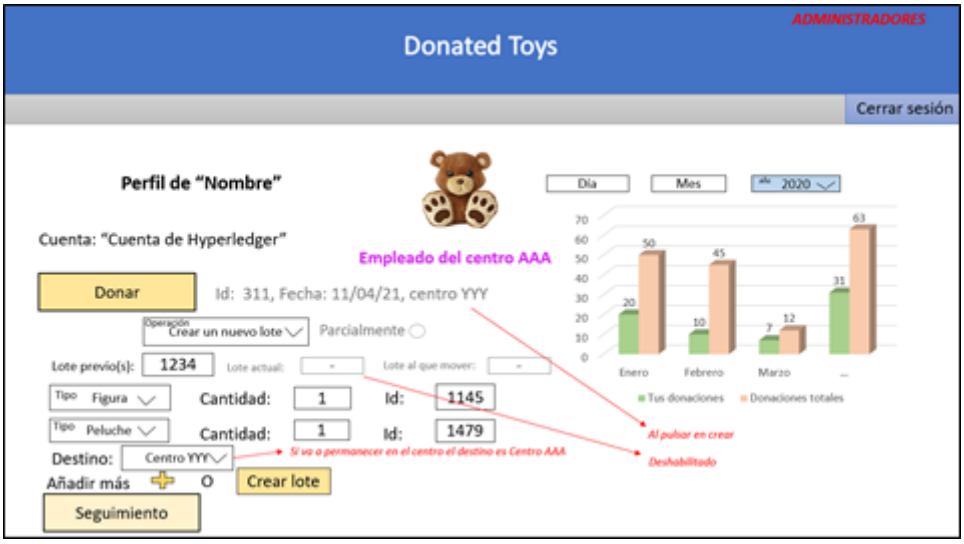

Figura 5.9: Vista de administrador: donar – Crear (prototipo)

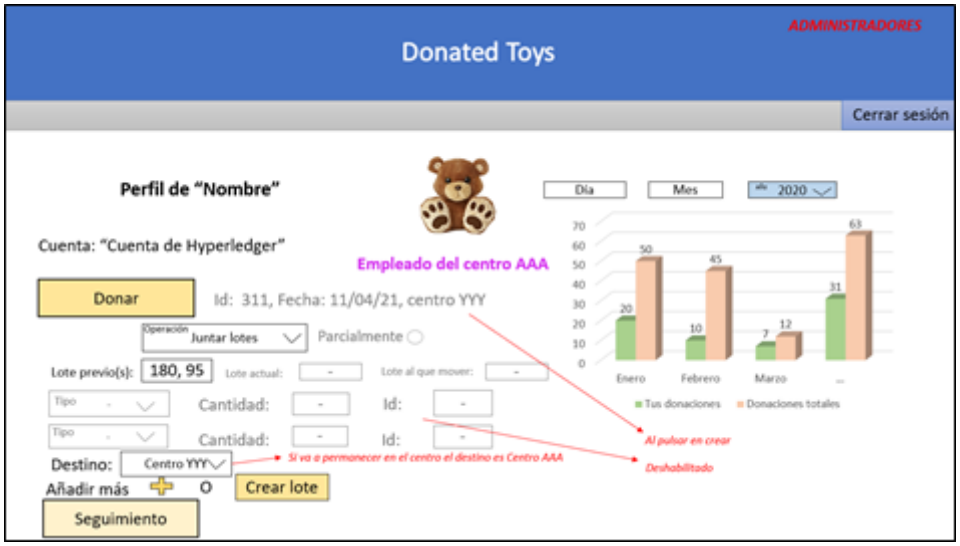

Figura 5.10: Vista de administrador: donar – Juntar totalmente (prototipo)

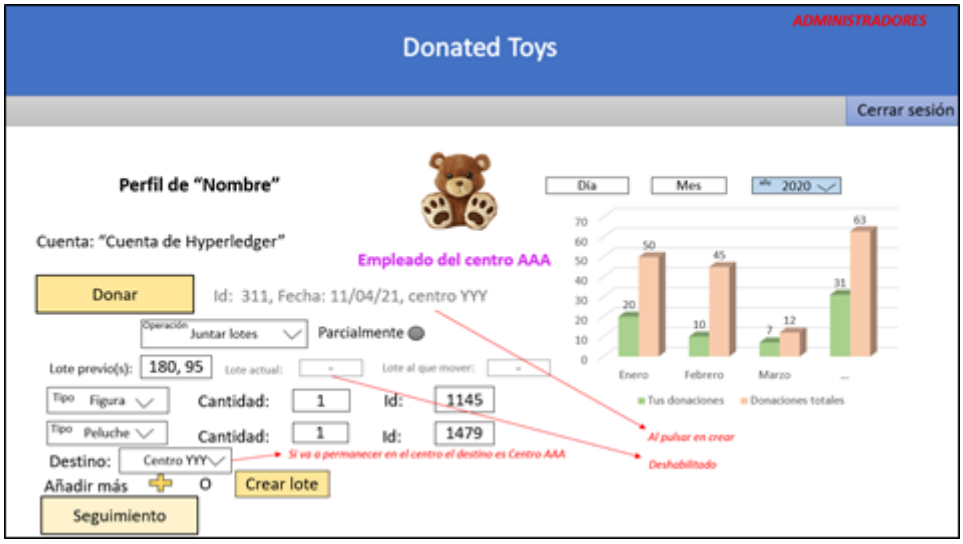

Figura 5.11: Vista de administrador: donar – Juntar parcialmente (prototipo)

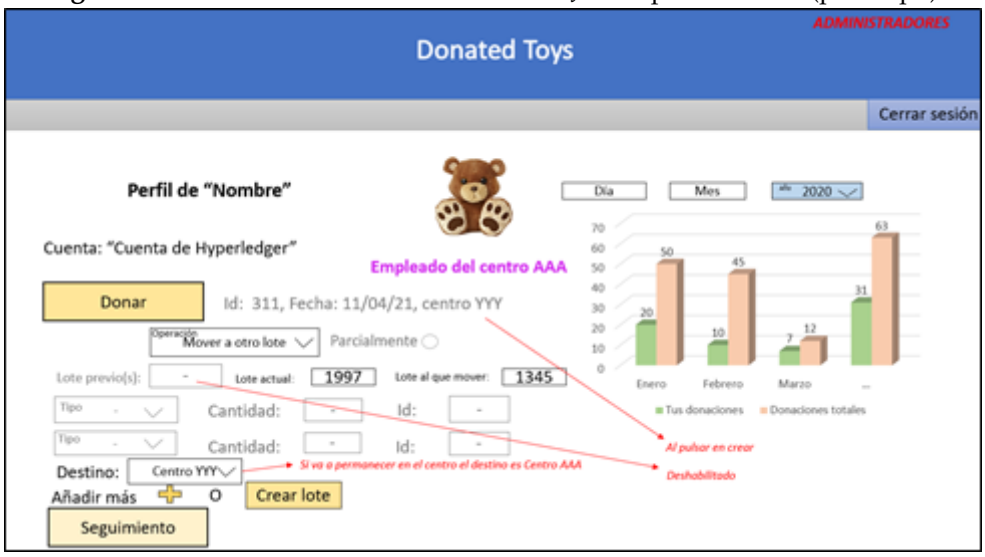

Figura 5.12: Vista de administrador: donar – Mover totalmente (prototipo)

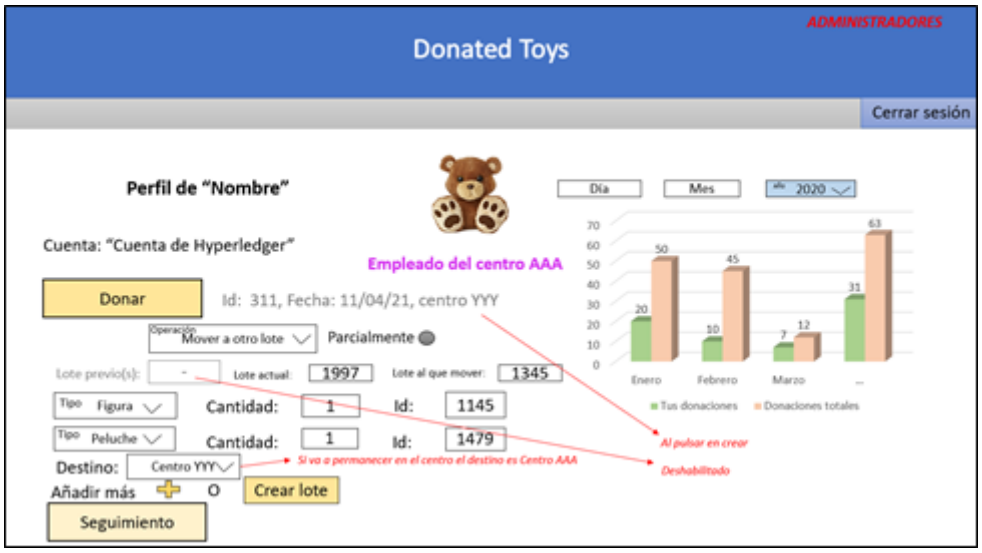

Figura 5.13: Vista de administrador: donar – Mover parcialmente (prototipo)

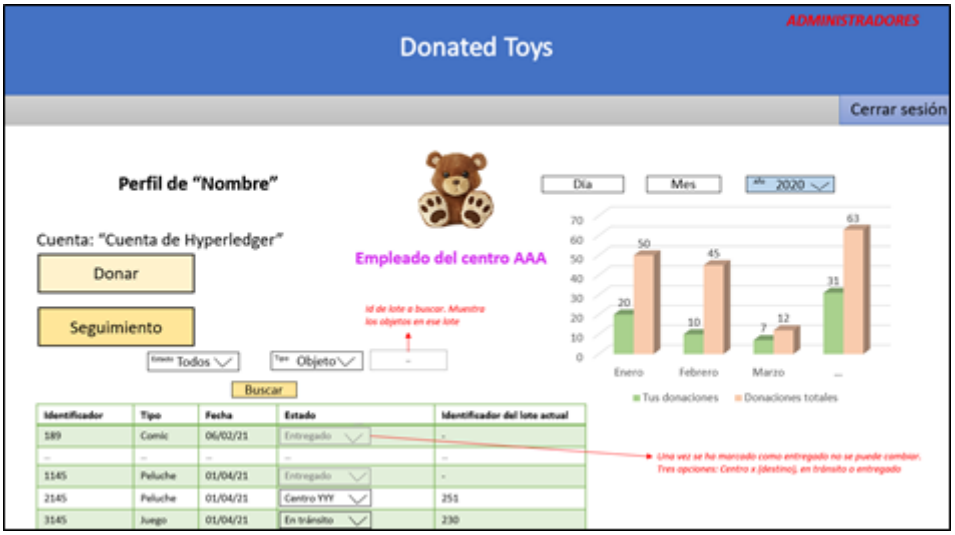

Figura 5.14: Vista de administrador: seguimiento (prototipo)

5.2 Módulo 1

Este módulo o fase 1 se corresponde con la implementación del *backend*, de la red híbrida, del *chaincode* y de la conexión con MongoDB. Por tanto, vamos a dividir este subapartado en 5 apartados.

5.2.1. Pre-requisitos y configuración del entorno

En este apartado se comentarán los requisitos necesarios para hacer funcionar la red de Hyperledger Fabric.

En primer lugar, se debe de instalar Golang, el lenguaje de programación que emplea Hyperledger para su correcto funcionamiento. Para ello se debe ejecutar el siguiente comando en una terminal de la WSL (Ubuntu): "sudo apt-get install golang-go".

A continuación, se instala Docker Desktop para Windows y se ejecutan los siguientes comandos en Ubuntu para poder utilizar Docker desde la WSL:

- sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable"
- sudo apt-get install -y docker-ce
- sudo apt-get install docker-compose

Posteriormente, se deben de realizar las siguientes instrucciones en caso de que se encuentren problemas relacionados con la versión del *kernel* al utilizar la WSL:

- En una PowerShell escribir los comandos:
  - wsl.exe -l -v
  - wsl –set-version Ubuntu 2 (Ubuntu o el nombre completo que tenga el sistema operativo de la WSL que se seleccione)

- En una terminal de Ubuntu escribir los comandos:
  - sudo usermod -a -G docker username
  - newgrp docker
- Reiniciar Ubuntu

En el siguiente paso se instala la última versión de Hyperledger Fabric junto con sus dependencias usando el comando: “curl -sSL https://bit.ly/2ysbOFE | bash -s”.

Por último, se instala MongoDB para Ubunbtu mediante el comando: "sudo apt-get install -y mongodb". Y en el directorio previo a home se deben ejecutar los comandos:

- sudo mkdir -p data/db
- sudo chown -R ` id -un` data/db

Cabe decir que para que el código del *frontend* y *backend* funcione correctamente hace falta tener instalados NPM, NestJS y Angular:

- Escribir en una terminal el comando:
  - curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.35.3/install.sh | bash
- Cerrar y reabrir la terminal. Posteriormente ejecutar los comandos:
  - nvm install node
  - nvm install v12.21.0
  - nvm use v12.21.0
- En una terminal situada en el directorio del *frontend* ejecutar los comandos:
  - npm install -g @angular/cli@latest
  - npm install
- En una terminal situada en el directorio del *backend* ejecutar el comando:
  - npm install

5.2.2. Red híbrida (Hyperledger Fabric)

Para representar los elementos de la red de Hyperledger Fabric (vistos en el apartado 4.3), se han modificado los ficheros de la red de ejemplo test-network y cuyo esquema se muestran en la figura 5.15.

El fichero configtx.yaml es el encargado de definir que organizaciones conforman la red (recepcionista y distribuidor), además del nodo Orderer que se encarga de ordenar las transacciones realizadas. Para lo cual se define cada nodo, se especifica su id, el conjunto de políticas relacionadas (si puede leer, escribir y si es administrador), el *endpoint* y el directorio MSP (para que puedan identificarse como parte de la red). Además, consta de un apartado que configura algunos parámetros extra y permite ponerle un nombre al bloque génesis (bloque inicial de la cadena), en este caso ha sido llamado DonacionesGenesis.

Seguidamente, el fichero docker-compose-ca.yaml se encarga de desplegar los contenedores Docker de certificados en la red, docker-compose-couch.yaml despliega los

contenedores relacionados con la base de datos de Fabric (CouchDB) y docker-compose-test-net.yaml despliega los contenedores que hacen referencia a las organizaciones especificadas en configtx.yaml.

Posteriormente se crean los certificados para las organizaciones de la red mediante el archivo cryptogen que el propio Hyperledger provee, pasándole como argumentos los archivos crypto-config-recepcionista.yaml, crypto-config-distribuidor.yaml y crypto-config-orderer.yaml. En estos archivos se especifica el nombre de la organización, el dominio y la cantidad de nodos que se le asignan (en este caso uno para simplificar el sistema).

Por otro lado, al tratarse de una cadena de bloques, es esencial la creación del bloque génesis (el primer bloque de la cadena), además de un canal para que las distintas organizaciones puedan comunicarse entre sí. Para esto último el script createChannel.sh de la figura 5.16 junto con configUpdate.sh y setAnchorPeer.sh se encargan de crear dicho canal y de unir a él a los participantes necesarios.

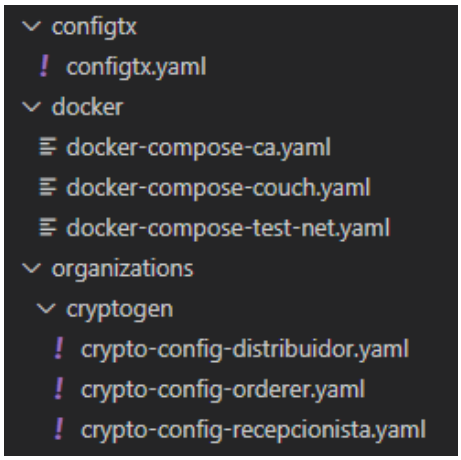

Figura 5.15: Parte del esquema del directorio de la red de Fabric (test-network)

Cabe decir que para poder poner en marcha todo lo que se ha comentado hasta el momento se emplea en el directorio test-network el comando: "source ./network.sh up createChannel -ca -c mychannel" (en donde "-ca" permite crear los certificdos para las organizaciones, "createChannel -c mychannel" el canal y el resto del comando inicia la red), [24]. El cual utiliza el script deployCC.sh de la figura 5.16 para instalar y deplegar la red en los nodos. El resultado de su ejecución se puede ver en las figuras: 5.17, 5.18, 5.19 y 5.20 .

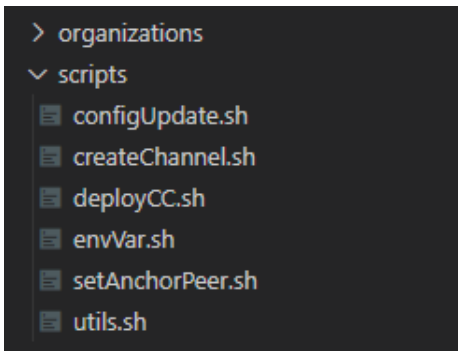

Figura 5.16: Parte del esquema del directorio de la red de Fabric (test-network) (2)

```
Creating channel 'mychannel'.
If network is not up, starting nodes with CLI timeout of '5' tries and CLI delay of '3' seconds and using database 'le
Bringing up network
LOCAL_VERSION=2.3.1
DOCKER_IMAGE_VERSION=2.3.1
CA_LOCAL_VERSION=1.4.9
CA_DOCKER_IMAGE_VERSION=1.4.9
Generating certificates using Fabric CA
Creating network "fabric_test" with the default driver
Creating ca_distribuidor  ... done
Creating ca_recepcionista ... done
Creating ca_orderer       ... done
Creating recepcionista Identities
Enrolling the CA admin
++ fabric-ca-client enroll -u https://admin:adminpw@localhost:7054 --caname ca-recepcionista --tls.certfiles /home/lev
cepcionista/tls-cert.pem
2021/04/19 17:51:57 [INFO] Created a default configuration file at /home/levan/tempblock/fabric-samples/test-network/c
```

Figura 5.17: Creación de certificados para los nodos de la red

```
Creating orderer.example.com            ... done
Creating peer0.distribuidor.example.com ... done
Creating peer0.recepcionista.example.com ... done
Creating cli                            ... done
CONTAINER ID   IMAGE                               COMMAND                  CREATED              STATUS                  PORT
318c595afc6c   hyperledger/fabric-tools:latest     "/bin/bash"              Less than a second ago   Up Less than a second
9f734e538ae7   hyperledger/fabric-peer:latest      "peer node start"        1 second ago         Up Less than a second   0.0.
91d323073986   hyperledger/fabric-peer:latest      "peer node start"        1 second ago         Up Less than a second   705
4fbe7d53c8ad   hyperledger/fabric-orderer:latest   "orderer"                1 second ago         Up Less than a second   0.0.
5cfea32cce9d   hyperledger/fabric-ca:latest        "sh -c 'fabric-ca-se…"   7 seconds ago        Up 6 seconds            705
0d784036a9a8   hyperledger/fabric-ca:latest        "sh -c 'fabric-ca-se…"   7 seconds ago        Up 6 seconds            0.0.
d14780d034ca   hyperledger/fabric-ca:latest        "sh -c 'fabric-ca-se…"   7 seconds ago        Up 6 seconds            705
```

Figura 5.18: Despliegue de los nodos y certificados en Docker

```
Generating channel genesis block 'mychannel.block'
/home/levan/tempblock/fabric-samples/test-network/../bin/configtxgen
+++ configtxgen -profile DonacionesGenesis -outputBlock ./channel-artifacts/mychannel.block -channelID mychannel
2021-04-19 17:52:02.985 CEST [common.tools.configtxgen] main -> INFO 001 Loading configuration
2021-04-19 17:52:02.995 CEST [common.tools.configtxgen.localconfig] completeInitialization -> INFO 002 orderer type:
2021-04-19 17:52:02.996 CEST [common.tools.configtxgen.localconfig] completeInitialization -> INFO 003 Orderer.EtcdRa
nflight_blocks:5 snapshot_interval_size:16777216
2021-04-19 17:52:02.996 CEST [common.tools.configtxgen.localconfig] Load -> INFO 004 Loaded configuration: /home/leva
2021-04-19 17:52:02.997 CEST [common.tools.configtxgen] doOutputBlock -> INFO 005 Generating genesis block
2021-04-19 17:52:02.997 CEST [common.tools.configtxgen] doOutputBlock -> INFO 006 Creating application channel genesi
2021-04-19 17:52:02.998 CEST [common.tools.configtxgen] doOutputBlock -> INFO 007 Writing genesis block
```

Figura 5.19: Creación del bloque génesis

```
Creating channel mychannel
Using organization recepcionista
+++ osnadmin channel join --channelID mychannel --config-block ./channel-artifacts/mychannel.block -o localhost:7053 --ca-file
ions/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem --client-cert /home/levan/tempblock/fa
/orderer.example.com/tls/server.crt --client-key /home/levan/tempblock/fabric-samples/test-network/organizations/ordererOrgani
+++ res=0
Status: 201
{
        "name": "mychannel",
        "url": "/participation/v1/channels/mychannel",
        "consensusRelation": "consenter",
        "status": "active",
        "height": 1
}

Channel 'mychannel' created
Joining recepcionista peer to the channel...
Using organization recepcionista
+++ peer channel join -b ./channel-artifacts/mychannel.block
+++ res=0
2021-04-19 17:52:09.374 CEST [channelCmd] InitCmdFactory -> INFO 001 Endorser and orderer connections initialized
2021-04-19 17:52:09.412 CEST [channelCmd] executeJoin -> INFO 002 Successfully submitted proposal to join channel
Joining distribuidor peer to the channel...
Using organization distribuidor
```

Figura 5.20: Creación y registro de las organizaciones en el canal mychannel

Finalmente, se procede a instalar el *chaincode* en los nodos que representan las organizaciones, para así interactuar con la red a través de él (figuras 5.21 y 5.22). Esto se realiza en el directorio test-network mediante el comando: "source ./network.sh deployCC -ccn basic -ccp ../chaincodeTest -ccl javascript", [25]. En donde, "-ccn basic" indica que el contrato se llamará basic, "-ccp ../chaincodeTest" es la ruta donde se encuentra el *chaincode*,

"-ccl javascript" indica el lenguaje en el que está implementado el contrato y el resto del comando inicia su despliegue en los nodos de las organizaciones.

```
deploying chaincode on channel 'mychannel'
executing with the following
- CHANNEL_NAME: mychannel
- CC_NAME: basic
- CC_SRC_PATH: ../chaincodeTest
- CC_SRC_LANGUAGE: javascript
- CC_VERSION: 1.0
- CC_SEQUENCE: 1
- CC_END_POLICY: NA
- CC_COLL_CONFIG: NA
- CC_INIT_FCN: NA
- DELAY: 3
- MAX_RETRY: 5
- VERBOSE: false
+++ peer lifecycle chaincode package basic.tar.gz --path ../chaincodeTest --lang node --label basic_1.0
+++ res=0
Chaincode is packaged
Installing chaincode on peer0.recepcionista...
Using organization recepcionista
+++ peer lifecycle chaincode install basic.tar.gz
+++ res=0
2021-04-19 17:55:52.024 CEST [cli.lifecycle.chaincode] submitInstallProposal -> INFO 001 Installed remote
d12cee1e6\022\tbasic_1.0" >
2021-04-19 17:55:52.024 CEST [cli.lifecycle.chaincode] submitInstallProposal -> INFO 002 Chaincode code pa
Chaincode is installed on peer0.recepcionista
Install chaincode on peer0.distribuidor...
Using organization distribuidor
```

Figura 5.21: Despliegue del *chaincode* (1)

```
Committed chaincode definition for chaincode 'basic' on channel 'mychannel':
Version: 1.0, Sequence: 1, Endorsement Plugin: escc, Validation Plugin: vscc, Appr
```

Figura 5.22: Despliegue del *chaincode* (2)

Si en algún momento se quiere parar la red o apagarla, y borrar todos los contenedores desplegados, monedero y archivos generados, se debe de borrar manualmente la carpeta del monedero y de ejecutar el siguiente comando: "source ./network.sh down" (figura 5.23).

```
Stopping network
Stopping cli                              ... done
Stopping peer0.recepcionista.example.com  ... done
Stopping peer0.distribuidor.example.com   ... done
Stopping orderer.example.com              ... done
Stopping ca_orderer                       ... done
Stopping ca_recepcionista                 ... done
Stopping ca_distribuidor                  ... done
Removing cli                              ... done
Removing peer0.recepcionista.example.com  ... done
Removing peer0.distribuidor.example.com   ... done
Removing orderer.example.com              ... done
Removing ca_orderer                       ... done
Removing ca_recepcionista                 ... done
Removing ca_distribuidor                  ... done
Removing network fabric_test
Removing volume docker_orderer.example.com
Removing volume docker_peer0.recepcionista.example.com
Removing volume docker_peer0.distribuidor.example.com
Removing remaining containers
Removing generated chaincode docker images
Untagged: dev-peer0.distribuidor.example.com-basic_1.0-3d3deb056b52243d0d7c5042b2ce
Deleted: sha256:03597a657d40e2f62d1f48f6947cda105920014f674a93535ecabf21e3873b8b
Deleted: sha256:019052eecf1fad60669ffefbd94f5903c3cd3852b673d76f0b21d418ef2b16f4
Deleted: sha256:5eca848760ca69bd82ac063f08556261e9895b0583aefb2e1e24d2dc44b18da1
Deleted: sha256:3ee0ebc20fbc052ce42e6907f708d16d2804784bd261d9d3b850dbc98752dd49
Untagged: dev-peer0.recepcionista.example.com-basic_1.0-3d3deb056b52243d0d7c5042b2c
Deleted: sha256:01a422224a55f527d5261243c7946435bea1d1079ad43cf91fa761b98e7670be
Deleted: sha256:e76e8067cc72d5fe4706f7c681302029b52e54f565bd59775eb1245bac1d815e
Deleted: sha256:5dcd4159fa5df48742b8ad707c16cf16d766571211303392020ddca2b19b3a19
Deleted: sha256:444d386af9bc529a56f8ccfa9049b0506053f40e9b83158c10a2672ca92c2abd
```

Figura 5.23: Parar la red y borrar los archivos creados durante su ejecución

5.2.3. MongoDB

En cuanto a MongoDB, se ha creado la base de datos donaciones con la colección *users* para guardar los usuarios de la página web.

Dichos usuarios poseen el siguiente esquema definido en el *backend* (figura 5.24):

```
// Interfaz del usuario
export interface IUser extends Document {
    readonly Nombre: string;
    readonly Contraseña: string;
    readonly Centro: string | undefined;
```

Figura 5.24: Esquema de usuario para MongoDB

Como se puede ver la variable *Centro* puede ser undefined, esto es así para el caso de los donantes que solo necesitan especificar un nombre de usuario y una contraseña.

5.2.4. *Chaincode*

La estructura de los lotes que se guardan en la *Blockchain* es la siguiente (figura 5.25):

```
export interface Obj {
    ID: string; // Identificación del objeto
    Estado: string; // Puede ser entregado o el nombre del centro donde está
    Lote: string; // Identificación del lote donde se encuentra
    Camino: string; // Ruta seguida por el momento
    Tipo: string; // Clasificación: peluche, juego o juguete
    Particular: string; // Persona que lo ha donado
    Fecha: string; // Fecha en la que se donó
}

export class createLoteDto {
    ID: string; // Identificación del lote en la BLOCKCHAIN
    Estado: string; // Centro de destino o donde se guardará
    Lotes_previos: [string]; // Identificaciones de los lotes de los que procede
    Lote_actual: string; // Identificación real del LOTE
    Lote_mover: string; // Identificación del lote al que se va a mover el contenido
    ACargo: string; // Administrador que crea o realiza una operación con el contenido del lote
    Objetos: [Obj] // Lista de objetos donados que tiene el lote
}
```

Figura 5.25: Esquema de lote para el *chaincode*

El esquema del lote contiene a su vez una lista de objetos que puede estar vacía o no, en el caso de no estarla la interfaz *Obj* define todos los atributos que tendría cada uno de dichos objetos (artículos donados).

Por otro lado, los métodos que realiza el *chaincode* sobre la red son los que se detallan a continuación:

- **InitLedger:** inicializa la cadena de bloques, en este caso dispone de un ejemplo básico (figuras 5.26 y 5.27) con datos falsos (contiene una variable para los lotes y otra para los usuarios) a partir del cual proceder.

```
async initLedger(ctx) {
    const assets = [
        {
            ID: '00000000',
            Estado: 'UPV',
            Lotes_previos: "-1",
            Lote_actual: "00000000",
            Lote_mover: "-1",
            ACargo: "admin",
            Objetos: [{
                ID: '10',
                Estado: 'UPV',
                Lote: '00000000',
                Camino: 'UPV',
                Tipo: 'peluche',
                Particular: 'user',
                Fecha: '08-03-2021',
            },
            {
                ID: '20',
                Estado: 'UPV',
                Lote: '00000000',
                Camino: 'UPV',
                Tipo: 'peluche',
                Particular: 'user',
                Fecha: '08-03-2021',
            },
            {
                ID: '30',
                Estado: 'UPV',
                Lote: '00000000',
                Camino: 'UPV',
                Tipo: 'juego',
                Particular: 'user',
                Fecha: '08-03-2021',
            }],
        },
```

Figura 5.26: Esquema inicial de la cadena de bloques (lotes)

```
];

const users = [
    {
        Particular: 'user',
        Donaciones: '3',
    },
    {
        Particular: 'user2',
        Donaciones: '1',
    },
];
```

Figura 5.27: Esquema inicial de la cadena de bloques (usuarios)

- **CreateAsset:** a partir de un tipo de operación los administradores pueden crear, juntar o mover lotes. Su funcionamiento es el siguiente:
    - **Crear:** crea un nuevo lote con los contenidos especificados en *Objetos*, dentro del JSON pasado al método. Además, si el usuario donante ya está guardado en la *Blockchain*, su número de donaciones se incrementará en el número de objetos donados por él que lleve el lote. Si no está guardado en la variable *users*, entonces se almacenará junto con el número de donaciones a su nombre que hay en el lote. Finalmente devuelve un mensaje que contiene la variable *numberAssets* (devuelve la id del nuevo lote que se ha creado).

- **Mover:** mueve los contenidos especificados en *Objetos*, dentro del JSON pasado al método, a otro lote especificado, para lo cual comprueba que el lote que se va a mover contenga los objetos (ids) especificados, posteriormente mueve el contenido que coincide y crea un nuevo lote como "copia" del lote original, pero conteniendo solo los objetos que no se han movido. Sin embargo, si *Objetos* estuviese vacío significaría que se ha escogido la opción de realizar la operación totalmente, por lo que se movería el contenido completo comprobando únicamente que el lote fuese un lote activo, pero sin comprobar las ids de los objetos pues no se especificarían. Finalmente, devuelve un mensaje que puede ser:
  - "Alguna o algunas de las ids especificadas para los objetos son incorrectas.": si alguna id a mover no está en el lote indicado.
  - "Todas las ids de los objetos son incorrectas, no se ha realizado la acción.": en el caso de que ninguna id esté en el lote.
  - "Una o varias de las ids especificadas para los lotes son incorrectas. Puede que se haya movido el contenido del lote o que haya sido entregado.": si las ids de los lotes no existen o se trata de lotes cuyos contenidos han sido movidos a otro.
- **Juntar:** junta los contenidos especificados en *Objetos*, dentro del JSON pasado al método, de dos lotes. Para ello comprueba que las ids pertenezcan a uno o a otro lote, de ser así mueve los objetos con esas ids a un lote nuevo y crea dos lotes como "copia" de los lotes previos que contendrían los objetos cuyas ids no se han indicado, es decir, que no se ha realizado ninguna acción con estos artículos. Sin embargo, si *Objetos* estuviese vacío significaría que se ha escogido la opción de realizar la operación totalmente, por lo que se juntaría el contenido completo comprobando únicamente que los lotes fuesen lotes activos, pero sin comprobar las ids de los objetos pues no se especificarían. Finalmente, devuelve un mensaje que puede ser:
  - "Alguna o algunas de las ids especificadas para los objetos son incorrectas.": si alguna id a mover no está en el lote indicado.
  - "Todas las ids de los objetos son incorrectas, no se ha realizado la acción.": en el caso de que ninguna id esté en el lote.
  - "Una o varias de las ids especificadas para los lotes son incorrectas. Puede que se haya movido el contenido del lote o que haya sido entregado.": si las ids de los lotes no existen o se trata de lotes cuyos contenidos han sido movidos a otro (son por tanto lotes inactivos).
  - La variable numberAssets: devuelve la id del lote que se ha creado al juntar los dos previos en uno nuevo.

- **ReadUser:** devuelve un JSON con la información de un usuario de la *Blockchain* (particular (nombre) y donaciones), se utiliza como método interno al crear un lote (para aumentar el número de donaciones que ha realizado un usuario).

- **UpdateAssetEntrega:** actualiza el estado de un lote u objeto a "Entregado" y el camino o ruta a "...-Entregado". Esto solo puede ser realizado por los administradores del sistema.

- **GetAllAssets:** devuelve un JSON con todos los lotes de la *Blockchain*, se utiliza como método interno para los posteriores.

- **GetAllCurrentAssets:** devuelve un JSON con todos los lotes actuales que siguen activos, es decir, cuyos contenidos no han sido movidos a otro lote.

- **GetPath:** devuelve la traza de un lote concreto proporcionado a través de su id (figura 5.28) y emplea, para ello, los dos métodos posteriores que se irán llamando recursivamente hasta obtener un resultado que el método *getPath* juntará y devolverá como solución.

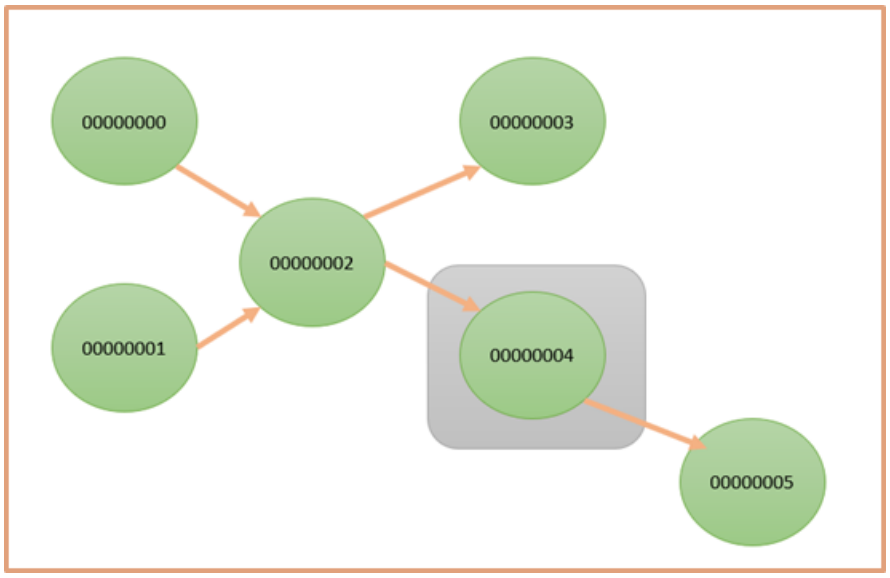

Figura 5.28: Ejemplo de diagrama en donde se busca el lote 00000004

- **GetPathPrev:** obtiene todos los lotes que coinciden con las ids contenidas en la variable *Lotes_previos* del lote actual. Es decir, los lotes anteriores a uno dado (figura 5.29). Posteriormente emplea las ids de estos lotes encontrados para seguir buscando a través de las ids de sus lotes anteriores, actuando de forma recursiva.

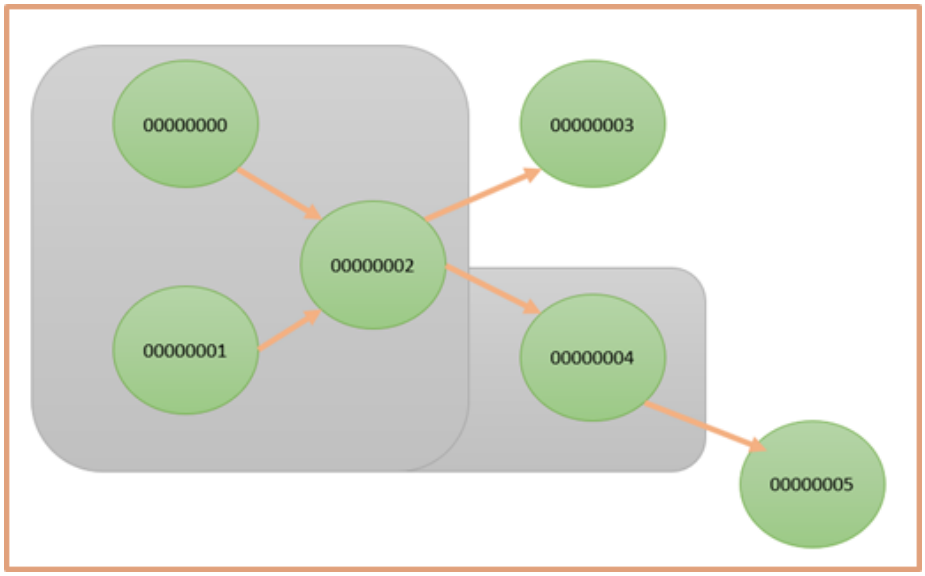

Figura 5.29: Resultado de aplicar getPathPrev al ejemplo de diagrama donde se busca el lote 00000004

- **GetPathPost:** obtiene todos los lotes cuya variable *Lotes_anteriores* contiene la id buscada actualmente. Es decir, los lotes posteriores a uno dado (figura 5.30). Una

vez haga esto, buscará lotes que tengan como lote anterior la id obtenida de la anterior búsqueda, actuando de forma recursiva.

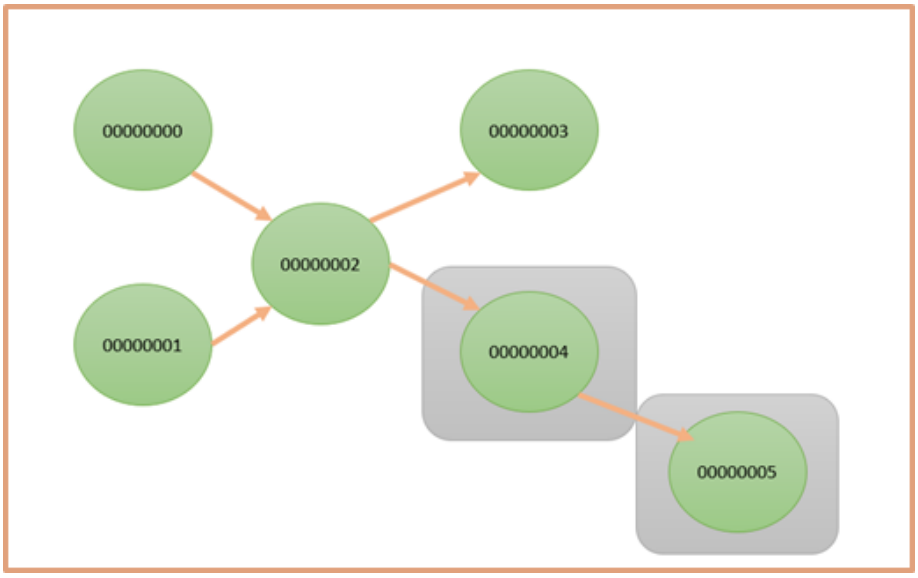

Figura 5.30: Resultado de aplicar getPathPost al ejemplo de diagrama donde se busca el lote 00000004

- **LengthChainAssets:** devuelve el número total de lotes almacenados en la *Blockchain*, con la función de asignar una nueva id a los lotes creados, pues éstas son secuenciales.
- **GetUserObj:** devuelve una lista con las donaciones realizadas por un usuario.

5.2.5. *Backend*

El *backend* se encarga de relacionar el *chaincode* y MongoDB con el *frontend* y aporta los servicios necesarios para que el sistema funcione como debe. Para ejecutarlo tan solo son necesarios el comando "npm run start:debug" (figura 5.31) justo dentro de la carpeta del *backend* y "mongod" (figura 6.5) en cualquier directorio (para iniciar MongoDB en Ubuntu).

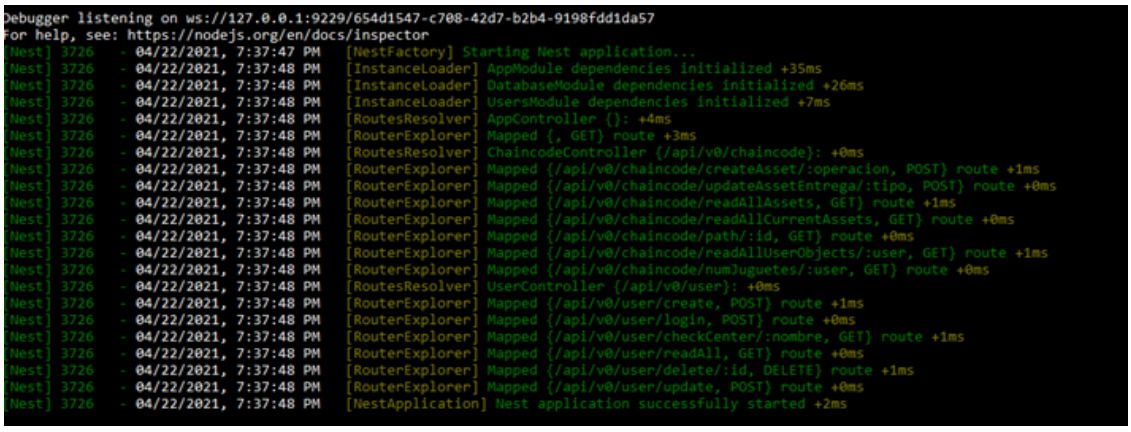

Figura 5.31: *Backend* en ejecución

Figura 5.32: Servicio de MongoDB en ejecución

5.2.5.1. Conexión con MongoDB

El controlador (API[1]) que hace de puente entre el *frontend* y la clase de servicios que conecta con MongoDB tienen los siguientes métodos:

URL	Métodos	Parámetros	Definición
api/v0/user/create	POST	JSON con la información del usuario. Ejemplo: { "Nombre": "Juan", "Contraseña": "abc", "Centro": "undefined" } - En este caso se ha creado un usuario de tipo normal por lo que el campo del centro permanece como undefined	Este *endpoint* crea un nuevo usuario en la base de datos usando para ello un JSON con la información a almacenar (siguiendo la estructura vista en la figura 5.24). Si el usuario no existe devolverá el mensaje "Usuario creado", de otra forma mostrará "El nombre de usuario ya existe"
api/v0/user/login	POST	JSON con la información del usuario. Ejemplo: { "Nombre": "Juan", "Contraseña": "abc", "Centro": "undefined" } - En este caso se comprueba la información del usuario Juan y cuya contraseña encriptada es abc	Este *endpoint* comprueba si las credenciales de inicio de sesión del usuario son correctas (nombre y contraseña encriptada). Si la información no coincide con la de la base de datos devolverá el mensaje "La información es incorrecta", en caso contrario mostrará "El usuario (o administrador) ha sido logeado"
api/v0/user/checkCenter/:nombre	GET	El nombre del usuario se especifica en la url. Ejemplo: api/v0/user/checkCenter/Juan - Este ejemplo devolvería el centro (de tenerlo) del usuario Juan	Este *endpoint* devuelve el centro al que pertenece un usuario dado

[1]Interfaz de Programación de Aplicaciones

api/v0/user/readAll	GET	No tiene	Este *endpoint* devuelve la lista completa de todos los usuarios. No se utiliza en la página web, pero se ha añadido por si hiciese falta en una versión futura
api/v0/user/delete/:id	DELETE	La id del usuario se especifica en la url. Ejemplo: api/v0/user/delete/0 - Este ejemplo borraría al usuario Juan	Este *endpoint* borra un usuario concreto de la base de datos. No se utiliza por el momento, pero se ha añadido para una versión futura en la que un usuario quisiera borrar su cuenta
api/v0/user/update	POST	JSON con la información del usuario modificada. Ejemplo: { "_id": "0", "Nombre": "Juan", "Contraseña": "abc", "Centro": "Caritas" } - En este caso se ha actualizado el usuario de nombre Juan, modificando el campo del centro	Este *endpoint* actualiza un usuario concreto de la base de datos. No se utiliza por el momento, pero se ha añadido para una versión futura en la que un usuario quisiera actualizar su cuenta

Tabla 5.1: *Endpoints* MongoDB

5.2.5.2. Conexión con *Blockchain* (*chaincode*)

El controlador (API) que hace de puente entre el *frontend* y la clase de servicios que conecta con el *chaincode* tienen los siguientes métodos:

URL	**Métodos**	**Parámetros**	**Definición**
api/v0/chaincode/updateAssetEntrega/:tipo	POST	JSON con una lista lista que contiene las ids de los objetos a actualizar (lote1, objeto1, lote2, objeto2, ...) o (lote, 0, ...) si es todo el lote lo que se actualiza. Ejemplo: { "lista": "2,11"} - En este caso se actualiza el objeto 11 del lote 2	Este *endpoint* actualiza el estado y camino de un objeto o lote a "Entregado" gracias a la información obtenida de un JSON
api/v0/chaincode/readAllCurrentAssets	GET	No tiene	Este *endpoint* devuelve todos los lotes de la *Blockchain* que permanecen activos, es decir, que no están vacíos

api/v0/chaincode/createAsset/:operacion	POST	JSON con la información del lote a crear y la indicación del tipo de operación (crear, juntar, mover) en la url. Ejemplo: api/v0/chaincode/createAsset/Crear { “ID”: “-”, “Estado”: “UPV”, “Lotes_previos”: “-1”, “Lote_actual”: “-1”, “Lote_mover”: “-1”, “ACargo”: “admin”, “Objetos”: [{ “ID”: “xrty6”, “Estado”: “UPV”, “Lote”: “-”, “Camino”: “-”, “Tipo”: “juego”, “Particular”: “Juan”, “Fecha”: “2021-03-15T23:00:00.000Z” }]} - En este caso se ha creado un lote con un objeto de tipo juego	Este *endpoint* realiza una operación sobre un lote, ya sea crear, juntar o mover el contenido, tanto parcial como totalmente. También actualiza el número de donaciones de cada usuario y devuelve el mensaje que recibe del *chaincode*
api/v0/chaincode/path/:id	GET	La id del lote se especifica en la url. Ejemplo: api/v0/chaincode/path/0 - Este ejemplo mostraría la traza del lote con id 0	Este *endpoint* devuelve los lotes anteriores o posteriores a uno dado, muestra por tanto la traza del lote buscado
api/v0/chaincode/readAllUserObjects/:user	GET	El nombre de usuario se especifica en la url. Ejemplo: api/v0/chaincode/readAllUserObjects/Juan - Este ejemplo mostraría todos los objetos donados por Juan	Este *endpoint* devuelve todos los objetos donados por un usuario
api/v0/chaincode/readAllAssets	GET	No tiene	Este *endpoint* devuelve todos los lotes de la *Blockchain*
api/v0/chaincode/numJuguetes/:user	GET	El nombre de usuario se especifica en la url. Ejemplo: api/v0/chaincode/numJuguetes/Juan - Este ejemplo mostraría el número de objetos donados por Juan	Este *endpoint* devuelve la cantidad de objetos donados por un usuario

Tabla 5.2: *Endpoints chaincode*

Cabe decir que existe un método en la clase de servicios llamado *init()* que sirve para iniciar la interacción con la red, crear el monedero, registrar al usuario y obtener el contrato, entre otras cosas.

5.3 Módulo 2

Este módulo tiene como objetivo la implementación del *frontend* y su conexión con el *backend*, por lo que se va a mostrar el resultado de la página web para apreciar su diferencia con el prototipado del apartado 5.1.

Cabe decir que para que el *frontend* se ejecute hace falta escribir en la terminal el comando: "ng serve -o".

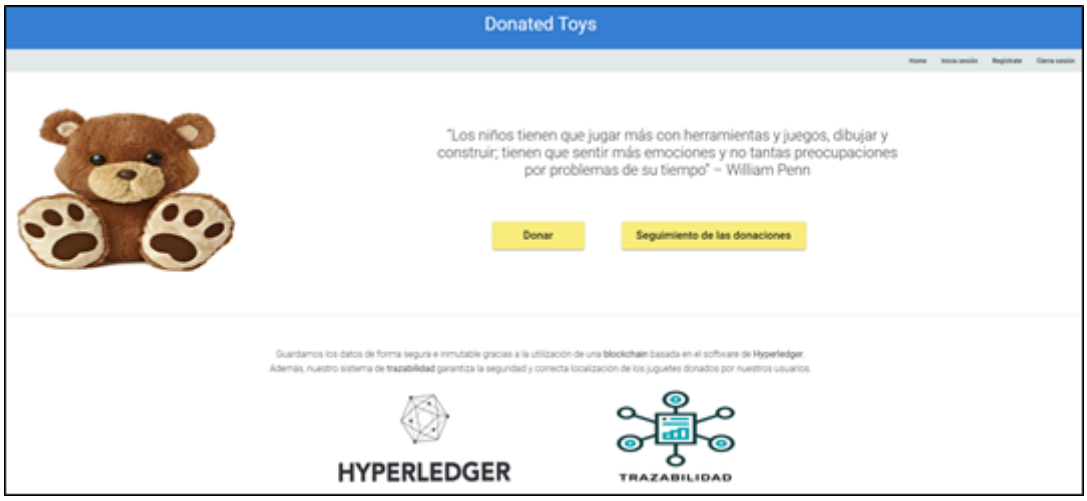

Figura 5.33: Pantalla de inicio (resultado final)

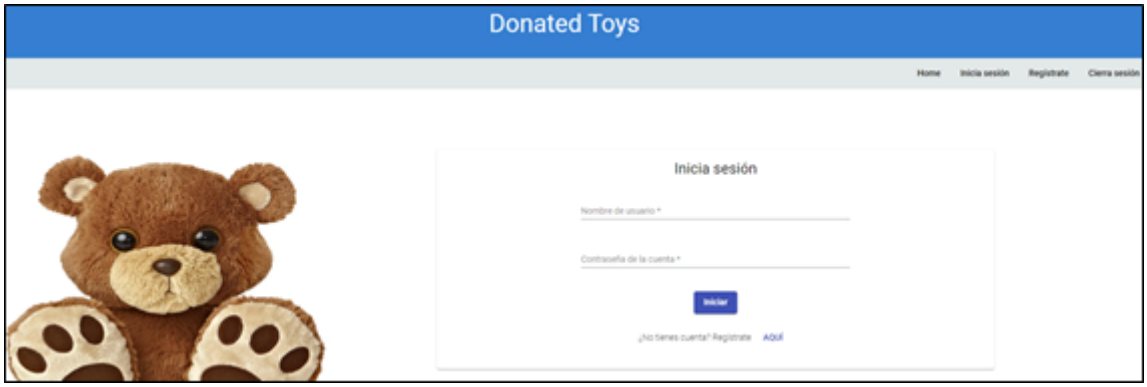

Figura 5.34: Inicio de sesión (resultado final)

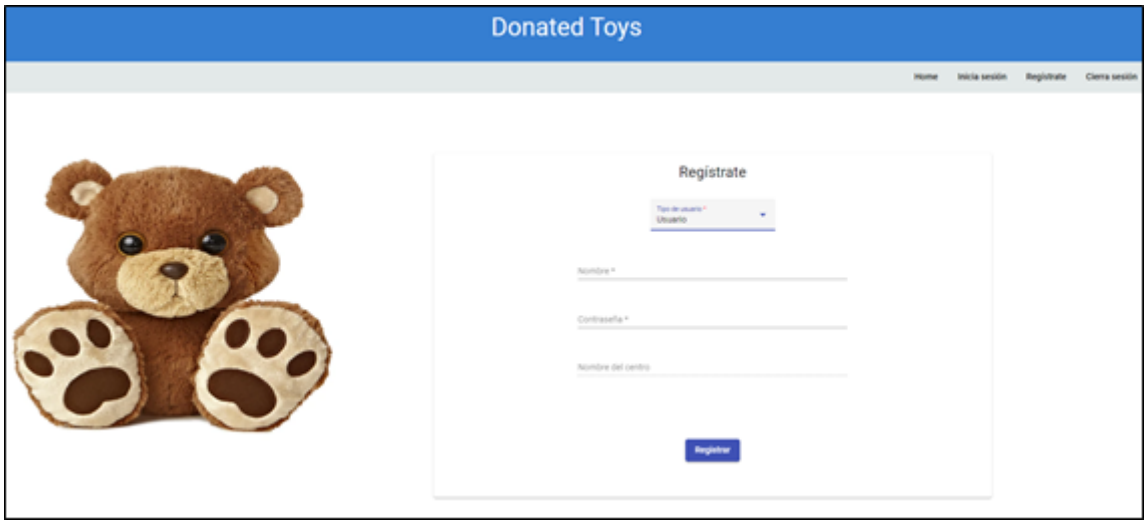

Figura 5.35: Registro de usuario (resultado final)

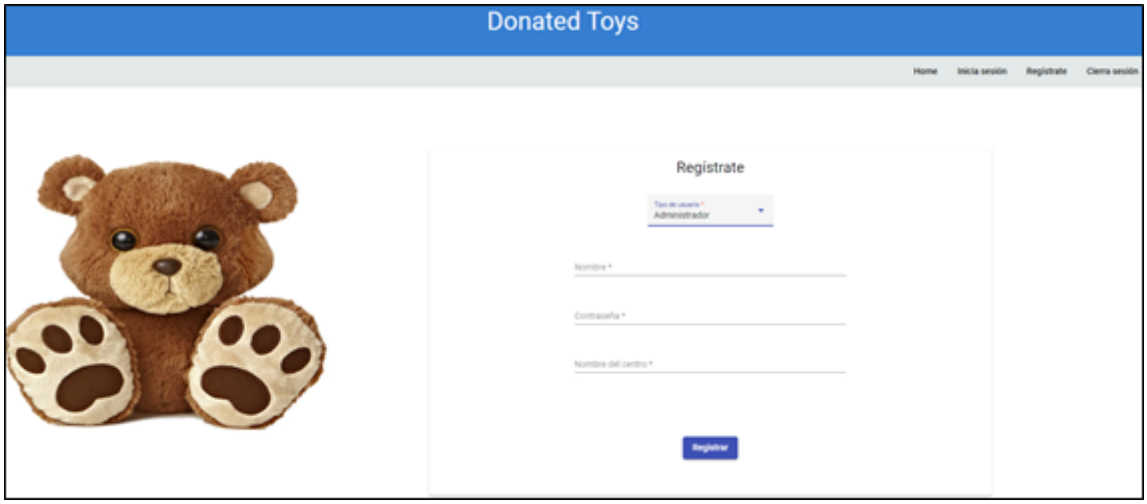

Figura 5.36: Registro de administrador (resultado final)

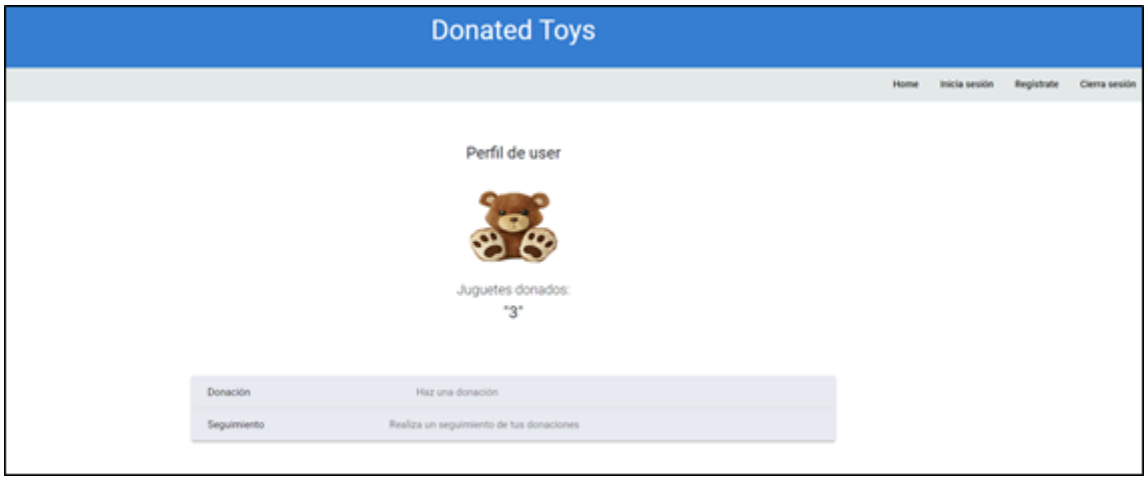

Figura 5.37: Vista de usuario (resultado final)

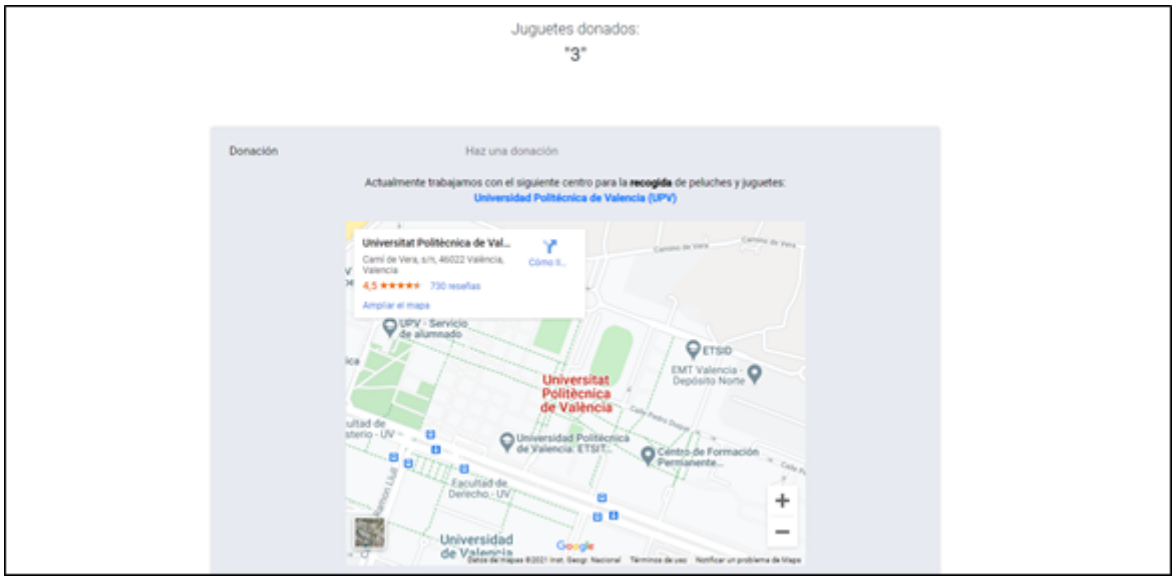

Figura 5.38: Vista de usuario: donar (resultado final)

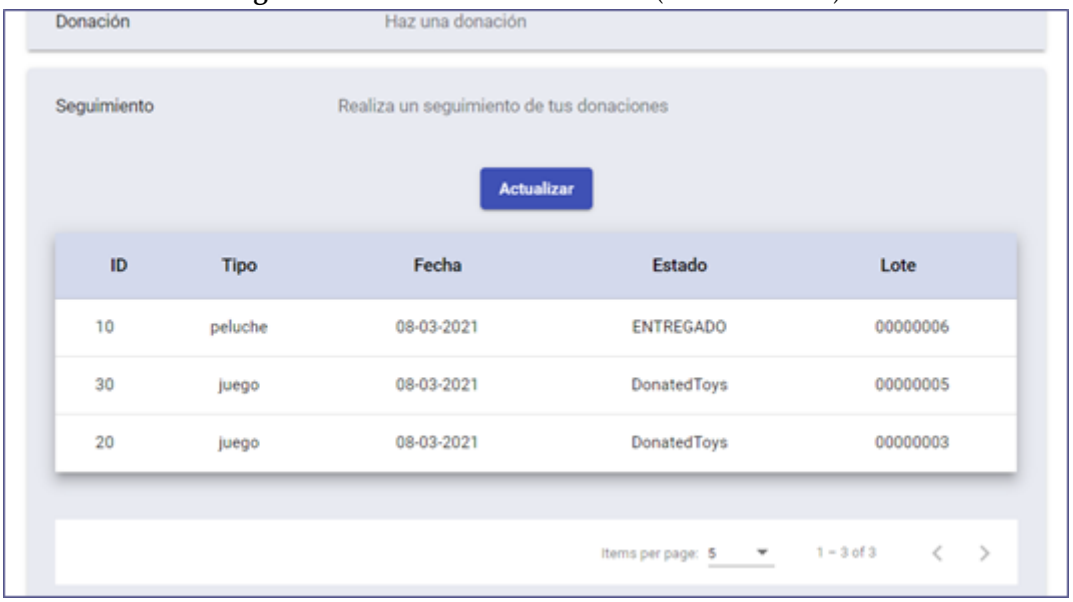

Figura 5.39: Vista de usuario: seguimiento (resultado final)

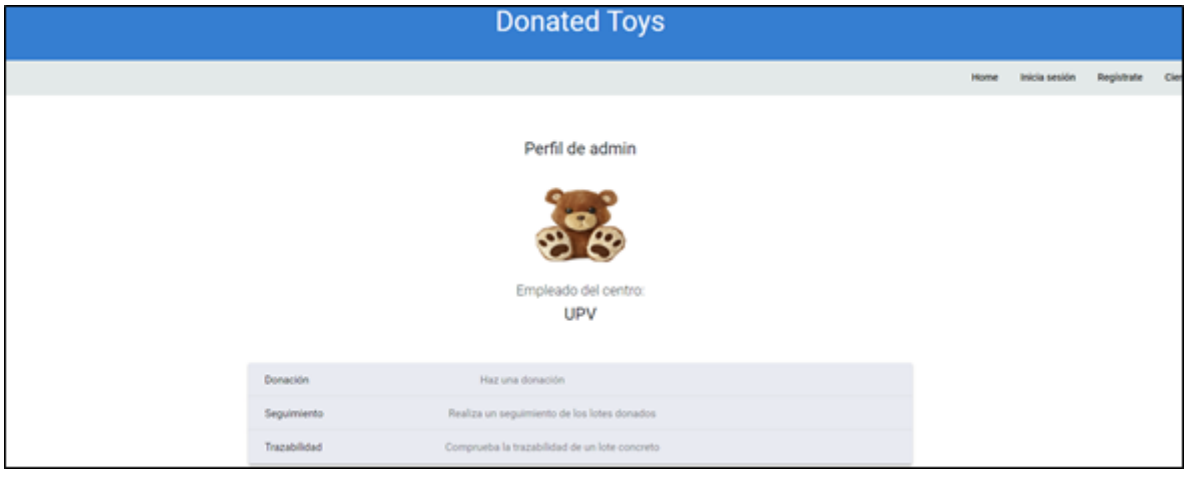

Figura 5.40: Vista de administrador (resultado final)

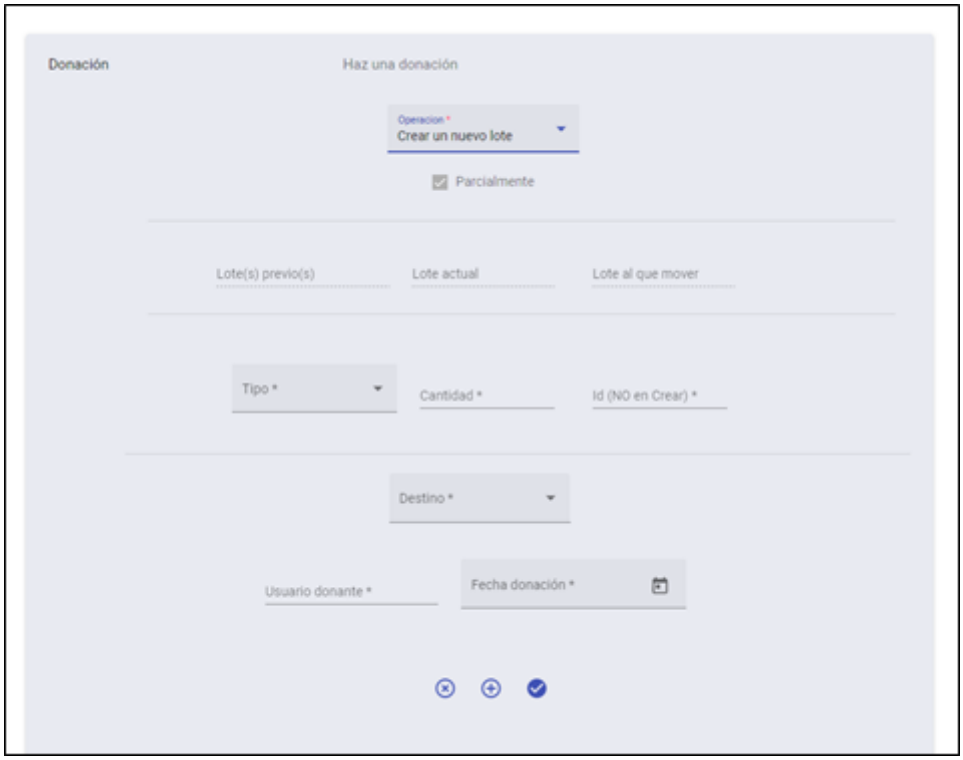

Figura 5.41: Vista de administrador: crear – Crear (resultado final)

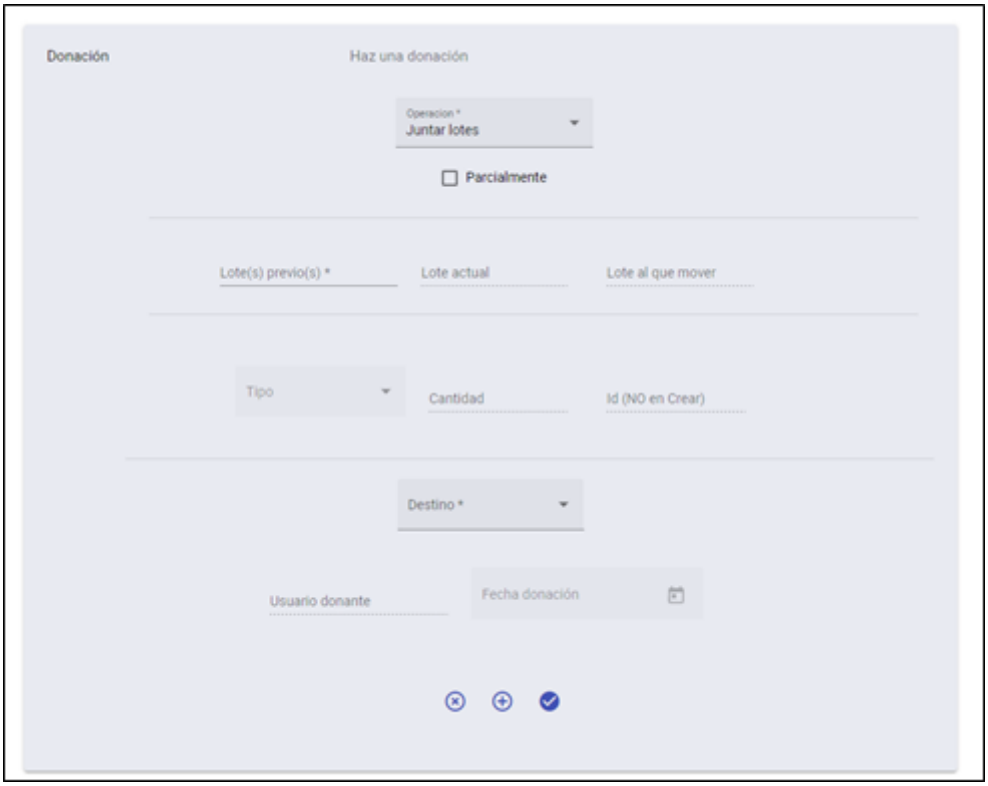

Figura 5.42: Vista de administrador: crear – Juntar totalmente (resultado final)

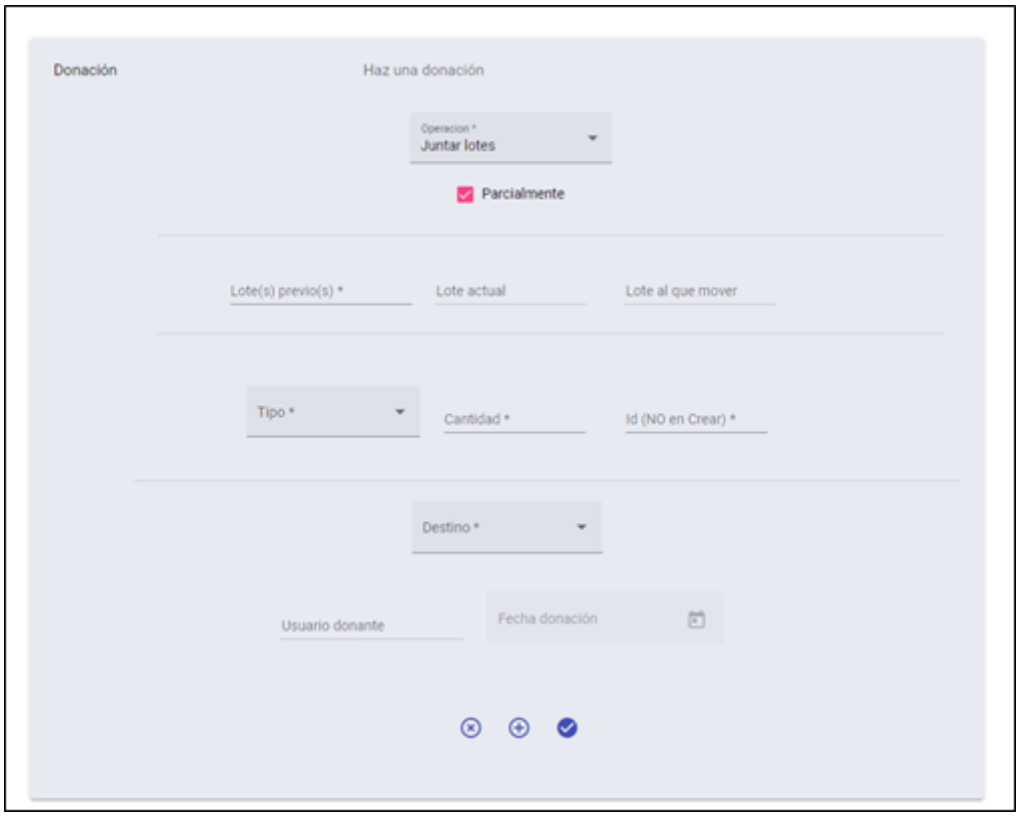

Figura 5.43: Vista de administrador: crear – Juntar parcialmente (resultado final)

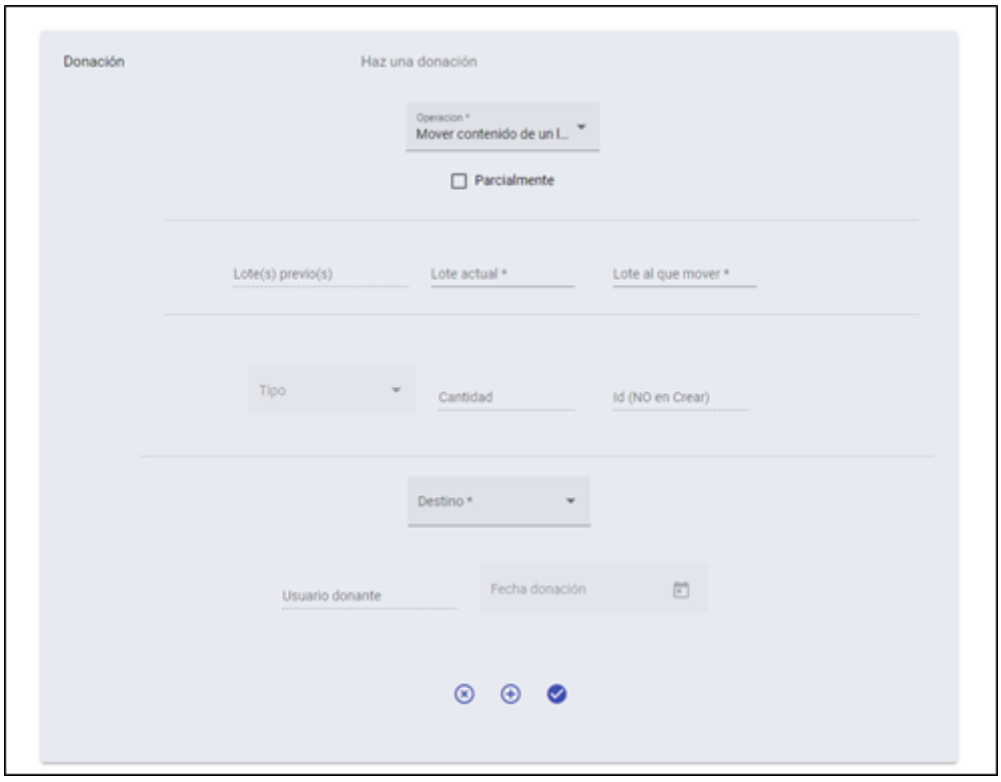

Figura 5.44: Vista de administrador: crear – Mover totalmente (resultado final)

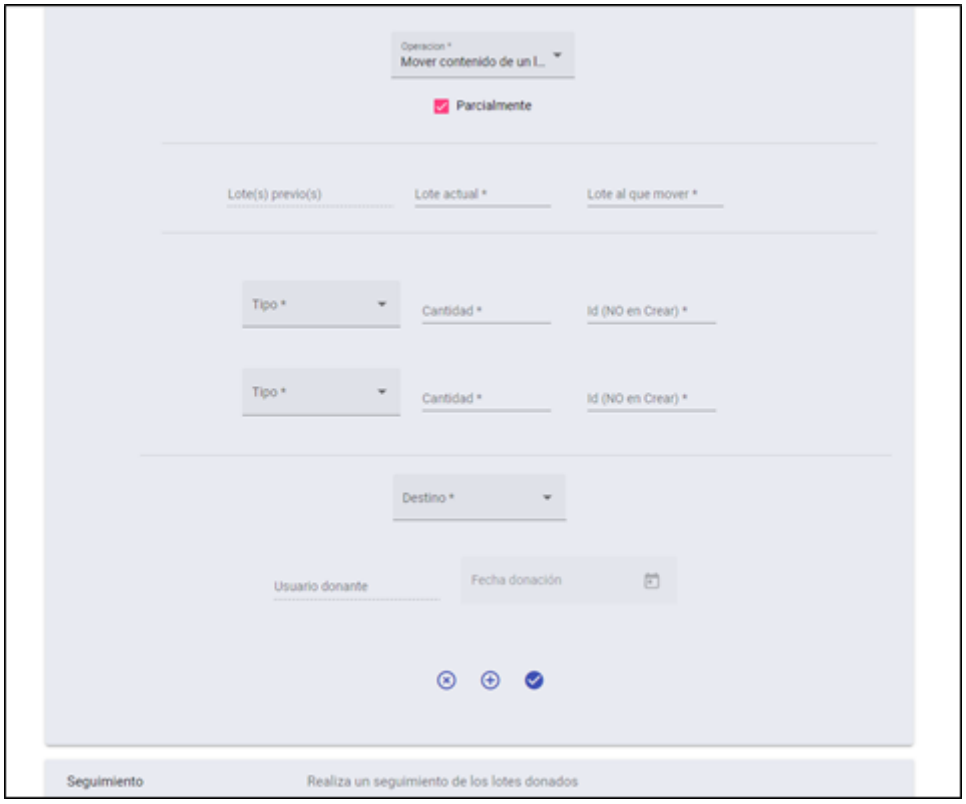

Figura 5.45: Vista de administrador: crear – Mover parcialmente (resultado final)

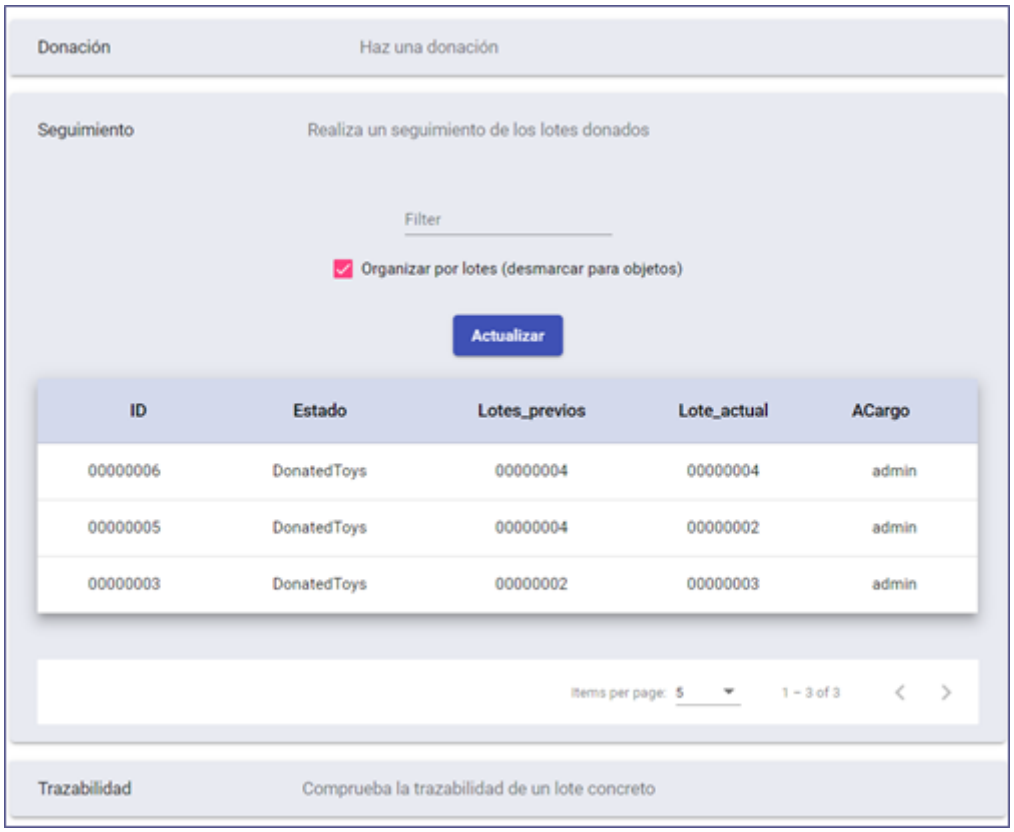

Figura 5.46: Vista de administrador: seguimiento - Lotes (resultado final)

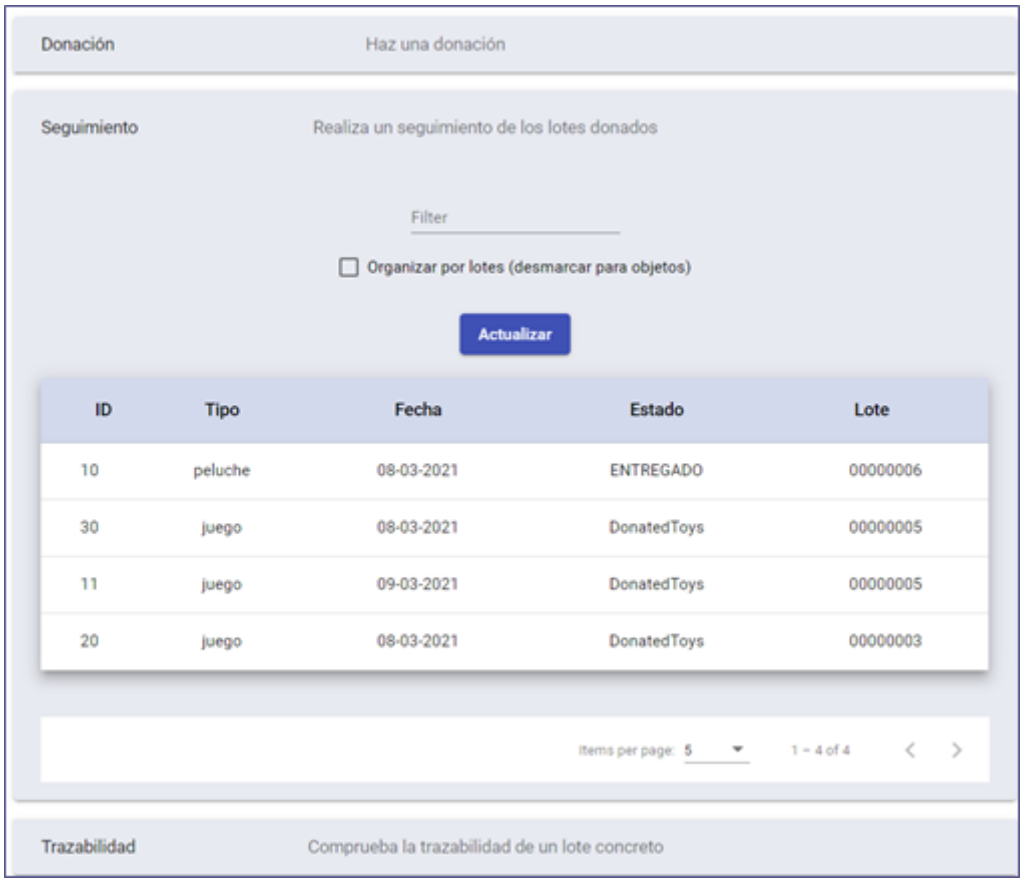

Figura 5.47: Vista de administrador: seguimiento - Objetos (resultado final)

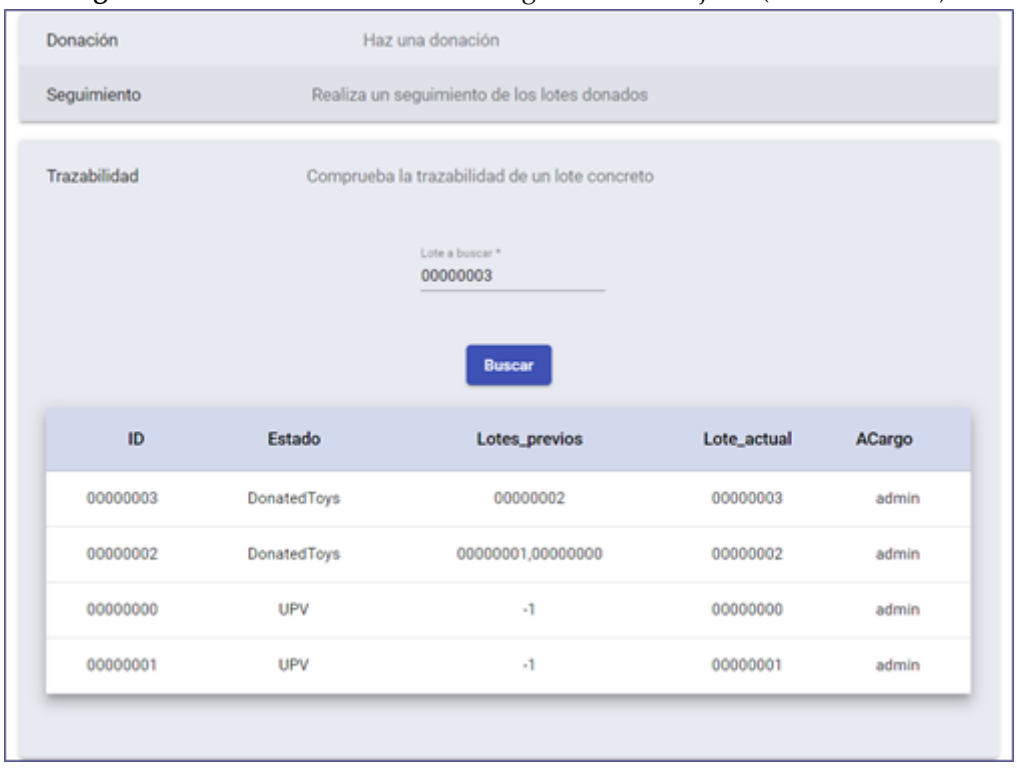

Figura 5.48: Vista de administrador: trazabilidad (resultado final)

5.4 Módulo 3

Este último módulo o fase 3 se encarga de realizar pruebas unitarias con los usuarios y administradores, estimar tiempos y analizar los resultados obtenidos de dichas pruebas, además de proponer posibles mejoras que se puedan desarrollar en un futuro. Por lo que se recogerá en los capítulos posteriores (el capítulo 6 de pruebas y validaciones, el capítulo 7 de conclusiones y el 8 de trabajos futuros).

CAPÍTULO 6
Validación y pruebas

En este capítulo se mostrarán las pruebas que se han hecho con la red, a fin de comprobar si su funcionamiento es correcto y estimar los tiempos de respuesta de las distintas acciones que un usuario (donante o administrador) puede realizar.

6.1 Registro e inicio de sesión

El primer experimento consiste en realizar un registro de un usuario en la web (figuras 6.1 y 6.2) y posteriormente (tras cerrar sesión) comprobar que se puede iniciar sesión (figuras 6.3 y 6.4) correctamente en la página.

El tiempo de respuesta para el registro de un usuario en MongoDB (sea del tipo que sea) es de menos de un segundo.

Figura 6.1: Registro de usuario (Pruebas)

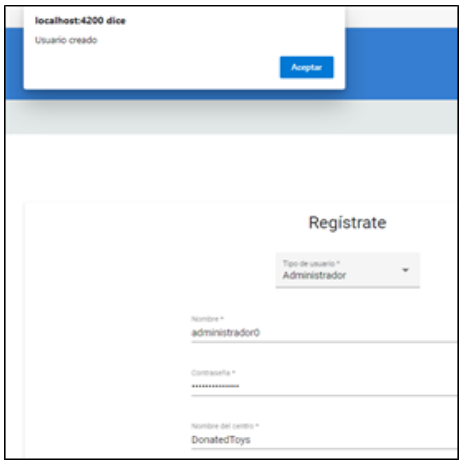

Figura 6.2: Registro de administrador (Pruebas)

Figura 6.3: Inicio de sesión de usuario (Pruebas)

Figura 6.4: Inicio de sesión de administrador (Pruebas)

En el caso del inicio de sesión el tiempo de espera coincide con el de registro, inferior a un segundo. A continuación, se mostrarán en la figura 6.5 los datos de MongoDB para corroborar la creación de los usuarios.

Figura 6.5: Base de datos de MongoDB (En Ubuntu)

6.2 Caso 1: Donación

A continuación, se tratarán ejemplos de creación de lotes de donaciones, así como las variantes de mover y juntar contenidos.

6.2.1. Caso 1.1: Nueva donacióin

En este caso el planteamiento es el de un usuario que acude a un centro a realizar una donación. Para simplificar se va a utilizar el usuario de nombre “usuario0” quien donará 2 juguetes, 3 peluches y 1 juego siendo que no ha donado nada previamente.

En primer lugar, se puede ver que en la figura 6.6 el usuario0 ha realizado 0 donaciones (“Juguetes donados: 0”), dicha cantidad aparece tras 3 segundos o menos de espera.

Figura 6.6: Caso 1.1: perfil de usuario (pruebas)

Seguidamente, el usuario acude a un centro (en la figura 6.7 es DonatedToys) y un recepcionista que ha iniciado sesión como administrador (véase el administrador0) procede a guardar la información del lote inicial (las donaciones del usuario que luego se moverán a su respectivo lugar) en la *Blockchain*.

Figura 6.7: Caso 1.1: perfil de administrador (pruebas)

El tiempo de espera para que salga el centro del empleado es inmediato.

Por otro lado, el tiempo que tarda en añadirse un nuevo lote a la *Blockchain* y notificar al usuario es de unos 4 segundos (es aproximadamente igual para todas las operaciones, independientemente de que se hagan con el contenido parcial o total).

Además, como se aprecia en la figura 6.8 la acción de crear un nuevo lote se ha realizado con éxito, también muestra las ids de los objetos añadidos y la propia del lote (00000007 porque en la *Blockchain* hay un esquema inicial de seis lotes, siendo el siguiente el número 7, por lo que la id devuelta es correcta).

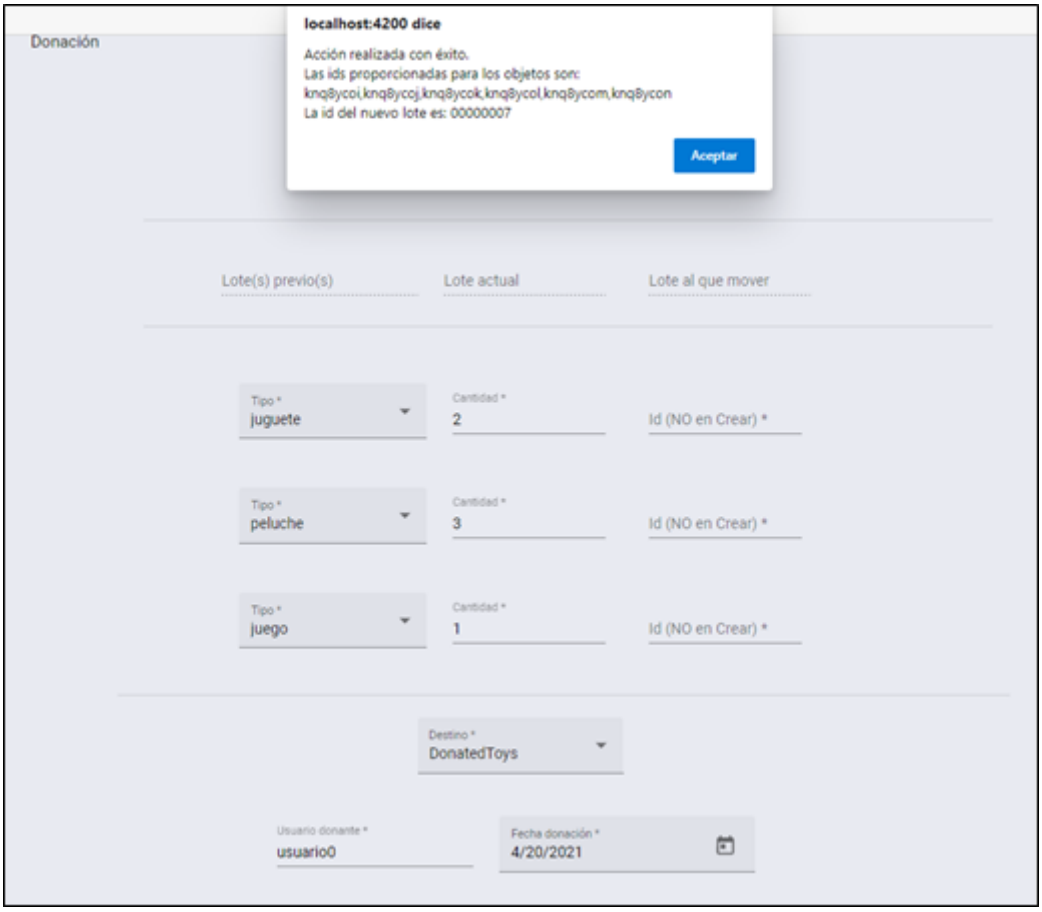

Figura 6.8: Caso 1.1: creación de un lote (pruebas)

A continuación, se obtienen los lotes activos como en el caso 3 para comprobar que dichas ids están en el lote 00000007 (figura 6.9).

Figura 6.9: Caso 1.1: seguimiento del lote creado (pruebas)

6.2.2. Caso 1.2: Mover contenido a otro lote (parcialmente)

En este nuevo caso (figura 6.10) un administrador decide mover los 3 peluches de usuario0 a otro lote (para ello elige el lote 00000005 que contiene los artículos con las ids 11 y 30). Esto se podría dar en el caso de que se separasen los objetos donados por tipo.

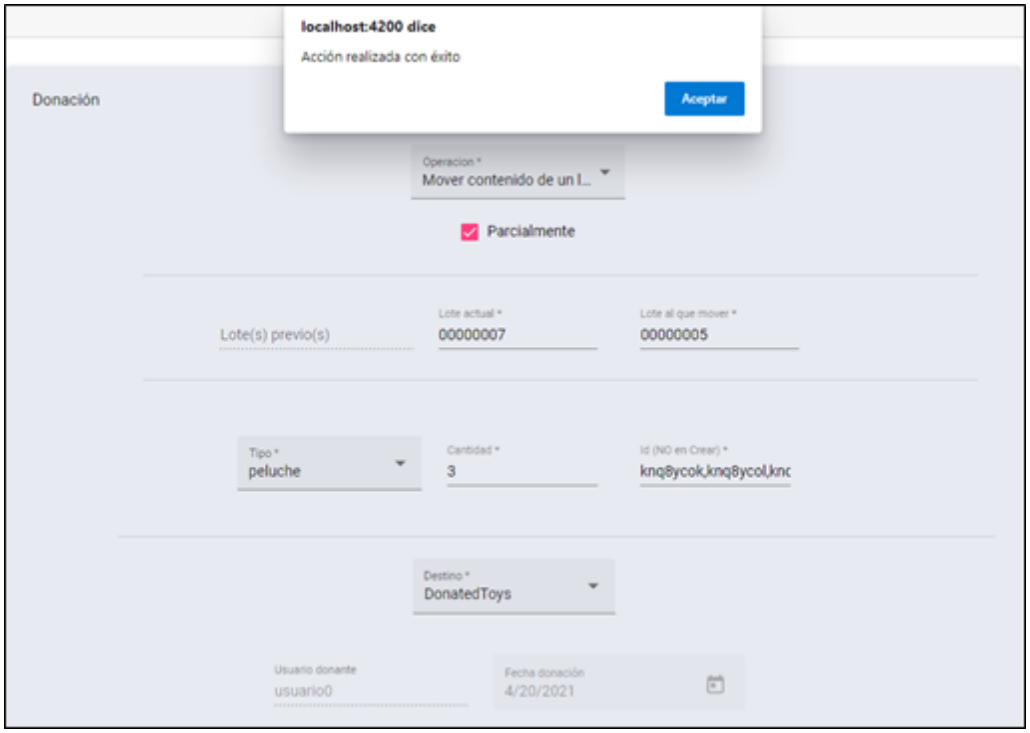

Figura 6.10: Caso 1.2: traslado del contenido parcialmente (pruebas)

Nuevamente el tiempo de espera es de unos 4 segundos aproximadamente. A continuación, el administrador obtiene los lotes activos como en el caso 3 para comprobar que dichas ids están en el lote 00000005.

Como se aprecia en la posterior figura 6.11, el lote 00000008 alias el lote 00000005 contiene los objetos con las ids 30 y 11 de este lote y las 3 nuevas que se han añadido. Cabe decir que, el campo *ID* de la tabla no es 00000005 porque en la *Blockchain* no se pueden guardar dos lotes con el mismo id, por lo que el campo que identifica realmente al lote es *Lote_actual*, tampoco es conveniente modificar los datos del lote 00000005 para que se pueda guardar con esta id, porque entonces se podría manipular más fácilmente la información que contiene cada lote, borrando, creando o moviendo del registro (es decir, de la cadena de bloques) artículos donados.

Figura 6.11: Caso 1.2: seguimiento del contenido trasladado (pruebas)

También se ha generado un nuevo lote como copia del lote 00000007 (su nuevo ID en la *Blockchain* es 00000009) con los objetos que no se han movido (figura 6.12).

Figura 6.12: Caso 1.2: seguimiento del contenido no trasladado (pruebas)

6.2.3. Caso 1.3: Mover contenido a otro lote (totalmente)

En este caso (figura 6.13) el administrador0 procede a mover el contenido del lote con id 00000003 (contiene el objeto con id 20) al lote 00000006 (alberga el objeto con id 10). Esto se podría utilizar cuando se quisiera mover el contenido completo de un lote a un contenedor para ser trasladado a otro centro, en el caso que se va a tratar el administrador especificará que dicho lote con el nuevo contenido tendrá como destino la UPV.

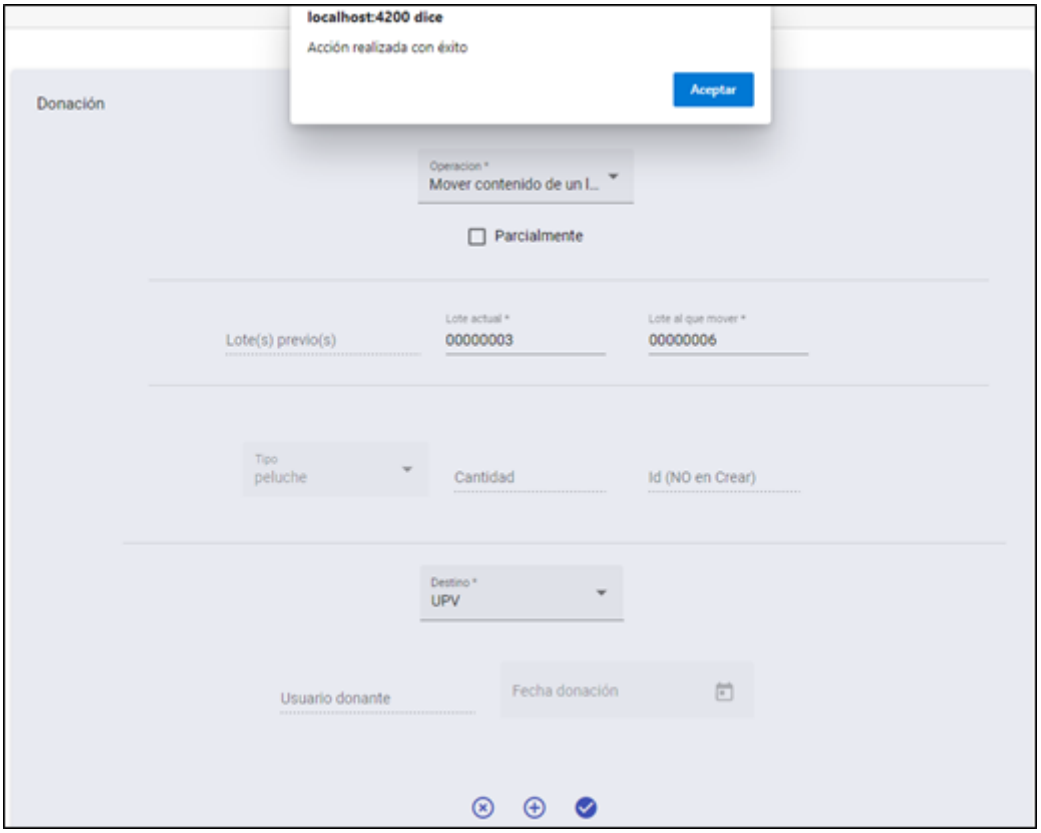

Figura 6.13: Caso 1.3: traslado del contenido totalmente (pruebas)

Al comprobar el resultado en seguimiento se puede ver en la figura 6.14 que se ha creado un nuevo lote con id 00000010 que tiene como lotes previos los lotes anteriores y cuyo lote actual es el 00000006.

Figura 6.14: Caso 1.3: seguimiento del contenido trasladado (pruebas)

6.2.4. Caso 1.4: Juntar contenido (parcialmente)

En este caso (figura 6.15) el administrador decide juntar el contenido de dos lotes en uno nuevo, para lo cual el sistema le devuelve la id que tendrá el nuevo lote. En el ejemplo el administrador procede a crear dos nuevos lotes (00000011 con dos juegos con ids knq6gp9z y knq6gpa0 devueltas por el sistema; y 00000012 con dos peluches con ids knq6h5uc y knq6h5ud, y un juego con id knq6h5ue) y seleccionar del primero el objeto con la id knq6gp9z y del segundo el objeto con la id knq6h5ue.

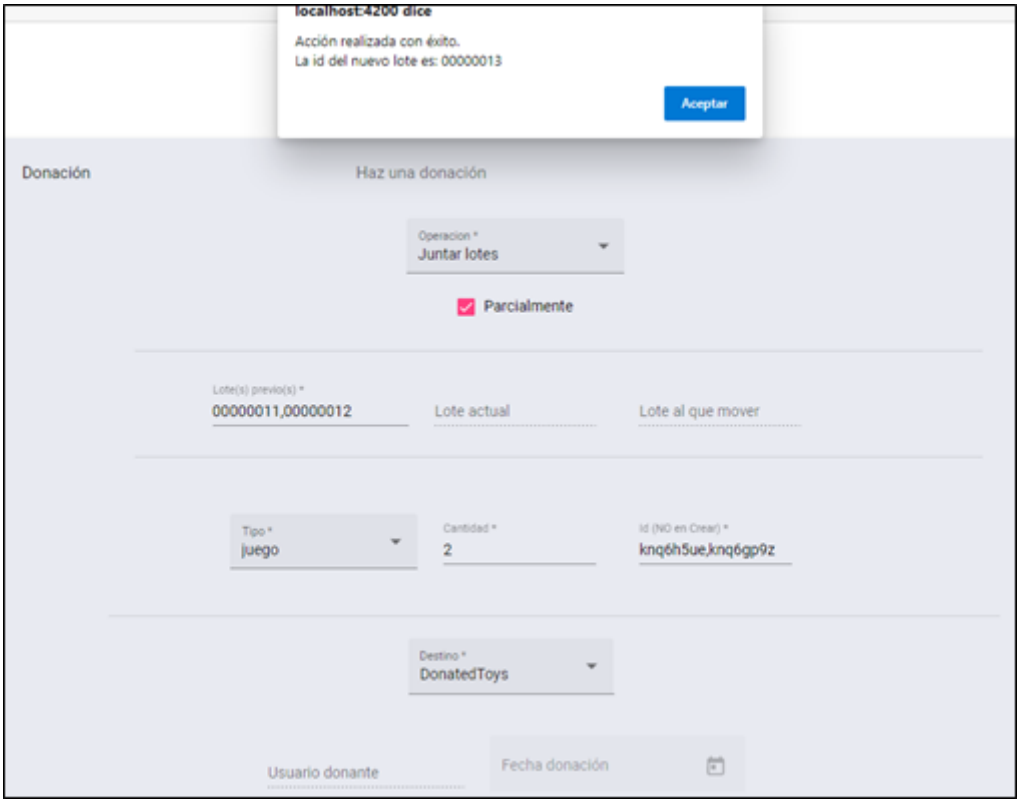

Figura 6.15: Caso 1.4: mezcla del contenido de dos lotes parcialmente (pruebas)

Seguidamente se comprueba si el resultado es correcto. En la figura 6.16 se muestra como el nuevo lote con id 00000013 contiene los objetos juntados pertenecientes a los lotes anteriores (únicamente lo que se especificaron que iban a moverse).

Figura 6.16: Caso 1.4: seguimiento del contenido mezclado (pruebas)

En las posteriores figuras (6.17 y 6.18) se muestran los lotes previos con el contenido que no se ha trasladado.

Figura 6.17: Caso 1.4: seguimiento del contenido no movido del primer lote (pruebas)

Figura 6.18: Caso 1.4: seguimiento del contenido no movido del segundo lote (pruebas)

6.2.5. Caso 1.5: Juntar contenido (totalmente)

En este caso el administrador junta totalmente el contenido de dos lotes, para ello selecciona los lotes con las ids 00000015 y 00000014 de las figuras previas (6.17 y 6.18).

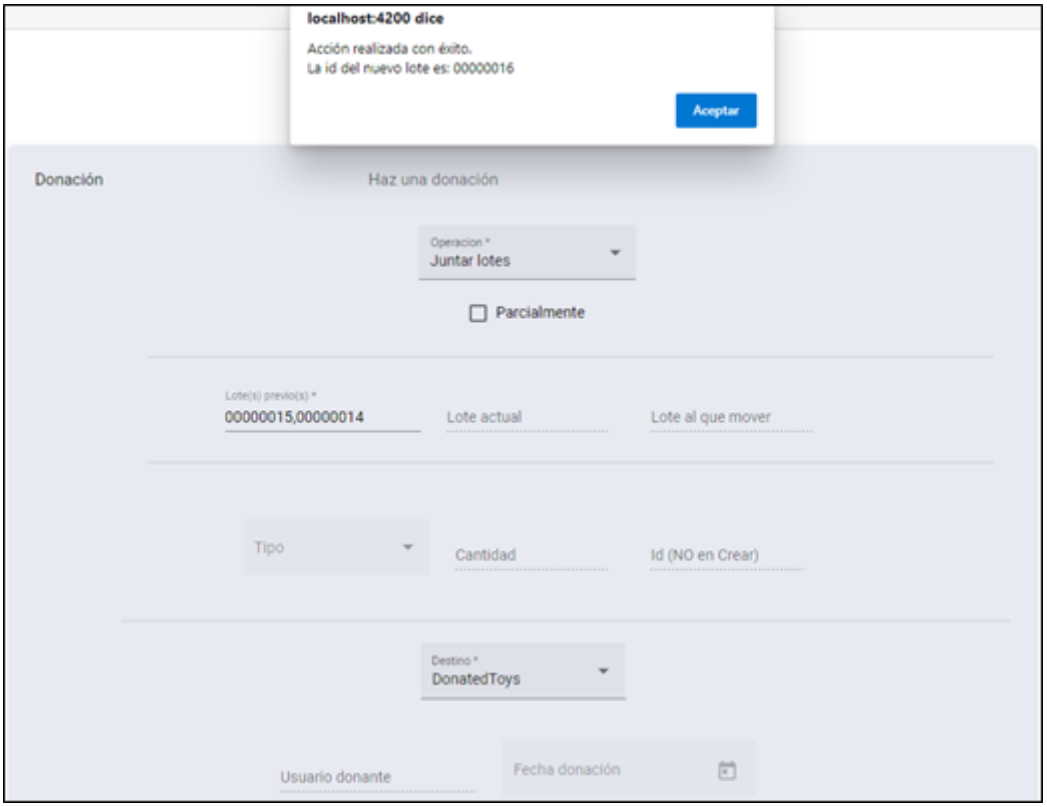

Figura 6.19: Caso 1.5: mezcla del contenido de dos lotes totalmente (pruebas)

A continuación, se muestra en la figura 6.20 la prueba de que se ha creado un nuevo lote con los objetos de los dos lotes especificados. Cabe decir que, estos mismos lotes desaparecen de la tabla de seguimiento puesto que están inactivos (su contenido se ha movido, pese a que en la *Blockchain* siguen apareciendo no están realmente en uso).

Figura 6.20: Caso 1.5: seguimiento del contenido mezclado (pruebas)

6.3 Caso 2: Seguimiento por parte del administrador

Se han hecho distintas pruebas a lo largo del caso 1 y todas han tardado menos de 4 segundos. Para la obtención de objetos en la *Blockchain* el tiempo es el mismo, en la siguiente figura (6.21) se muestra el resultado.

ID	Tipo	Fecha	Estado	Lote
knq6h5uc	peluche	2021-04-19T22:00:00.000Z	DonatedToys	00000016
knq6h5ud	peluche	2021-04-19T22:00:00.000Z	DonatedToys	00000016
knq6gpa0	juego	2021-04-19T22:00:00.000Z	DonatedToys	00000016
knq6h5ue	juego	2021-04-19T22:00:00.000Z	DonatedToys	00000013
knq6gp9z	juego	2021-04-19T22:00:00.000Z	DonatedToys	00000013

Figura 6.21: Caso 2: seguimiento de los artículos donados (pruebas)

Ahora se procede a marcar el segundo de estos objetos como entregado, el resultado es el que se muestra a continuación (figura 6.22).

ID	Tipo	Fecha	Estado	Lote
knq6h5uc	peluche	2021-04-19T22:00:00.000Z	DonatedToys	00000016
knq6h5ud	peluche	2021-04-19T22:00:00.000Z	ENTREGADO	00000016
knq6gpa0	juego	2021-04-19T22:00:00.000Z	DonatedToys	00000016
knq6h5ue	juego	2021-04-19T22:00:00.000Z	DonatedToys	00000013
knq6gp9z	juego	2021-04-19T22:00:00.000Z	DonatedToys	00000013

Figura 6.22: Caso 2: marcado un artículo como “Entregado” (pruebas)

A partir de ahora el peluche de la segunda fila pasará a estar “Entregado”, por lo que el usuario podrá ver que su donación le ha llegado a alguien.

6.4 Caso 3: Seguimiento por parte del usuario

En este caso de uso un usuario desea comprobar el estado de sus donaciones. Como se muestra en la posterior figura (6.23) ha realizado un total de 11 donaciones y de estas una de ellas (la marcada como entregada en el caso 2) ha sido entregada. El tiempo de espera ha sido de 4 segundos.

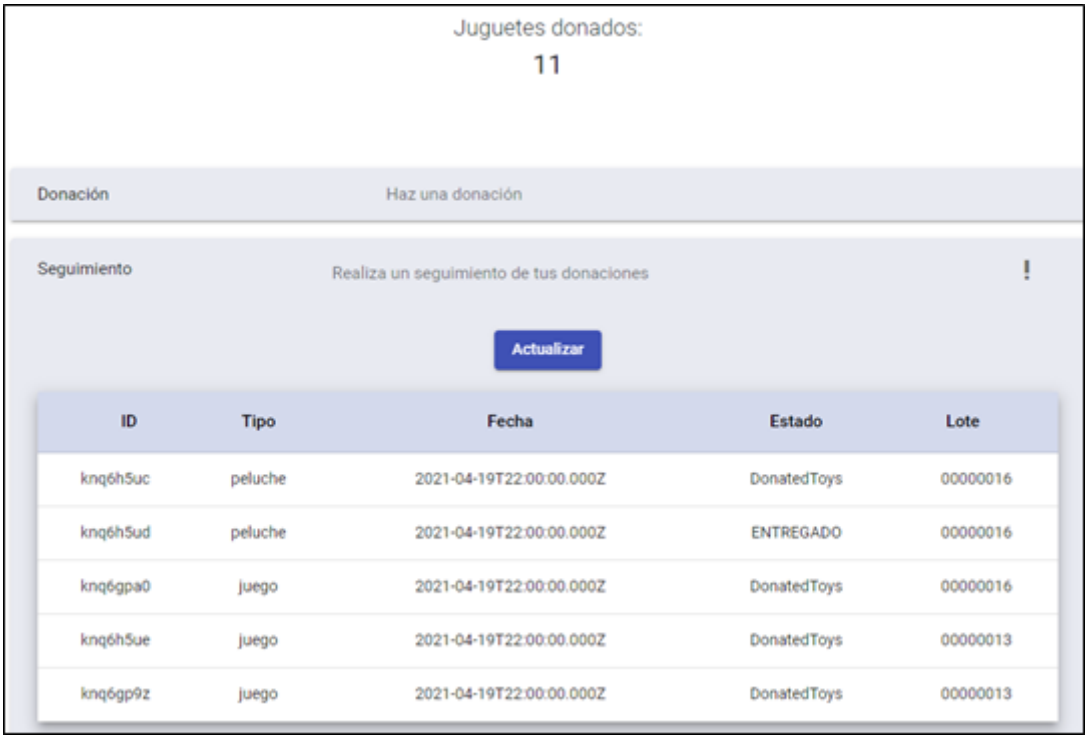

Figura 6.23: Caso 3: seguimiento de las donaciones del usuario (pruebas)

6.5 Caso 4: Trazabilidad de un lote

En este caso el administrador decide comprobar la traza de un lote porque sospecha que parte de los objetos que almacena han podido "extraviarse". Para este ejemplo se ha escogido el lote con id 00000011.

En la figura 6.24 se aprecian cuatro resultados obtenidos en aproximadamente 4 segundos, entre ellos el propio lote 00000011 (lote inicial) que contiene dos objetos.

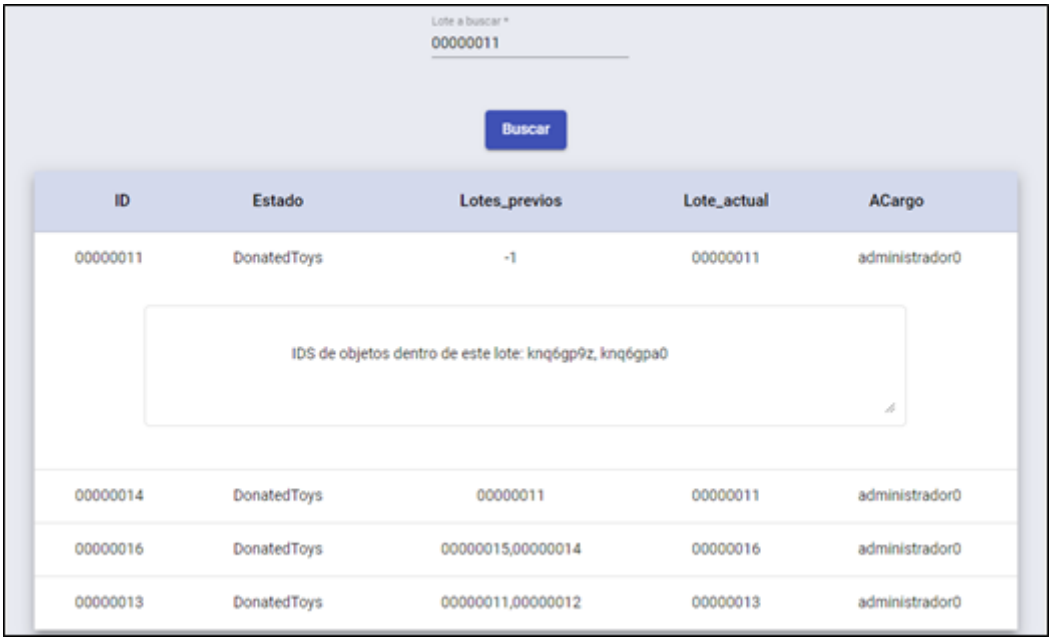

Figura 6.24: Caso 4: trazabilidad de un lote (pruebas)

También está el lote 00000014 que es realmente una copia del lote 00000011, por tanto, contiene los objetos que no se han trasladado a ningún otro lote (figura 6.25). El lote 00000013 contiene los objetos que se trasladaron inicialmente junto con los de otro lote (figura 6.26). Y, finalmente, el lote 0000016 que alberga los objetos del lote copia (00000014) y de otro lote (figura 6.27).

Figura 6.25: Caso 4: trazabilidad de un lote (1) (pruebas)

Figura 6.26: Caso 4: trazabilidad de un lote (2) (pruebas)

Figura 6.27: Caso 4: trazabilidad de un lote (3) (pruebas)

En la figura posterior (6.28) se aprecia como el lote 00000011 tras una operación de juntar parcialmente (véase el caso 1.4) se ha trasladado su contenido al lote 00000013 que está de color azul puesto que es un lote activo, y que el contenido restante se ha quedado en el mismo lote bajo una nueva id en la *Blockchain* (00000014). Además, el contenido de esa copia se ha juntado totalmente con el del lote 00000015 (véase el caso 1.5) dando lugar a un nuevo lote activo con la id 00000016.

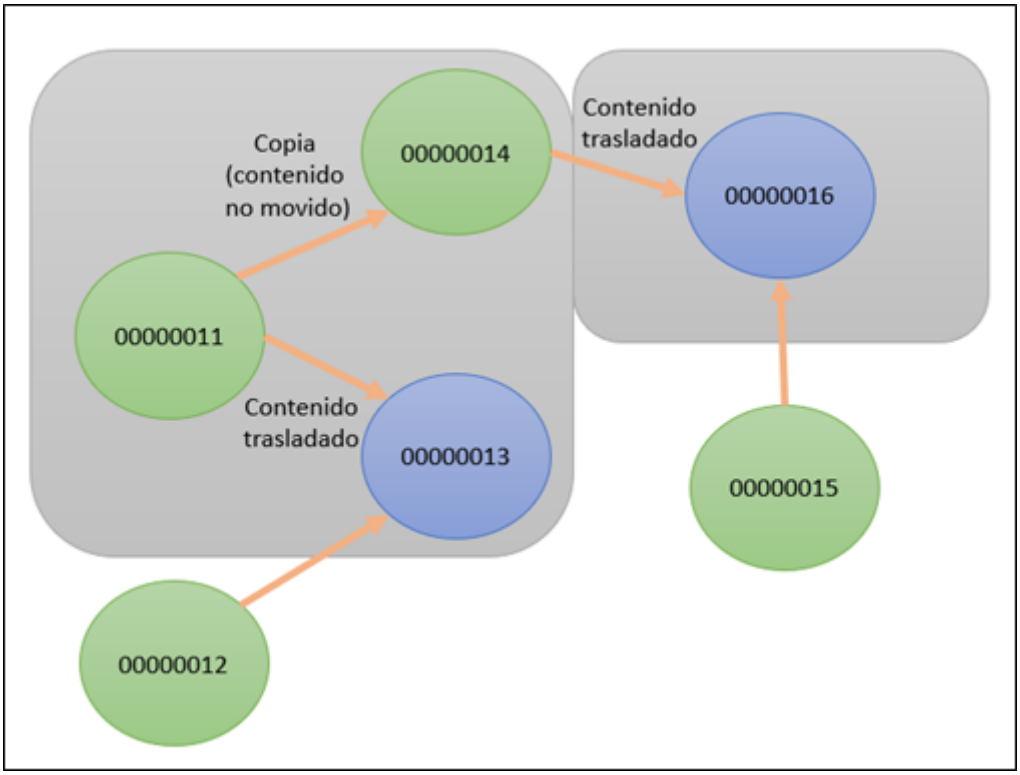

Figura 6.28: Caso 4: esquema de la traza de un lote (pruebas)

CAPÍTULO 7
Conclusiones

En este trabajo se ha realizado un sistema de trazabilidad de lotes de juguetes donados utilizando para ello la tecnología de Hyperledger Fabric, es decir, tecnología *Blockchain*. En cuanto al objetivo secundario, se ha desarrollado una página web, basada en Angular y NestJS, que permite la interacción del usuario con la *Blockchain* a través de un Contrato Inteligente que en Fabric se conoce como *chaincode*, para lo cual se ha realizado una distinción entre los usuarios normales que son los donantes, y los administradores del sitio web que comprenden los recepcionistas de los centros de recogida y los distribuidores de los centros de distribución.

Con todo esto, empleando para ello la tecnología seleccionada tras una búsqueda bibliográfica exhaustiva, se ha obtenido un sistema que relaciona las acciones web de los usuarios totales (usuarios y administradores) con un *backend* que se conecta a su vez con la base de datos en MongoDB para almacenar las credenciales de los usuarios y, con el *chaincode* para interactuar con una red híbrida en Hyperledger. De esta forma se cumplen los objetivos de los módulos o fases 0, 1 y 2.

Por lo que respecta a la fase 3, en el capítulo 6 se comentan las pruebas finales y mediciones que se han realizado. De aquí se extrae que, si comparamos el tiempo de respuesta de las acciones ligadas con la base de datos de MongoDB y el tiempo de respuesta de la *Blockchain*, obviamente MongoDB es más rápido. Sin embargo, el tiempo medido para la *Blockchain* es en torno a 4 segundos, por lo que tampoco supone un problema muy grande si a cambio de dichos segundos se gana en privacidad, seguridad, inmutabilidad y trazabilidad. Por otro lado, las pruebas demuestran el correcto funcionamiento de la red y las ventajas que tiene el sistema para seguir o localizar un producto, además de la flexibilidad que posee al poderse adaptar a otros casos de trazabilidad de artículos tanto a nivel empresarial, como de cooperación al desarrollo como es este caso.

En conclusión, se puede considerar que se ha conseguido llevar a cabo el sistema diseñado previamente para cubrir las carencias detectadas en la trazabilidad de donaciones de juguetes, además, posee un gran potencial para ser aplicado a otros campos evitando así tramas de corrupción, apropiación indebida de productos, mejora logística, etc.

7.1 Relación del trabajo desarrollado con los estudios cursados

Este trabajo ha sido realizado gracias a los conocimientos adquiridos de gran cantidad de materias cursadas a lo largo del Grado de Ingeniería Informática. Seguidamente, se va a mencionar y detallar dónde se ve reflejada dicha influencia.

En primer lugar, cabe destacar que se emplean conocimientos de diversas asignaturas del ámbito de programación, tanto asignaturas básicas como otras avanzadas relaciona-

das con el manejo de sistemas distribuidos, véase "Introducción a la Informática y la Programación" (IIP), "Programación" (PRG), "Concurrencia de Sistemas Distribuidos" (CSD) y "Tecnologías de los Sistemas de Información en la Red" (TSR).

Asimismo, este proyecto también cuenta con gran influencia de asignaturas como "Ingeniería de Software" (ISW), "Interfaces Persona Computador" (IPC) y "Gestión de Proyectos" (GPR). Puesto que esas materias han servido para estructurar, organizar, documentar y mostrar un producto que trata de ser intuitivo y fácil de utilizar para el usuario final.

En segundo lugar, se ve claramente reflejado el peso de la rama de Computación, escogida en la carrera, ya que se trata de un sistema dirigido al consumo cotidiano (cualquier usuario podría donar en cualquier momento o revisar el estado de sus donaciones a través de la página web), también introduce mecanismos de automatización como lo son los Contratos Inteligentes (*chaincode*) y la aplicación resultante realiza un procesamiento y extracción de grandes volúmenes de información textual al tratar con una base de datos como lo es *Blockchain*. Además, se han utilizado conocimientos derivados de la asignatura "Algorítmica" (ALG), propia de la especialización.

Por otro lado, las competencias transversales de la Universidad Politécnica de Valencia (UPV) se ven reflejadas en los siguientes casos:

- **CT-01. Comprensión e integración:** al diseñar el sistema pues es necesario utilizar tanto los conocimientos adquiridos a lo largo del grado como los de fuentes externas a la universidad.
- **CT-02. Aplicación y pensamiento práctico:** en la argumentación que se realiza en los apartados 3.3 y 3.4 sobre que soluciones existen al problema planteado, y cuál es la que supone un mayor beneficio para los usuarios del sistema final.
- **CT-03. Análisis y resolución de problemas:** al analizar el problema (capítulo 3) y posteriormente diseñar la solución (capítulo 4).
- **CT-04. Innovación, creatividad y emprendimiento:** la idea de este proyecto surgió como solución para un problema que se encontró y para el que tras una larga búsqueda no se encontraron soluciones que ofreciesen a la vez inmutabilidad de la información almacenada, seguridad y trazabilidad.
- **CT-05. Diseño y proyecto:** al diseñar y estructurar el trabajo del proyecto se han utilizado técnicas vistas en la asignatura de "Gestión de Proyectos" (GPR).
- **CT-06. Trabajo en equipo y liderazgo:** al realizarse diversas reuniones con varias personas para comprobar que el sistema fuese intuitivo, útil y agradable de cara al usuario.
- **CT-07. Responsabilidad ética, medioambiental y profesional:** al analizar los aspectos legales y éticos que conlleva este proyecto en el apartado 3.2.
- **CT-08. Comunicación efectiva:** al redactar esta memoria se ha respetado el formato requerido y se ha intentado organizar la información de la forma más clara posible.
- **CT-09. Pensamiento crítico:** en el capítulo 2, al analizar el contexto tecnológico para el problema planteado en el capítulo 1 y al tomar decisiones acerca del diseño e implementación del sistema en el capítulo 3.
- **CT-10. Conocimiento de problemas contemporáneos:** el uso de tecnología *Blockchain* para crear un sistema de trazabilidad que resuelve un problema atemporal

como son las donaciones de juguetes hace que se proponga una solución contemporánea y novedosa.

- **CT-11. Aprendizaje permanente:** al investigar y comprender tecnologías que no se habían estudiado en el grado, lo cual ha supuesto un autoaprendizaje de ciertas herramientas para poder desarrollar el proyecto final.
- **CT-12. Planificación y gestión del tiempo:** al dividir el trabajo a realizar mediante el esquema de la EDT (figura 3.1), estimando a su vez los tiempos requeridos para cada tarea y subtarea (apartado 3.5).
- **CT-13. Instrumental específica:** al utilizar tecnologías especializadas como lo es, por ejemplo, Hyperledger Fabric.

CAPÍTULO 8
Trabajos futuros

A lo largo del desarrollo del proyecto se han detectado algunas posibles mejoras y ampliaciones para este sistema que proceden a detallarse en los siguientes párrafos.

En primer lugar, existen una serie de mejoras realizables a nivel de funcionalidad de la página web de cara al usuario, como son la inclusión de un correo electrónico y número de teléfono en el registro de los usuarios para poder enviar un correo o mensaje de recuperación de cuenta si han olvidado la contraseña, la posibilidad de actualizar los parámetros que se registran de un usuario en la web (por ejemplo, el centro en el que trabaja, la contraseña, móvil, etc.) y por último, el permitir darse de baja a un usuario que ya no quisiera seguir registrado en el sistema.

Asimismo, también referente a la web, sería interesante añadir un apartado que permitiese editar el tamaño de letra, resaltar los enlaces, cambiar el color... Es decir, un apartado que hiciese el sitio web inclusivo y accesible para llegar a un público mayor.

Otra posible mejora detectada es el uso de un código de barras en vez de la id que genera el sistema para identificar los artículos donados. Esto dotaría el sistema de mayor flexibilidad, transparencia y seguridad, ofreciendo también la posibilidad de escanearlos para mostrar información acerca de su trazabilidad.

Por último, cabe decir que las pruebas realizadas se han hecho con datos de ejemplo en un entorno controlado, pero puliendo ciertos detalles, ya comentados en los párrafos previos, el sistema posee una gran capacidad para ser implantado y utilizado por la población para hacer realidad la misión de este, para que ningún niño se quede sin juguete, para que nadie pueda sacar un beneficio propio de una buena acción.

Glosario

API interfaz de programación de aplicaciones, en inglés Application Programming Interface.

Backend es la parte del desarrollo web que se encarga de que toda la lógica de una página web funcione y se corresponde con la capa de acceso a datos.

Blockchain es una DLT cuyos bloques que conforman la base de datos están unidos entre sí mediante un mecanismo criptográfico llamado hash para formar una cadena. En castellano también se la conoce como cadena de bloques.

DLT es una base de datos no centralizada, trasparente y con un registro distribuido, en inglés Distributed Ledger Technology.

EDT Esquema de Desglose de Trabajo.

Framework entorno o marco de trabajo.

Frontend es la parte de una web que conecta e interactúa con los usuarios que la visitan y que se corresponde con la capa de presentación de la aplicación desarrollada.

Hyperledger Fabric es una plataforma de código abierto para la creación de redes *Blockchain* basadas en Hyperledger.

JSON enotación literal de objeto de JavaScript, en inglés JavaScript Object Notation.

Bibliografía

[1] Laurie Hughes, Yogesh K. Dwivedi, Santosh K Misra, Nriprenda Rana, Vishnupriya Raghavan & Viswanadh Akella. Blockchain Research, Practice and Policy: Applications, Benefits, Limitations, Emerging Research Themes and Research Agenda. *International Journal of Information Management*, 39:114-129, diciembre, 2019.

[2] Edvard Tijan, Saša Aksentijevic, Katarina Ivanic & Mladen Jardas. Blockchain Technology Implementation in Logistics. *Sustainability*, febrero, 2019.

[3] Distributed Ledger Technology (DLT). Consultado en `https://es.wikipedia.org/wiki/Distributed_Ledger_Technology_(DLT)`.

[4] Advait Deshpande, Katherine Stewart, Louise Lepetit, Salil Gunashekar. Distributed Ledger Technologies/Blockchain: Challenges, opportunities, and the prospects for standards. Mayo, 2019.

[5] Tecnología Blockchain: qué es y cómo funciona. Consultado en `https://estrategafinanciero.com/tecnologia-blockchain-funciona/`.

[6] Julija Golosova & Andrejs Romanovs. The Advantages and Disadvantages of the Blockchain Technology. *IEEE*, 2018.

[7] Cuántos tipos de Blockchain hay. Consultado en `https://academy.bit2me.com/cuantos-tipos-de-blockchain-hay/`.

[8] Conoce los diferentes tipos de Blockchains. Consultado en `https://www.blockchainservices.es/novedades/conoce-los-diferentes-tipos-de-blockchain/`.

[9] Contrato Inteligente. Consultado en `https://es.wikipedia.org/wiki/Contrato_inteligente`.

[10] Hyperledger: la Blockchain privada que todos tenemos que conocer. Consultado en `https://www.eleconomista.es/economia/noticias/8899454/01/18/Hyperledger-la-Blockchain-privada-que-todos-tenemos-que-conocer.html`.

[11] Hyperledger. Consultado en `https://aprendeblockchain.wordpress.com/hyperledger/`.

[12] ¿Qué es Ethereum? Consultado en `https://ethereum.org/es/what-is-ethereum/`.

[13] MultiChain te permite crear tu propia Blockchain en 90 segundos. Consultado en `https://www.criptonoticias.com/negocios/multichain-te-permite-crear-tu-propia-blockchain-en-90-segundos/`.

[14] ¿Qué es Corda Blockchain? ¿Cómo ayuda a las empresas? Consultado en `https://es.crypto-economy.com/corda-blockchain/`.

[15] Blockchain: qué es, cómo funciona y cómo se está usando en el mercado. Consultado en https://www.welivesecurity.com/la-es/2018/09/04/blockchainBlockchain-que-es-como-funciona-y-como-se-esta-usando-en-el-mercado/.

[16] Trazabilidad en Blockchain - Proyecto PorkChain. Consultado en https://criptoblog.tutellus.com/trazabilidad-en-blockchain-proyecto-porkchain/.

[17] Veritas, nuevo proyecto de Blockchain para la trazabilidad de los envases. Consultado en https://techpress.es/veritas-nuevo-proyecto-de-blockchain-para-la-trazabilidad-de-los-envases/.

[18] How Walmart Using Blockchain Technology? Consultado en https://blog.coincodecap.com/how-walmart-is-using-blockchain.

[19] La ONU sustituye a los bancos por la Blockchain para el Programa Mundial de Alimentos. Consultado en https://www.fin-tech.es/2018/02/blockchain-programa-mundial-alimentos.html.

[20] Salvation Army executive guilty of massive toy-for-profit fraud. Consultado en https://www.thestar.com/news/crime/2017/04/26/salvation-army-executive-guilty-of-massive-toy-for-profit-fraud.html.

[21] Goodwill Executives Arrested After Years Of Skimming Donated Goods Off Top. Consultado en https://www.theonion.com/goodwill-executives-arrested-after-years-of-skimming-do-1819578191.

[22] Detenido el presidente de la ONG Anesvad por apropiación indebida de fondos. Consultado en https://www.20minutos.es/noticia/210427/0/anesvad/detenido/bilbao/?autoref=true.

[23] Jorge Justo Giménez Duro. Sistema de trazabilidad basado en DLT (Distributed Ledger Technology). 2020.

[24] Using the Fabric test network. Consultado en https://hyperledger-fabric.readthedocs.io/en/latest/test_network.html.

[25] Deploying a smart contract to a channel. Consultado en https://hyperledger-fabric.readthedocs.io/en/release-2.2/deploy_chaincode.html.

[26] Donghee Shin, Yujong Hwang. The effects of security and traceability of Blockchain on digital affordance. Junio, 2019.