



**Escuela de Doctorado
y Estudios de Posgrado**
Universidad de La Laguna

Trabajo Fin de Máster

Aplicación de *Blockchain* en Logística

Blockchain application in Logistics

Daute Rodríguez Rodríguez

Máster Universitario en Ciberseguridad e Inteligencia de Datos

La Laguna, 17 de junio de 2020

Dña. **Pino Caballero Gil**, con NIF 45534310Z Catedrática de Universidad adscrita al Departamento de Ingeniería Informática y de Sistemas, como tutora

D. **Néstor García Moreno**, con NIF 79085553F contratado por proyecto de investigación adscrito al Departamento de Ingeniería Informática y de Sistemas, como cotutor

C E R T I F I C A (N)

Que la presente memoria titulada:

“Aplicación de Blockchain en Logística”

ha sido realizada bajo su dirección por D. **Daute Rodríguez Rodríguez**, con N.I.F. 79151574H.

Y para que así conste, en cumplimiento de la legislación vigente y a los efectos oportunos firman la presente en La Laguna a 17 de junio de 2020

Agradecimientos

A mi tutora Pino y cotutor Néstor por orientarme y ayudarme durante la realización del trabajo.

A mis compañeros por hacer de esta etapa de mi vida una experiencia tan enriquecedora.

A mis familiares y personas de mi entorno cercano por apoyarme y animarme incondicionalmente.

Licencia



© Esta obra está bajo una licencia de Creative Commons Reconocimiento-NoComercial-CompartirIgual 4.0 Internacional.

Resumen

La irrupción de la tecnología de cadena de bloques y la aparición de los contratos inteligentes han supuesto un gran avance tecnológico en diversos sectores de la sociedad. La posibilidad de poder llevar a cabo la verificación, el registro y la coordinación de transacciones de manera autónoma sin necesidad de una tercera parte de confianza supone una gran ventaja respecto a los sistemas de transacción tradicionales. Al aplicar correctamente esta tecnología se consigue aportar transparencia, inmutabilidad y descentralización a los procesos que componen la actividad del ámbito de aplicación, mejorándolos significativamente. Uno de los sectores que más puede beneficiarse de la correcta aplicación de la *Blockchain* es el de la Logística, en este Trabajo Fin de Máster se propone una aplicación web descentralizada para la plataforma *Ethereum* que permite el control y la gestión de cadenas de suministro.

Palabras clave: *Blockchain*, Logística, Cadena de suministro, *Ethereum*, Contrato inteligente

Abstract

Blockchain technology irruption and the emergence of Smart contracts have led to a major technological breakthrough in several sectors. Being able to carry out verification, registration and coordination of transactions autonomously without the need for a trusted third party is a great advantage over traditional transaction systems. By correctly applying this technology it is possible to provide transparency, immutability and decentralisation to the processes that make up the activity of the field of application, improving them significantly. Logistics is one of the sectors that can benefit most from the correct application of Blockchain. In this Master's Degree final project an Ethereum decentralized web application for controlling and managing supply chains is proposed.

Keywords: *Blockchain, Logistics, Supply chain, Ethereum, Smart contract*

Índice general

1. Introducción	1
1.1. Motivación	1
1.2. Objetivo	2
1.3. Especificación del problema	3
1.4. Solución propuesta	5
1.5. Fases del desarrollo	6
1.6. Estructura de la memoria	7
2. Antecedentes	8
2.1. Criptografía	8
2.1.1. Función <i>hash</i> criptográfica	8
2.1.2. Criptografía asimétrica	9
2.1.3. Árbol de Merkle	10
2.2. <i>Blockchain</i>	11
2.2.1. Bloques	11
2.2.2. Problema del consenso	13
2.2.3. Transacciones	14
2.2.4. Tipos de <i>blockchain</i>	16
2.2.5. Desafíos y limitaciones	17
2.3. <i>Ethereum</i> y los <i>smart contracts</i>	17
2.3.1. <i>Blockchain</i> como máquina de estados	18
2.3.2. <i>Ethereum Virtual Machine</i>	19
3. Desarrollo de la solución propuesta	20
3.1. Selección de tecnologías	20
3.1.1. <i>Typescript</i>	20
3.1.2. <i>React</i> y <i>Material-UI</i>	20
3.1.3. <i>Truffle</i> y <i>Ganache</i>	21
3.1.4. <i>Metamask</i>	22
3.2. Aplicación desarrollada	22
3.2.1. Particularidades	22
3.2.2. Contratos inteligentes	24
3.2.3. Vistas de la aplicación	27
4. Conclusiones y líneas futuras	35
4.1. Conclusiones	35
4.2. Líneas futuras	35
5. Summary and Conclusions	36
5.1. Conclusions	36
5.2. Future work	36

6. Presupuesto	37
Glosario	38
Acrónimos	40

Índice de figuras

1.1. Logo de <i>Bitcoin</i> , primera aplicación real de la tecnología <i>blockchain</i>	1
1.2. Ilustración de una cadena de suministro simplificada	3
1.3. Promedio del porcentaje de envíos realizados tal y como se han programado . .	4
1.4. Diagrama de transición de estados de un producto	6
2.1. Ilustración de un ejemplo de función <i>hash</i> criptográfica	9
2.2. Empleo de un par de claves para obtener confidencialidad en el envío de un mensaje	9
2.3. Empleo de un par de claves para lograr la autenticidad de un mensaje	10
2.4. Ilustración de ejemplo de un árbol de Merkle	10
2.5. Ilustración de una cadena de bloques simplificada	11
2.6. Ilustración de una transacción <i>blockchain</i> simplificada	15
2.7. Ilustración de ejemplo de <i>blockchains</i> privada y pública	16
2.8. Logos de <i>Ethereum</i> y <i>Solidity</i>	17
3.1. Logo de <i>Typescript</i>	20
3.2. Logos de <i>React</i> y <i>Material-UI</i>	21
3.3. Logos de <i>Truffle</i> y <i>Ganache</i>	21
3.4. Logo de <i>Metamask</i>	22
3.5. Logo de <i>SupplyBlocks</i>	22
3.6. <i>Struct EntityData</i> , representa los datos de un usuario	24
3.7. <i>Struct ProductsData</i> , representa los datos de un producto/envío	25
3.8. <i>Enums EntityType</i> y <i>ProductState</i> , representan el tipo de usuario o entidad y el estado de un producto respectivamente	25
3.9. Miembros del contrato inteligente <i>Product</i>	26
3.10. Miembros relevantes del contrato inteligente <i>Manager.png</i>	27
3.11. Parte superior de la vista de <i>Inicio</i> de <i>SupplyBlocks</i>	28
3.12. Parte inferior de la vista de <i>Inicio</i> de <i>SupplyBlocks</i>	29
3.13. Vista de <i>Registro</i> de <i>SupplyBlocks</i>	29
3.14. Menú <i>Compañías</i> desde el punto de vista del usuario administrador	30
3.15. Menú <i>Productos</i> desde el punto de vista de un usuario tipo <i>F</i>	31
3.16. Menú <i>Productos</i> desde el punto de vista de un usuario tipo <i>M</i>	32
3.17. Menú <i>Envíos</i> desde el punto de vista de un usuario tipo <i>M</i>	33
3.18. Menú <i>Envíos</i> . Línea de tiempo de un envío no finalizado	33
3.19. Menú <i>Envíos</i> . Línea de tiempo de un envío finalizado	34

Índice de cuadros

2.1. Estructura de un bloque	12
2.2. Estructura de la cabecera de un bloque	12
3.1. Métodos relevantes del contrato inteligente <i>Product</i>	27
3.2. Métodos relevantes del contrato inteligente <i>Manager</i>	27
6.1. Presupuesto estimado del proyecto	37

Capítulo 1

Introducción

1.1. Motivación

La irrupción de la tecnología *blockchain* ha provocado cambios significativos en un amplio número de sectores durante los últimos años. Su origen se remonta a 1991, año en el que Stuart Haber y W. Scott Stornetta publican el artículo *How to time-stamp a digital document*, introduciendo el concepto de **cadena de bloques**. A pesar de la corta edad de esta tecnología, se han conseguido importantes avances y resultados que han servido como prueba del gran potencial que presenta. Por esta razón, expertos como Ginni Rometty, anterior *CEO* de *IBM*, establecen que la *blockchain* supondrá para las transacciones lo que ha supuesto internet para la información. La capacidad de esta tecnología de garantizar la seguridad en redes distribuidas *Peer-to-peer* posibilita el poder llevar a cabo transacciones fiables sin la necesidad de una tercera parte de confianza.

Uno de los hitos más relevantes acaecidos durante la breve historia de la *blockchain* ha sido la aparición de los contratos inteligentes o *smart contracts*. Estos contratos se ejecutan de forma automática y autónoma como parte de una transacción. En ellos se recogen los términos del acuerdo al que han llegado las partes interesadas (escritos directamente en líneas de código). Estos términos se almacenan en la cadena de bloques, de forma que los contratos ejecutados son totalmente transparentes, rastreables e irreversibles. Una vez se cumplen las condiciones del acuerdo recogido en el *smart contract*, las acciones preestablecidas se llevan a cabo de manera automática. Mediante el uso de *smart contracts* se permite la ejecución de cualquier tipo de acuerdo entre partes sin necesidad de un organismo regulador.



Figura 1.1: Logo de *Bitcoin*, primera aplicación real de la tecnología *blockchain*

Desde la publicación del artículo *Bitcoin: A peer-to-peer electronic cash system* [16] (Figura 1.1), se han adoptado diferentes aproximaciones de uso de la tecnología de cadena de bloques. La principal y más popular aplicación ha sido la criptomoneda, sin embargo, sus deseables características (inmutabilidad, transparencia, ...) junto a la invención de los *smart contracts* han logrado que se aplique de manera satisfactoria en numerosos y diversos ámbitos. A continuación se presenta una lista de ejemplos de sectores en los que se han podido adoptar exitosamente soluciones basadas en la tecnología *blockchain*:

- Cuidado de la salud: *BurstIQ* [3].
- Compañías aseguradoras: *Etherisc* [6].
- Transporte público: *Banco Santander* y *Vottun* [24].
- Logística: *ShipChain* [19].
- Identidad digital: *ID2020* [9].
- Consumo responsable, ético y sostenible: *Provenance* [17].

La clave de tal amplia aplicabilidad reside en la naturaleza de la tecnología. La posibilidad de poder llevar a cabo la verificación, el registro y la coordinación de transacciones de manera autónoma sin necesidad de terceras partes supone una gran ventaja respecto a los sistemas de transacción tradicionales [11]. Estas ventajas en un sector como el de la logística, supondrían la eliminación de los intermediarios y los respectivos costes. Si a este factor se le añade el gran incremento de transparencia y trazabilidad del que se podría dotar a las cadenas de suministro al adoptar soluciones basadas en *smart contracts*, es posible entender de manera general cómo al aplicar *blockchain* en los procesos se impulsaría y mejoraría el sector en gran medida.

Pese a la gran cantidad de ventajas que ofrece, se trata de una tecnología con poca madurez para la que siguen surgiendo numerosos desafíos. El hecho de que la primera aplicación de la misma fueran las criptodivisas ha ayudado a la difusión de la tecnología que, en la actualidad, está en auge. Sin embargo, se ha de seguir desarrollando y adoptando progresivamente antes de que se pueda aplicar a escala en los diversos sectores para los que a día de hoy se han propuesto soluciones y en otros que aún no han sido contemplados. Por este motivo, la realización de trabajos de esta índole y la formación en materia de posibles aplicaciones basadas en el uso de cadena de bloques ocupan un importante papel en el desarrollo de esta prometedora e interesante tecnología.

1.2. Objetivo

El principal objetivo de este Trabajo Fin de Máster (TFM) consiste en llevar a cabo el **desarrollo de una aplicación web en la que se aplique la tecnología *blockchain* al ámbito de la logística**. En concreto se pretende orientar el funcionamiento de la aplicación hacia la gestión de la cadena de suministro y el control de la adquisición, envío y seguimiento de productos. En la próxima sección de la memoria se introducirá el concepto cadena de suministro y cómo aplicando la tecnología de cadena de bloques se pueden obtener numerosos beneficios.

A pesar de la existencia en el mercado de aplicaciones y herramientas que abordan este problema, se ha creído oportuno enfocar este TFM hacia la resolución del objetivo anteriormente expuesto con el propósito de adquirir los conocimientos teóricos y fundamentos básicos de una de las tecnologías más prometedoras actualmente. Este trabajo ha resultado ser una oportunidad excelente para llevar a la práctica de manera sencilla pero eficaz dichos conocimientos y fundamentos.

1.3. Especificación del problema

La logística empresarial es el concepto general que engloba los numerosos y diversos requisitos de logística de una empresa. La logística, en el ámbito de la administración comercial, queda definida como el conjunto de procesos de planificación, gestión, ejecución y control de mercancías e información que se producen en las diferentes partes que conforman la **cadena de suministro**. La capacidad logística de un país está directamente relacionada con su nivel de desarrollo general, entre el Índice de Desempeño Logístico (*LPI*) y el Producto Interior Bruto (*PIB*) existe una fuerte correlación positiva [18]. A su vez, el desempeño logístico está fuertemente relacionado con la fiabilidad de la cadena de suministro y la predictibilidad de las entregas de los envíos [2].

La cadena de suministro (Figura 1.2) se define como el conjunto de organizaciones, personas, actividades, información y recursos que participan en el suministro de un servicio o producto a un consumidor. Las actividades de la cadena de suministro implican la transformación de recursos naturales y materias primas en el producto final ofrecido a los clientes [10]. A partir de una definición tan amplia es posible deducir que se trata de un concepto complejo que engloba numerosos procesos.



Figura 1.2: Ilustración de una cadena de suministro simplificada

La fiabilidad de la cadena de suministro y, por consiguiente, la calidad del servicio o producto ofertado, es un aspecto de gran relevancia que ha de considerarse en el intento de mejorar el desempeño logístico de una organización. Tal y como puede apreciarse en la Figura 1.3, las compañías que cumplen en mayor medida con la planificación y programación de los envíos tienen, en promedio, un mayor Índice de Desempeño Logístico.

Como en cualquier proceso complejo en el que están involucrados múltiples agentes, en la cadena de suministro existen una serie de riesgos y amenazas que se han de atender con el fin de asegurar que los productos llegan con la calidad requerida y el tiempo adecuado al consumidor, o dicho de otra forma, con el propósito de asegurar la continuidad de la empresa en cuestión. Algunas de estas amenazas pueden ser los robos, las pérdidas injustificadas de inventario, la falsificación y el contrabando de productos.

Si a los ya mencionados riesgos se le añade el hecho de que los numerosos agentes que participan en la cadena de suministro tienen intereses propios y, por norma general, no suelen mantener entre ellos un nivel de cooperación adecuado, es posible concluir que se necesitan herramientas y mecanismos que consigan asegurar la integridad, el funcionamiento y la fiabilidad de la cadena de suministro aumentando su resiliencia ante las amenazas. Entre las posibles medidas que deberían adoptarse para el cumplimiento de dicho objetivo podrían contemplarse el aumento de la transparencia, seguridad, visibilidad y trazabilidad, y la estandarización de los procesos mediante los cuales se comunican y comparten información los diferentes agentes que forman parte de la cadena de suministro.

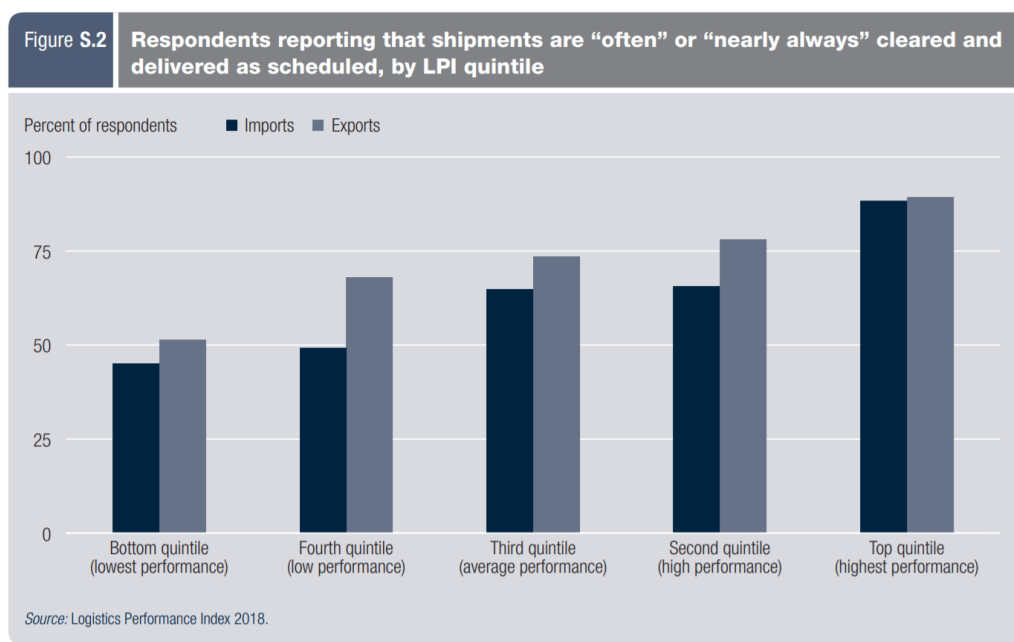


Figura 1.3: Promedio del porcentaje de envíos realizados tal y como se han programado. Valores agrupados por quintiles del Índice de Desempeño Logístico de las compañías. Extraído de *Connecting to Compete 2018* [2]

Afortunadamente, las actividades que conforman las cadenas de suministro “convencionales” parecen ir en consonancia con los tipos de procesos que pueden ser adaptados a la cadena de bloques, las necesidades y requisitos anteriormente expuestos pueden abordarse haciendo uso de esta tecnología. A continuación se presenta una lista de ventajas que, a priori, podrían obtenerse al aplicar la tecnología para la gestión y el control de la cadena de suministro:

- **Trazabilidad:** al registrar los distintos movimientos que se realizan para hacer llegar los productos al cliente final, es posible conocer el punto exacto de la cadena en el que se encuentran los mismos en un momento específico. Por otra parte, se conseguiría un aumento en la predictibilidad de los envíos y se dispondría de información con alta granularidad en lo relativo a los puntos de la cadena de suministro en los que se están produciendo retrasos. Esta información podría ser analizada para ayudar en la toma de decisiones para llevar a cabo medidas correctoras. A su vez, estas posibles medidas serían más precisas, eficaces y podrían atacar directamente a la raíz del problema.
- **Origen de los productos:** identificando únicamente cada producto, se conseguiría distinguir inequívocamente el origen de los mismos y qué camino han recorrido hasta llegar a las manos del cliente. Además, se evitarían problemas relacionados con la falsificación de productos y el contrabando, pues sólo aquellos que fueran originales podrían haber sido registrados en la cadena de bloques (en teoría).
- **Transparencia:** se conseguiría un aumento de la transparencia de cara al cliente. Éste podría conocer con total certeza numerosos aspectos de los productos o servicios que adquiere. También se podrían identificar con mayor facilidad los problemas de pérdidas injustificadas de inventario y robos en caso de que se produjeran.
- *Single source of truth:* si todos los participantes de la cadena de suministro utilizan en sus procesos la tecnología *blockchain* se crearía una única fuente de información (en cuanto a

la información que han de compartir). Los diversos agentes tendrían acceso a la información sin duplicidades (peligrosas para la integridad) y cooperar de manera más sencilla, obteniendo una estandarización indirecta de los procesos que llevan a cabo.

- Seguridad y confianza: el uso de la *blockchain* supondría un incremento general de la seguridad y la confianza entre los agentes de la cadena de suministro. Cualquier agente podría auditar y comprobar el estado de cualquier transacción, es decir, de sus actividades. Además, contaría con la certeza de que lo que se recoge en la cadena de bloques es cierto.
- Reducción de costes: al eliminar la necesidad de terceras partes de confianza se produce un decremento de los costes asociados a las actividades de los participantes de la cadena de suministro.

1.4. Solución propuesta

Una vez presentada la naturaleza del problema que se quiere abordar y cómo la tecnología *blockchain* podría ayudar en dicha labor, conviene especificar de manera más detallada los objetivos específicos del Trabajo Fin de Máster. Se pretende construir una aplicación web para la plataforma *Ethereum* (sección 2.3) que por medio de la ejecución de contratos inteligentes provea los mecanismos necesarios para crear una red de negocios privada en la que se cubran los procesos de adquisición y seguimiento de productos a la venta. A dicha red se unirán diversos tipos de usuarios en representación de una compañía. Los tipos de compañías o entidades contempladas son los siguientes:

- Fabricantes (tipo F): Pueden crear productos y exponerlos a los usuarios de la red de tipo M .
- Empresas de transporte (tipo T): Encargadas de realizar los envíos de los productos adquiridos por las compañías de tipo M . Estos envíos pueden tener como origen un almacén (usuario tipo A) o una fábrica (usuario tipo F), y como destino, un almacén (usuario tipo A) o el comprador (usuario tipo M).
- Empresas de almacenaje (tipo A): Guardan temporalmente los productos trasladados por las compañías de transporte.
- Minoristas y pequeños comercios (tipo M): Adquieren los productos que los usuarios tipo F han ofrecido a la red.

A los productos registrados en la red por los usuarios de tipo F se les asigna un identificador que indica su estado. Las transiciones entre los distintos estados de cada producto se registran con una marca de tiempo que representa el momento en el que se produjo el cambio de estado. Los posibles estados de un producto se enumeran a continuación:

- *Creado* (C): Estado inicial de todos los productos registrados. Simboliza la creación del producto por parte del fabricante.
- *Preparado* (P): Indica que un producto está preparado para ser enviado a un usuario tipo M .
- *Tránsito* (T): Representa que el producto lo posee un usuario tipo T , o dicho de otra forma, el producto está en tránsito para ser entregado al usuario tipo M correspondiente.

- *Almacenado (A)*: Indica que el producto se encuentra almacenado por un usuario tipo *A*.
- *Entregado (E)*: Pone en manifiesto que el producto ha sido entregado a un usuario tipo *M*, representa el estado final del producto.

En la Figura 1.4 se presenta un diagrama con los posibles estados de un producto y qué tipo de usuario es capaz de ocasionar una transición entre estados.

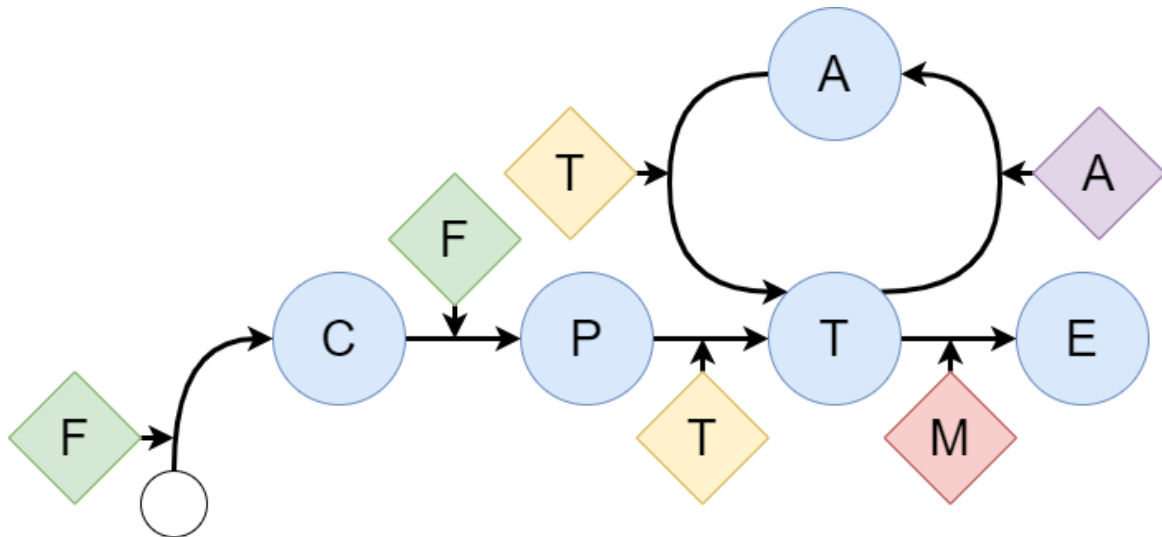


Figura 1.4: Diagrama de transición de estados de un producto. Los círculos azules representan un estado, es posible diferenciarlos por medio de la letra que se encuentra en su interior. Las flechas con las que se unen estados representan transiciones. Los rombos indican tipos de usuario, es posible diferenciarlos por el color y la letra interior. Las flechas con origen en un rombo y destino en una transición simbolizan la posibilidad de dicho tipo de usuario (rombo) de desencadenar dicha transición.

En la sección 3.2 se detallarán y expondrán los detalles de implementación de la solución propuesta, pero cabe adelantar algunos aspectos de gran relevancia:

- Dado que existen diversos tipos de usuario y cada uno puede realizar distintas acciones, la aplicación ha de incluir algún mecanismo de control de acceso basado en roles (*RBAC*).
- Para crear una red de negocios privada resultará necesario implementar un sistema al que se puedan unir nuevos usuarios tras haber sido aceptados, o dicho de otra forma, se ha de permitir el acceso únicamente a usuarios conocidos. Tal y cómo se expondrá en la sección 2.2.4 las *blockchain* permissionadas son ideales para este tipo de necesidades.

1.5. Fases del desarrollo

A continuación se presentan las fases que han conformado el desarrollo del Trabajo Fin de Máster.

1. Estudio e investigación iniciales: Adquisición de los conocimientos básicos y fundamentos sobre *blockchain* y desarrollo de aplicaciones web que incorporan esta tecnología. Estudio del estado del arte en cuanto a plataformas que incorporan esta tecnología y búsqueda de soluciones actuales implantadas con éxito en diversos sectores.

2. Definición de requisitos: Especificación de características y funcionalidades que ha de incorporar la aplicación a desarrollar de acuerdo con lo establecido como objetivo del TFM.
3. Selección de tecnologías: Elección de las tecnologías a utilizar para el desarrollo de la aplicación teniendo en cuenta los requisitos establecidos en la fase previa.
4. Desarrollo: Fase de desarrollo de la aplicación web. Ésta ha de incorporar los requisitos recogidos en la segunda fase y se ha de desarrollar incluyendo las tecnologías seleccionadas en la tercera fase.
5. Redacción de la memoria: Redacción del trabajo realizado y los resultados obtenidos durante el desarrollo del TFM.

El periodo de tiempo dedicado a la elaboración del Trabajo Fin de Máster abarca el intervalo comprendido entre el 16 de Marzo de 2020 y el 15 de Junio de 2020.

1.6. Estructura de la memoria

El resto de los contenidos de esta memoria se presentan siguiendo la siguiente estructura. En el Capítulo 2 se recoge la definición y explicación del funcionamiento de la tecnología *blockchain*, se profundiza en el concepto de los *smart contracts* y se expone la plataforma para la que se ha desarrollado la aplicación (*Ethereum*). Las tecnologías y herramientas seleccionadas para la creación de la aplicación se presentan en el Capítulo 3. Junto a las tecnologías, se exponen los *smart contracts* implementados para abordar el objetivo del proyecto y el funcionamiento de la aplicación web construida. El Capítulo 6 muestra un presupuesto estimado para cada una de las tareas que han conformado el desarrollo del proyecto. Por último, en los capítulos 4 y 5 se aglutinan las conclusiones alcanzadas tras el desarrollo del Trabajo Fin de Máster y unas posibles líneas futuras en los idiomas español e inglés respectivamente.

Capítulo 2

Antecedentes

A lo largo de este capítulo se presentan de manera detallada las definiciones y el funcionamiento de los conceptos fundamentales en torno a los que gira el Trabajo Fin de Máster. En primer lugar se expondrán algunas herramientas criptográficas en las que se basa la tecnología *blockchain*. Seguidamente, se explicará el funcionamiento de esta tecnología y de qué manera se consigue que cuente con las deseables características que ofrece. Además, se ahondará en el concepto de *smart contract* y en la importancia de los mismos en cuanto al gran crecimiento de las posibilidades de aplicación de la cadena de bloques. Por último, se introducirá *Ethereum*, plataforma para la que se ha desarrollado la aplicación web objetivo del proyecto.

2.1. Criptografía

Antes de explicar el funcionamiento de la tecnología *blockchain* resulta necesario exponer una serie de conceptos criptográficos en los que se basa.

2.1.1. Función *hash* criptográfica

Una función *hash* criptográfica es una función trampa que recibe como entrada un mensaje y retorna un resumen o *hash* de tamaño fijo. Además, las funciones *hash* criptográficas han de cumplir con una serie de propiedades:

- Debe ser capaz de calcular el resumen de manera eficaz independientemente del mensaje de entrada.
- Dado un resumen, ha de ser extremadamente complejo computacionalmente generar o calcular un mensaje de entrada que produzca dicho resumen.
- La probabilidad de que dos textos de entrada ligeramente distintos produzcan un mismo resumen ha de ser extremadamente baja.
- Ha de ser una función determinista, esto es, para el mismo mensaje ha de devolver siempre el mismo resumen.

La Figura 2.1 presenta una ilustración de ejemplo del funcionamiento de una función *hash* criptográfica. A partir de dos textos de entrada cualesquiera, se generan dos resúmenes de mismo tamaño.

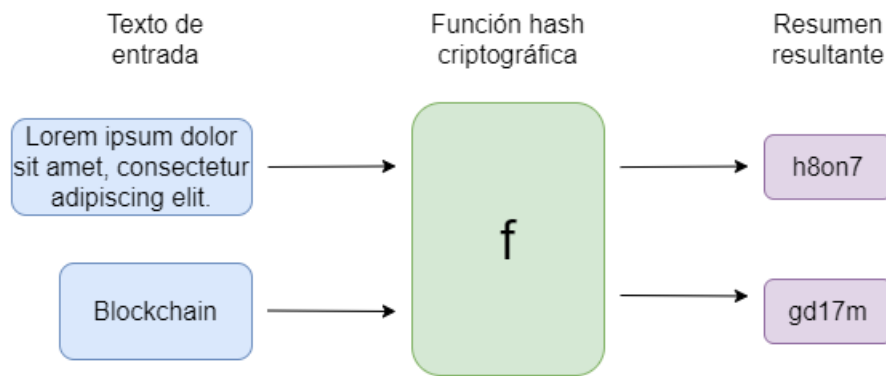


Figura 2.1: Ilustración de un ejemplo de función *hash* criptográfica

2.1.2. Criptografía asimétrica

La criptografía asimétrica o criptografía de clave pública es un sistema criptográfico en el que se utilizan un par de claves para el envío de mensajes. Cada persona o entidad dispone de una clave pública y una clave privada. La clave pública puede ser compartida con las demás personas o entidades del sistema sin comprometer la seguridad del mismo, mientras que la clave privada debe ser almacenada cuidadosamente por su propietario. La generación de las claves se realiza de manera que ambas están estrechamente relacionadas, gracias a esto resulta posible combinar operaciones de cifrado de maneras específica para conseguir distintos resultados:

- Suponiendo que Alice (*A*) quiere enviar un mensaje a Bob (*B*). *A* utilizará la clave pública de *B* para cifrar el mensaje. El criptograma generado solo podrá descifrarse con la clave privada de *B*. De esta manera se consigue la **confidencialidad** de la comunicación, ideal cuando se ha de establecer por medio de canales en los que no es posible garantizar la seguridad (Figura 2.2).
- En esta ocasión Alice quiere enviar un mensaje firmado digitalmente. Para ello, *A* utiliza su clave privada para cifrar el mensaje, de manera que cualquier persona con acceso a su clave pública podrá verificar que el remitente del mensaje es quien dice ser, además de asegurarse de que el mensaje no ha sido alterado por un agente externo y representa el mensaje original que *A* deseaba enviar. De esta forma se consigue garantizar la **autenticidad e integridad** del mensaje (Figura 2.3).

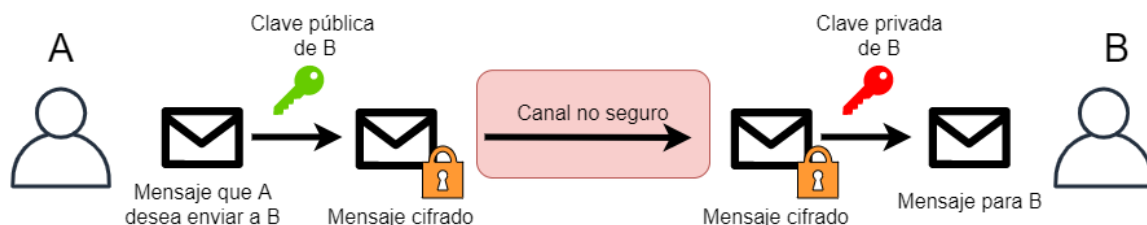


Figura 2.2: Empleo de un par de claves para obtener confidencialidad en el envío de un mensaje

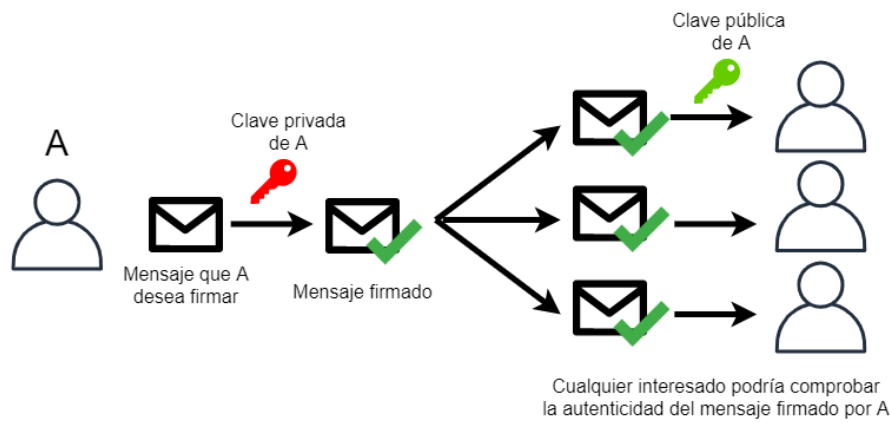


Figura 2.3: Empleo de un par de claves para lograr la autenticidad de un mensaje

2.1.3. Árbol de Merkle

Un árbol de Merkle es una estructura de datos en forma de árbol en la que cada nodo hoja se etiqueta con el valor de un *hash* criptográfico de un bloque de datos y cada nodo interno se etiqueta con el *hash* criptográfico de la concatenación de las etiquetas de sus nodos hijo. Esta forma de almacenar los datos permite asociar un gran número de éstos a un único *hash*, el correspondiente al nodo raíz del árbol. Gracias a esto es posible verificar de forma eficiente y segura grandes cantidades de datos. La Figura 2.4 presenta un ejemplo de árbol de Merkle.

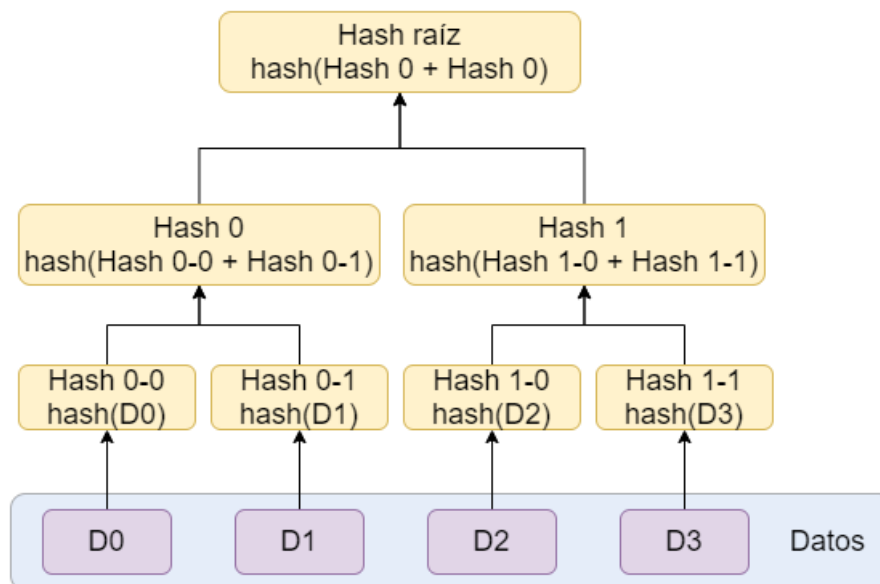


Figura 2.4: Ilustración de ejemplo de un árbol de Merkle

Atendiendo a la Figura 2.4, para llevar a cabo la verificación de los cuatro bloques de datos tan sólo haría falta comprobar el *hash* raíz. Si el ejemplo se extrapola a un árbol mucho más complejo con un gran número de nodos hoja es posible comprender la utilidad de esta estructura de datos.

2.2. Blockchain

Blockchain es una **tecnología de registro distribuido** (*Distributed Ledger Technology*) que permite el registro de transacciones entre partes de manera segura y permanente [13], puede entenderse como una base de datos descentralizada, consensuada y distribuida. Como ya se ha expuesto anteriormente, su origen se remonta a 1991, año en el que por primera vez se propone la idea de encadenar bloques de información asegurados criptográficamente. Sin embargo, no fue hasta el año 2008 cuando comenzó a ganar popularidad.

La primera aplicación real de esta tecnología fue el sistema de pago de criptomonedas *Bitcoin*. La publicación del artículo *Bitcoin: A peer-to-peer electronic cash system* [16] en 2008 asentó la primera propuesta de un sistema de pago electrónico basado totalmente en criptografía y en el que no se produce el problema del doble gasto. Para abordar dicho problema, los sistemas tradicionales implementan un modelo basado en confianza en el que una tercera parte asegura la fiabilidad de las transacciones. En esta nueva propuesta, cualquier par de interesados tiene la capacidad de llevar a cabo transacciones de manera directa entre ellos, sin necesidad de una tercera parte de confianza.

Además de esta posibilidad, la tecnología ofrece una serie de características sumamente deseables: descentralización, inmutabilidad, seguridad y transparencia. En las siguientes subsecciones se irán exponiendo los fundamentos básicos de la cadena de bloques y cómo es capaz de ofrecer estas ventajas. Cabe destacar que existen variantes a la *blockchain* inicialmente propuesta para el sistema *Bitcoin*. Cada implementación puede hacer uso de distintas alternativas en cuanto a los distintos componentes, como por ejemplo, el algoritmo de consenso o la estructura de datos en la que se almacenan las transacciones. A pesar de ello, en esta memoria se han intentado plasmar las ideas principales en las que se basa la tecnología. Por este motivo, se ha utilizado como referencia la primera implementación real de la cadena de bloques, es decir, el *Bitcoin*.

2.2.1. Bloques

Un *bloque* es una estructura de datos en la que se almacenan las transacciones que se desean almacenar en la cadena de bloques. Concretamente, cada bloque se compone de dos partes, una cabecera en la que se almacenan metadatos y una lista de transacciones. Uno de los campos de la cabecera es el *hash* criptográfico (sección 2.1.1) del bloque anterior. Gracias a esto, cada bloque se “encadena” con el bloque previo, formando una lista enlazada hacia atrás de bloques de información (transacciones) (Figura 2.5), de ahí el nombre de cadena de bloques. La estructura de un bloque queda recogida en el Cuadro 2.1, mientras que los distintos campos de la cabecera se presentan en el Cuadro 2.2.

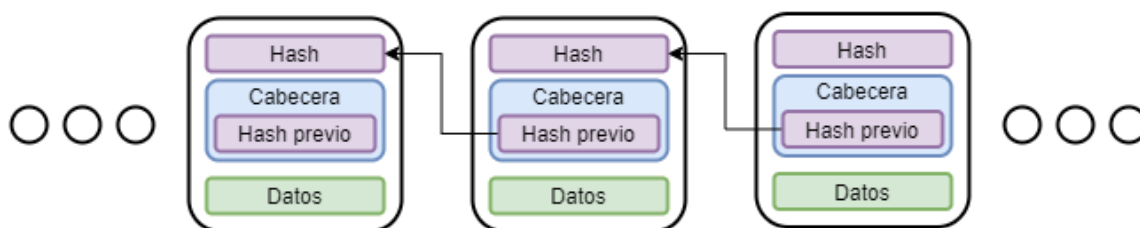


Figura 2.5: Ilustración de una cadena de bloques simplificada

El *hash* de cada bloque se calcula sobre la cabecera del mismo. A partir de lo comentado hasta el momento, de esta afirmación pueden extraerse dos conclusiones sumamente importantes:

1. En la cabecera de un bloque se almacena el *hash* del bloque previo. Si el bloque $n - 1$ sufre un cambio, su *hash* variará y no coincidirá con el *hash* almacenado en la cabecera del bloque n , esto implicará la actualización del bloque n para que los *hashes* coincidan. A su vez, la cabecera del bloque n variará y el nuevo *hash* resultante no coincidirá con el *hash* almacenado en la cabecera del bloque $n + 1$, y así sucesivamente. Gracias a este efecto en cascada, el cambio en un bloque supondría volver a recalcular los subsiguientes bloques. Por el momento no se ha llegado a explicar, pero la creación de un bloque requiere capacidad de cómputo. Este hecho junto con el anterior deriva en que modificar un bloque de la cadena sea impracticable, pues requeriría un inmenso esfuerzo computacional.
2. De igual forma, la cabecera de cada bloque también dispone de un campo denominado raíz Merkle. El valor de este campo se establece al valor del nodo raíz del árbol de Merkle formado a partir de las transacciones almacenadas en el bloque. Recordando lo expuesto en la sección 2.1.3, este valor representa un “resumen eficiente” del total de nodos hoja del árbol (los datos), de manera que si alguno sufre un cambio, el valor de la raíz también varía. Incluyendo este valor en la cabecera y teniendo en cuenta el punto anterior, se deduce directamente que de esta forma se consigue la inmutabilidad e integridad de los datos almacenados. Dado que la única operación que se acepta sobre la cadena es la de añadir bloques, las transacciones realizadas son irreversibles.

Tamaño	Campo	Descripción
4 bytes	Tamaño	Tamaño del bloque, en bytes
80 bytes	Cabecera	Cabecera del bloque, contiene metadatos
1-9 bytes	Contador	Cantidad de transacciones almacenadas en el bloque
Variable	Transacciones	Transacciones almacenadas en el bloque

Cuadro 2.1: Estructura de un bloque, extraída de *Mastering Bitcoin: unlocking digital cryptocurrencies* [1]

Tamaño	Campo	Descripción
4 bytes	Versión	Número de versión
32 bytes	<i>Hash</i> previo	<i>Hash</i> del bloque anterior en la cadena
32 bytes	Raíz Merkle	Nodo raíz del árbol de Merkle de las transacciones del bloque
4 bytes	Timestamp	Marca de tiempo del instante en el que el bloque se creó
4 bytes	Dificultad	Dificultad del algoritmo de consenso (Proof-of-work)
4 bytes	<i>Nonce</i>	Contador necesario para el algoritmo de consenso

Cuadro 2.2: Estructura de la cabecera de un bloque, extraída de *Mastering Bitcoin: unlocking digital cryptocurrencies* [1]

Como se ha podido ir deduciendo a lo largo de la lectura de esta sección, el primer bloque creado en cada *blockchain* tiene una particularidad, no contiene en su cabecera el *hash* del bloque previo. Esta distinción es suficiente como para que a este bloque se le asigne un nombre específico: *genesis block*. Otro aspecto que ha de comentarse es que cada bloque de la cadena puede identificarse de dos maneras, a partir de su *hash* y a partir de su *height* (posición que ocupa en la cadena).

Por último, cabe destacar que algunos elementos de la tabla 2.2 no se han explicado por el momento. En el siguiente apartado de la memoria se abordarán los conceptos algoritmo de consenso, dificultad del algoritmo y *nonce*.

2.2.2. Problema del consenso

Tal y como se ha expuesto con anterioridad, la tecnología de bloques se asenta sobre redes distribuidas *peer-to-peer*. En computación distribuida surge un concepto denominado problema del consenso. Normalmente, resulta necesario que los numerosos procesos que se ejecutan de forma distribuida entre los nodos de la red lleguen a un acuerdo sobre algo en específico. En el caso de la *blockchain*, los nodos no dependen de una autoridad central que tome las decisiones por lo que han de ponerse de acuerdo por su cuenta.

La capacidad de alcanzar el consenso entre los nodos de la red sin necesidad de un organismo regulador central que orqueste la toma de decisiones es una de las características más importantes de la tecnología. Este es el motivo por el que se trata de una tecnología totalmente descentralizada. A continuación, se expondrá cómo se consigue abordar este problema y para qué decisiones se ha de alcanzar el consenso entre los nodos de la red.

Para la resolución del problema del consenso se aplican los algoritmos de consenso. Este tipo de mecanismos aseguran que los usuarios que participan en la *blockchain* (nodos de la red) alcancen un acuerdo en cuanto al estado de la cadena y la validez de los nuevos bloques que se vayan creando. Cada nodo de la red mantiene una copia local de la cadena de bloques, por lo que a medida que aparecen nuevos bloques las copias han de ir actualizándose. Antes de extender la copia local con un bloque, cada nodo ha de estar seguro de que se trata de un bloque válido.

En concreto, el algoritmo de consenso utilizado para la primera aplicación real de la *blockchain* fue el denominado *Proof-of-work* (*PoW*). En este algoritmo, los usuarios de la red *blockchain* participan en el proceso de creación de bloques aportando potencia computacional. Para la creación de un nuevo bloque se ha de buscar un valor para el campo *nonce* de su cabecera (tabla 2.2) que consiga que el resumen resultante al aplicar una función *hash* sobre la misma comience por un número determinado de ceros, a este proceso se le da el nombre de “minado”. Dicho esfuerzo representa una “prueba de trabajo” por parte del nodo que acabó encontrando un valor del *nonce* válido. Por convención, al nodo ganador se le otorga una recompensa, incentivando el esfuerzo de apoyar la red. Una vez llevado a cabo el esfuerzo computacional requerido para la creación de un nuevo bloque, los contenidos del bloque no pueden alterarse sin antes volver a encontrar un valor adecuado para el campo *nonce*. A medida que se encadenan bloques, el esfuerzo requerido para cambiar un bloque aumentaría proporcionalmente al esfuerzo necesario para volver a minar ese bloque y todos los siguientes.

Dado que todos estos procesos tienen lugar en un entorno distribuido, puede darse el caso en el que dos nodos honestos minan un bloque de manera simultánea. Ambos bloques se transmitirían por la red y serían aceptados por distintos nodos. Se daría la indeseable situación en la que existen en la red diferentes versiones válidas de la cadena de bloques. Por suerte, el algoritmo *Proof-of-work* también propone una solución para este problema. La única versión válida de la cadena de bloques es aquella de mayor longitud, o dicho de otra forma, aquella en la que se ha invertido más esfuerzo computacional (*pruebas de trabajo*). Si la mayoría del esfuerzo computacional está controlado por usuarios honestos, la cadena de bloques “honesta” crecerá a mayor velocidad que las posibles cadenas fraudulentas.

Para la validación de un nuevo bloque recién creado, cada nodo de la red ha de realizar dos comprobaciones:

1. El valor del campo *Hash* previo de la cabecera del nuevo bloque es igual al *hash* del último bloque de su cadena. En caso contrario, o el nodo tiene una copia desactualizada de la cadena o el bloque nuevo no es válido.
2. Aplicando la función *hash* sobre la cabecera del nuevo bloque se obtiene un *hash* que empieza con el número requerido de ceros.

El esfuerzo computacional medio requerido para encontrar el valor del campo *nonce* aumenta exponencialmente con la cantidad de ceros necesarios al comienzo del *hash* resultante. Por este motivo a esta cantidad se le denomina dificultad del algoritmo o dificultad de minado, y su valor varía en el tiempo con el propósito de ajustar la velocidad a la que se pueden crear nuevos bloques. El hecho de que resulte tan complejo crear nuevos bloques junto a la posibilidad de ganar una recompensa si se participa en la red de forma legal, sirve como medida disuasoria para usuarios malintencionados y mejora la seguridad general del proceso.

A pesar de que en esta sección solo se ha explicado un algoritmo de consenso existen numerosas alternativas, a continuación se describen brevemente las dos más comunes:

- *Proof of Stake (PoS)*: Resuelve el problema de escalabilidad que sufren las redes *blockchain* que implementan *PoW*. Con este algoritmo, la probabilidad de seleccionar un nodo como creador de un nuevo bloque es directamente proporcional a la riqueza que dicho nodo haya acumulado.
- *Proof of Authority (PoA)*: este algoritmo se presentó como una alternativa eficiente especialmente dirigida a redes privadas. Se basa en el uso de las identidades reales de los usuarios de cada nodo y sólo permite que un conjunto de estos actúen como validadores dentro de la red. Cada validador pone su identidad real y reputación como garantía de transparencia. Gracias a esto presenta una mejor escalabilidad que los algoritmos *PoW* y *PoS*.

2.2.3. Transacciones

Hasta el momento se han definido y explicado los conceptos de bloque y algoritmo de consenso, resta exponer cómo se llevan a cabo las transacciones entre los usuarios de la red. Cabe destacar que en las operaciones de creación y verificación de las transacciones resultan necesarias las claves asociadas a cada usuario (criptografía asimétrica, sección 2.1.2). En la Figura 2.6 puede verse una ilustración simplificada de una transacción de ejemplo entre los usuarios *A* y *B*. Además, a continuación se presenta una lista con los pasos que componen el proceso:

1. El usuario *A* (emisor) desea llevar a cabo una transacción con el usuario *B* (receptor).
2. Los datos de la transacción junto con la clave pública del receptor y la clave privada del emisor conforman la entrada a una función de firma. La salida de dicha función es la propia firma de la transacción.

3. La firma de la transacción, junto con la clave pública del emisor y los datos de la transacción son transmitidos a través de la red. Todos los nodos encargados de crear nuevos bloques (mineros) y el receptor llevan a cabo la comprobación de la validez de la nueva transacción, comprueban que el emisor es quien dice ser y que cumple los requisitos necesarios para poder asumir la ejecución de la transacción (así se evita el problema del doble gasto). Para ello, utilizan los 3 elementos recibidos, es decir, la clave pública del emisor, la firma de la transacción y los propios datos de la transacción. Por medio de la comprobación de la firma son capaces de determinar si la transacción es válida.
4. En caso de que la transacción sea válida, los mineros la añaden al bloque en construcción. Cuando se termine de construir el bloque este es transmitido a la red.
5. Los distintos nodos de la red aceptan el bloque sí y solo sí todas las transacciones que contienen son válidas y pueden ejecutarse.
6. Los nodos que han comprobado la validez del bloque lo añaden a su copia local de la cadena y continúan trabajando en construir el siguiente, utilizando como *hash* previo el *hash* del nuevo bloque.
7. En este instante la transacción ya se ha ejecutado.

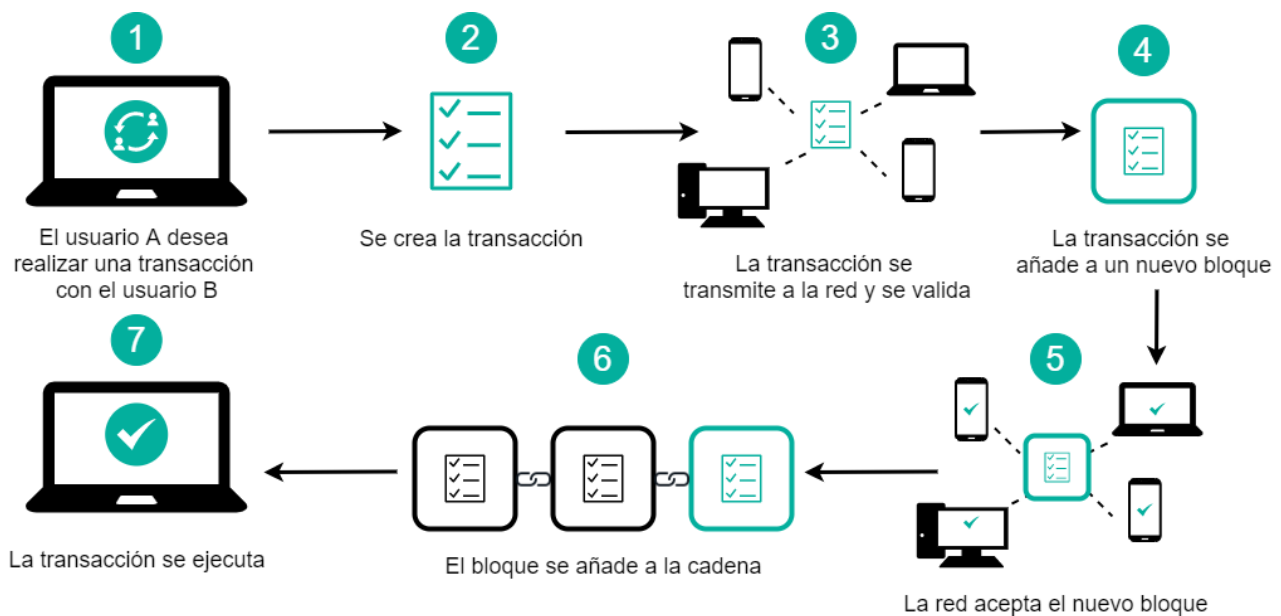


Figura 2.6: Ilustración de una transacción *blockchain* simplificada

Mediante la utilización del par de claves de cada usuario es posible que los nodos comprueben que las transacciones se han iniciado por la persona correcta, de manera legal. El sistema de firma basado en criptografía asimétrica permite garantizar la seguridad de las transacciones. Esto junto al hecho de que se requiere al menos el 51 % de la capacidad de cómputo de la red para poder burlar la confianza y credibilidad que se obtiene con el algoritmo de consenso, hace que la seguridad general de la tecnología sea muy alta.

2.2.4. Tipos de *blockchain*

En la actualidad se suelen definir 3 tipos diferentes de cadenas de bloques. En la siguiente lista se presenta cada uno de estos tipos así como sus particularidades:

- **Públicas:** las cadenas de bloques públicas son las más comunes y las más ampliamente utilizadas. Son de código abierto y permiten que cualquier persona se una a la red (equipos conectados a internet), ya sea como usuario básico o como minero, es decir, cualquier persona es capaz de realizar transacciones o de ofrecer la potencia de cómputo de su nodo para participar en el minado de nuevos bloques. Típicamente incluyen algún tipo de *token* para recompensar a los participantes de la red. Otro aspecto a tener en cuenta es que todas las transacciones realizadas se transmiten públicamente entre los nodos, de manera que cualquier usuario es capaz de obtener información acerca del histórico de transacciones realizadas (transparencia). Tanto *Bitcoin* como *Ethereum* son ejemplos de cadenas de bloques públicas.
- **Privadas o permissionadas:** la aplicación de este tipo de *blockchains* es más común en ámbitos empresariales puesto que únicamente se permite la unión a la red a organizaciones conocidas, de manera que se forma una especie de red de negocios privada sólo para miembros. Esto implica que la información almacenada en la cadena de bloques es presuntamente confidencial y sólo ciertas personas han de poder acceder a ella. Este tipo de cadenas, a diferencia de las públicas, no están construidas en torno al principio del anonimato, sino que se apoyan en la identidad de los miembros para confirmar los privilegios de acceso a la información. De esta manera, los participantes en la red saben exactamente con quién están tratando. La verificación de las transacciones sólo puede llevarse a cabo por usuarios conocidos, por ello, en este tipo de redes los algoritmos de consenso como el *Proof-of-authority* encajan a la perfección. El mejor ejemplo de *blockchain* privada es *Hyperledger*, propiedad de la compañía *IBM*.
- **Híbridas:** las cadenas de bloques híbridas combinan los beneficios de privacidad de los que disponen las cadenas permissionadas con los beneficios de seguridad y transparencia de las cadenas públicas. Esta posibilidad da a las empresas una gran flexibilidad en cuanto a la elección de qué datos quieren hacer públicos y transparentes y qué datos quieren mantener privados.

En la Figura 2.7 es posible apreciar ilustraciones de ejemplo de una *blockchain* privada y otra pública. La del lado izquierdo es la cadena de bloques pública, cualquier usuario puede formar parte de la red. La ilustración de la derecha representa una *blockchain* privada, solo usuarios específicos pueden acceder a la red.

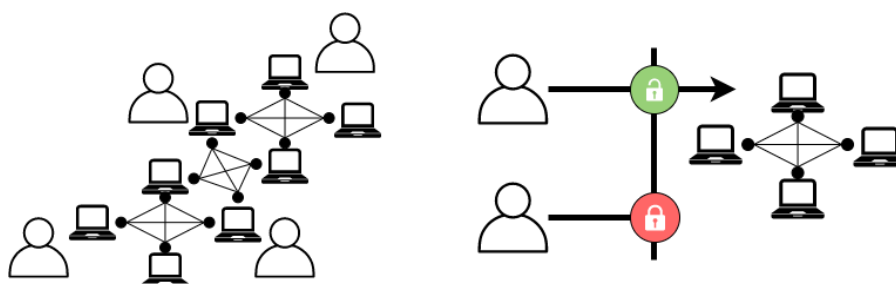


Figura 2.7: Ilustración de ejemplo de *blockchains* privada y pública

2.2.5. Desafíos y limitaciones

Tal y cómo se ha comentado en el Capítulo 1 de la memoria, la tecnología de cadena de bloques tiene muy poca madurez. Por ello, existen desafíos y limitaciones que han de abordarse con el propósito de incrementar las posibilidades de aplicación de la misma. A continuación, se enumeran algunos de estos desafíos y limitaciones:

- Escalabilidad y eficiencia: la tecnología no es escalable y presenta problemas de eficiencia. Las cadenas que utilizan el algoritmo de consenso *Proof-of-work* son especialmente ineficientes por el gran esfuerzo computacional que se desperdicia. El rendimiento de la red (velocidad a la que se realizan las transacciones) está inversamente relacionado con la cantidad de nodos, de manera que a más nodos, menor rendimiento. A pesar de esto, se han presentado numerosas propuestas para solucionar estos problemas, desde nuevos algoritmos de consenso a los diferentes tipos de *blockchain*.
- Almacenamiento: a medida que transcurre el tiempo el tamaño de la cadena de bloques seguirá aumentando. La velocidad con la que aumentan las cadenas de bloques más populares es mayor a la velocidad a la que se consigue aumentar la capacidad de los discos duros. Además existe el peligro de que la cadena llegue a ocupar tanto espacio que los nodos convencionales (ordenadores personales) no sean capaces de almacenar la copia de la cadena, esto supondría la pérdida de un gran número de nodos.

2.3. *Ethereum y los smart contracts*

Ethereum es una plataforma que ofrece una *blockchain* en la que se incorpora *Solidity*, un lenguaje de programación *Turing completo* diseñado para la creación de contratos inteligentes. El uso de este tipo de contratos permite a los desarrolladores implementar lo que se conoce como aplicaciones descentralizadas o *dapps*, en las que es posible definir reglas de posesión y formatos de transacción propios totalmente personalizados para el ámbito de aplicación específico.

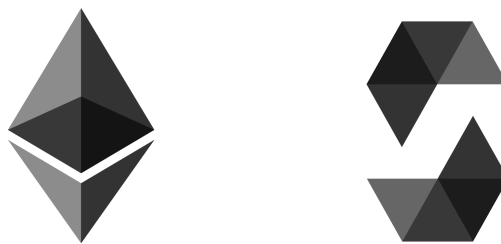


Figura 2.8: Logos de *Ethereum* y *Solidity*

El origen de los *smart contracts* se remonta a 1997, año en el que Nick Szabo publica el artículo *Formalizing and Securing Relationships on Public Networks* [23] en el que se presenta el concepto de *smart contract* como protocolo de transacción que ejecuta los términos de un contrato. Vitalik Buterin y Gavin Wood crean *Ethereum* en 2014 [4], aunando los conceptos de *smart contract* y *blockchain* en una única plataforma. Este momento se denominó el nacimiento de la *blockchain* 2.0 a raíz del gran incremento de posibilidades de aplicación que supuso la combinación de ambas ideas.

Extendiendo las ventajas que aporta la tecnología de cadena de bloques con la versatilidad y el amplio abanico de posibilidades del que disfrutaban los contratos inteligentes se permitió que cualquier par de interesados pudiera firmar cualquier tipo de acuerdo sin necesidad de intermediarios, quedando registrados de manera inmutable los términos de dicho acuerdo y pasando a ser irreversible y transparente.

Uno de los aspectos más relevantes de *Ethereum* reside en la facilidad con la que es posible crear aplicaciones descentralizadas que se aprovechen de todas las ventajas que ofrece la cadena de bloques. Al crear un *smart contract*, el desarrollador define su comportamiento y los términos del acuerdo directamente en líneas de código. Una vez que se incluye el contrato en la *blockchain* no puede ser modificado, por este motivo se puede asegurar que la ejecución del contrato se llevará a cabo tal y como fue programada. Los contratos inteligentes pueden ser vistos como programas independientes almacenados en la *blockchain* que llevarán a cabo transacciones, operaciones u acciones de manera autónoma si ciertas condiciones se cumplen. El desarrollador es el que especifica dichas acciones y condiciones dependiendo del ámbito y propósito de la aplicación que esté implementando.

Las oportunidades empresariales que surgieron a partir de tal abstracción de la tecnología *blockchain* contribuyeron enormemente a su difusión, uso y aceptación. Mediante la implantación exitosa de soluciones eficaces basadas en *blockchain* y los *smart contracts* se han conseguido abordar numerosos problemas en diversos ámbitos de la sociedad, tal y como se ejemplificó en la sección 1.1. Es por esto por lo que la aparición de los *smart contracts* en 2014 se considera un hito tan importante en la breve historia de la tecnología.

Antes de cerrar este capítulo de la memoria conviene abordar de manera simplificada algunos de los componentes y particularidades más importantes de *Ethereum*.

2.3.1. *Blockchain* como máquina de estados

La principal característica que diferencia las *blockchain* tradicionales (como *Bitcoin*) con la implementación de *Ethereum* es la asunción de que la cadena de bloques es equivalente a una máquina de estados basada en transacciones [4]. Un estado queda definido por objetos denominados cuentas (*accounts*). A su vez, cada cuenta está asociada a una dirección de 20 *bytes*. Las transiciones entre estados se representan por medio de la transferencia de valor e información entre cuentas, es decir, por medio de la ejecución de transacciones. La información necesaria para llevar a cabo la validación de los términos recogidos en cada transacción se encuentra disponible en el estado previo.

Existen dos tipos de cuenta en *Ethereum*, las asociadas a usuarios externos (controladas mediante claves privadas) y las asociadas a contratos inteligentes (controladas por el código de contrato). Los bloques en *Ethereum* almacenan información sobre las transacciones realizadas y el estado más reciente de cada cuenta de contrato y usuario. A continuación se listan los distintos campos que posee una cuenta de contrato, cabe destacar que las cuentas de usuarios quedan conformadas por los dos primeros:

- *Nonce*, contador que asegura la unicidad de las transacciones.
- Balance de *ether*. *Ether* es la criptomoneda interna de *Ethereum*, necesaria para el pago de las cuotas de las transacciones realizadas.
- Código de contrato de la cuenta

- Almacenamiento de la cuenta

A su vez, *Ethereum* establece una diferenciación entre las operaciones de lectura y las operaciones de escritura sobre la cadena de bloques. Esta distinción es sumamente importante a la hora de invocar los contratos inteligentes y durante el desarrollo de los mismos. A la operación de escritura sobre la cadena se la denomina *transacción*, mientras que a la operación de lectura se la denomina *llamada*. Es importante conocer sus diferencias:

- *Transacción*: las transacciones cambian el estado de la red, escriben o modifican los datos almacenados en la cadena. Para ejecutarlas se ha de pagar una cuota conocida como *Gas*, mediante la cual se consigue limitar la capacidad de cómputo total que se le asignará a la ejecución de la transacción, por cada paso computacional se irá decrementando esta cantidad. Al llegar a cero la transacción terminará con un fallo, de esta manera se abordan los problemas de *spam* y de ejecuciones potencialmente infinitos (bucles sin condición de parada). La ejecución de una función de un contrato inteligente por medio de una transacción no devuelve el valor de retorno puesto que no se puede asegurar que la transacción se ejecute inmediatamente. Por este motivo, las funciones de los *smart contracts* que se ejecutan con transacciones (que escriben datos) no suelen devolver valores.
- *Llamada*: las llamadas no realizan cambios permanentes sobre el estado de la red, por este motivo son gratuitas y suelen utilizarse para ejecutar funciones que leen datos. Cuando se ejecuta una función mediante una llamada se obtiene el valor de retorno inmediatamente.

Por último, resta mencionar que para la implementación inicial de *Ethereum* se decidió hacer uso de una variante de árbol de Merkle denominada *Patricia tree* y de *Ethash*, un algoritmo de consenso de prueba de trabajo propio [25]. Sin embargo, actualmente se pretende llevar a cabo una sustitución del algoritmo de consenso por una nueva implementación de tipo *Proof-of-stake* denominada *Casper* [5].

2.3.2. *Ethereum Virtual Machine*

La *EVM* es la máquina virtual que provee *Ethereum* para llevar a cabo la ejecución del *bytecode* asociado a los contratos inteligentes. El *bytecode* o el *EVM code* es el código de bajo nivel al que se compila el código de los *smart contracts* escritos en lenguajes de alto nivel como *Solidity*. Cada nodo de la red *Ethereum* ha de contar con una “instancia” de la *EVM*. Esta máquina virtual otorga un entorno de ejecución seguro y controlado en el que es posible asegurar la correcta ejecución de las transacciones, y por consiguiente, de los contratos inteligentes.

Capítulo 3

Desarrollo de la solución propuesta

En este capítulo de la memoria se exponen las tecnologías seleccionadas para llevar a cabo el desarrollo de la aplicación web. A su vez, se explican y presentan los detalles de implementación y particularidades de dicha aplicación.

3.1. Selección de tecnologías

A lo largo de esta sección se expondrán de manera resumida las tecnologías seleccionadas, el porqué de dicha elección y una explicación del papel que juegan en el conjunto de la aplicación web desarrollada.

3.1.1. *Typescript*

Para el desarrollo de la aplicación web se optó por hacer uso del lenguaje de programación *Typescript* [15] (Figura 3.1). *Typescript* es un *superset* fuertemente tipado de *Javascript*, el lenguaje por excelencia para el desarrollo web. Mediante la adición de tipos estáticos al código *Javascript* se consigue agilizar y mejorar el desarrollo. Al ser fuertemente tipado, permite a las herramientas de desarrollo la detección de errores de manera anticipada. Además, otorga un mayor control y un incremento en la coherencia del código desarrollado.



Figura 3.1: Logo de *Typescript*

3.1.2. *React* y *Material-UI*

Se creyó conveniente utilizar *React* [7] (3.2) como librería *Javascript* de desarrollo de interfaces gráficas de usuario. Está basada en componentes encapsulados con estado propio y permite

la creación de interfaces de usuario interactivas de forma sencilla, y tiene soporte total para *Typescript*. El hecho de que la librería esté escrita en *Javascript* facilita la interacción entre las partes gráfica (apariencia visual) y lógica (datos y funcionamiento) de la aplicación.

Mediante la declaración de componentes se favorece la reutilización de partes de la interfaz gráfica, permitiendo diseñar vistas simples para cada estado de la aplicación. *React* se encarga de actualizar y renderizar de manera eficiente los componentes correctos cuando los datos han sufrido cambios. Al crear las vistas de la aplicación de manera declarativa el código se vuelve más predecible y fácil de depurar.

Por último cabe destacar que gracias a la naturaleza basada en componentes de *React* existen numerosas librerías de componentes gráficos ya preparados. Tal es el caso de *Material-UI* [12] (3.2), que ofrece numerosos componentes de *React* ya preparados para agilizar la construcción de interfaces gráficas siguiendo la normativa de diseño *Material Design*.

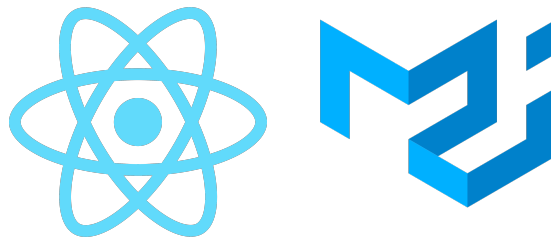


Figura 3.2: Logos de *React* y *Material-UI*

3.1.3. *Truffle* y *Ganache*

Truffle [21] y *Ganache* [20] (Figura 3.3) forman parte de lo que se conoce como *Truffle Suite*, un conjunto de herramientas que facilitan el desarrollo de aplicaciones descentralizadas. En concreto, *Truffle* proporciona un entorno de desarrollo y marco de pruebas para *blockchain* utilizando la *Ethereum Virtual Machine*. Permite la compilación, depuración y el despliegue de contratos inteligentes por línea de comandos o a través de una consola interactiva de manera sencilla. Por otro lado, *Ganache* ofrece la posibilidad de desplegar en local una *blockchain* local de prueba para poder comprobar el funcionamiento de las aplicaciones descentralizadas en desarrollo.



Figura 3.3: Logos de *Truffle* y *Ganache*

3.1.4. Metamask

Metamask [14] (Figura 3.4) es una extensión para el navegador que permite la ejecución de aplicaciones descentralizadas sin necesidad de un nodo *Ethereum*, o dicho de otra forma, sin ser parte de la red *Ethereum*. En concreto se conecta a un nodo de la red externo denominado *INFURA* y ejecuta los *smart contracts* en dicho nodo. Da la posibilidad de crear cuentas de usuario asociadas a la *blockchain* a la que se conecte, almacenando su clave privada de forma segura y permitiendo a dichas cuentas recibir y enviar transacciones y *Ether* a las aplicaciones aprobadas.



Figura 3.4: Logo de *Metamask*

3.2. Aplicación desarrollada

La aplicación creada para el desarrollo de este Trabajo de Fin de Máster tiene como nombre *SupplyBlocks* [22] (Figura 3.5). En esta sección se explican al detalle las particularidades de su implementación.

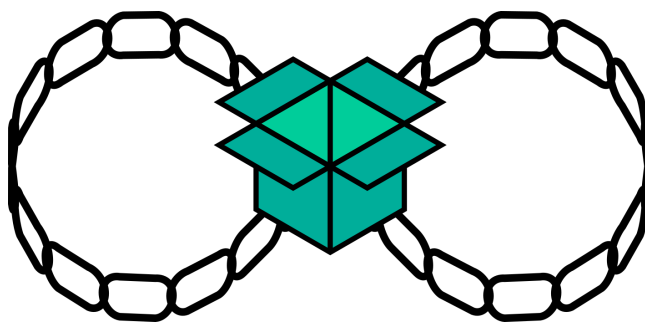


Figura 3.5: Logo de *SupplyBlocks*

3.2.1. Particularidades

La principal y más importante particularidad de *SupplyBlocks* reside en el hecho de que es totalmente descentralizada, es decir, no se apoya en ningún tipo de servicio (más allá del servidor web utilizado para desplegarla) y es posible utilizarla únicamente con un navegador (y la *blockchain* desplegada en local). En las siguientes secciones de este capítulo se entrará en detalle, pero cabe adelantar que uno de los contratos inteligentes desarrollados (denominado

Manager) actúa como servicio de la aplicación. Esta ventaja es posible gracias a lo que se puede considerar un gran inconveniente en la *blockchain* de *Ethereum*, **la cadena de bloques ha de almacenar toda la información sobre los productos y las compañías registradas** para el correcto funcionamiento de la aplicación.

En la *blockchain* pública de *Ethereum*, almacenar mucha información en la cadena de bloques acarrea un alto costo y se penaliza, a más información es necesario crear más bloques, y para la creación de éstos hay que aportar potencia computacional extra. Por ello, el guardar datos de manera indiscriminada está considerado como una mala práctica, sin embargo, ante las cadenas de bloques permissionadas se plantea otro escenario. Al tratarse de una red privada de acceso restringido y más controlado, almacenar una mayor cantidad de información no supone un gran problema, pues se parte de la asunción de que la cadena de bloques privada tendrá un crecimiento menor al de una pública. El deseo de mantener el carácter descentralizado junto a la necesidad de restringir el acceso a las funcionalidades de la aplicación supusieron los principales motivos por los que se decidió que el funcionamiento de la *SupplyBlocks* requiriera una *blockchain* privada.

Para desplegar la aplicación y poder llevar a cabo pruebas y observar los resultados del desarrollo fue necesario realizar los pasos que a continuación se enumeran:

1. Desplegar una *blockchain* de prueba con *Ganache*. Al desplegar la cadena de bloques, *Ganache* provee un número de cuentas de prueba con *Ether* ficticio. En el caso de la aplicación desarrollada para este trabajo, el *Ether* es necesario para poder pagar los costos asociados a la ejecución de las transacciones de los contratos inteligentes (*Gas*).
2. ConFigurar *Metamask* desde el ordenador para hacer uso de la *blockchain* privada y poder acceder a las cuentas de usuario de prueba que provee *Ganache*.
3. Compilar y desplegar los contratos inteligentes haciendo uso de *Truffle*.

Una vez seguidos estos pasos, la aplicación es totalmente funcional. Es sumamente relevante aclarar que la gestión de cuentas de *SupplyBlocks* se delega a *Metamask* que, como se ha expuesto con anterioridad, es capaz de almacenar de forma segura y distribuida las claves privadas de las cuentas de usuarios de la *blockchain*. Al iniciar la aplicación por primera vez habrá que dar permisos a *Metamask* para que interactúe con ella. El usuario actual de la aplicación queda determinado por la cuenta seleccionada en *Metamask*. Cada vez que el usuario realiza una acción que deriva en la invocación de una función de un *smart contract* por medio de una *transacción* (modificación de datos) *Metamask* abrirá una pequeña ventana para confirmar dicha transacción.

Al desplegar la aplicación se crea una cuenta especial única, denominada cuenta del administrador. Ésta tiene es sumamente importante pues es la única cuenta capaz de habilitar otras cuentas. Cuando una compañía desea usar la aplicación cumplimenta un formulario y se crea una nueva cuenta, tras esto, deberá esperar a que el administrador apruebe su cuenta para comenzar a disfrutar de *SupplyBlocks*. Este proceso de espera de aprobación de la cuenta equivale a la aceptación de un nuevo miembro para la red privada de negocios.

Otra de las particularidades más importantes de *SupplyBlocks* es su adaptabilidad, se ha procurado conseguir que cualquier acción del usuario que suponga un cambio significativo (cambio de cuenta en *Metmask*, creación de un nuevo producto, ...) produzca un nuevo renderizado de los elementos de la interfaz gráfica correspondientes. Esto se ha conseguido en gran medida

gracias a la facilidad que otorga *React* para ello. Cuando se lleva a cabo una acción que requiere la ejecución de una transacción

El proceso de cálculo de la ruta de un pedido es simulado por la propia aplicación web. Ésta escoge de manera aleatoria un conjunto de usuarios de tipo *T* y la cantidad correspondiente de usuarios de tipo *A* (cantidad de usuarios de tipo *T* escogidos menos uno). Y por último, aunque pueda parecer obvio, cabe recalcar que el control de las acciones que los usuarios son capaces de realizar se lleva a cabo íntegramente desde los *smart contracts*, asegurando el buen funcionamiento de la aplicación.

3.2.2. Contratos inteligentes

En esta sección se exponen los distintos contratos inteligentes implementados para el desarrollo de *SupplyBlocks*. Para cada uno se comentarán y mostrarán los métodos y miembros más relevantes, junto con su funcionamiento u objetivo. A pesar de que no se profundice totalmente en los detalles de implementación de cada método por cuestiones de longitud de la memoria, se ha de destacar que se han utilizado cláusulas *require* de *Solidity* con el propósito de garantizar el correcto funcionamiento de los contratos inteligentes, teniendo especial cautela con los parámetros recibidos.

TypesLibrary

Contiene la declaración de los *structs* *EntityData* (Figura 3.6) y *ProductData* (Figura 3.7) utilizados para el almacenamiento de la información sobre los usuarios (entidades) y los productos/envíos respectivamente. Por otro lado, también se declaran los *enums* auxiliares (Figura 3.8) que definen el tipo de una entidad *EntityType* y el estado de un producto *ProductState*.

The image shows a code editor window with a dark background and light-colored text. The code defines a library named 'TypesLib' which contains a struct named 'EntityData'. The struct has several members: 'address id', 'string name', 'string email', 'string phoneNumber', 'TypesLib.EntityType entityType', 'bool set', and 'bool approved'. The code is enclosed in curly braces for the struct and the library.

Figura 3.6: *Struct EntityData*, representa los datos de un usuario

Entity

Smart contract que únicamente cuenta con un *struct* miembro del tipo *EntityData*. Este contrato resulta necesario para el almacenamiento de la información de los usuarios. Cada

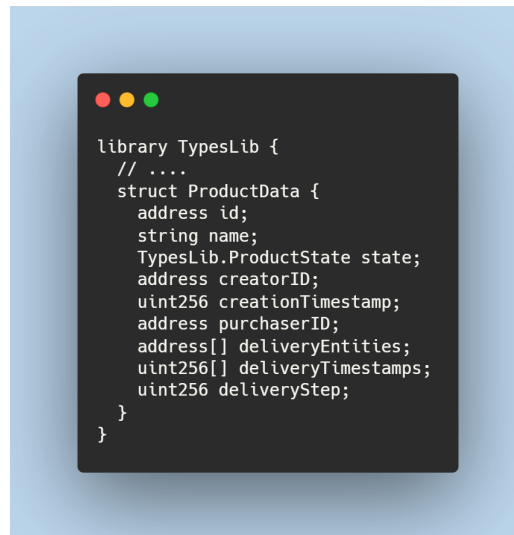


Figura 3.7: *Struct ProductsData*, representa los datos de un producto/envío

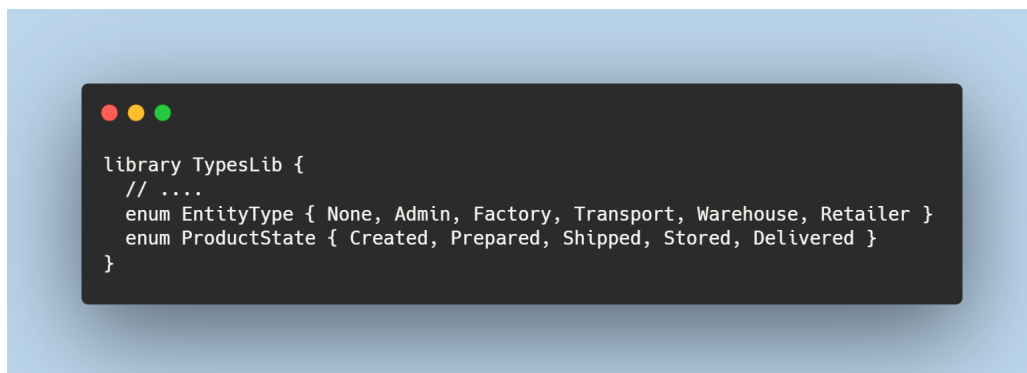


Figura 3.8: *Enums EntityType y ProductState*, representan el tipo de usuario o entidad y el estado de un producto respectivamente

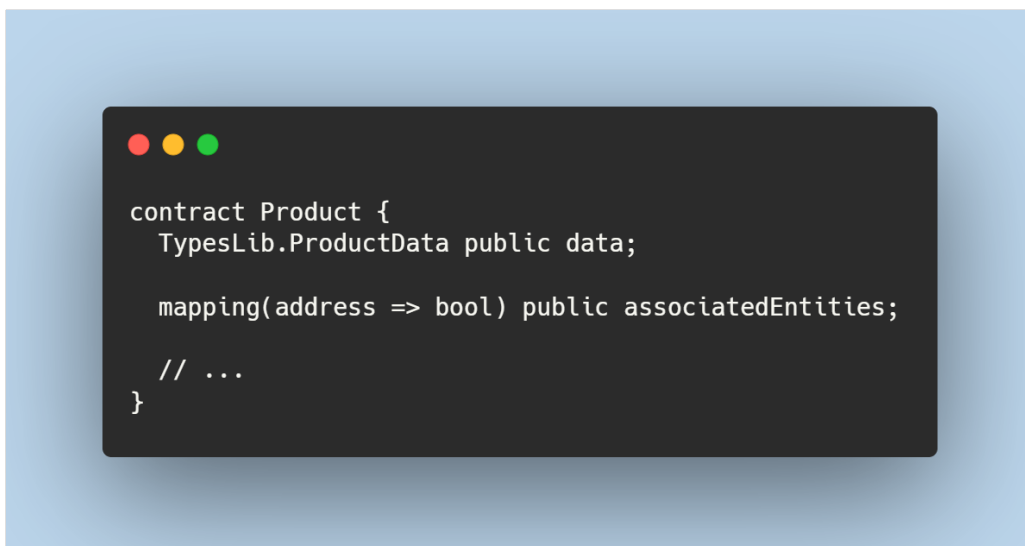
usuario creado en *SupplyBlocks* tiene un único contrato de este tipo asociado. A continuación se muestra la explicación de cada una de las propiedades del *struct EntityData*:

- *address id*: Dirección de la cuenta de usuario asociada a la entidad
- *string name*: Nombre de la compañía
- *string name*: Email de contacto de la compañía
- *string name*: Teléfono de contacto de la compañía
- *TypesLib.EntityType entityType*: Tipo de la compañía
- *bool set*: Indica si el contrato ha sido inicializado. Junto con *bool approved* especifican si el usuario está pendiente de la aprobación del usuario administrador o ya posee una cuenta aprobada.
- *bool approved*: Indica si la entidad ha sido aprobada por el usuario administrador.

Product

Almacena los datos asociados a un producto y las entidades relacionadas con éste (Figura 3.9). Por entidad relacionada se entiende que ha participado en la creación o en el envío de un producto a un usuario de tipo *M*. A continuación se muestra la explicación de cada una de las propiedades del *struct ProductData*:

- *address id*: Dirección de este contrato
- *string name*: Nombre del producto
- *TypesLib.ProductState state*: Estado del producto
- *address creatorID*: Dirección del contrato inteligente *Entity* del usuario de tipo *F* que creó el producto
- *uint256 creationTimestamp*: Marca de tiempo de la creación del producto
- *address purchaserID*: Dirección del contrato inteligente *Entity* del usuario de tipo *M* que creó el producto *bool approved* especifican si el usuario está pendiente de la aprobación del usuario administrador o ya posee una cuenta aprobada
- *address[] deliveryEntities*: Array con las direcciones de los contratos inteligentes *Entity* asociados a las compañías que conforman la ruta de la entrega
- *uint256[] deliveryTimestamps*: Array con las marcas de tiempo de los cambios de estado del producto, o dicho de otra manera, del momento en el que se completan las diferentes etapas de la entrega del producto
- *uint256 deliveryStep*: Apunta a la etapa actual de la entrega del producto

A screenshot of a code editor with a dark background and light blue text. The code defines a Solidity contract named 'Product'. It includes a public variable 'data' of type 'TypesLib.ProductData', a public function 'associatedEntities' that takes 'address' as an argument and returns a 'bool', and a comment '// ...'. The code is enclosed in curly braces.

```
contract Product {  
    TypesLib.ProductData public data;  
  
    mapping(address => bool) public associatedEntities;  
  
    // ...  
}
```

Figura 3.9: Miembros del contrato inteligente *Product*

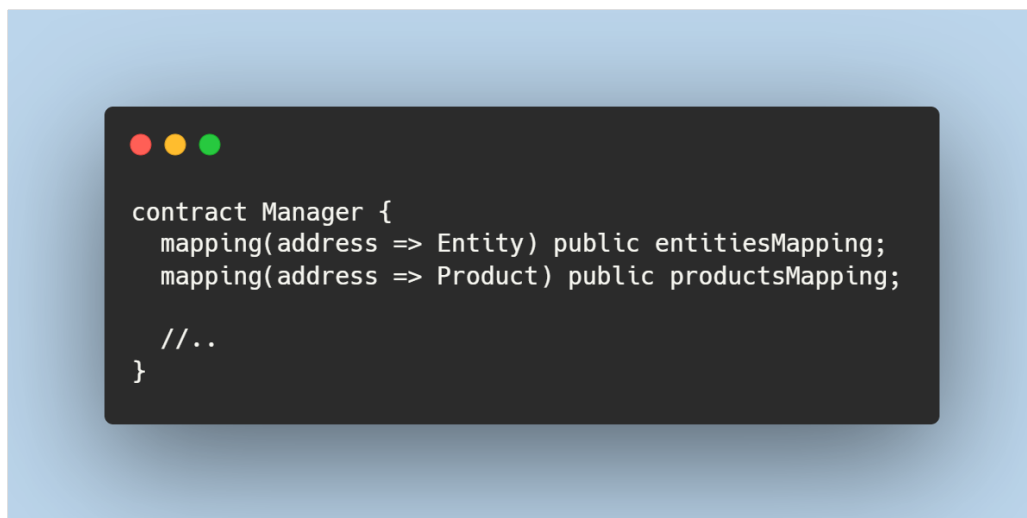
Los métodos de especial relevancia del contrato inteligente *Product* pueden encontrarse en el cuadro 3.1.

Método	Descripción
<i>prepareDelivery</i>	Establece el estado del producto a <i>Prepared (P)</i>
<i>timestampDeliveryStep</i>	Cambia el estado y establece una marca de tiempo
<i>purchase</i>	Marca el producto como comprado y establece una ruta de envío

Cuadro 3.1: Métodos relevantes del contrato inteligente *Product*

Manager

El contrato inteligente *Manager* ocupa una posición importante en el funcionamiento de *SupplyBlocks* pues actúa como “servicio coordinador dentro de la *blockchain*”. Se trata del único contrato inteligente de entre los desarrollados que es desplegado directamente por *Truffle*. La Figura 3.10 presenta los miembros relevantes de este contrato: *entitiesMapping* y *productsMapping*, necesarios para llevar un registro de las entidades y los productos creados. En el Cuadro 3.2 se muestran los métodos de especial relevancia de este contrato inteligente:

Figura 3.10: Miembros relevantes del contrato inteligente *Manager.png*

Método	Descripción
<i>createEntity</i>	Crea una nueva entidad (usuario)
<i>createProduct</i>	Crea un nuevo producto
<i>approveEntity</i>	Aprueba una entidad (solicitud de nuevo usuario)
<i>getEntities</i>	Devuelve las entidades registradas
<i>getProducts</i>	Devuelve los productos registrados

Cuadro 3.2: Métodos relevantes del contrato inteligente *Manager*

3.2.3. Vistas de la aplicación

La aplicación web está conformada por tres vistas principales: *Inicio*, *Registro* y *Cuadro de mando*. A continuación se describirán las acciones que se pueden realizar desde cada una de estas vistas y sus diferentes partes junto con una serie de capturas de pantalla de las partes

más relevantes de *SupplyBlocks* para que se pueda apreciar su aspecto general. Además de las vistas principales, la aplicación incluye mecanismos auxiliares para notificación de errores (vistas secundarias, mensajes en pantalla, ...) que no podrán mostrarse en la presente memoria por cuestiones de longitud e interactividad. A pesar de esto, siguiendo las instrucciones descritas en el repositorio de la aplicación [22] es posible desplegar *SupplyBlocks* localmente para poder realizar las pruebas y comprobaciones que se consideren oportunas.

Inicio

La vista de *Inicio* presenta el mensaje de bienvenida a la aplicación, enumera las ventajas que ofrece *SupplyBlocks*, a qué tipo de compañías se dirige su funcionamiento y muestra una línea temporal explicativa del proceso llevado a cabo para registrar de manera inmutable los envíos de productos desde los fabricantes hasta los pequeños negocios. La Figura 3.11 muestra la parte superior de la vista de *Inicio* y la Figura 3.12 la parte inferior.

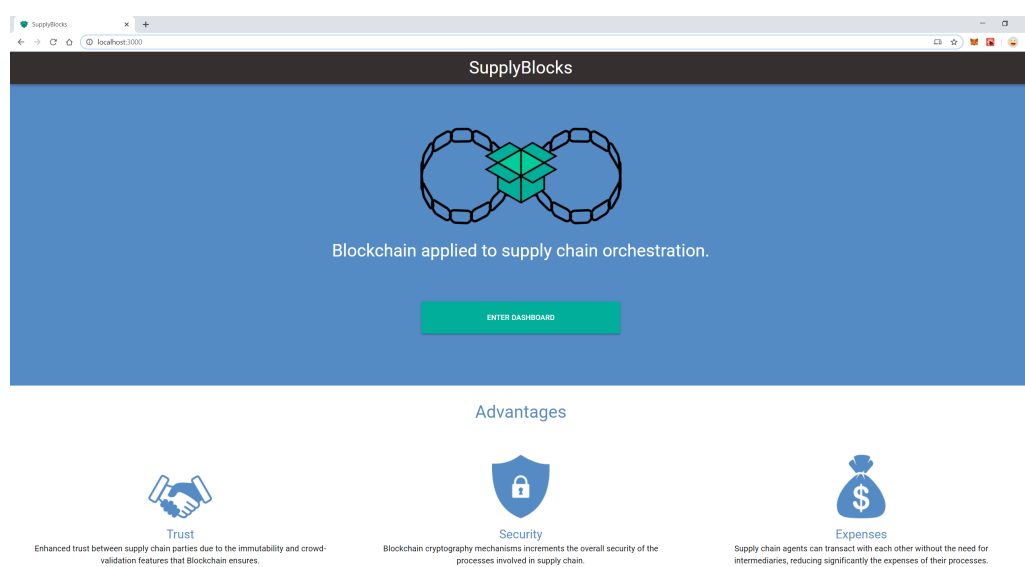
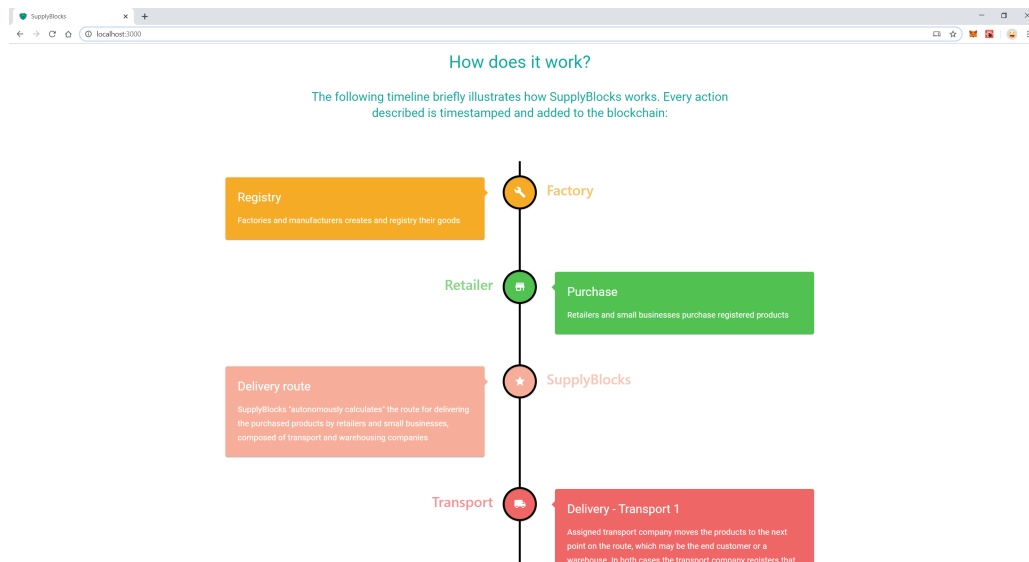


Figura 3.11: Parte superior de la vista de *Inicio* de *SupplyBlocks*

El componente más importante de esta vista es el botón central de la parte superior de la página. Su comportamiento y texto cambia en función de si el usuario actual (cuenta activa en *Metamask*) se ha registrado y ha sido aprobado por el administrador, lo que equivale a ser un usuario válido de *SupplyBlocks*. Si el usuario que ha accedido a la aplicación es un usuario válido al pulsar el botón se le redirigirá a la vista *Cuadro de mando*, y en caso contrario se le redirigirá a la vista de *Registro*.

Registro

La vista de *Registro* presenta un formulario para que las compañías que deseen unirse a *SupplyBlocks* cumplimenten una solicitud. La Figura 3.13 muestra la apariencia visual de esta vista.

Figura 3.12: Parte inferior de la vista de *Inicio* de *SupplyBlocks*

Send a petition for joining SupplyBlocks

Company Name *
Example

Email *
example@example.com

Phone number *
123456789

Company type
Warehouse

☒ I'm aware that registering an entity the active Metamask account is associated permanently to that entity.

SEND PETITION

CANCEL

Figura 3.13: Vista de *Registro* de *SupplyBlocks*

Cuadro de mando

La vista *Cuadro de mando* es la más importante de *SupplyBlocks*. Se trata de la página desde la cual las diferentes compañías realizarán las acciones permitidas y podrán comprobar el estado de sus productos y/o los envíos asociados a los mismos. Está dividida en tres menús: *Compañías*, *Productos* y *Envíos*. A continuación se expone cuál es el propósito de cada una de estas partes y qué acciones pueden realizar desde ellas los usuarios. Por último, se debe mencionar que la navegación entre los distintos menús se consigue pulsando los botones que aparecen en la parte derecha de la vista, junto con un resumen del usuario actual en el que se presenta la información asociada al usuario.

Compañías

El menú *Compañías* muestra las compañías registradas en *SupplyBlocks*. Por cada usuario se muestra su nombre, el tipo de compañía, la dirección del contrato inteligente *Entity* asociado, su dirección de correo electrónico y su número de contacto. En caso de que el usuario actual sea el administrador, la interfaz también mostrará las peticiones para unirse a *Supplyblocks*. Junto a cada petición aparecerá un botón para aprobar la petición y registrar un nuevo usuario. En la Figura 3.14 se muestra el menú desde el punto de vista del usuario administrador, y puede apreciarse cómo hay registradas tres entidades y una compañía de transportes está pendiente de confirmación.

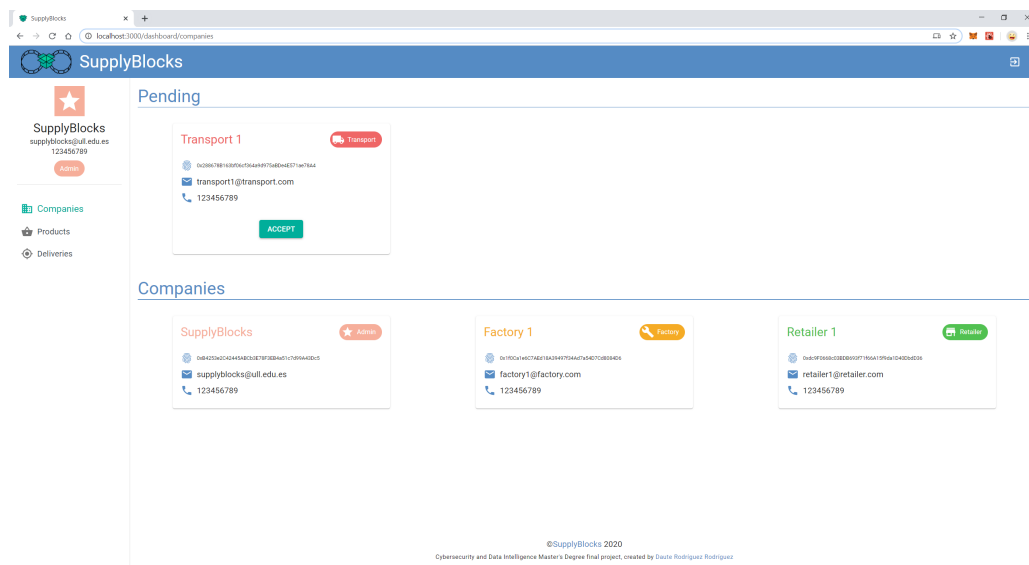


Figura 3.14: Menú *Compañías* desde el punto de vista del usuario administrador

Productos

El menú *Productos* muestra los productos asociados a la cuenta actual. Por cada producto se presentan los siguientes campos:

- Nombre del producto
- Estado del producto
- Dirección del contrato inteligente *Product* asociado
- Dirección del contrato inteligente *Entity* del usuario tipo *F* que lo creó
- Fecha de creación
- Dirección del contrato inteligente *Entity* del usuario tipo *M* que lo adquirió, en caso de haberse adquirido
- Fecha de entrega, en caso de haberse entregado

A su vez, esta vista también se comporta de manera diferente atendiendo al tipo de usuario actual:

- Administrador: se muestran todos los productos registrados.
- Fabricante: se muestra un formulario para añadir nuevos productos. Si un producto ha sido adquirido, junto a su tarjeta aparecerá un botón para preparar el envío, o lo que es lo mismo, para cambiar el estado del producto de C a P .
- Minorista: aparecen todos los productos registrados y un botón junto a ellos para adquirirlo. Este tipo de usuario es el único que puede adquirir productos.
- Compañía de transporte: se muestran los productos que se han de transportar o los productos transportados en el pasado.
- Compañía de almacenaje: se muestran los productos almacenados actualmente o en el pasado.

La Figura 3.15 muestra el menú *Productos* desde la perspectiva de un usuario tipo F que está a punto de crear uno nuevo. Por otro lado, en la Figura 3.16 pueden verse numerosos productos desde una cuenta de usuario tipo M . Además, es posible apreciar los diferentes estados de los productos, cómo aparece habilitado el botón de compra para los productos no adquiridos y cómo el producto *Product 2* (ya entregado) muestra la fecha de entrega.

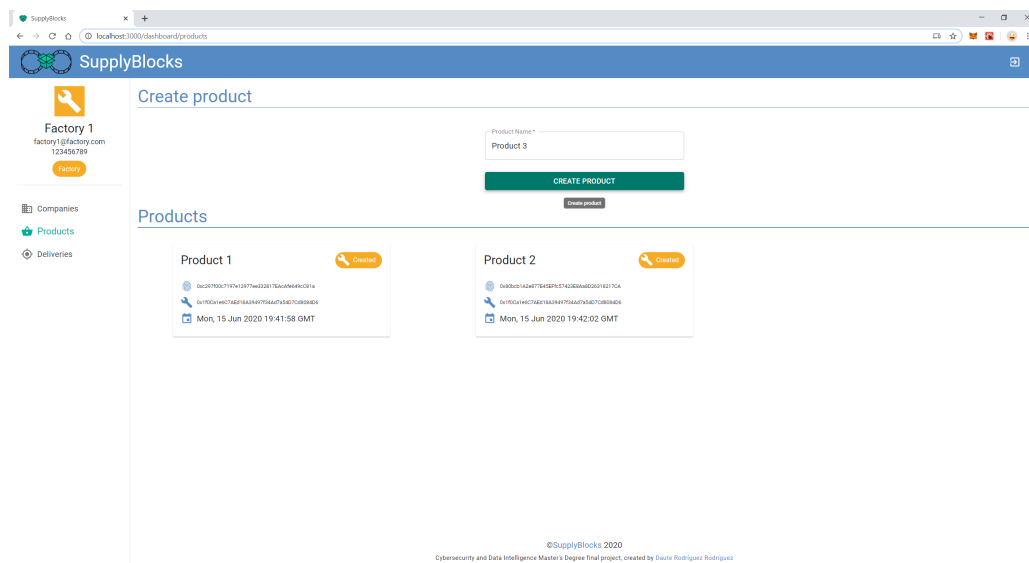


Figura 3.15: Menú *Productos* desde el punto de vista de un usuario tipo F

Envíos

En el último menú del cuadro de mandos se muestra el histórico de pedidos asociados a productos relacionados con la cuenta actual. Por cada pedido se muestra un panel desplegable, la información mínima que contiene dicho panel es el estado actual del producto y su nombre. Además, si el envío del producto requiere ser confirmado por el usuario actual para así cambiar

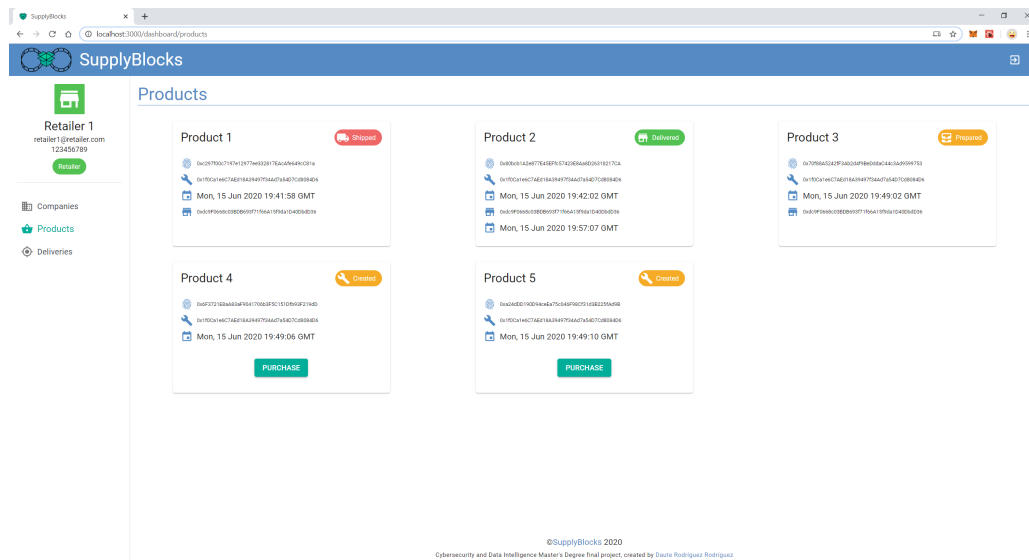


Figura 3.16: Menú *Productos* desde el punto de vista de un usuario tipo *M*

su estado, aparecerá un botón para llevar a cabo dicha confirmación. Por ejemplo, si el usuario actual es un usuario tipo *T* y está llevando el producto al próximo destino de la ruta, en el momento en el que lo entregue deberá pulsar el botón para confirmar que el producto ha sido entregado. Este ejemplo se aplica a los demás tipos de usuario atendiendo al diagrama presente en la Figura 1.4.

Al desplegar el panel se mostrará una línea temporal con los diferentes pasos del envío. Por cada paso se muestra un elemento con el sello de tiempo en el que se completó y la dirección del contrato inteligente *Entity* del usuario que lo registró. En caso de no haberse completado el envío los elementos correspondientes de la línea temporal aparecerán en gris. En la Figura 3.17 se presenta un ejemplo de los paneles desplegables anteriormente comentados. Se corresponden con los pedidos de los productos *Product 1*, *Product 2* y *Product 3*. A su vez, es posible apreciar el estado de dichos productos: *Shipped* (*Tránsito*), *Delivered* (*Entregado*) y *Prepared* (*Preparado*) respectivamente.

En la Figura 3.18 se puede visualizar la línea de tiempo asociada al envío del producto *Product 2*, el cual no ha finalizado. Por otro lado, la Figura 3.18 muestra la misma línea de tiempo una vez ha finalizado el envío.

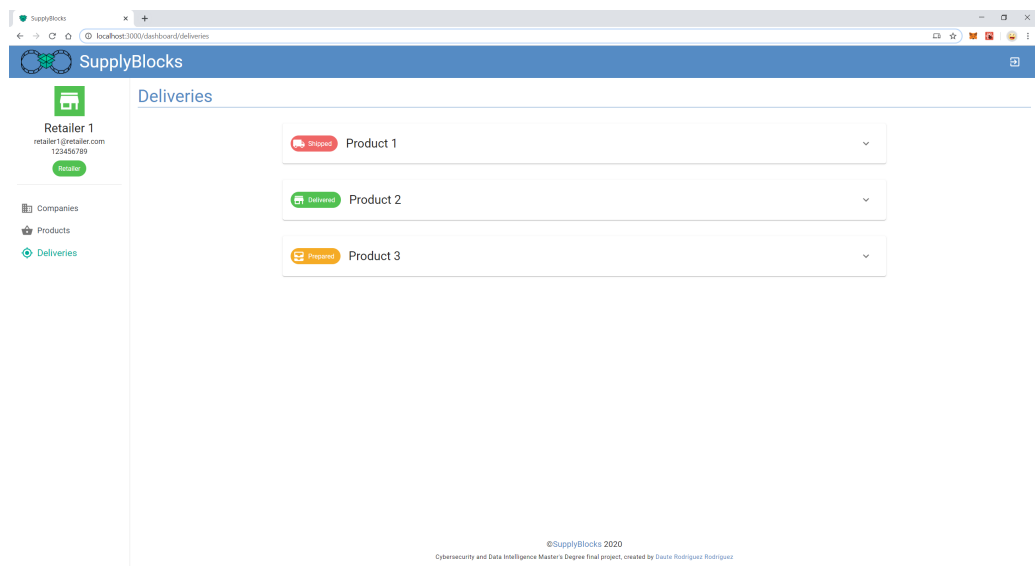


Figura 3.17: Menú *Envíos* desde el punto de vista de un usuario tipo *M*

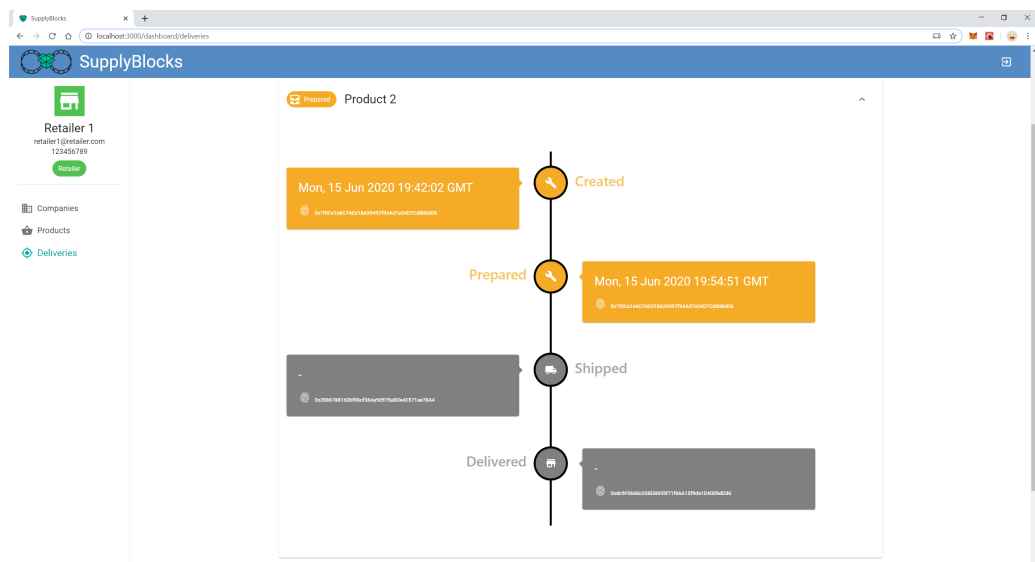


Figura 3.18: Menú *Envíos*. Línea de tiempo de un envío no finalizado



Figura 3.19: Menú *Envíos*. Línea de tiempo de un envío finalizado

Capítulo 4

Conclusiones y líneas futuras

En este Capítulo se presentan las conclusiones alcanzadas, así como las posibles líneas futuras

4.1. Conclusiones

Partiendo de un problema que atañe al sector de la logística, se ha realizado una investigación con el propósito de comprender el funcionamiento de una de las tecnologías más prometedoras de la actualidad: la cadena de bloques. Gracias al estudio realizado, ha sido posible entender cómo al aplicar correctamente esta tecnología se consiguen resultados ventajosos.

La implementación de la aplicación web descentralizada para la plataforma *Ethereum* ha servido para asentar los conocimientos teóricos adquiridos durante las fases iniciales de investigación del proyecto. A pesar del grado de simplicidad de las acciones que los usuarios de *SupplyBlocks* pueden llevar a cabo haciendo uso de la aplicación, se ha demostrado que esta tecnología es capaz de alterar sustancialmente el modo en el que se realizan y controlan los procesos de numerosos sectores, logrando un estado de confianza máxima entre los interesados del mismo.

El hecho de que no se hayan contemplado algunos de los factores más importantes de la gestión de la cadena de suministro como los costes, las condiciones de los traslados de los productos o el control de calidad se deben al corto alcance de este Trabajo Fin de Grado. Sin embargo, es posible considerar que tal y como se ha concebido la aplicación web y su funcionamiento, no sería excesivamente complicado ampliar sus funcionalidades con aspectos de esta índole. A continuación, se presentan algunas de las posibles líneas futuras a considerar tras la elaboración del proyecto.

4.2. Líneas futuras

Estas son algunas de las posibles ampliaciones que se deberían contemplar como continuación del proyecto:

- Estudiar hacer uso de algún protocolo como *IPFS* para el almacenamiento distribuido de datos, evitando su almacenamiento en la *blockchain* y mejorando el rendimiento a largo plazo.
- Incluir en el modelo aspectos como precios, condiciones de transporte y control de calidad.
- Añadir nuevos tipos de agentes de la cadena de suministro como entidades certificadoras.

Capítulo 5

Summary and Conclusions

This Chapter presents the conclusions, as well as some research and experimentation proposals to take into account in the future.

5.1. Conclusions

Starting from a problem concerning the logistics sector, an investigation has been carried out in order to understand how one of the most promising technologies works: blockchain. The study carried out has made possible to understand how correctly applying this technology can achieve advantageous results.

The implementation of the decentralized web application for *Ethereum* platform led to consolidation of theoretical knowledge acquired during the initial research phases of the project. Despite the simplicity of the actions that users of *SupplyBlocks* can carry out by using the application, it has been demonstrated that blockchain technology is able to substantially altering the way in which processes are carried out and managed in several fields and environments, obtaining maximum confidence among the stakeholders of this processes.

The fact that some of the most important factors of supply chain orchestration such as costs, shipment conditions and quality control have not been included *SupplyBlocks* is due to short term of this Master's Degree Final Project. However, it is possible to consider that the manner in which the web application have been conceived ease the tasks of extending its functionalities with those factors. In the next section some examples of possible future lines to be considered are presented.

5.2. Future work

These are some of the possible extensions that should be considered as a continuation of the project:

- Study the use of some protocol like IPFS for storing in a distributed way *SupplyBlocks* data improving the blockchain performance in the long term.
- Extend *SupplyBlocks* for take into consideration aspects such as prices, shipment conditions and quality control.
- Add new types of supply chain agents as products certification authorities.

Capítulo 6

Presupuesto

En este Capítulo se expone una estimación del presupuesto del proyecto. El Cuadro 6.1 muestra el coste de cada una de las actividades de investigación e implementación llevadas a cabo.

Actividad	Precio / Hora	Horas invertidas	Coste
Investigación <i>blockchain</i>	21 €	50	1050 €
Desarrollo <i>smart contracts</i>	21 €	25	525 €
Desarrollo aplicación web	16 €	25	400 €
Integración aplicación web - <i>smart contracts</i>	16 €	35	560 €
Total	-	135	2.535 €

Cuadro 6.1: Presupuesto estimado del proyecto

Glosario

Bytecode Lenguaje intermedio al que se compilan los programas escritos en algunos lenguajes de alto nivel para posteriormente ser interpretados eficientemente por un intérprete.

Casper Nueva propuesta de algoritmo de consenso para la plataforma *Ethereum*. Se trata de un algoritmo de consenso de tipo *Proof-of-stake*.

Control de acceso basado en roles Sistema de control de acceso en el que se establecen permisos por grupos de usuario o roles.

Criptomoneda Medio digital de intercambio, se utiliza criptografía para asegurar las transacciones, controlar la creación de unidades adicionales y verificar la transferencia de activos usando tecnologías de registro distribuido (*DLT*).

Dapps Aplicación descentralizada, tipo de aplicación cuyo funcionamiento se basa en una red descentralizada de nodos que actúan entre sí sin necesidad de un organismo regulador central.

Distributed Ledger Technology Tecnología de registro distribuido, mediante el uso de este tipo de tecnologías se permite la creación de registros inalterables gestionados de forma descentralizada.

Ether Criptomoneda nativa de la plataforma *Ethereum*.

Función trampa Función matemática cuyo cálculo directo es sencillo, pero en la que el cálculo de la función inversa es muy complejo, es decir, involucra un elevado número de operaciones. Por ejemplo, el producto de números primos.

Gas Nombre de la cuota que ha de abonarse en la plataforma *Ethereum* al llevar a cabo transacciones.

Lenguaje de programación *Turing* completo Lenguaje de programación mediante el que es posible emular el modelo computacional máquina de *Turing*.

Logistics Performance Index Indicador utilizado para la evaluación comparativa de las oportunidades y el desempeño en materia de logística comercial de los países.

Minero Nodo que participa en una red *blockchain* aportando su potencia de cómputo para la creación o “minado” de nuevos bloques para la cadena.

Máquina de estados Modelo computacional que realiza cálculos automáticamente sobre una entrada para producir una salida.

Peer-to-peer Red de ordenadores en la que todos o algunos aspectos funcionan sin clientes ni servidores fijos, de manera que todos los nodos se comportan como iguales entre sí.

Problema del doble gasto Defecto potencial del dinero digital por el que una misma moneda digital (*token*) puede gastarse más de una vez.

Producto Interior Bruto Magnitud macroeconómica que expresa el valor monetario de la producción de bienes y servicios de demanda final de un país o región durante un período determinado.

Acrónimos

CEO Chief Executive Officer, director ejecutivo.

DLT Distributed Ledger Technology, Tecnología de Registro Distribuida.

EVM Ethereum Virtual Machine, máquina virtual de Ethereum.

LPI Logistics Performance Index, Índice de Desempeño Logístico.

P2P Peer-to-Peer, red de pares.

PIB Producto Interior Bruto.

PoA Proof-of-authority, prueba de autoridad.

PoS Proof-of-stake, prueba de participación.

PoW Proof-of-work, prueba de trabajo.

RBAC Role Based Access Control, control de acceso basado en roles.

Bibliografía

- [1] Andreas M Antonopoulos. *Mastering Bitcoin: unlocking digital cryptocurrencies*. .O'Reilly Media, Inc.", 2014.
- [2] Jean-François Arvis, Lauri Ojala, Christina Wiederer, Ben Shepherd, Anasuya Raj, Karlygash Dairabayeva, and Tuomas Kiiski. Connecting to compete 2018. 2018.
- [3] BurstIQ. *Blockchain based healthcare data solutions*. <https://www.burstiq.com/>, 2020. [Online; accessed 24-May-2020].
- [4] Vitalik Buterin et al. A next-generation smart contract and decentralized application platform. *white paper*, 3(37), 2014.
- [5] Vitalik Buterin and Virgil Griffith. Casper the friendly finality gadget. *arXiv preprint arXiv:1710.09437*, 2017.
- [6] Etherisc. Decentralized insurance protocol to collectively build insurance products. <https://etherisc.com/>, 2020. [Online; accessed 24-May-2020].
- [7] Facebook. React. <https://reactjs.org/>, 2020. [Online; accessed 15-Jul-2020].
- [8] Stuart Haber and W Scott Stornetta. How to time-stamp a digital document. In *Conference on the Theory and Application of Cryptography*, pages 437–455. Springer, 1990.
- [9] ID2020. Digital identity alliance. <https://id2020.org/>, 2020. [Online; accessed 24-May-2020].
- [10] Irina V Kozlenkova, G Tomas M Hult, Donald J Lund, Jeannette A Mena, and Pinar Kekec. The role of marketing channels in supply chain management. *Journal of Retailing*, 91(4):586–609, 2015.
- [11] Tiana Laurence. *Blockchain for dummies*. John Wiley & Sons, 2019.
- [12] Material-UI. Material-ui. <https://material-ui.com/>, 2020. [Online; accessed 15-Jul-2020].
- [13] Dr. Markus Kückelhaus Matthias Heutger. Blockchain in logistics - perspectives on the upcoming impact of blockchain technology and use cases for the logistics industry. <https://www.dhl.com/content/dam/dhl/global/core/documents/pdf/glo-core-blockchain-trend-report.pdf>, 2020. [Online; accessed 24-May-2020].
- [14] Metamask. Metamask. <https://metamask.io/>, 2020. [Online; accessed 15-Jul-2020].
- [15] Microsoft. Typescript. <https://www.typescriptlang.org/>, 2020. [Online; accessed 15-Jul-2020].

- [16] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. Technical report, Manubot, 2019.
- [17] Provenance. Every product has a story. <https://www.provenance.org/>, 2020. [Online; accessed 24-May-2020].
- [18] K Sharipbekova and Z Raimbekov. Influence of logistics efficiency on economic growth of the cis countries. *European Research Studies Journal*, 2018.
- [19] ShipChain. The end-to-end logistics platform of the future: trustless, transparent tracking. <https://shipchain.io/>, 2020. [Online; accessed 24-May-2020].
- [20] Truffle Suite. Ganache. <https://www.trufflesuite.com/>, 2020. [Online; accessed 15-Jul-2020].
- [21] Truffle Suite. Truffle. <https://www.trufflesuite.com/>, 2020. [Online; accessed 15-Jul-2020].
- [22] Supplyblocks. <https://github.com/DauteRR/SupplyBlocks>, 2020. [Online; accessed 15-Jul-2020].
- [23] Nick Szabo. Formalizing and securing relationships on public networks. *First Monday*, 2(9), 1997.
- [24] Vottun. Public transport *Blockchain* application. <https://businessblockchainhq.com/business-blockchain-news/madrid-blockchain-public-transit-app/>, 2020. [Online; accessed 24-May-2020].
- [25] Gavin Wood et al. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum project yellow paper*, 151(2014):1–32, 2014.