



**Escuela Superior
de Ingeniería y Tecnología**
Universidad de La Laguna

Trabajo de Fin de Grado

**Técnicas para mejorar la seguridad de
una aplicación web**

Techniques to improve the security of a web application

Juan Jesús Padrón Hernández

La Laguna, 9 de junio de 2019

D. **Pino Caballero Gil**, con N.I.F. 45.534.310-Z Catedrática de Universidad adscrita al Departamento de Ingeniería Informática y de Sistemas de la Universidad de La Laguna, como tutora

D. **Carlos B. Rosa Remedios**, con N.I.F. 43.786.084-H responsable de la Unidad de Tecnologías de la Información y la Comunicación de Gestión de Servicios para la Salud y Seguridad en Canarias, como cotutor

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

Que la presente memoria titulada:

"Técnicas para mejorar la seguridad de una aplicación web"

ha sido realizada bajo su dirección por D. **Juan Jesús Padrón Hernández**, con N.I.F. 78.644.518-C.

Y para que así conste, en cumplimiento de la legislación vigente y a los efectos oportunos firman la presente en La Laguna a 9 de junio de 2019

Agradecimientos

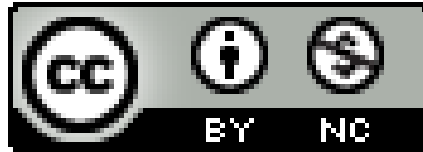
A mi tutora Pino Caballero Gil, por su esfuerzo, preocupación y dedicación.

A mi cotutor y tutor externo de prácticas, Carlos B. Rosa Remedios, por sus consejos y las oportunidades que me ha dado.

A mi familia, tanto a mis padres como a mi hermana, porque son los pilares de mi vida y sin ellos no estaría donde estoy.

Y a todas las personas que me ha apoyado durante estos años.

Licencia



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

Resumen

El objetivo de este trabajo ha sido mejorar la seguridad del aplicativo web empleado por el 112 Canarias para dar información a empresas de transporte sobre el movimiento de pacientes a través de la Mesa de Transporte Sanitario No Urgente.

Para ello, se ha desarrollado un método de autenticación en el servidor web empleando blockchain. Esta tecnología permite tener una base de datos distribuida, segura e inmutable, de forma que mejora tanto la fiabilidad de la red, como su trazabilidad.

Se ha investigado la tecnología blockchain, analizando su funcionamiento y principales características, con el fin de determinar cómo aplicar esta tecnología en un entorno concreto para decidir qué tipo de blockchain es el más adecuado para el sistema a desarrollar, y determinar cuáles son las ventajas e inconvenientes de cada opción.

Finalmente, se ha desarrollado el prototipo de un método de autenticación utilizando la cadena de bloques Ethereum, que ofrece la posibilidad de trabajar con contratos inteligentes. Para administrar las claves privadas de los/as usuarios/as registrados/as de forma segura y distribuida se ha utilizado Metamask, que es una extensión clave para comunicar el navegador con la blockchain. Además, como valor añadido de este trabajo, se ha implementado un sistema de control de acceso a archivos para ir más allá en las operaciones de blockchain.

Palabras clave: Seguridad, autenticación, blockchain, Ethereum, contrato inteligente, Metamask

Abstract

The objective of this Final Degree Project has been to improve the security of the web application that 112 Canarias uses to provide information to transport companies about the movements of patients through the Non-Urgent Sanitary Transport Bureau.

A novel authentication method has been developed in the webserver using blockchain. This technology allows having a distributed, secure and immutable database, in a way that improves both the reliability of the network and its traceability.

The blockchain technology has been researched, analyzing its operation and main characteristics, in order to determine how to apply this technology in an environment to decide which type of blockchain is the most appropriate for the system to be developed, and determine what the advantages and inconveniences of each option.

Finally, the prototype of an authentication method has been developed using the Ethereum blockchain, which offers the possibility of working with Smart Contracts. Metamask has been used to manage the private keys of registered users in a secure and distributed way, being a key extension to communicate the browser with the blockchain. In addition, as an added value of this work, a file access control system has been implemented to go further in the blockchain operations.

Keywords: Security, authentication, blockchain, Ethereum, Smart Contract, Metamask

Índice general

1. Introducción	1
1.1. Motivación	1
1.2. Objetivos	2
1.3. Fases del desarrollo	2
1.4. Estructura de la memoria	3
2. Antecedentes y estado del arte	5
2.1. Antecedentes	5
2.2. Introducción a blockchain	6
2.3. Trabajos relacionados	6
2.4. Criptografía	7
2.4.1. Funciones hash	7
2.4.2. Infraestructura de clave pública y firma digital	8
2.4.3. Árboles de Merkle	8
2.5. Funcionamiento de blockchain	9
2.5.1. Bloques	10
2.5.2. Transacciones	11
2.5.3. Distribución	11
2.5.4. Tipos de blockchain	12
2.5.5. Algoritmos de consenso	13
3. Solución propuesta	15
3.1. Tecnologías	15
3.1.1. Ethereum	15
3.1.2. Truffle	16
3.1.3. Ganache	16
3.1.4. Web3	17
3.1.5. Drizzle	17
3.1.6. React	17
3.1.7. JSON Web Token	17
3.1.8. Metamask	18
3.2. Funcionamiento	19
3.2.1. Smart Contract: ControlAcceso	19
3.2.2. Fase de registro	19
3.2.3. Fase de validación	21
3.2.4. Fase de acceso	22
3.3. Sistema de control de acceso a ficheros	24
3.3.1. Usuarios	25

3.3.2. Recursos	26
3.3.3. Peticiones	27
3.3.4. Panel de recursos	28
3.4. Diseño de la blockchain	29
3.4.1. Privacidad de la blockchain	29
3.4.2. Algoritmo de consenso	30
4. Presupuesto	31
4.1. Personal y división de tareas	31
4.2. Presupuesto personal	31
4.3. Presupuesto material	31
4.4. Planificación temporal	32
5. Conclusiones y líneas futuras	33
5.1. Conclusiones	33
5.1.1. Método de autenticación	33
5.2. Líneas futuras	34
6. Summary and Conclusions	36
6.1. Conclusions	36
6.2. Future Works	37

Índice de Figuras

2.1. Métodos de identificación	5
2.2. Ejemplo función hash	8
2.3. Esquema firma digital	8
2.4. Ejemplo árbol de Merkle	9
2.5. Ejemplo de bloque simplificado	11
2.6. Esquema de verificación de firmas	12
2.7. Esquema de funcionamiento de una blockchain	12
3.1. Ejemplo de funcionamiento de Smart Contract	16
3.2. Truffle Suite	16
3.3. Estructura de un JSON Web Token	18
3.4. Estructura de un usuario	19
3.5. Crear cuenta en Metamask	20
3.6. Método POST para enviar datos de registro	20
3.7. Método AddUser del contrato inteligente	21
3.8. Método setUserState del contrato inteligente	21
3.9. Manejo de la aceptación de un usuario	22
3.10 Petición POST del reto	22
3.11 Firma del reto con web3	23
3.12 Verificación de la firma en el servidor	23
3.13 Decodificación del token	24
3.14 Nueva estructura del usuario	25
3.15 Estructura de los recursos	26
3.16 Interfaz añadir recursos	26
3.17 Transacción creada en Metamask	27
3.18 Función addResource del contrato inteligente	27
3.19 Estructura de las peticiones de recursos	28
3.20 Interfaz de las solicitudes	28
3.21 Panel de recursos	29
4.1. Tareas y duración estimada	32
4.2. Diagrama de Gantt	32

Índice de Tablas

4.1. Estimación de presupuestos 32

Capítulo 1

Introducción

1.1. Motivación

En un mundo cada día más digital, la ciberseguridad se ha convertido en un área de gran relevancia. Muchas grandes empresas se ven cada día comprometidas por vulnerabilidades informáticas, como el caso del ransomware Wannacry en 2017 [1], que se estima infectó a más de 300.000 máquinas en unos 150 países, dejando en jaque a empresas como Telefónica [2]. Uno de los ataques más populares son los que implican robo de identidad, también conocidos como spoofing [3]. Por ello, disponer de un método de autenticación seguro se ha convertido en algo primordial, en especial en servicios críticos o que tratan con datos sensibles.

Por otro lado, la blockchain se está situando como una de las tecnologías más disruptivas de los últimos años. La posibilidad de asegurar transacciones sin depender de una tercera parte de confianza hace que, más allá de su uso habitual en las criptomonedas, encuentre otras aplicaciones que se benefician de su naturaleza distribuida e inmutable.

En este documento se presenta un método de autenticación para mejorar la seguridad del aplicativo web que ofrece la empresa pública Gestión de Servicios para la Salud y Seguridad en Canarias (GSC) para dar información a las empresas de transporte sobre el movimiento de pacientes a través de la Mesa de Transporte Sanitario No Urgente [4]. La propuesta se apoya en blockchain y contratos inteligentes para registrar a los/as usuarios/as y llevar un control de sus accesos, así como para la firma digital que permite identificar unívocamente a cada usuario/a en el servidor web. Gracias al uso de blockchain, cada acceso se registra de forma inmutable. Como por privacidad no se ha podido mostrar su aplicación en la aplicación del GSC, para ilustrar el funcionamiento se ha desarrollado un sistema de acceso a recursos en el que los/as usuarios/as pueden añadir recursos, de forma que el resto de usuarios/as puedan solicitar el acceso a ellos.

El presente trabajo ha sido redactado tratando de hacer un uso inclusivo del lenguaje, utilizando allí donde fuera posible términos colectivos. Sin embargo, tal como se observa en el párrafo anterior, hay términos, como usuarios/as, para los que no existe análogo colectivo. Esa es la razón por la que, a nuestro pesar, utilizaremos en este trabajo, por economía lingüística, la palabra usuarios para referirnos a las personas que usen el sistema, independientemente de su género.

1.2. Objetivos

La meta del presente Trabajo Fin de Grado ha sido el desarrollo de un método de autenticación en un servidor web empleando la tecnología blockchain, de forma que se garantiza la seguridad y trazabilidad del mismo.

Con este método, el usuario puede iniciar sesión en el aplicativo web sin necesidad de introducir contraseña. La autenticación se realiza mediante la firma de un reto que le proporciona el servidor. Esta firma se verifica, así como si dicho usuario está registrado en la blockchain, para asegurar que, además de que el usuario es quien dice ser, tiene permitido acceder al servicio.

El objetivo principal de esta prueba de concepto ha sido entender los principales conceptos de la tecnología blockchain. Para ello se llevó a cabo un completo análisis de su aplicación actual, identificando el tipo de red, el nivel de privacidad y el algoritmo de consenso, entre otras características que determinan el funcionamiento de las cadenas de bloques.

Por último, se estudió la viabilidad del sistema y se analizaron tanto las ventajas como los inconvenientes, así como los costes asociados a su implementación.

1.3. Fases del desarrollo

La primera parte del desarrollo ha consistido en el estudio e investigación de la tecnología blockchain, su funcionamiento, conceptos básicos (algoritmos esenciales, tipologías de blockchains, minado...), así como todos los conceptos de criptografía necesarios para entender la tecnología. Así mismo, se han estudiado aplicaciones de la tecnología en diferentes campos, como el de la medicina para gestionar historiales clínicos, o en el mercado para la trazabilidad de productos.

La segunda fase ha conllevado el aprendizaje de las tecnologías em-

pleadas en el desarrollo de la solución. Entre ellas, se encuentran las relacionadas con blockchain, como la Suite Truffle, compuesta por 3 herramientas, Truffle, Ganache y Drizzle, las cuales explicaremos más adelante; además, se ha estudiado el lenguaje de programación Solidity, empleado para la implementación de Smart Contracts. Por otro lado, están las herramientas para el desarrollo de una aplicación web de prueba. En este caso se ha usado Node JS para el desarrollo del backend y React como framework de frontend.

A continuación, se ha planificado el desarrollo del prototipo, analizando las características que ha de tener la red de blockchain sobre la que se desplegará, planificando cada una de los componentes React del frontend, y cada ruta del backend. Tras dicha planificación, se ha procedido a su implementación. Se ha desarrollado un servidor de backend, que funciona de API para las operaciones que se comunican con la blockchain. En cuanto al frontend, se han implementado los componentes establecidos, así como la comunicación con las funciones requeridas de la blockchain.

Por último, se ha analizado el resultado final, indicando las ventajas e inconvenientes del mismo, así como las dificultades encontradas en el desarrollo. Se detalla un presupuesto aproximado de implementación, incluyendo los costes del despliegue de la blockchain, y una planificación temporal del proyecto.

1.4. Estructura de la memoria

La presente memoria consta de seis capítulos. El primero de ellos introduce el contenido del resto del documento, presentando la motivación, los objetivos propuestos, las fases del desarrollo para alcanzarlos, y, por último, el apartado actual, un análisis de la estructura de la memoria.

El segundo capítulo incluye un marco teórico tanto de blockchain como de los métodos criptográficos empleados. En este no solo se presentan los antecedentes y los trabajos relacionados, sino que también se explica el funcionamiento de la tecnología blockchain, desgranando cada uno de sus componentes, así como los conceptos criptográficos, vitales tanto para entender las cadenas de bloques como para sus otros usos dentro de la solución propuesta.

En tercer lugar encontramos el punto central del presente documento, que es una completa descripción de la solución propuesta. Se explican las tecnologías empleadas para el desarrollo, el diseño de la solución y su funcionamiento. Además se explicarán las partes esenciales del código

empleado.

A continuación se desglosa un presupuesto estimado del desarrollo de este proyecto. En él se contemplan tanto los costes de desarrollo como los de despliegue de la red de blockchain. También se hace una estimación temporal del proceso.

Por último se incluyen algunas conclusiones, comentando ventajas, inconvenientes, mejoras, y dificultades en el desarrollo. Se detallan también posibles líneas futuras tanto del proyecto como de la tecnología blockchain.

Capítulo 2

Antecedentes y estado del arte

En este capítulo se introduce el estado actual de los métodos de autenticación en general, y del empleo actualmente por GSC actualmente en su aplicativo. Se explica detalladamente el funcionamiento de la tecnología blockchain, así como los conceptos criptográficos que intervienen en ella.

2.1. Antecedentes

Actualmente son varias las alternativas disponibles para la autenticación [5] de un usuario en un servidor. Podemos clasificarlos en cuatro campos, atendiendo a qué se utiliza para verificar que el usuario es quien dice ser (ver Figura 2.1).

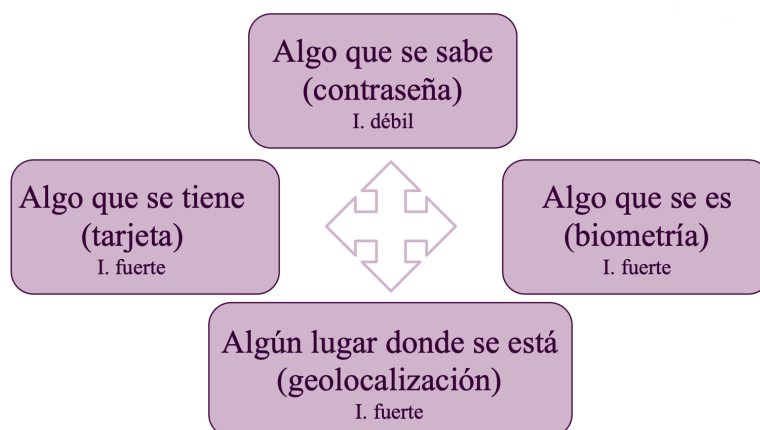


Figura 2.1: Métodos de identificación

La primera aproximación es confiar en algo que sabe el usuario, por ejemplo, una contraseña. En segundo lugar, podemos considerar algo que es el usuario. Aquí entrarían los métodos biométricos, como la huella dactilar. También podemos emplear algo que el usuario tenga, ya sea una tarjeta o un certificado. Por último, tenemos el lugar donde está el usuario, que emplea técnicas de geolocalización.

En el caso del aplicativo web actual del GSC, la identificación de los equipos que acceden al mismo requiere habilitar las IPs públicas de la empresa adjudicataria del servicio para que acceda a través del Firewall perimetral de la empresa pública GSC hacia sus servidores internos. Dicho entorno se está migrando a un sistema que no requiere el acceso directo de la empresa externa hacia los servidores de backend de GSC, colocando un servidor web alojado externamente que actuará de frontal de acceso, de forma que dicho servidor será el único que tendrá acceso hacia el backend.

2.2. Introducción a blockchain

Blockchain surge en 2008 cuando, bajo el pseudónimo “Satoshi Nakamoto”, se publica el whitepaper “Bitcoin: a peer-to-peer electronic cash system” [6]. En ese documento se presentó un método de pago online sin el problema del doble gasto y donde no intervienen terceras partes de confianza que verificasen las transacciones. El problema del doble gasto, inherente a las monedas virtuales, permitía que estas pudiesen replicarse o usarse más de una vez y, por ello, en las operaciones financieras, se recurre a intermediarios de confianza que garantizan que esto no ocurra. Para evitar depender de estos agentes externos, se planteó en ese paper emplear una base de datos distribuida, donde cada transacción se asegura con métodos criptográficos y queda registrada permanentemente en la misma. A esta tecnología se le dio el nombre de blockchain.

Actualmente se enmarca dentro de las Tecnologías de Registro Distribuidas (Distributed Ledger Technologies, DLT) [7], siendo predecesora de las mismas. En el caso particular de blockchain, como su propio nombre indica, presenta una estructura de bloques enlazados, en los que no solo van los datos inherentes a la operación que se desee realizar, sino que se incluyen una serie de metadatos que permiten comprobar si ha habido alguna modificación en algún bloque de la cadena y llevar un control de cada cambio realizado, asegurando la inmutabilidad de la misma.

2.3. Trabajos relacionados

Actualmente existen varios proyectos conocidos que se apoyan en la tecnología blockchain para autenticación. Hay que tener en cuenta que la mayoría se centran en la privacidad, puesto que están pensados para su uso en redes públicas en las que todo usuario puede leer los datos de la cadena de bloques.

Uno de los trabajos más relevantes es el los Identificadores Descentralizados (DIDs) [8], que son unos archivos que recogen una serie de datos que se emplean para identidad digital verificable y auto-soberana. Este sistema es utilizado por diferentes métodos de identificación distribuida y podrían llegar a estandarizarse en el futuro.

Uno de estos sistemas que se apoyan en los DIDs es uPort [9], la solución más popular para la identidad soberana. Propone que cada usuario sea dueño de su propia identidad sin necesidad de terceras partes de confianza, es decir, sin depender de una entidad que nos emita los certificados necesarios para identificarnos [10].

Por otro lado, en el ámbito del Internet de las Cosas (IoT), se han desarrollado diferentes propuestas de métodos de control de acceso o autenticación empleando blockchain. Entre estas podemos encontrar FairAccess [11], un framework de autenticación en dispositivos IoT totalmente descentralizado donde se mantiene la privacidad de las identidades de cada dispositivo, y permitiendo a los usuarios empoderarse de sus datos. Para esto, emplea las transacciones en la cadena de bloques para conceder, conseguir, delegar y revocar el acceso a dichos dispositivos.

2.4. Criptografía

Para comprender cómo funciona blockchain y conocer su seguridad, en gran parte basada en la inmutabilidad de sus datos y la integridad de los mismos, es necesaria una introducción de los conceptos criptográficos en los que se apoya esta tecnología.

2.4.1. Funciones hash

Una función hash [12] es una operación criptográfica que genera una cadena arbitraria a partir de un conjunto de datos de entrada (ver Figura 2.2). Normalmente, esta cadena es de longitud fija, establecida por el algoritmo que se emplee. En el caso de blockchain, se utiliza SHA-256 [13], donde el tamaño de la cadena resultante, también denominada hash, es de 256 bits.

Algunas características de SHA-256, como función hash criptográfica, son:

- su determinismo, es decir, que el hash de dos entradas exactamente iguales, siempre será el mismo;

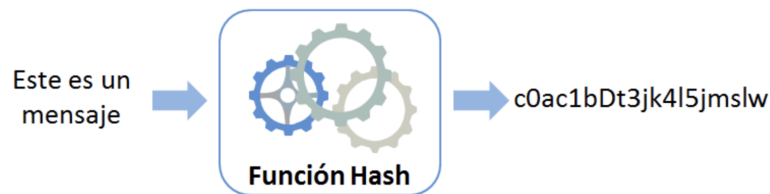


Figura 2.2: Ejemplo función hash

- su dificultad para ser revertida, pues hasta ahora no se ha encontrado método para obtener los datos iniciales a partir de su hash; y
- su resistencia a las colisiones, pues es muy poco probable que dos cadenas diferentes tomen el mismo hash utilizando este algoritmo.

Gracias a estas características, el hash se emplea dentro de la cadena de bloques para asegurar la integridad de los datos, pues el hash de cada bloque es almacenado en el siguiente de la cadena, de forma que, si un bloque es modificado, el hash del mismo no coincidirá con el que contiene el siguiente.

2.4.2. Infraestructura de clave pública y firma digital

Cada usuario de la blockchain dispone de una clave pública [14], la cual es compartida con el resto de nodos de la red, e identifica al mismo, y una clave privada, a la que solo él tiene acceso. Ésta se utiliza para firmar cada transacciones y verificar que se han realizado por dicho usuario. Es decir, cada vez que se añade una transacción a un bloque, esta es firmada por el usuario con su clave privada (ver Figura 2.3).



Figura 2.3: Esquema firma digital

El resto de usuarios de la red pueden verificar esta transacción empleando la correspondiente clave pública, de modo que si se cambia cualquier dato de la misma, la firma se modifica y se detecta el fraude.

2.4.3. Árboles de Merkle

Los árboles de Merkle [15] son estructuras de datos donde cada nodo interno (que no es hoja) almacena un hash de su nodo hijo, de forma

que estos se van concatenando hasta llegar al nodo raíz (ver Figura 2.4). El hash del nodo raíz, al estar ligado al de todos sus hijos, permite la verificación de los mismos de forma segura y eficiente. Blockchain utiliza los árboles de Merkle para almacenar las transacciones dentro de los bloques.

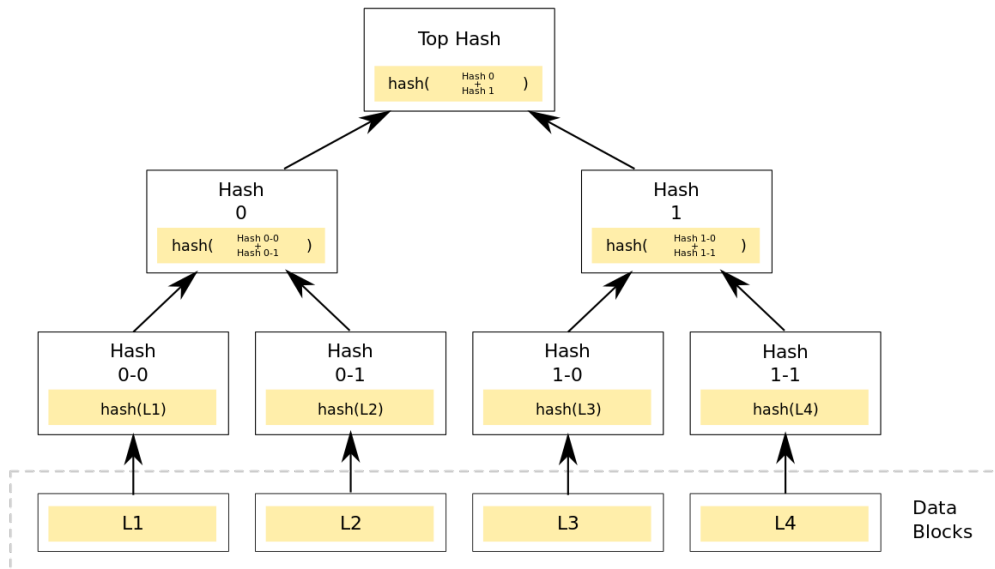


Figura 2.4: Ejemplo árbol de Merkle

2.5. Funcionamiento de blockchain

A diferencia de los sistemas tradicionales de almacenamiento de datos, en los cuales toda la información está centralizada y sólo existe una fuente de la verdad, blockchain presenta una base de datos distribuida, de modo que cada participante de la red mantiene una copia de la misma. Esta base de datos se compone de bloques enlazados, los cuales, entre otras cosas, almacenan los datos que queremos preservar y una referencia al bloque anterior de la cadena, de forma que si cualquier bloque se modifica, el cambio se propaga a los siguientes, detectándose dicho cambio.

Gracias a esto, y a las operaciones criptográficas sobre las que se sustenta, se garantiza la seguridad e inmutabilidad de los datos almacenados, además de que cada operación realizada sobre la blockchain queda registrada permanentemente, permitiendo un control sobre las mismas.

2.5.1. Bloques

Los bloques de la blockchain, además de las transacciones realizadas, almacenan una serie de datos esenciales para el funcionamiento de la misma [16]. Estos datos varían entre las diferentes cadenas de bloques, de forma general podemos destacar los siguientes campos (ver Figura 2.5):

- **Height:** Indica la altura a la que se encuentra el bloque dentro de la cadena, es decir, indica el número de bloque en la blockchain.
- **Timestamp:** Marca temporal que indica fecha y hora en la que se ha creado el bloque.
- **Datos o transacciones:** Operaciones realizadas en la blockchain. Además del envío de algún token de un usuario a otro, pueden representar varias cosas, como pueden ser operaciones con los smart contracts. Cada transacción está firmada por el usuario que la realiza. Estas se almacenan en forma de árbol de Merkle para facilitar la verificación de las mismas.
- **Hash:** El resultado de aplicar una función hash al bloque. Este, dependiendo del tipo de blockchain, tendrá determinado requisito, que puede ser que comience con un número determinado de ceros. Gracias a esto se regula la dificultad de minado.
- **Hash previo:** Es el hash del bloque inmediatamente anterior en la cadena de bloques.
- **Nonce:** En las redes de blockchain que utilizan como algoritmo de consenso la prueba de trabajo, el nonce es el número que se ha de calcular para que, al aplicar la función hash sobre el bloque, se cumpla el requisito del hash establecido por la dificultad de minado.

Como cada bloque almacena una copia del hash anterior, si un bloque es modificado, el siguiente ya no será válido, pues el hash almacenado no coincidirá. En caso de que un atacante quiera realizar un cambio, tendría que modificar el bloque que le interesa y todos los siguientes, con el consiguiente coste computacional que esto conlleva.

Este coste computacional, asociado al algoritmo de consenso de prueba de trabajo, viene de calcular el previamente mencionado nonce, lo que comúnmente se conoce como minado. Para que un bloque sea incluido en la blockchain, primero ha de minarse, es decir, hallar el valor del nonce que

Block: # 1

Nonce: 139358

Tx:

\$	25.00	From:	Darcy	->	Bingley
\$	4.27	From:	Elizabeth	->	Jane
\$	19.22	From:	Wickham	->	Lydia
\$	106.44	From:	Lady Catherine	->	Collin
\$	6.42	From:	Charlotte	->	Elizabeth

Prev: 00

Hash: 00000c52990ee86de55ec4b9b32beefd745d71675dc

Mine

Figura 2.5: Ejemplo de bloque simplificado

hace que el hash de dicho bloque comience con el número determinado de ceros establecido por la dificultad de minado actual del bloque.

2.5.2. Transacciones

Como se ha mencionado previamente, las transacciones dentro del bloque están firmadas por el usuario y se almacenan en forma de árbol de Merkle para agilizar la verificación de las mismas (ver Figura 2.6). Esto hace que si un usuario malintencionado quiere alterar una, aunque consiga modificar toda la cadena de bloques para que estos sean válidos, minando cada uno de ellos, este cambio será detectado en las propias transacciones por no verificarse la firma.

2.5.3. Distribución

Además de todas las medidas de seguridad expuestas, tenemos que la cadena de bloques se encuentra replicada en cada nodo de la red. Cualquier cambio se puede comprobar en el resto de copias de la cadena (ver Figura 2.7), tomando como cadena válida la que coincida en más nodos como

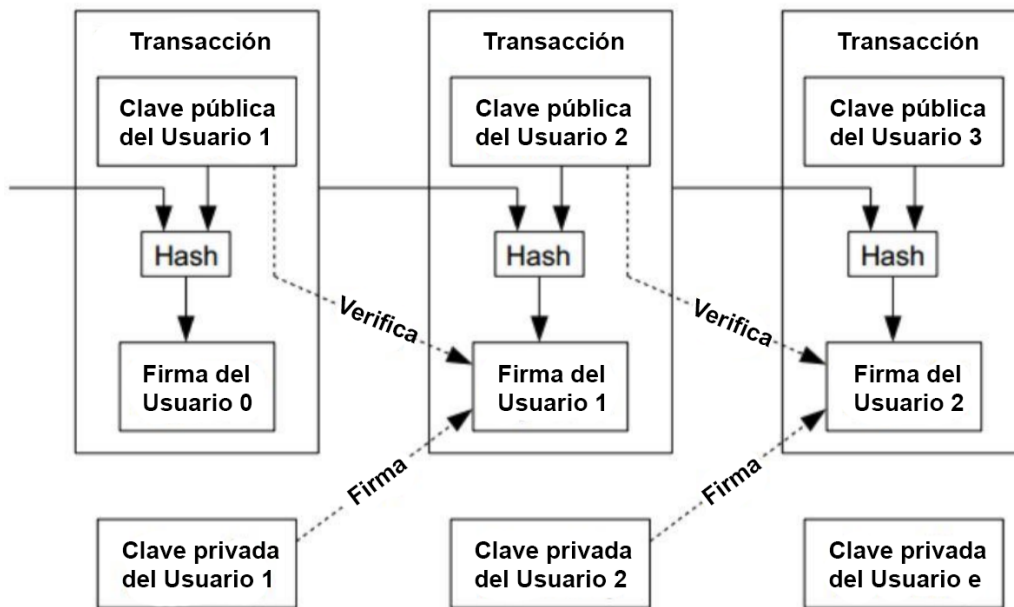


Figura 2.6: Esquema de verificación de firmas

correcta.

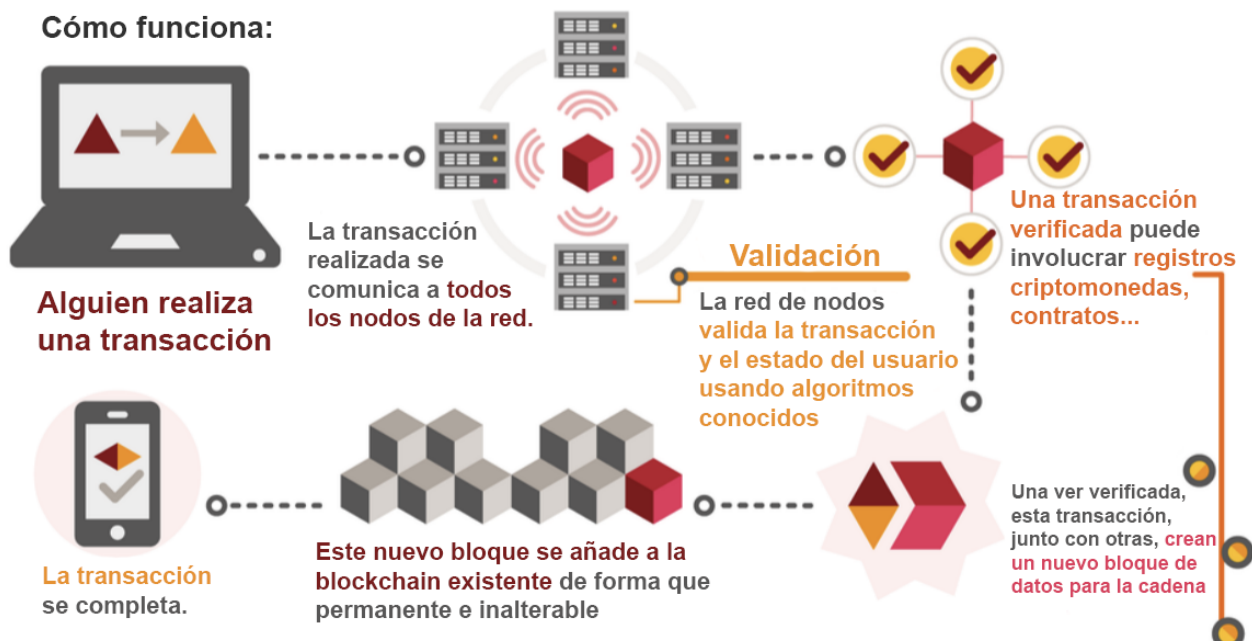


Figura 2.7: Esquema de funcionamiento de una blockchain

2.5.4. Tipos de blockchain

Podemos diferenciar entre tres tipos de blockchain, atendiendo a la accesibilidad que tienen los usuarios [17].

En primer lugar tenemos las *blockchain públicas*, las cuales son accesibles por cualquier usuario. Es decir, cualquier usuario podría participar en la red y leer las transacciones realizadas en la cadena. Bitcoin o Ethereum

son ejemplos de estas.

Por otra parte, las *blockchain privadas* establecen que sólo los usuarios escogidos formen parte de la red, y además son los únicos que pueden leer las transacciones. Hyperledger y R3 son los ejemplos más conocidos de las cadenas de bloques privadas.

Por último, las *blockchain híbridas* combinan los conceptos de pública y privada, de modo que sólo determinados usuarios pertenecen a la red y pueden realizar transacciones, pero estas son públicas y accesibles por cualquier usuario. Un ejemplo de este tipo de blockchain es BigchainDB.

2.5.5. Algoritmos de consenso

Los algoritmos de consenso son mecanismos que permiten confirmar las transacciones, garantizando su integridad e inmutabilidad. Cada nodo de la red tiene que aplicar el algoritmo de consenso establecido para llegar a un acuerdo con el resto de nodos, de forma que compruebe que el nodo y las transacciones del mismo son válidas. Una vez se hayan realizado estas comprobaciones, se añade el bloque a la cadena y se replica en el resto de nodos.

El algoritmo de consenso más empleado es la prueba de trabajo (Proof of Work) [18]. Fue el primero que surgió, y aún se utiliza en la gran mayoría de redes de blockchain públicas, como son el caso de Bitcoin y de Ethereum. La explicación previa del funcionamiento de la cadena de bloques se ha realizado teniendo en cuenta este algoritmo, donde cada nodo de la red ha de verificar cada bloque, calculando el nonce atendiendo a la dificultad de minado. El gran inconveniente de la prueba de concepto es el gran coste computacional que conlleva, siendo este cada vez mayor a medida que crece la red, afectando a la escalabilidad de la misma.

Con el objetivo de disminuir este coste computacional surge la prueba de participación (Proof of Stake) [18], donde se busca llegar al consenso por medio de operaciones computacionalmente más sencillas. Esto se consigue porque se elige al creador del nodo de la red de manera determinista, en función de la participación o riqueza que este haya acumulado en la red.

Por otro lado, tenemos la prueba de autoridad (Proof of Authority) [19]. Este algoritmo está pensado para su uso en redes privadas, y su rendimiento es más eficiente que el de la prueba de trabajo o la prueba de participación. En este se tienen en cuenta las identidades reales de los dueños de los nodos, de forma que se elige un conjunto de estos en los que se confía para que sean los únicos que puedan añadir bloques en

la blockchain. Este algoritmo no solo es el más eficiente, sino que es el que proporciona más escalabilidad a la red y es imprescindible en redes privadas donde la velocidad de las operaciones sea vital.

Capítulo 3

Solución propuesta

El método de autenticación diseñado se basa en el concepto de firma digital.

Se ha empleado un wallet distribuido como Metamask para la gestión de cuentas de usuario y claves privadas.

Se ha desarrollado un servidor web utilizando Node que proporcionará una API para el registro e inicio de sesión de los usuarios. Éste se comunicará con la blockchain utilizando la librería Web3.

Por otro lado, se ha implementado el frontend utilizando el framework React, y se ha apoyado en Drizzle para facilitar las funciones que requieren interactuar con la blockchain o con Metamask.

Apoyándose en esas mismas tecnologías, se ha desarrollado un sistema de control de acceso a ficheros para contextualizar el funcionamiento del sistema de autenticación.

Se ha desplegado una blockchain de pruebas con Ganache, y para la migración y el testeo de los contratos inteligentes se ha empleado el framework Truffle.

3.1. Tecnologías

El primer paso en el diseño de la solución fue determinar las herramientas a emplear en el desarrollo. Entender qué papel juegan estas en la solución es fundamental, sobre todo cuando trabajamos con una tecnología como blockchain.

3.1.1. Ethereum

Ethereum [20] es una plataforma open source descentralizada y basada en una blockchain, que permite ejecutar programas en un entorno controlado. Esto se consigue gracias a que provee una máquina virtual, la

Ethereum Virtual Machine (EVM), la cual garantiza que el código que ejecutemos en la blockchain funcione por igual en todos los nodos de la red. Al código que se ejecuta en la blockchain se le denomina Contrato Inteligente (Smart Contract) (ver Figura 3.1), y se implementa en el lenguaje Solidity.



Figura 3.1: Ejemplo de funcionamiento de Smart Contract

La posibilidad de desarrollar contratos inteligentes es por lo que se ha utilizado una blockchain de Ethereum para nuestro desarrollo. Además, la gran mayoría de herramientas y frameworks se basan en esta por las facilidad que dan a los desarrolladores.

3.1.2. Truffle

Truffle [21] es un entorno de desarrollo para Ethereum, que facilita el desarrollo de Smart Contracts y de aplicaciones que se comuniquen con la blockchain, denominadas DApps. Truffle es una de las tres herramientas que pertenecen a la Suite de Truffle (ver Figura 3.2).



Figura 3.2: Truffle Suite

Gracias a este entorno podremos testear los contratos inteligentes, compilarlos y migrarlos a la blockchain de una manera cómoda y sencilla.

3.1.3. Ganache

Dentro de las herramientas que incluye la Suite de Truffle, nos encontramos Ganache [22]. Gracias a esta, podemos lanzar una blockchain de

pruebas de forma local, donde nos mostrará una serie de usuarios que podremos utilizar para simular una red distribuida. Además, dentro de esta podremos ver las transacciones realizadas, los bloques de la cadena, los contratos migrados, y todo tipo de registro que nos ayude a comprobar errores.

3.1.4. Web3

Web3 [23] es una colección de librerías que permiten interactuar con un nodo de una blockchain de Ethereum, utilizando una conexión HTTP, WebSocket o IPC.

Se ha utilizado web3.js, la versión de JavaScript, pues el servidor web que queremos conectar con la blockchain ha sido desarrollado con NodeJS.

3.1.5. Drizzle

Es la tercera y última herramienta de la Suite de Truffle. Drizzle [24] es una colección de librerías de frontend que facilitan el desarrollo de interfaces para comunicar las aplicaciones web con la blockchain. Internamente emplea web3.js, pero su uso es más sencillo.

3.1.6. React

React [25] es una librería de JavaScript desarrollada por Facebook. Está orientada a crear interfaces de usuario de una sola página. Además, permite llevar un control de los datos que cambian en el tiempo, actualizándolos sin tener que recargar la página. Todo esto, apoyándose en un modelo de componentes que facilita y hace muy cómodo el desarrollo de la interfaz.

Además de por todas las ventajas ya mencionadas, la posibilidad de incluir componentes de react en páginas no desarrolladas con React hace que sea idóneo para nuestro proyecto.

3.1.7. JSON Web Token

JSON Web Token (JWT) [26] es un estándar propuesto para la creación de tokens de acceso, de forma que permiten la transmisión de información, normalmente referente a la identidad o privilegios, de forma segura. Esta información puede ser verificada al estar firmada.

En la estructura de un JWT (ver Figura 3.3) pueden distinguirse tres partes: la cabecera, la carga útil y la firma (ver Figura 3.3).

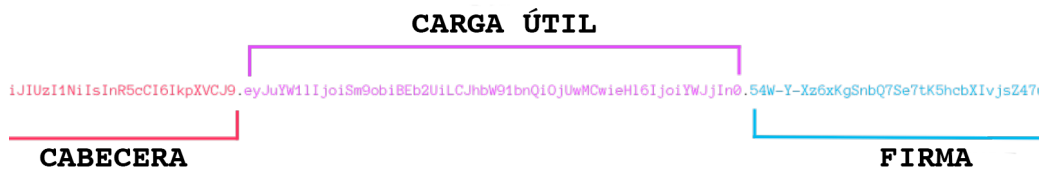


Figura 3.3: Estructura de un JSON Web Token

La cabecera (header) contiene dos atributos: el tipo de token (typ), el cuál se indica como JWT; y el algoritmo de firma digital empleado (sig).

La carga útil (payload) contiene declaraciones referentes a la entidad (normalmente el usuario), y otra información adicional.

La firma (signature) se crea a partir de la cabecera y la carga útil codificadas y un secreto, aplicando el algoritmo indicado en la cabecera.

3.1.8. Metamask

Metamask [27] es una extensión de navegador disponible para navegadores basados en Chromium (Google Chrome, Vivaldi, Opera, etc.) y Mozilla Firefox. Ofrece una cartera o wallet de Ethereum, es decir, un almacén desde el cual podemos manejar Ether, la criptomoneda empleada por Ethereum. Esta cartera funciona de forma similar a una cuenta bancaria, permitiendo la gestión segura de cada transacción.

Aunque el uso más común de esta extensión es para gestionar operaciones monetarias, cabe destacar la posibilidad que brinda para almacenar claves privadas de forma segura. Metamask permite crear usuarios, de forma que les genera un par de claves pública-privada, almacenando esta última de forma segura y distribuida, siguiendo los principios de la tecnología blockchain.

Además, esta extensión inyecta la librería Web3 en nuestro navegador, de forma que podamos interactuar con aplicaciones que se comuniquen con la blockchain a la que tengamos conectada Metamask.

El principal motivo de usar Metamask es la facilidad que brinda para almacenar las claves privadas, pudiendo usar las funciones disponibles en Web3 para realizar la firma digital de mensajes, imprescindible para el método de autenticación desarrollado. Por otro lado, en el sistema de ficheros implementado, cada nodo usuario interactúa directamente con la blockchain, y es esta misma extensión la que gestiona cada transacción que el usuario realice.

3.2. Funcionamiento

El proceso de autenticación desarrollado presenta tres fases bien diferenciadas: el registro, la validación y el acceso.

3.2.1. Smart Contract: ControlAcceso

Con el fin de almacenar los usuarios registrados en la blockchain, se ha desarrollado el contrato inteligente denominado *ControlAcceso*. En este se ha creado una estructura *User* (ver Figura 3.4), la cual representa a los usuarios. Allí se contempla la dirección o clave pública del usuario, el nombre introducido, un identificador, un timestamp del momento en el que se ha creado el usuario, y un estado cuyos valores pueden ser:

- rechazado, indicado con un 0;
- aceptado, indicado con un 1; y
- pendiente, indicado con un 2.

```
struct User {  
    address userAddress;  
    string username;  
    uint id;  
    bool admin;  
    uint creationDate;  
    uint state;  
    uint nonce;  
}
```

Figura 3.4: Estructura de un usuario

Además, incluye el atributo denominado *nonce*, el cual es un número aleatorio que irá variando a medida que el usuario se autentique en el sistema, pues este es el mensaje que ha de firmar el usuario en cada acceso, para verificar su identidad.

3.2.2. Fase de registro

En la fase de registro, el usuario ha de crearse una cuenta en Metamask. Para ello, tendrá que tener instalada la extensión en su navegador. Se

abre y, en el menú de gestión de cuentas, está la opción de crear una (ver Figura 3.5). Al crear una cuenta, se generan una clave privada, la cual se almacena en la extensión, y una pública.

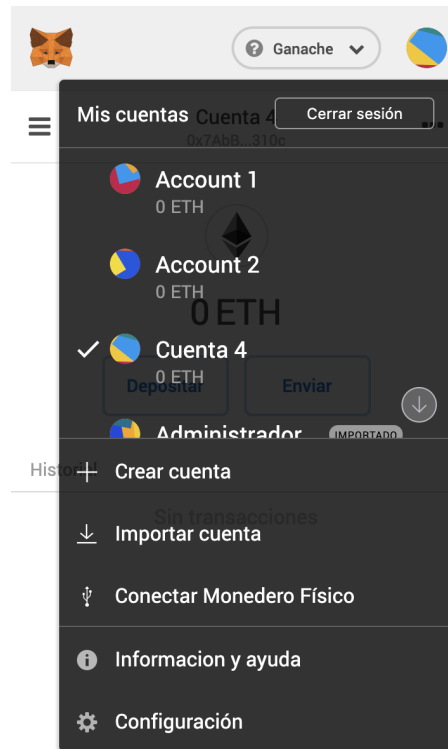


Figura 3.5: Crear cuenta en Metamask

Con esta clave pública se rellena el formulario de registro, el cual se envía al servidor por medio de una petición POST de HTTP (ver Figura 3.6). Los métodos de petición de HTTP se definen con el fin de determinar qué se desea realizar con un recurso determinado. La peculiaridad de este método es que los datos están ocultos en el cuerpo de la petición, a diferencia que con el método GET, el cual muestra los datos de la petición en la URL.

```
fetch('/api/register', {
  method: 'POST',
  headers: {
    'Accept': 'application/json',
    'Content-Type': 'application/json',
  },
  body: JSON.stringify({
    hostAddress: this.newUserAddress.current.value,
    hostname: this.newUserName.current.value
  })
});
this.input
```

Figura 3.6: Método POST para enviar datos de registro

Al recibir el formulario de registro, el servidor lo almacena en la blockchain, junto a un número aleatorio el cual se utiliza como mensaje a firmar cuando el usuario se quiera autenticar. Para almacenar al usuario en la cadena de bloques, se utiliza la función del contrato inteligente *AddUser* (ver Figura 3.7). Este número se actualiza cada vez que se utilice.

```
function addUser(address _userAddress, string memory _username, bool _admin, uint _nonce) public onlyAdmin {
    require(checkUsername(_username) && addressToUser[_userAddress].id == 0, "This username already exist");
    userCount++;
    actualUserCount++;
    User memory _newUser = User(_userAddress, _username, userCount, _admin, now, 2, _nonce);
    addressToUser[_userAddress] = _newUser;
    usersArray.push(_userAddress);
    emit CreateUser(_userAddress, _username, userCount, _admin, 2);
}
```

Figura 3.7: Método AddUser del contrato inteligente

Cada usuario creado en la blockchain tiene un atributo estado o state, representado por un entero que puede tomar tres posibles valores: 2 (pendiente), 1 (aceptado) y 0 (rechazado.) Cuando se registra al usuario en la cadena de bloques se le asigna un estado de pendiente (2), de forma que solo los usuarios con un estado de aceptado puedan iniciar sesión.

3.2.3. Fase de validación

Cuando un usuario envía sus datos, los administradores ven esta petición en su panel de administración. En este aparecen todos los usuarios que hayan realizado una solicitud de registro. Aquellos que estén en un estado de pendiente, podrán ser aceptados o rechazados. Los usuarios que hayan sido aceptados podrán ser eliminados del sistema. Los usuarios rechazados se mostrarán para llevar una trazabilidad completa de las peticiones.

La función que gestiona los cambios de estados de los usuarios en el contrato inteligente es *setUserState* (ver Figura 3.8), la cual recibe como parámetros la clave pública (o address) del usuario y el estado que se le quiere asignar. Esta función solo puede ser ejecutada por administradores.

```
function setUserState(address _userAddress, uint _newState) public onlyAdmin {
    addressToUser[_userAddress].state = _newState;
    emit ChangeUserState(_userAddress, _newState);
}
```

Figura 3.8: Método setUserState del contrato inteligente

Cuando un usuario es aceptado, desde el propio frontend del administrador se realiza una transacción en la blockchain, de tal forma que se modifica el estado del usuario de pendiente (2) a aceptado (1). En caso de que sea rechazado, la función es similar, pero poniendo el estado del usuario a rechazado (0) (ver Figura 3.9).

```
handleAccept(id, event) {
  event.preventDefault();
  const { drizzle, drizzleState } = this.props;
  const contract = drizzle.contracts.ControlAcceso;
  const stackId = contract.methods.setUserState.cacheSend(id[1], 1, { gas: 300000 });
  this.setState({ stackId });
}
```

Figura 3.9: Manejo de la aceptación de un usuario

Al usuario se le notificará que su registro se ha validado. Esto se podrá realizar por medio de un email a la cuenta introducida en el formulario, siendo posible la automatización de este proceso.

3.2.4. Fase de acceso

Una vez el administrador valida al usuario, este podrá autenticarse en el sistema. Para ello no necesitará introducir contraseñas, sino que bastará con hacer clic sobre el botón de inicio de sesión. Al hacerlo, solicita por medio de una petición POST al servidor web un reto (ver Figura 3.10), que el usuario deberá firmar para verificar su identidad. Este reto es el número aleatorio almacenado junto al nombre del usuario en la blockchain al registrarse, que se modificará una vez que el usuario lo haya firmado.

```
fetch('/api/getNonce', {
  method: 'POST',
  headers: {
    Accept: 'application/json',
    'Content-Type': 'application/json',
  },
  body: JSON.stringify({
    a: address
  })
})
.then((
  ...
```

Figura 3.10: Petición POST del reto

En la petición POST donde se reclama el reto, lo único que se indicará es la clave pública del usuario. Al recibir el reto, se procederá a firmarlo. Para ello se utiliza la función de Web3 `web3.eth.personal.sign` (ver Figura 3.11), la cual recibe como parámetro el mensaje, y se encarga de tomar de Metamask la clave privada de la cuenta para realizar la firma.

```
web3.eth.personal.sign(web3.utils.utf8ToHex(`${result1.nonce}`), address, 'testpassword')
  .then((signedMessage) => {
    fetch('/api/login', {
      method: 'POST',
      headers: {
        Accept: 'application/json',
        'Content-Type': 'application/json',
      },
      body: JSON.stringify({
        m: message,
        sm: signedMessage,
        a: address
      })
    })
  })
  .then((
    ...
```

Figura 3.11: Firma del reto con web3

El servidor recibirá el reto firmado y se encargará de verificar la firma. Para esto, lo primero que hace es comprobar que el usuario tenga un estado de aceptado en la blockchain, en caso contrario, el usuario no sería válido y no podrá autenticarse. A continuación, se emplea la función de Web3 `web3.eth.accounts.recover` (ver Figura 3.12), la cual recibe como parámetros el reto y el mensaje firmado, y devuelve la clave pública del usuario que ha firmado el mensaje. Si la clave obtenida en este punto coincide con la del usuario, la autenticación será válida, actualizará el próximo reto del usuario en la blockchain y devolverá un JSON Web Token.

```
web3.eth.personal.sign(web3.utils.utf8ToHex(`${result1.nonce}`), address, 'testpassword')
  .then((signedMessage) => {
    fetch('/api/login', {
      method: 'POST',
      headers: {
        Accept: 'application/json',
        'Content-Type': 'application/json',
      },
      body: JSON.stringify({
        m: message,
        sm: signedMessage,
        a: address
      })
    })
  })
  .then((
    ...
```

Figura 3.12: Verificación de la firma en el servidor

La gestión de JSON Web Token se ha implementado con la librería *jwt-simple*. Con esta se ha desarrollado una función para crear el token y otra para decodificarlo (ver Figura 3.13).

```
function decodeToken(token) {
  const decoded = new Promise((resolve, reject) => {
    try {
      const payload = jwt.decode(token, config.secret);
      if (payload.exp <= moment().unix()) {
        reject ({
          status: 401,
          message: 'El token ha expirado'
        });
      }
      // console.log(`Sub: ${payload.sub}`)
      resolve(payload.sub);
    } catch (err) {
      reject ({
        status: 500,
        message: 'invalid token'
      });
    }
  });
  return decoded;
}
```

Figura 3.13: Decodificación del token

3.3. Sistema de control de acceso a ficheros

Puesto que por motivos de privacidad del GSC no se ha podido mostrar la aplicación web para la que se ha desarrollado la autenticación, se ha desarrollado un sistema de control de acceso a ficheros o recursos almacenados en un servidor. Este, al igual que el método de autenticación propuesto, emplea la tecnología blockchain, de forma que cada recurso se almacena en la cadena de bloques, así como las peticiones, los permisos asignados, etc.

En este sistema, los usuarios pueden añadir recursos, representados,

para simplificar, por su nombre y descripción. El resto de usuarios los verán desde el panel de recursos, y podrán solicitar acceso a esos recursos. Cuando se solicita el acceso a un recurso, el dueño del mismo, es decir, quien lo ha creado en el sistema, verá esta petición y podrá aceptarla o rechazarla.

3.3.1. Usuarios

El modelo de usuarios, descrito en la autenticación e implementado en el contrato inteligente, se ha ampliado añadiendo tres nuevos atributos (ver Figura 3.14):

- *resourcesCount*, el cual indica la cantidad de recursos que este usuario ha añadido;
- *requestsCount*, que indica la cantidad de peticiones de acceso que otros usuarios han hecho a un recurso de este usuario; y por último,
- *allowedResources*, vector que contiene los recursos a los que este usuario tiene acceso.

```
struct User {
    address userAddress;
    string username;
    uint id;
    bool admin;
    uint creationDate;
    uint state;
    uint nonce;
    uint resourcesCount;
    uint requestsCount;
    uint[] allowedResources;
}
```

Figura 3.14: Nueva estructura del usuario

3.3.2. Recursos

En el contrato inteligente se ha creado una nueva estructura para representar los recursos (ver Figura 3.15). Siguiendo esta estructura, cada recurso contiene:

- el *id*, un entero que se utilizará de identificador del recursos;
- el atributo *emphname*, es decir, su nombre;
- *description*, la descripción que el creador del recurso le ha otorgado;
- *creationDate*, sello temporal del momento en que se creó el recurso; y
- *creator* y *creatorName*, haciendo referencia a la clave pública y al nombre del creador del recurso, respectivamente.

```
struct Resource {  
    uint id;  
    string name;  
    string description;  
    uint creationDate;  
    address creator;  
    string creatorName;  
}
```

Figura 3.15: Estructura de los recursos

Para añadir recursos se ha creado una sencilla interfaz (ver Figura 3.16), en la que se solicita el nombre del mismo y una descripción. Esta información no se envía por medio de una petición POST al servidor, sino que directamente se comunica con la blockchain, creando una nueva transacción, la cual deberemos aceptar en Metamask (ver Figura 3.17).

Añadir recurso

Nombre	Camillas
Descripción	Description
<input type="button" value="Añadir"/>	

Figura 3.16: Interfaz añadir recursos

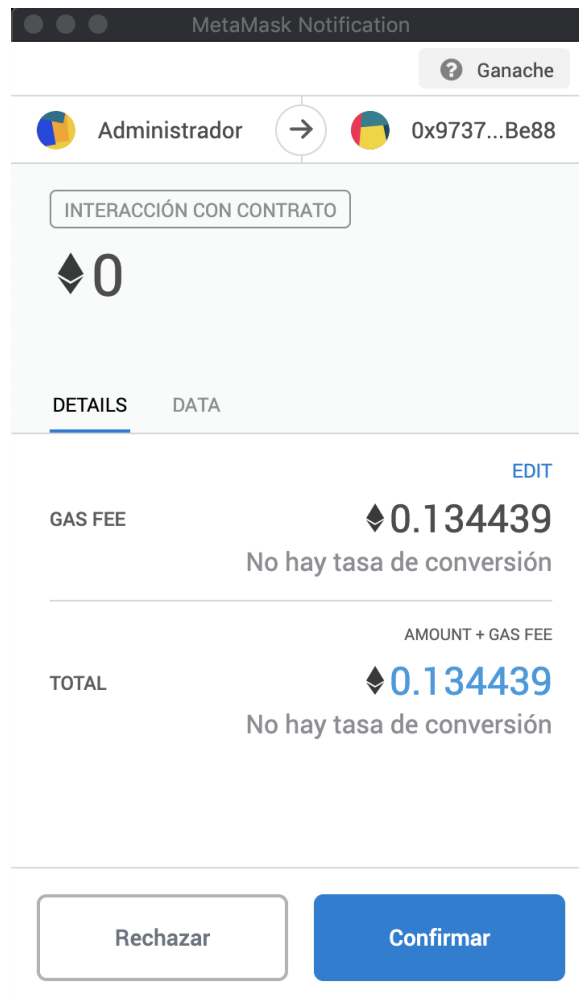


Figura 3.17: Transacción creada en Metamask

Esta transacción ejecuta la función del contrato inteligente `emphaddResource` (ver Figura 3.18), la cual toma los datos necesarios y crea el recurso de acuerdo con la estructura definida previamente.

```
function addResource(string memory _name, string memory _description) public {
    resourceCount++;
    actualResourceId++;
    Resource memory _newResource = Resource(actualResourceId, _name, _description, now, msg.sender, addressToUser[msg.sender].username);
    idToResource[actualResourceId] = _newResource;
    addressToUser[msg.sender].resourcesCount++;
    addressToResourcesId[msg.sender].push(actualResourceId);
    resourcesArray.push(actualResourceId);
    emit CreateResource(actualResourceId, _name, _description, msg.sender);
}
```

Figura 3.18: Función `addResource` del contrato inteligente

3.3.3. Peticiones

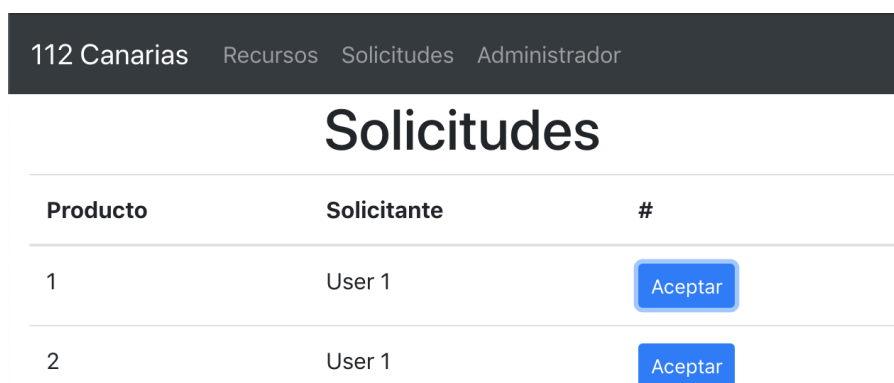
Para manejar de una forma más cómoda las peticiones de recursos en el contrato inteligente, se ha creado una estructura (ver Figura 3.19), la cual indica el usuario que ha realizado la petición, con el atributo `user`; el identificador del recurso utilizado, `resource`; y `state`, que indica el estado

de la petición. Este último puede tomar 3 posibles valores: 0 si se ha rechazado, 1 si se ha aceptado y 2 si está pendiente de responder.

```
struct Request {
    address user;
    uint resource;
    uint state;
}
```

Figura 3.19: Estructura de las peticiones de recursos

Cuando un usuario solicita un recurso, se realiza una transacción en la blockchain, de forma que el contrato inteligente añade crea una petición asociada al usuario creador del recurso solicitado. El usuario creador podrá ver esta solicitud en la pestaña de Solicitudes (ver Figura 3.20), y decidir si aceptarla o no.



Producto	Solicitante	#
1	User 1	<input type="button" value="Aceptar"/>
2	User 1	<input type="button" value="Aceptar"/>

Figura 3.20: Interfaz de las solicitudes

3.3.4. Panel de recursos

En el panel de recursos (ver Figura 3.21) se mostrarán todos los recursos creados. En este, se muestran los datos de interés de cada uno de ellos, como pueden ser el identificador, el nombre, y la descripción del recurso, así como el nombre de su creador. Por otra parte, es desde aquí desde donde se pueden solicitar recursos a otros usuarios y eliminar los recursos propios. Cuando realizamos la solicitud de un recurso, se nos indica que el estado de la misma es de *pendiente*. Si esta es aceptada por el creador del recurso, se indicará que está *disponible*.

Recursos

#	Producto	Descripción	Usuario	#
1	Recurso 1	Descripción del recurso 1	admin	Disponible
2	Recurso 2	Descripción del recurso 2	admin	Pendiente
3	Recurso 3	Descripción del recurso 3	admin	Solicitar
4	Recurso 1	Creado por nuevo user	User 1	Eliminar

Figura 3.21: Panel de recursos

3.4. Diseño de la blockchain

Una parte importante del desarrollo de una aplicación que emplee la tecnología blockchain es analizar la red de blockchain sobre la que se desplegará. Determinar tanto la privacidad de la misma como los algoritmos de consenso que se emplearán es fundamental, no solo para optimizar el funcionamiento del aplicativo, sino también para garantizar la seguridad, fiabilidad y escalabilidad del mismo.

3.4.1. Privacidad de la blockchain

En un capítulo anterior del presente documento se han descrito los diferentes tipos de blockchain atendiendo a la accesibilidad o privacidad.

Como se indicó, las *blockchain públicas* implican que cualquier usuario puede acceder a ellas. En estas, los usuarios pueden acceder a cada bloque, observando cada una de las transacciones realizadas en la red. En un aplicativo que trata con datos sensibles, esta opción no será adecuada y se optará por una *blockchain privada*, de forma que solo determinados nodos puedan acceder a la información de esta.

Por otro lado, es interesante tener en cuenta el concepto de *blockchain híbrida*, pues, aunque para el método de autenticación descrito si convenga que se trate de forma privada, al sistema de ficheros se le podría dar un carácter público. El sistema de ficheros, por su parte, podría funcionar de manera independiente del método de autenticación, aprovechándose una blockchain puramente pública.

Por último, cabe mencionar que en todo caso se descarta el uso de la red pública de Ethereum, pues cada transacción que se realice en este tiene un coste monetario en Ether [28], lo cual conllevaría un gran gasto

de dinero, ya sea por parte de los usuarios que emplean la tecnología o por la empresa que la implante.

3.4.2. Algoritmo de consenso

El algoritmo de consenso más adecuado para esta implementación es la *prueba de autoridad*. Teniendo en cuenta que se trabajará con una *blockchain privada*, los nodos de la red serán propiedad de la empresa, y por tanto, se confiará en estos. Además, este algoritmo es el más óptimo, tanto en rendimiento como en escalabilidad, agilizando la velocidad de las transacciones, disminuyendo el coste energético y permitiendo el futuro crecimiento de la red.

Capítulo 4

Presupuesto

A continuación se detalla el presupuesto estimado para el desarrollo y despliegue de la solución propuesta. En este presupuesto se ha tenido en cuenta tanto el gasto en personal como en hardware y componentes.

Por otro lado, hay que pensar que, debido a la escasez que hay tanto de analistas como de programadores de blockchain, su salario es superior a la media.

4.1. Personal y división de tareas

Para desarrollar completamente este proyecto se considera necesario contar con dos analistas, uno de blockchain, para todo lo que conlleva la aplicación de esta tecnología, y otro de programación, el cual contemplaría el análisis del desarrollo del aplicativo web sin entrar en los detalles de la cadena de bloques.

Por otro lado, debe contarse con dos programadores de blockchain, un senior y un junior, además de un desarrollador senior de React y otro programador junior, el cual trabajaría junto al senior de React. Por último, para el despliegue de la blockchain, se contaría con dos técnicos de redes.

4.2. Presupuesto personal

Se detallan las horas de trabajo, el precio por hora y el total de cada miembro del personal implicado (ver Tabla 4.1).

4.3. Presupuesto material

En cuanto al presupuesto material, este dependerá del alcance al que se quiera llegar con la red. Como cifra estimada, el coste de un servidor blockchain junto con la instalación del mismo puede llegar a ser de unos

Recurso	Horas	Precio/hora (€)	Total (€)
Analista blockchain	40	18	720
Analista programador	40	15	600
Programador senior blockchain full-stack	80	16	1280
Programador junior blockchain	80	12	960
Desarrollador senior React	80	14	1120
Desarrollador junior	80	12	960
Técnico de redes	80	10	800
Técnico de redes	80	10	800
			7240

Tabla 4.1: Estimación de presupuestos

2000€ aproximadamente.

4.4. Planificación temporal

A continuación se presenta una estimación temporal del proyecto. En primer lugar, se pueden observar las tareas y duración de las mismas (ver Figura 4.1), y, por otro lado, tenemos la organización de las mismas en forma de diagrama Gantt (ver Figura 4.2).

Nombre	Duración	Inicio	Terminado
Fase de análisis	14 days	12/03/19 8:00	29/03/19 17:00
Despliegue	14 days	1/04/19 8:00	18/04/19 17:00
Desarrollo blockchain b	14 days	1/04/19 8:00	18/04/19 17:00
Desarrollo blockchain fr	14 days	10/04/19 8:00	29/04/19 17:00
Desarrollo frontend	14 days	18/04/19 14:00	8/05/19 14:00
Fase de puesta en marc	7 days	8/05/19 14:00	17/05/19 14:00

Figura 4.1: Tareas y duración estimada

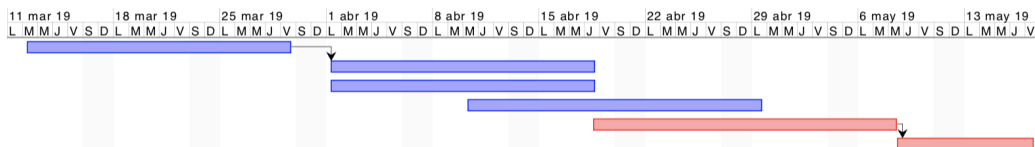


Figura 4.2: Diagrama de Gantt

Capítulo 5

Conclusiones y líneas futuras

En este capítulo se presentan algunas conclusiones obtenidas del desarrollo del presente Trabajo de Fin de Grado y posibles líneas futuras a partir de la solución implementada y del estado de la tecnología blockchain.

En primer lugar se ofrece una opinión sobre la tecnología blockchain y cómo sus características han influido en la solución propuesta. Se comentan varias ventajas e inconvenientes, así como otras consideraciones.

Por último, se comenta el futuro tanto del método de autenticación y el sistema de control de acceso a ficheros desarrollados, como de la tecnología blockchain.

5.1. Conclusiones

5.1.1. Método de autenticación

El estudio de blockchain llevado a cabo para el desarrollo de este proyecto no solo ha conducido a una solución que aprovecha cada una de las ventajas que proporciona esta tecnología, sino que ha permitido ver muchas de las posibilidades de futuro que tiene. Aunque cada día salen a la luz nuevas noticias sobre blockchain, y las criptomonedas están a la orden del día, es aún una tecnología con mucho potencial por explotar, y que podría conllevar una revolución informática.

Entre las ventajas del uso de blockchain en nuestra aplicación podemos destacar:

- **Distribución de la información:** No solo contribuye a la seguridad de la cadena de bloques, sino que también aumenta la fiabilidad de la red. Si un nodo se cae, la blockchain sigue funcionando pues hay más nodos con los datos.
- **Inmutabilidad:** Gracias al empleo de la tecnología blockchain, es

seguro que cada operación realizada queda grabada de forma inalterable, pues se tiene control de todo lo que ha ocurrido en la red.

Por otra parte, esta tecnología también tiene sus inconvenientes. La dificultad de actualizar los contratos inteligentes en la red hace que su diseño tenga grandes exigencias. Una vez que un contrato inteligente se despliega, cualquier cambio en el mismo conlleva que las transacciones referentes al anterior de alguna forma se actualicen en el nuevo, lo que es bastante complejo.

Este Trabajo de Fin de Grado ha sido presentado en las VII Jornadas sobre Tecnología y Nuevas Emergencias [29] organizadas por el 1-1-2 Canarias. Estas han tenido lugar el día 8 de mayo de 2019 en el salón de actos de Presidencia del Gobierno de Canarias, contando con una lista de 175 inscritos. Además, se presentará el día 15 de julio de 2019 en la TLP Innova [30]. Por otro lado, los conocimientos adquiridos de la tecnología blockchain han permitido la participación en el reto OHDapp [31], propuesto en conjunto por DEMOLA Canarias y Open Canarias. En este reto he trabajado con un equipo multidisciplinar para diseñar una aplicación médica que, usando blockchain, registre los historiales médicos de los pacientes, de forma que estos sean accesibles cuando y donde se desee [32].

5.2. Líneas futuras

Aunque la tecnología blockchain aún es joven en lo que respecta a sus usos más allá de las criptomonedas, cada vez son más las aplicaciones desarrolladas empleándola. La capacidad de no depender de terceras partes de confianza para garantizar la integridad de los datos abre un mundo de posibilidades que algún día verán la luz.

En lo referente a la solución propuesta:

- El método de autenticación, al no tener que recordar contraseñas, podría implementarse en más aplicativos, haciendo más cómodo el acceso a ellos.
- El uso de un wallet físico para gestionar las claves privadas mejoraría su seguridad, lo que sería conveniente en empresas en las que la privacidad es imprescindible.
- El sistema de control de acceso a ficheros podría complementarse con el sistema de ficheros compartidos distribuidos IPFS, gestionando los

permisos de los archivos subidos a esta plataforma.

- Una posible aplicación que aproveche esta solución sería un gestor de historiales médicos, en el cual se almacenen la información médica de los pacientes, ya sean del sector público o privado, empoderando a los ciudadanos de sus datos.

Capítulo 6

Summary and Conclusions

This chapter presents the conclusions obtained from the development of this Final Degree Project and future lines of the implemented solution and blockchain technology.

Firstly, an opinion is given on the blockchain technology and how it has influenced the proposed solution. Some advantages and disadvantages are discussed, as well as other considerations.

Finally, we talk about the future of both the authentication method and the file access control system that has been developed, as well as the blockchain technology.

6.1. Conclusions

The blockchain study carried out for the development of this project has not only led to a solution that takes advantage of each one of the advantages that this technology provides, but it has allowed to see some of the future possibilities that it has. Although every day new blockchain news come to light, and cryptocurrencies are the order of the day, it is still a technology with a lot of potential to exploit that could lead to a revolution.

Among the advantages of blockchain in our application, we can distinguish:

- **Distributed information:** That not only contributes to the security of the blockchain, but also increases the network's reliability. If a node of the networks crash, the blockchain will still working because there will be more nodes with the data.
- **Immutability:** Thanks to the use of the blockchain technology, we guarantee that each transaction is recorded unalterably, keeping track of everything that has happened on the network.

On the other hand, this technology also has some disadvantages. The difficulty of updating the Smart Contracts in the network means that a great design of them is required. Once a Smart Contract is deployed, every change in his code will entail that the transactions referring to the previous Smart Contract should be updated in the new contract, something quite complex.

This Final Degree Project has been presented at the VII Conference on Technology and New Emergencies [29] organized by the 1-1-2 Canarias. These took place on May 8, 2019 in the auditorium of the Presidency of the Canary Government, with a list of 175 registered. In addition, it will be presented on July 15, 2019 in the TLP Innova.

On the other hand, the knowledge acquired from blockchain technology has allowed to participate in the OHDapp [31] challenge, proposed by DEMOLA Canarias and Open Canarias. In this project I have worked with a multidisciplinary team to design a medical application that uses blockchain to record patients' medical histories. That makes the medical data accessible when and where desired. [32].

6.2. Future Works

Although the blockchain technology is still young in terms of its uses beyond cryptocurrencies, more and more applications are developed using it. The ability not to depend on trusted third parties to guarantee the integrity of the data opens a world of possibilities that will one day see the light.

Regarding the proposed solution:

- With the authentication method developed, you don't have to remember passwords. It could be implemented in more applications, making access to them more convenient.
- The use of a physical wallet to manage private keys would improve the security of these, and would be convenient in companies where privacy is essential.
- The access control system to files could be complemented with the distributed file system distributed IPFS, managing the permissions of the files uploaded to this platform.
- A possible application that takes advantage of this solution would be a manager of medical records, in which the medical information

of the patients is stored, whether from the public or private sector, empowering the citizens of their data.

Bibliografía

- [1] *Wannacy, un años después* <https://www.xataka.com/seguridad/wannacry-un-ano-despues>. Último acceso 2019-05-15.
- [2] *Así está conquistando el ransomware WannaCry el mundo: FedEx, Rusia, Ucrania, Taiwán, España y más* <https://www.xataka.com/seguridad/asi-esta-conquistando-el-ransomware-wannacry-el-mundo-fedex-rusia-ucrania-taiwan-y-mas>. Último acceso 2019-05-15.
- [3] *Suplantación de identidad (spoofing, suplantación de dirección IP)* [https://docs.microsoft.com/es-es/previous-versions/office/communications-server/dd572566\(v=office.13\)](https://docs.microsoft.com/es-es/previous-versions/office/communications-server/dd572566(v=office.13)). Último acceso 2019-05-15
- [4] *Transporte Sanitario No Urgente* <https://www.gscanarias.com/web/index.php/laempresa/division-servicio-de-urgencias-canario-suc/unidad-de-transporte-sanitario-no-urgente-tsnu>. Último acceso 2019-06-06
- [5] *Los 7 métodos de autenticación más utilizados* <https://www.evidian.com/pdf/wp-strongauth-es.pdf>. Último acceso 2019-05-16
- [6] *Bitcoin: A Peer-to-Peer Electronic Cash System* <https://bitcoin.org/bitcoin.pdf>. Último acceso 2019-05-20
- [7] *¿Cuál es la diferencia entre una DLT y 'blockchain'?* <https://www.bbva.com/es/diferencia-dlt-blockchain/>. Último acceso 2019-05-28.
- [8] *Decentralized Identifiers (DIDs) v0.13* <https://w3c-ccg.github.io/did-spec/>. Último acceso 2019-05-29.
- [9] *What is a uPort identity?* <https://medium.com/uport/what-is-a-uport-identity-b790b065809c>. Último acceso 2019-05-2019.
- [10] *uPort, una Solución para la Identidad Soberana en Blockchain* <https://medium.com/astec/uport-una-solucion-para-la-identidad-soberana-en-blockchain-bb1509e5f8>. Último acceso 2019-05-30
- [11] *FairAccess: a new Blockchain-based access control framework for the Internet of Things* <https://onlinelibrary.wiley.com/doi/full/10.1002/sec.1748>. Último acceso 2019-06-08
- [12] *¿Qué son y para qué sirven los hash?: funciones de resumen y firmas digitales* <https://www.genbeta.com/desarrollo/que-son-y-para-que-sirven-los-hash-funciones-de-resumen-y-firmas-digitales>. Último acceso 2019-05-28.

- [13] *SHA-2* <https://es.wikipedia.org/wiki/SHA-2>. Último acceso 2019-05-28.
- [14] *Criptografía asimétrica* <https://academy.bit2me.com/que-es-criptografia-asimetrica/>. Último acceso 2019-05-28.
- [15] *Qué es el Árbol de Merkle* <https://academy.bit2me.com/que-es-arbol-de-merkle/>. Último acceso 2019-05-28.
- [16] *Cómo funciona la blockchain y sus partes* <https://www.miethereum.com/blockchain/como-funciona/>. Último acceso 2019-05-28.
- [17] *Los tipos de Blockchain: públicas, privadas e híbridas* <https://www.iecisa.com/es/blog/Post/Los-tipos-de-Blockchain-publicas-privadas-e-hibridas-y-II/>. Último acceso 2019-05-29.
- [18] *Prueba de Trabajo vs Prueba de Participación* <https://criptotario.com/prueba-de-trabajo-vs-prueba-de-participacion>. Último acceso 2019-05-29.
- [19] *Qué es PoA (Proof of Authority - Prueba de Autoridad)* <https://academy.bit2me.com/que-es-poa/>. Último acceso 2019-05-29.
- [20] *Ethereum Overview* <https://truffleframework.com/tutorials/ethereum-overview>. Último acceso 2019-06-01.
- [21] *Truffle Suite* <https://truffleframework.com/>. Último acceso 2019-06-01.
- [22] *Ganache* <https://www.trufflesuite.com/ganache>. Último acceso 2019-06-01.
- [23] *web3.js - Ethereum JavaScript API* <https://web3js.readthedocs.io/en/1.0/index.html>. Último acceso 2019-06-01.
- [24] *Drizzle* <https://www.trufflesuite.com/drizzle>. Último acceso 2019-06-01.
- [25] *React* <https://reactjs.org/>. Último acceso 2019-06-01.
- [26] *JSON Web Tokens* <https://jwt.io/>. Último acceso 2019-06-01.
- [27] *Metamask* <https://metamask.io>. Último acceso 2019-06-01.
- [28] *Transacciones de Ethereum: Qué es el Gas y para qué se utiliza* <https://criptomoneda.ninja/gas-ethereum/>. Último acceso 2019-06-04.
- [29] *VII Jornadas de Tecnología Y Nuevas Emergencias: Transformación Digital, ¿también en Emergencias?* <http://www.112canarias.com/info/index.php/divulgacion/congresos-jornadas/08-05-2019-vii-jornadas-de-tecnologia-y-nuevas-emergencias-transformacion-digital-tambien-en-emergencias>. Último acceso 2019-06-08
- [30] *TLP Innova 2019* <https://tlp-tenerife.com/tlpinnova/>. Último acceso 2019-06-08
- [31] *OHDAPP (Own Health Data app)* <https://applications.demola.net/cases/388>. Último acceso 2019-06-08
- [32] *Modelo finlandés con soluciones canarias* <https://www.eldia.es/sociedad/2019/05/16/modelo-finlandes-soluciones-canarias/975761.html>. Último acceso 2019-06-08