Escuela Superior
de Ingeniería y Tecnología
Universidad de La Laguna

Trabajo de Fin de Grado

Grado en Ingeniería Informática

Aplicación de Blockchain a situaciones de emergencias

Blockchain application for emergency situations

Sergio Ferrera de Diego

La Laguna, Junio de 2019

Dña. **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. **José Iván Santos González**, con N.I.F. 78.637.989-T Personal Investigador adscrito al Departamento de Ingeniería Informática y de Sistemas de la Universidad de La Laguna, como cotutor.

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

Que la presente memoria titulada:

"Aplicación de Blockchain a situaciones de emergencias"

ha sido realizada bajo su dirección por D. **Sergio Ferrera de Diego**,

con N.I.F. 45.867.750-T.

Y para que así conste, en cumplimiento de la legislación vigente y a los efectos oportunos firman la presente en La Laguna a 7 de junio de 2019

Agradecimientos

Me gustaría agradecer a mi tutora, Pino Caballero Gil, y a mi cotutor José Iván Santos González, por su implicación y ayuda en todo momento durante el desarrollo de este trabajo.

A mi familia, por creer en mí en todo momento y por su apoyo en las metas que me he propuesto.

Licencia

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

Resumen

El objetivo de este trabajo ha sido el desarrollo de una aplicación para decidir en tiempo real qué organizaciones pueden aportar recursos en una situación de emergencia, teniendo en cuenta aspectos como su distancia con respecto a la emergencia y si son capaces de aportar todos los recursos necesarios o bien por el contrario solo son capaces de ofrecer parte de los recursos solicitados o ninguno.

En el caso de abastecimiento parcial de recursos, la aplicación web permite al usuario solicitar esos recursos parciales a la organización más cercana y el resto a la siguiente que se encuentre más cerca de la emergencia, obteniendo así una respuesta más eficaz ante la situación de emergencia.

Cada interacción con la plataforma queda registrada en una Blockchain, empleando contratos inteligentes. Dichos contratos inteligentes representan las emergencias, de tal forma que permiten obtener información relativa a las organizaciones involucradas, fecha y hora de envío de un recurso a una emergencia específica.

Además, la aplicación facilita las comunicaciones entre el servicio de emergencias y las organizaciones que aportan los recursos, ya que todos los participantes hacen uso de la misma plataforma.

Palabras clave: Blockchain, Contratos Inteligentes, Ethereum, Javascript, aplicaciones descentralizadas, gestión de recursos, gestión de emergencias

Abstract

The main objective of this work has been the development of a web application for those organizations that want to provide resources in an emergency situation in real time taking into account aspects such as the distance to the emergency and if they can provide all the necessary resources or if they are only able to offer part of the requested or none.

In the case of partial supply of resources, the application will allow users to request that to the closest organization and the rest to the next closest to the emergency, obtaining in this way a faster response to the emergency situation.

Each interaction with the platform is registered in a Blockchain, using Smart Contracts, which in this case represent emergencies. The use of Smart Contracts allows the possibility to obtain information of the involved organizations, date and time where a resource has been sent to a specific emergency.

In addition, the application facilitates communications between emergency services and the organizations that provide resources, because both use the same platform.

Keywords: Blockchain, Smart Contracts, Ethereum, Javascript, decentralized applications, resource management, emergency management

Índice general

Índice de figuras

Índice de tablas

Capítulo 1 Introducción

1.1 Motivación

Gracias a la popularidad adquirida en estos años por la tecnología Blockchain, en gran medida obtenida inicialmente por el auge de las criptodivisas, poco a poco, distintos sectores de la sociedad han mostrado un gran interés por esta tecnología y por las posibilidades que ofrece.

Tras analizar la creación de nuevos proyectos basados en Blockchain mediante el uso de contratos inteligentes en distintas áreas como la sanidad o la logística, demostrando con ellos un gran número de posibles usos, resulta natural tener interés por el estudio y la profesionalización en esta tecnología, cuya base es la seguridad y la criptografía.

En este contexto, este Trabajo de Fin de Grado muestra uno de los posibles desarrollos que se pueden llevar a cabo con la tecnología Blockchain, mediante el uso de contratos inteligentes, garantizando así un sistema seguro y descentralizado. En este caso, los contratos inteligentes representan las emergencias, que han de ser abastecidas con una serie de recursos especificados. Su despliegue facilita la comunicación entre las distintas partes involucradas y el registro de las acciones realizadas de forma legítima.

1.2 Objetivos

El objetivo principal de este trabajo es el desarrollo de una plataforma web que permita a todos los involucrados en una emergencia, acciones diversas que cubren desde informar sobre la misma hasta abastecer de los recursos necesarios, todo ello haciendo uso de contratos inteligentes como representación de las emergencias, consiguiendo así una aplicación descentralizada y segura, basada en el token de Ethereum (Ether).

Para poder alcanzar el objetivo descrito, la aplicación web integra un sistema de autenticación y registro de distintos tipos de usuario, la posibilidad de informar usando la herramienta web sobre nuevas emergencias ocurridas, abastecer de los recursos solicitados en cada emergencia y un panel en el cual visualizar las emergencias ya satisfechas y las no satisfechas en lo que respecta a la recepción de recursos.

Otro de los objetivos es realizar una investigación de las herramientas y tecnologías existentes dentro del contexto de los contratos inteligentes, para su correcta implementación, para así obtener además una visión más precisa de las posibilidades que brindan los contratos inteligentes en distintos ámbitos.

1.3 Fases del desarrollo

Las fases del desarrollo del presente Trabajo Fin de Grado se pueden clasificar en seis.

Una primera etapa fue dedicada al estudio y análisis del funcionamiento de la tecnología Blockchain y sus conceptos clave (descentralización, bloques, nodos, minado, etc), las distintas plataformas existentes teniendo en cuenta la diferencia entre ellas y las tecnologías, así como las herramientas aplicables en este trabajo.

En lo que respecta a la segunda etapa, se procedió a consensuar una serie de ideas con relación al diseño de la aplicación web, teniendo en cuenta aspectos como la usabilidad.

Procediendo a la tercera etapa, tras haber definido correctamente la etapa anterior, se procedió al desarrollo e implementación de la aplicación web tanto en lo que respecta a su frontend como a su backend, haciendo uso de Bootstrap, MySQL, entre otros.

Llegados a la cuarta etapa, se procedió al desarrollo de un contrato inteligente cumpliendo los requisitos definidos. Para ello se empleó el lenguaje de programación Solidity con la suite de truffle para las pruebas y despliegue del mismo.

En la quinta etapa se integró la plataforma web con el sistema Blockchain definido mediante el contrato inteligente creado. Para ello se hizo uso de la API Web3js.

Finalmente, alcanzando la sexta y última etapa, se procedió a realizar un análisis general de las posibles vulnerabilidades de la plataforma en lo que respecta a las comunicaciones y datos tratados, con el objetivo de solventarlo haciendo uso de mecanismos de seguridad.

1.4 Estructura de la memoria

La estructura del presente documento se compone de siete capítulos. En el primero de introducción, se mencionan las motivaciones, objetivos, fases de desarrollo del trabajo, estructura de la memoria, la introducción a la aplicación, definiendo el problema a resolver por la misma, aplicaciones similares con el mismo cometido y finalmente una conceptualización de la propuesta.

El segundo capítulo abarca el estado del arte, con una introducción, y una explicación de conceptos que incluyen Ethereum, la criptografía implicada en la Blockchain y otros aspectos más específicos de la Blockchain.

En lo que respecta al tercer capítulo, dedicado a la descripción de la propuesta desarrollada en este trabajo. Primero se valoran las distintas herramientas empleadas para el desarrollo, sus características y aspectos de seguridad. Luego se mencionan aspectos más relacionados directamente con la aplicación web del trabajo, explicando el sistema Blockchain empleado, el sistema web y la combinación de ambos.

Con relación al desarrollo del trabajo y para garantizar su calidad del mismo, el cuarto capítulo incluye todas las labores de testeo ejercidas.

Recogemos en el quinto capítulo se incluye un presupuesto estimado de lo que conlleva el desarrollo e implementación de este Trabajo Fin de Grado en un contexto empresarial.

Finalmente, en el sexto y séptimo capítulos de esta memoria se mencionan algunas de

las conclusiones alcanzadas con este trabajo, tanto en español como en inglés.

1.5 Definición del problema

La gestión de los recursos en una emergencia suele gestionarse en plataformas distintas de la encargada del registro de una nueva emergencia ocurrida en el centro de emergencias. Sin embargo, ambas partes se ven involucradas de principio a fin, con un objetivo común, que es hacer llegar los recursos a su destino, que en este caso, es una emergencia.

Para garantizar lo anterior, es necesario obtener información como la ubicación en tiempo real de los recursos, tiempo estimado de llegada, recursos disponibles, etc. Dichos aspectos pueden ser requeridos por cualquier personal de emergencias, por lo que es imprescindible que estos estén disponibles para todos los involucrados o en su defecto, se pueda proporcionar acceso a los mismos de forma veloz, eficiente y segura.

En la práctica, el intercambio de esta información resulta complejo debido al uso de distinto software por parte de los diferentes participantes en la resolución de una emergencia.

Además, la seguridad y veracidad de los registros almacenados en lo que respecta a lo ocurrido en una emergencia quedan comprometidos debido a que se almacenan en sistemas de información, con capacidad de edición y por tanto, con posibilidad de alteración por parte de algún personal con autorización a ello.

1.6 Conceptualización de la propuesta

Para solventar los diversos problemas mencionados en el apartado anterior, se ha decidido proceder al desarrollo de una aplicación web, la cual es utilizada por todos los involucrados en una emergencia, desde el operador encargado de recibir las llamadas de los alertantes, los gestores de recursos hasta las organizaciones que proveen los recursos.

En lo que respecta a la adición de una nueva emergencia por parte del operador, se dispondrá de un formulario web para este cometido, en el cual tras introducir los datos de la emergencia proporcionados vía telefónica, procederá a enviarlos al sistema web, el cual se encargará a partir de ellos de generar un nuevo contrato inteligente, siendo este la representación de la emergencia dentro del sistema.

Tras añadir la emergencia en el panel correspondiente, los gestores de recursos podrán visualizarla, al igual que el resto a los que se les haya asignado, pendientes de satisfacer, en lo que a recursos se refiere.

Los tipos de recursos que intervienen en una emergencia se han basado en la realidad, gracias a la colaboración del 112 Canarias en mostrar el funcionamiento interno de la entidad, desde la recepción de una llamada hasta la movilización de los recursos correspondientes. Prestando atención en este punto, tal como se muestra en la Figura 1.1 los tipos de recursos que se han contemplado son:

- Ambulancias, sirviendo de representación dentro de la aplicación web del sector sanitario.
- Camiones de bomberos, representando al sector de extinción, salvamento y rescate.
- Cuerpos de policía, consiguiendo asi representación del sector de seguridad.

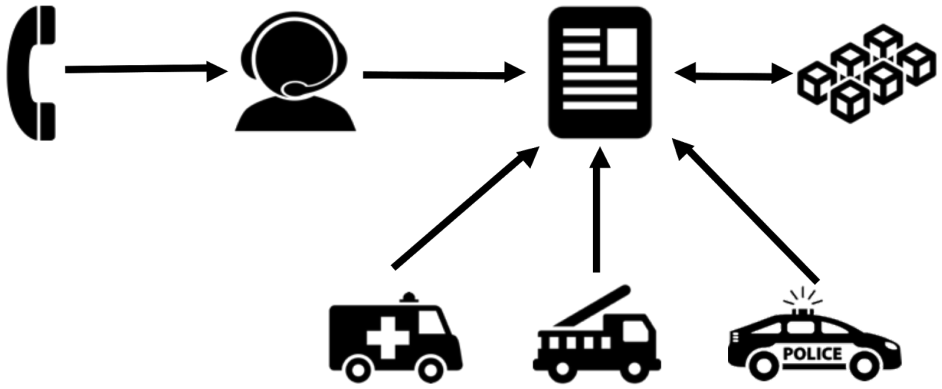


Figura 1.1: Esquema de la propuesta

Se ha contemplado la posibilidad, basándonos en la estructura de funcionamiento actual del 112 (ver Figura 1.2), donde se desea un usuario administrador dentro del sistema, pues simboliza una parte del cometido del coordinador multisectorial, quien desea visualizar todas las emergencias que se encuentren en la aplicación, tanto aquellas que ya han sido satisfechas como las que no. Para ello, a este tipo de usuario se le proporciona un panel con este cometido.

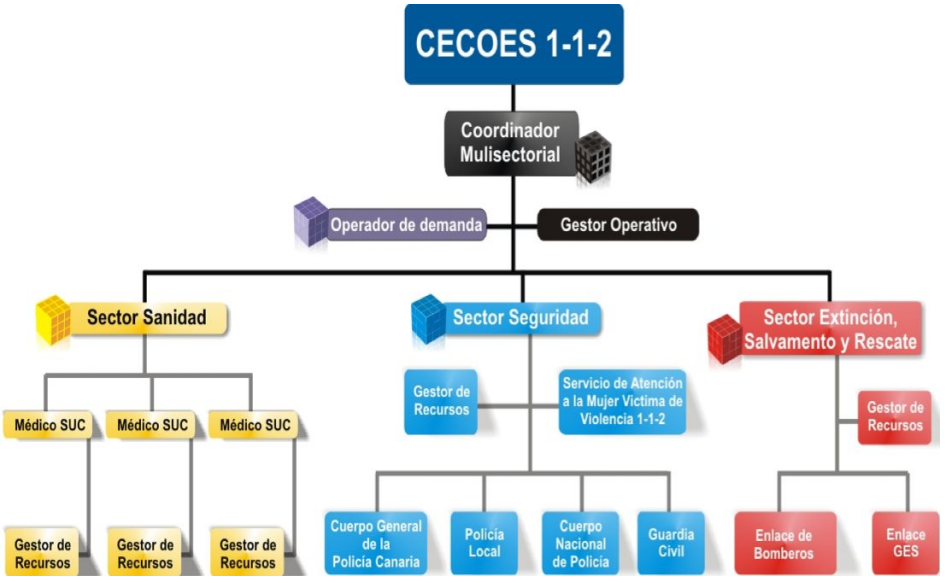


Figura 1.2: Estructura del 112 Canarias

Capítulo 2 Estado del arte

2.1 Introducción

La primera referencia que se tiene de la tecnología Blockchain tal y como es conocida hoy en día, se remonta en el año 1991 [1], de la mano de Stuart Haber y W. Scott, los cuales realizaron un primer trabajo basado en una cadena de bloques, cuyo objetivo era evitar la manipulación de las marcas de tiempo hacia los documentos.

Sin embargo, con el objetivo de mejorar la eficiencia, en 1992 actualizaron el sistema inicial, incorporando así arboles de Merkle, que son unas estructuras de datos especiales, que más adelante se explicarán. Esta mejora permite recopilar una mayor cantidad de documentos por bloque.

En el año 2008, Satoshi Nakamoto hizo público el whitepaper [2] sobre Bitcoins como sistema electrónico usuario-a-usuario. En realidad Satoshi Nakamoto es un seudónimo luego se desconoce la autoría de ese documento. En él, se define cómo funciona la criptomoneda Bitcoin (ver Figura 2.1), la cual impulsó la popularidad del sistema Blockchain, hasta como es conocido hoy en día.

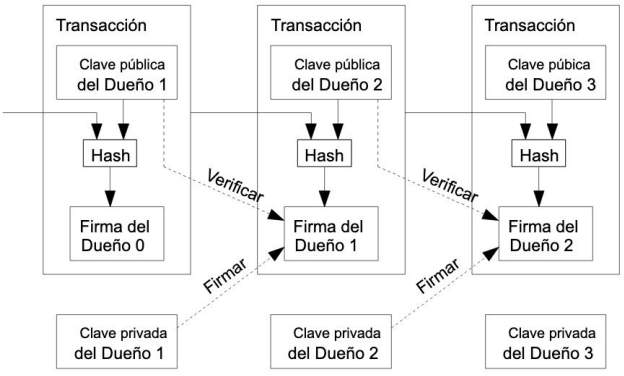


Figura 2.1: Diagrama de una transferencia de Bitcoin

Tras dicha publicación, en 2009, nuevamente el autor o autores del whitepaper, compartieron un informe, cuyo objetivo fue mostrar las pautas a seguir para conseguir un sistema que fuera capaz de generar confianza digital, basándose en la idea de la descentralización, evitando así, como ocurre hoy en día en una serie de sectores, que el control radique en un individuo o en una organización.

En el año 2013 se produce el nacimiento de Ethereum, una nueva blockchain, que en comparación con la existente usada en Bitcoin, permite una mayor cantidad de funciones aparte de ser una red punto a punto. Se abordará en profundidad todo lo relacionado con

Ethereum en el apartado correspondiente más adelante.

En 2015, con el objetivo de facilitar el desarrollo del Blockchain en el ámbito industrial, la Fundación Linux presentó el proyecto Hyperledger [3] (ver Figura 2.2), donde a día de hoy, se facilita código abierto Blockchain.

Figura 2.2: Herramientas y frameworks de Hyperledger

2.2 Ethereum

Aunque Ethereum [4] surge compartiendo varios aspectos de Bitcoin, sus diferencias son apreciables (ver Figura 2.3). Entre ellas, cabe mencionar el objetivo para el que se usan. Por un lado se encuentra Bitcoin cuyo objetivo es representar el dinero digital. Por otro lado, el enfoque de Ethereum es el de los contratos inteligentes [5].

	bitcoin	ethereum
concept	digital money	smart contracts
transaction	send from alice to bob	send from alice to bob if.. • date = jan 1, 2018 • bob's balance < 10 eth
market cap (as of feb 2017)	~$18 billion	~$1 billion
founder	satoshi nakamoto (unknown)	vitalik buterin and team
release date	jan 2009	july 2015
release method	early mining	presale raised $18M in bitcoin

Figura 2.3: Tabla comparativa entre Bitcoin y Ethereum

Centrándonos en el concepto de contrato inteligente, pues Ethereum gira en torno a él, no es más que un script, escrito en un lenguaje de programación, el cual es visible por cualquier individuo que conforme la red blockchain en la que se encuentre. Dicho código es ejecutado de forma automática y autónoma, sin necesidad de intermediarios.

Los posibles usos que pueden realizarse a base de contratos inteligentes (ver Figura 2.4), son los mismos que podemos encontrar a día de hoy, mediante contratos tradicionales, con la ventaja del tiempo y coste que suponen, debido a que en ellos no es necesaria la figura de una tercera persona, como un intermediario, facilitando así la realización de los mismos en cualquier parte del mundo.

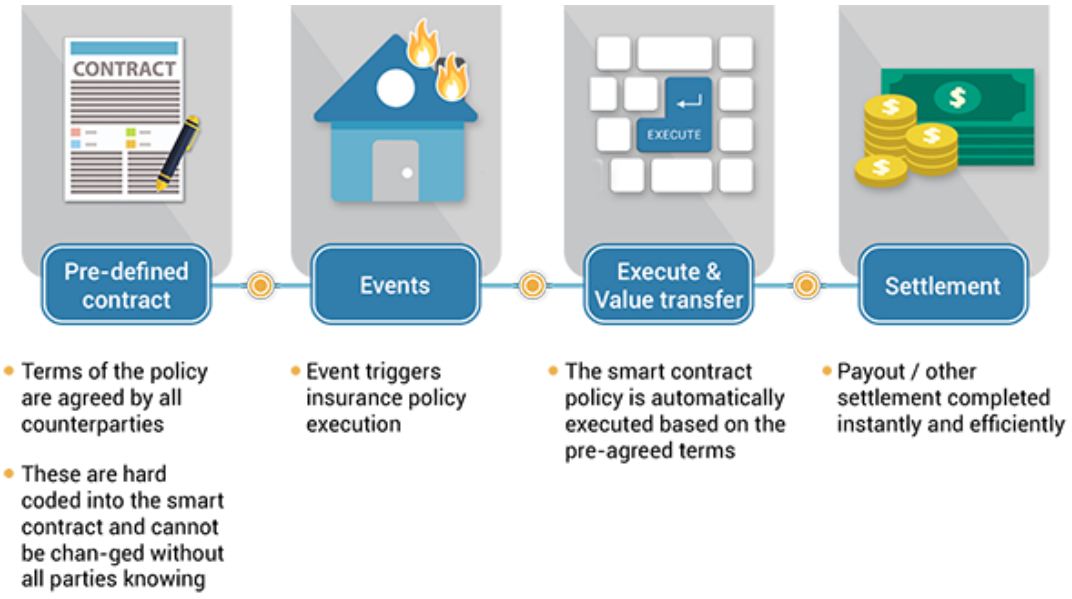


Figura 2.4: Ejemplo de contrato inteligente aplicado en pólizas de seguro

2.3 Criptografía

2.3.1 Función Hash

La aplicación de las funciones hash es diversa. Una de ellas es la firma digital de documentos, que permite garantizar la autenticidad de una acción.

Aplicando este concepto, en la Blockchain, su cometido es el de asegurar la autenticidad de las transacciones y así prevenir acciones maliciosas.

Tras haber mencionado una serie de usos y su implicación dentro del sistema Blockchain, procedemos a realizar una breve definición del mismo, pues es un procedimiento criptográfico, que haciendo uso de un algoritmo específico (en el caso del Bitcoin el estándar SHA-256 [6], perteneciente a la familia de los SHA-2) consigue convertir una información determinada, como por ejemplo texto, en una secuencia alfanumérica de longitud fija, siendo este el resultado, denominado hash (ver Figura 2.5).

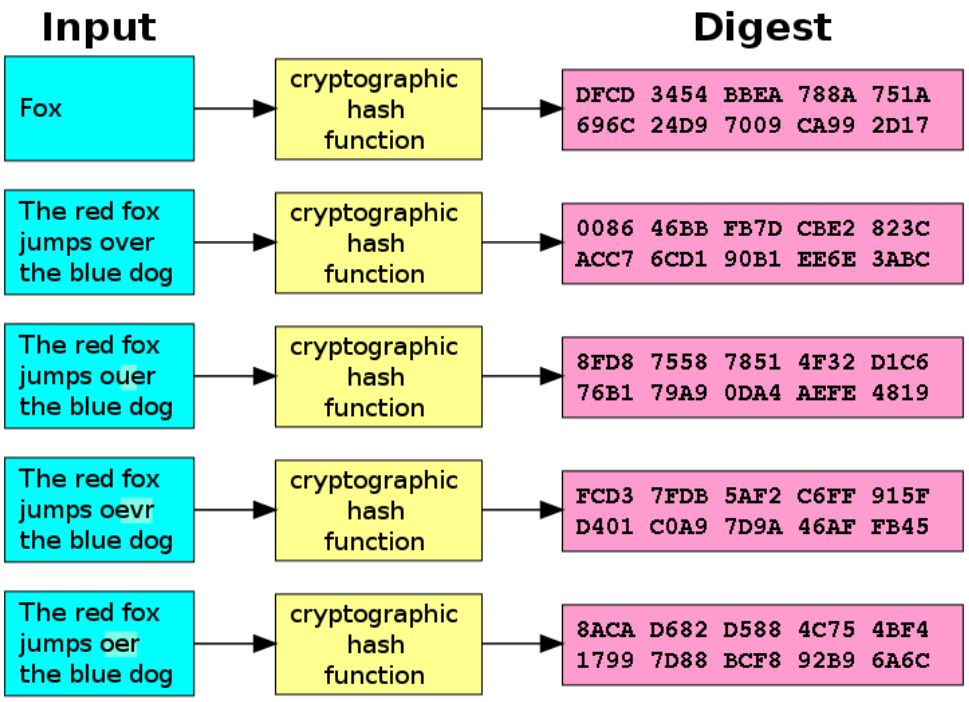


Figura 2.5: Ejemplo de función hash

2.3.2 Árbol de Merkle

Antes de entrar en detalles en lo que respecta al funcionamiento de un Árbol de Merkle, es importante aclarar lo que se persigue alcanzar mediante su uso, que es reducir el tiempo y consumo de recursos necesarios para verificar la integridad de una serie de datos, haciendo uso de un único hash, que se encontrará almacenado en la raíz.

Su desarrollador fue Ralph Merkle [7], siendo uno de informáticos y criptógrafos, más reconocidos en el mundo, pues además de desarrollar y patentar la estructura de árboles de Merkle en 1979, también fue el creador del cifrado de clave pública.

Esta estructura de datos es empleada en Blockchain para garantizar de forma eficiente y veloz, la verificación de las transacciones, pues debido a ella, si un hash es cambiado, el resto lo harán también hasta alcanzar el hash raíz, ya que en un Árbol de Merkle, los hashes son agrupados en pares, sin existir un límite de los mismos, partiendo del hash raíz, como podemos observar en la Figura 2.6.

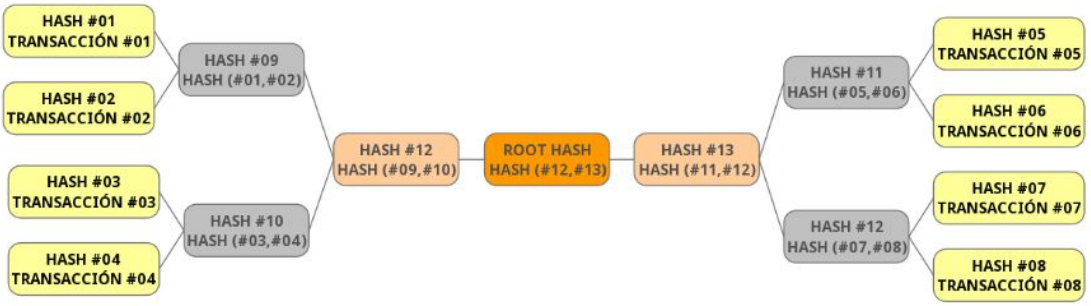


Figura 2.6: Árbol de Merkle aplicado en Blockchain

Tanto la adición de transacciones como el intento de modificación del hash raíz, ambas

acciones son invalidadas de forma automática, pues en ambos casos existe una dependencia por parte de otros hashes.

2.3.3 Clave pública

Tanto en Bitcoin como en Ethereum, como en una gran parte de los proyectos basados en la Blockchain en la actualidad, se hace uso de la criptografía asimétrica [8], siendo este una parte vital para el correcto funcionamiento del sistema y su seguridad.

Este tipo de criptografía se basa en el uso de un par de claves, denominadas clave privada y clave pública, que están relacionadas matemáticamente, pues la pública se obtiene fácilmente a partir de la privada. Por el contrario, el procedimiento inverso, es decir, obtener la clave privada a partir de la pública, no es tarea fácil.

Haciendo uso del algoritmo SHA256, en el caso de la Blockchain que estudiamos, se firman las transacciones mediante la clave privada, ya que con la pública se verifica el remitente (ver Figura 2.7).

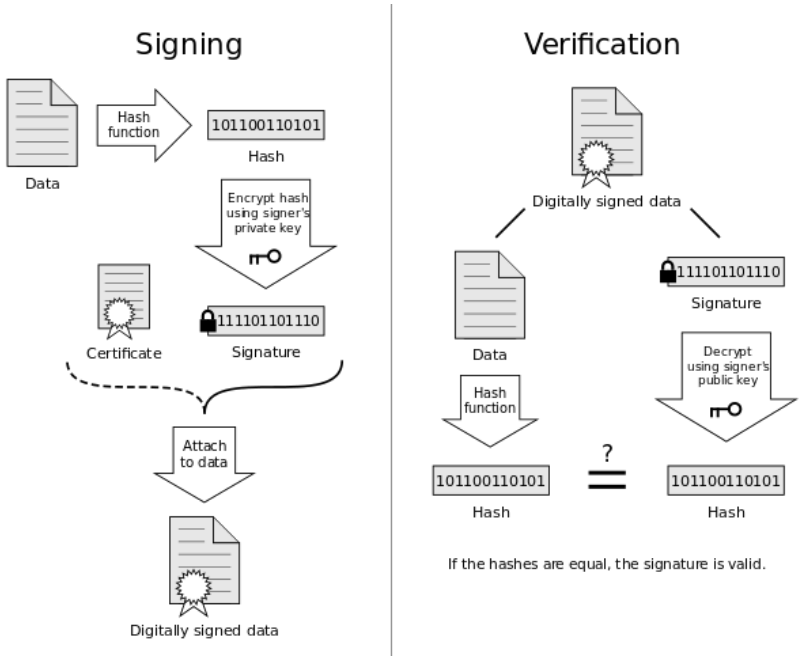


Figura 2.7: Uso de criptografía asimétrica en la firma digital

2.4 Blockchain

La Blockchain permite a los desarrolladores crear redes de dispositivos [9] sin la necesidad de un servidor o base de datos central para su interconexión. Ante una tarea específica, toda la potencia computacional de la suma de los nodos que conforman la red puede ser ejercida para la misma, para así conseguir realizarla de forma más rápida y eficiente.

Los dispositivos que deseen integrarse en la red solo deberán de hacer uso del software destinado a este cometido, por ejemplo en el caso de Bitcoin y de gran parte del resto de proyectos populares basados en Blockchain, estos son de carácter abierto y con licencias de software libre, permitiendo así transparencia y colaboración en los mismos

por parte de la comunidad.

La Blockchain es definida como una red distribuida (ver Figura 2.8) debido a su naturaleza de no disponer de una base de datos en un lugar único, pues más bien la información se encuentra replicada en cada uno de los nodos que participan.

Además, si más del 50% de los dispositivos que conforman la red no son propiedad de la misma persona u organización, también se le puede adjudicar la característica de ser una red descentralizada, pues el control de la misma no radica en una sola persona o empresa.

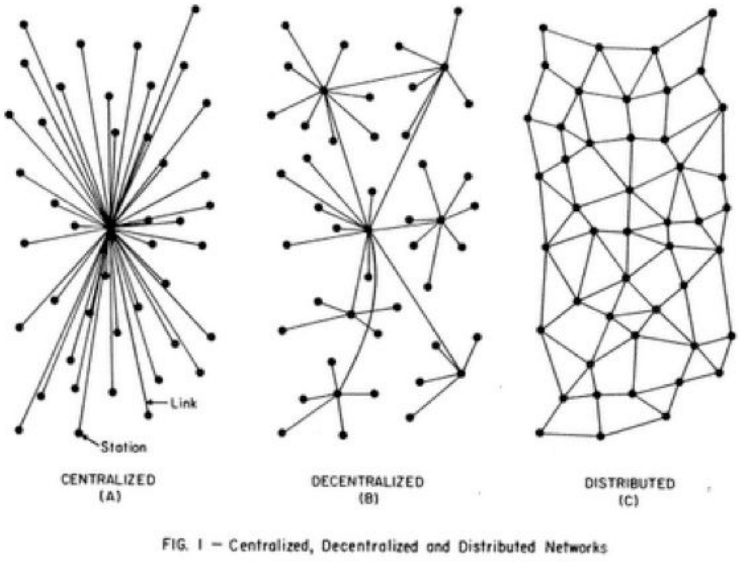


Figura 2.8: Redes centralizada, descentralizada y distribuida

2.4.1 Bloque

El concepto de bloque surge en la tecnología de Blockchain a partir de la idea de hacer uso de árboles de Merkle para optimizar el proceso de verificación de transacciones. Cada bloque es generado haciendo uso del sistema conocido como prueba de trabajo [10], que será explicado en detalle en el siguiente apartado.

Dentro de cada uno de estos bloques, se encuentra una subestructura definida mediante un Árbol de Merkle, que representa un resumen de todas las transacciones contenidas en cada uno (ver Figura 2.9).

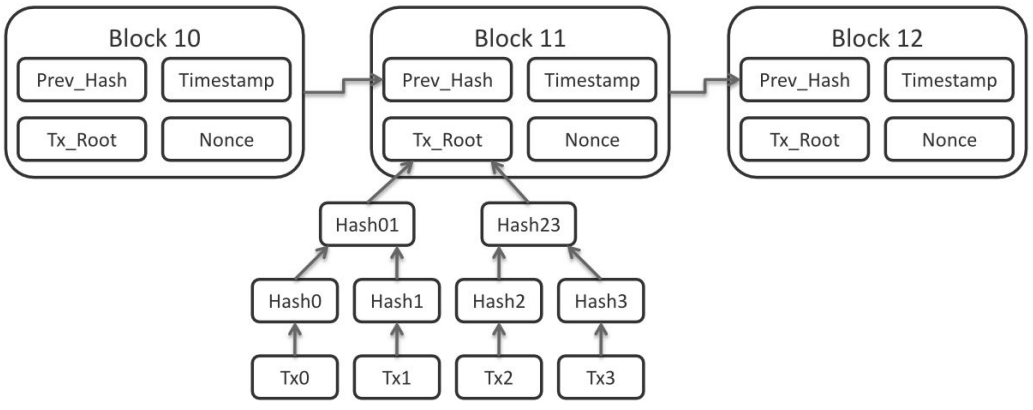


Figura 2.9: Representación de bloques dentro de un sistema Blockchain

Para facilitar la comprensión de la implicación de los bloques dentro de un sistema Blockchain, es usual hacer la analogía del mismo como si fuera un libro contable, pues cada bloque, siguiendo esta semejanza, representaría una página del libro, quedando escrita en cada una de ellas, las transacciones ocurridas en la red.

2.4.2 Prueba de trabajo (Proof of Work)

Con el cometido de evitar ataques informáticos [11], surge la tecnología llamada Proof of Work, la cual mediante una prueba de dificultad moderada que suele ser un problema matemático a realizar antes de una acción, se consigue el objetivo propuesto, pues si no es resuelta satisfactoriamente no se puede proseguir a la realización de la acción deseada (ver Figura 2.10).

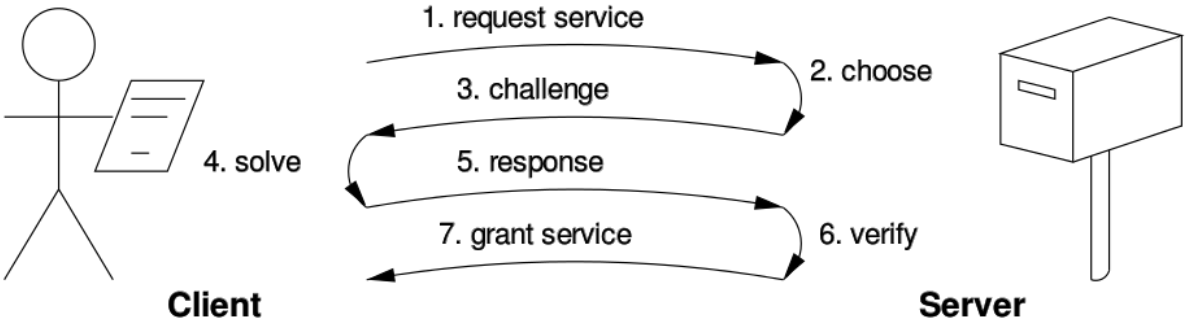


Figura 2.10: Diagrama de ejemplo del proceso de una prueba de trabajo

Aunque este concepto haya tenido su auge en su uso en el mundo de las criptomonedas, su invención se retoma en 1992, por parte de Cynthia Dwork y Moni Naor, cuyo objetivo en aquel entonces, era, haciendo uso de este sistema, reducir el correo no deseado o más conocido como SPAM, además de los ataques DDOS.

La implementación de la tecnología en la Blockchain dio como resultado evitar problemas como el del Doble Gasto o la manipulación de cuentas, pues brinda la posibilidad de confirmar transacciones y organizarlas en bloques de forma ordenada.

Aquellos nodos en la red que se encargan de solventar los problemas matemáticos planteados en el caso de por ejemplo en el Bitcoin, son denominados mineros. Para incentivar su aportación en lo que respecta a potencia computacional para este cometido, se les otorga una recompensa de forma automática y equitativa al trabajo realizado.

Capítulo 3 Propuesta desarrollada

3.1 Tecnologías

Con el objetivo de aportar una visión más clara de las tecnologías empleadas en este trabajo, procederemos a nombrarlas clasificándolas en base a si han sido empleadas para el frontend del proyecto o si por el contrario han sido usadas para el backend (ver Figura 3.1).

Figura 3.1: Visión general de tecnologías empleadas

Para cubrir la necesidad de una capa servidor, se ha optado por usar el entorno en tiempo ejecución NodeJS basado en JavaScript [12], debido a su adopción en este campo y a su uso en múltiples proyectos de prestigio como puede ser Netflix entre otros [13]. Además, es importante destacar el lenguaje de programación Javascript, pues cada vez está más presente en diversas partes de un proyecto, por lo que cuanto mayor sea su uso, mejor será su integración con las distintas partes del trabajo.

Debido a que datos como los de la autentificación de los usuarios tienen una estructura bien definida, hemos considerado que lo más acorde a ello es hacer uso de una base de datos relacional. Dentro de este tipo de base de datos, una de las soluciones más populares y con mayor soporte en la red es MySQL [14], y por ello que la hayamos escogido para este proyecto.

En lo que respecta al desarrollo de los contratos inteligentes, nos hemos decantado por el lenguaje de programación Solidity, debido a que su sintaxis se encuentra basada en Javascript [15], proporcionando así mayor cohesión con el resto de partes que conforman la aplicación web. Además cabe destacar la existencia de una documentación oficial completa pero de fácil comprensión [16], aún siendo un lenguaje de programación muy reciente en la red.

Buscando un entorno de desarrollo y testeo en lo que respecta a la programación de los contratos inteligentes con Solidity, se consideró que el más apropiado es Truffle Suite [17], debido a su popularidad e integración con NodeJS y Travis CI, del cual hablaremos

más adelante.

Relacionado con el frontend del trabajo, se ha optado por el conjunto de herramientas Boostrap [18], ya que nos permite hacer un desarrollo responsive, usable y ligero sin mayores complicaciones, y además la documentación de la misma es lo suficiente extensa para su uso.

Siguiendo la tendencia actual del uso de componentes web en el desarrollo frontend, nos hemos decantado por usar PugJS [19], antiguamente conocido como JadeJS, debido a que es el más veterano en su campo y por tanto de los que más documentación se encuentra en la red.

Con el objetivo de combinar el sistema web con de los contratos inteligentes, se hace uso del conjunto de librerías Web3JS [20], debido a su aparición recurrente en la documentación de Truffle Suite, pues este hecho facilita su aprendizaje debido a una integración ya conocida y explicada.

Para aportar integración continua en el proyecto y así garantizar una respuesta eficiente hacia posibles errores durante el desarrollo de la aplicación, se ha elegido TravisCI [21], ya que se disponía de experiencia previa con él, además ofrece una integración con Truffle Suite excelente para extender la integración continua a los contratos inteligentes.

3.2 Sistema de Blockchain

3.2.1 Estructura del contrato

Para que funcione el sistema blockchain de la aplicación es indispensable contar con el contrato inteligente desarrollado, que como hemos mencionado en capítulos anteriores, representa en este caso la emergencia. Con este objetivo en mente, la estructura del mismo es la siguiente.

Disponemos de diversos atributos (ver Figura 3.2), entre ellos, tres de tipo entero, que representan los distintos recursos necesarios por parte de la emergencia, denominados ambulances, firefighters y police.

Figura 3.2: Atributos del contrato

Por otra parte, disponemos de un vector de tipo cadena de caracteres, cuyo nombre es *organizations*, que contendrá las organizaciones que han sido seleccionadas por parte del operador para satisfacer los recursos necesarios.

El atributo booleano state representa con un true por defecto, la necesidad de

abastecimiento de recursos por parte de la emergencia y en caso contrario, es decir, cuando todos los recursos necesarios hayan sido satisfechos, se presenta con un false. Además dentro del contrato disponemos de un constructor por defecto, el cual establece como cantidad de recursos 0, el estado del contrato true y como propietario del contrato el creador del mismo.

Un método modificador con nombre onlyOwner, el cual nos permitirá más adelante establecer cuáles son los métodos que deseamos que solo puedan ser usados por el creador de la emergencia.

En el interior del contrato, también podremos observar las funciones set_resources y set_organization, las cuales permiten establecer la cantidad de recursos necesarios por la emergencia y las organizaciones partícipes respectivamente.

El método get_organizations_length, el cual devuelve el tamaño del vector de organizaciones, lo cual es necesario para poder recorrer el vector y así obtener las organizaciones partícipes en la emergencia.

Para permitir el envío de recursos a una emergencia se dispone de la función send_resources, por otra parte para poder comprobar si ya han sido satisfechos todos los recursos solicitados y así cambiar el estado de la emergencia, existe el método denominado check_request.

3.2.2 Ganache

Cualquier acción realizada sobre el contrato inteligente queda registrada en forma de transacción dentro de la Blockchain del proyecto. Para gestionarla se hace uso de la herramienta Ganache (ver Figura 3.3), la cual forma parte de la suite de Truffle.

Aún cuando solo se accede a la emergencia para visualizar aspectos como los recursos, este tipo de acciones se registran como cualquier otra dentro de la Blockchain, sin distinción alguna, almacenando una serie de datos como la dirección de la cartera (usuario) que ha realizado la acción pertinente y hacia qué contrato (emergencia).

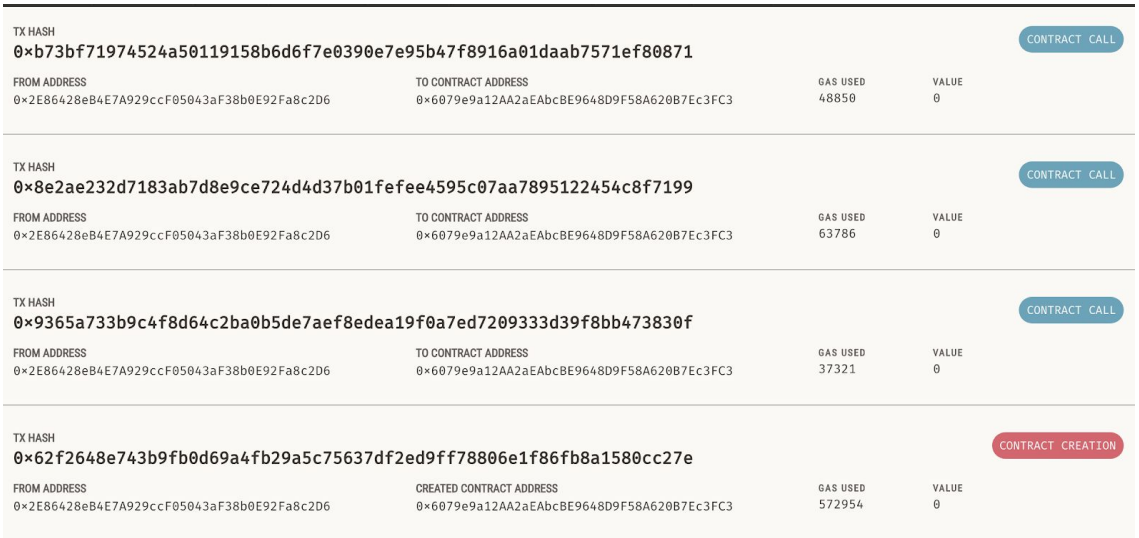


Figura 3.3: Ejemplo de transacciones del Blockchain con Ganache

3.3 Sistema web

3.3.1 Tipos de usuario

Los tipos de usuario contemplados (ver Figura 3.4) en este trabajo son:

- Un operador de emergencias, el cual tendrá como cometido informar al sistema de una nueva emergencia ocurrida y de una serie de datos del mismo, como la localización, datos del alertante, etc. También tendrá como cometido informar de los recursos necesarios para abarcar la emergencia. Finalmente, otra labor a realizar por su parte es la de seleccionar las organizaciones receptoras de la emergencia.
- Por otra parte, disponemos de los proveedores de recursos, cuya labor es gestionar los recursos que dispone entre las distintas emergencias que le han sido asignadas.
- Los solicitantes y proveedores de recursos, que tienen como funciones a poder desarrollar las mencionadas en los dos anteriores tipos de usuario en su conjunto, es decir, pueden añadir una nueva emergencia al sistema al igual que pueden proveer recursos a las emergencias que se les hayan sido asignadas.
- Finalmente, cabe mencionar a los administradores, que podrán disponer de una visión general de las emergencias, tanto aquellas que ya hayan sido satisfechas como las que no.

Figura 3.4: Tipos de usuario en el sistema web

3.3.2 Página principal, autenticación y registro

La primera experiencia del usuario ante la interfaz web es la página principal, en la cual se le brindara la posibilidad de realizar dos acciones diferentes. Por un lado, el inicio de sesión en caso de disponer de una cuenta y por otro el registro de un nuevo usuario en la plataforma. Ambas opciones se encuentran disponibles en el menú superior de la página principal como se muestra a continuación en la figura 3.5.

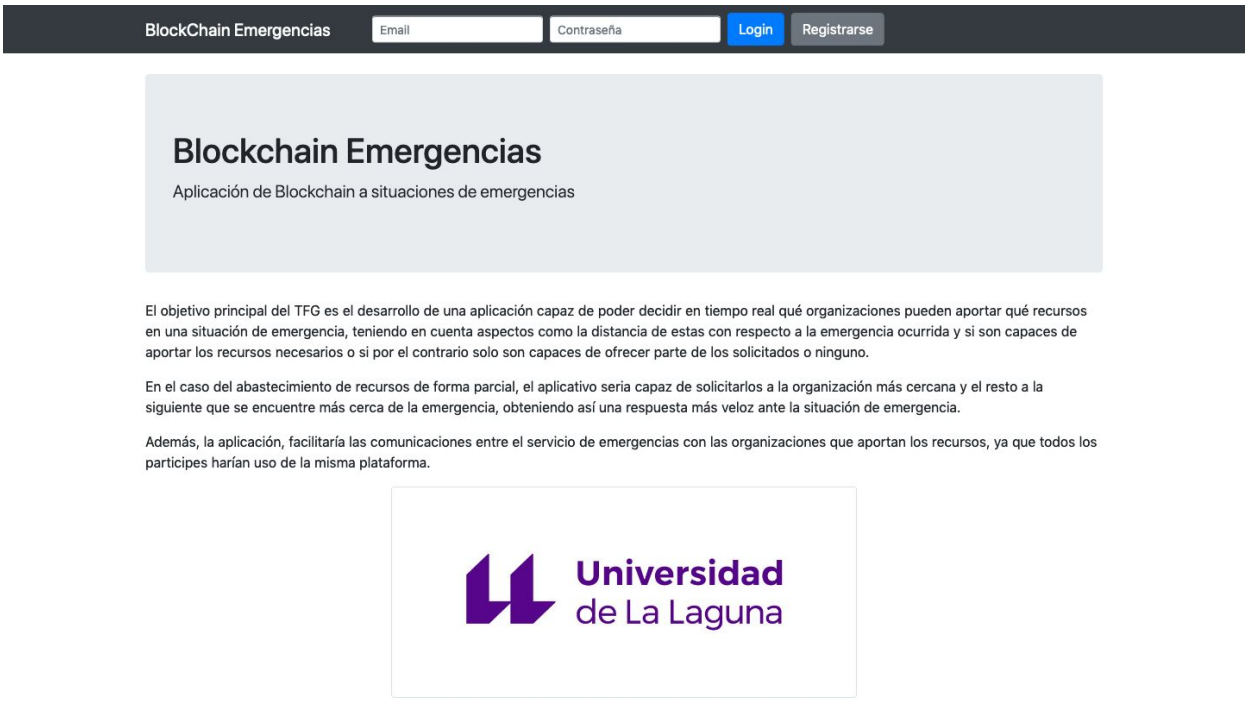


Figura 3.5: Página de inicio de la aplicación web

Si el cometido del usuario en la página de inicio es iniciar sesión, deberá de introducir el email de la cuenta y la contraseña establecida directamente en los campos indicados para ello en el menú principal.

Por otra parte, en el caso de que el individuo desee registrarse en la plataforma, deberá de acudir al botón denominado Registrarse. Tras hacer clic en él, le redirigirá al formulario de registro que podemos observar en la figura 3.6, mostrándole los siguientes campos a rellenar:

- Tipo de usuario (operador, proveedor o ambos a la vez).
- Email y contraseña, que posteriormente serán usados para la autenticación.
- Nombre de la organización a la que pertenece.
- Ubicación de la organización, con posibilidad de ser especificada mediante dirección postal, coordenadas o localización del dispositivo desde el que se accede.
- Flota de recursos, especificando la cantidad de cada tipo.

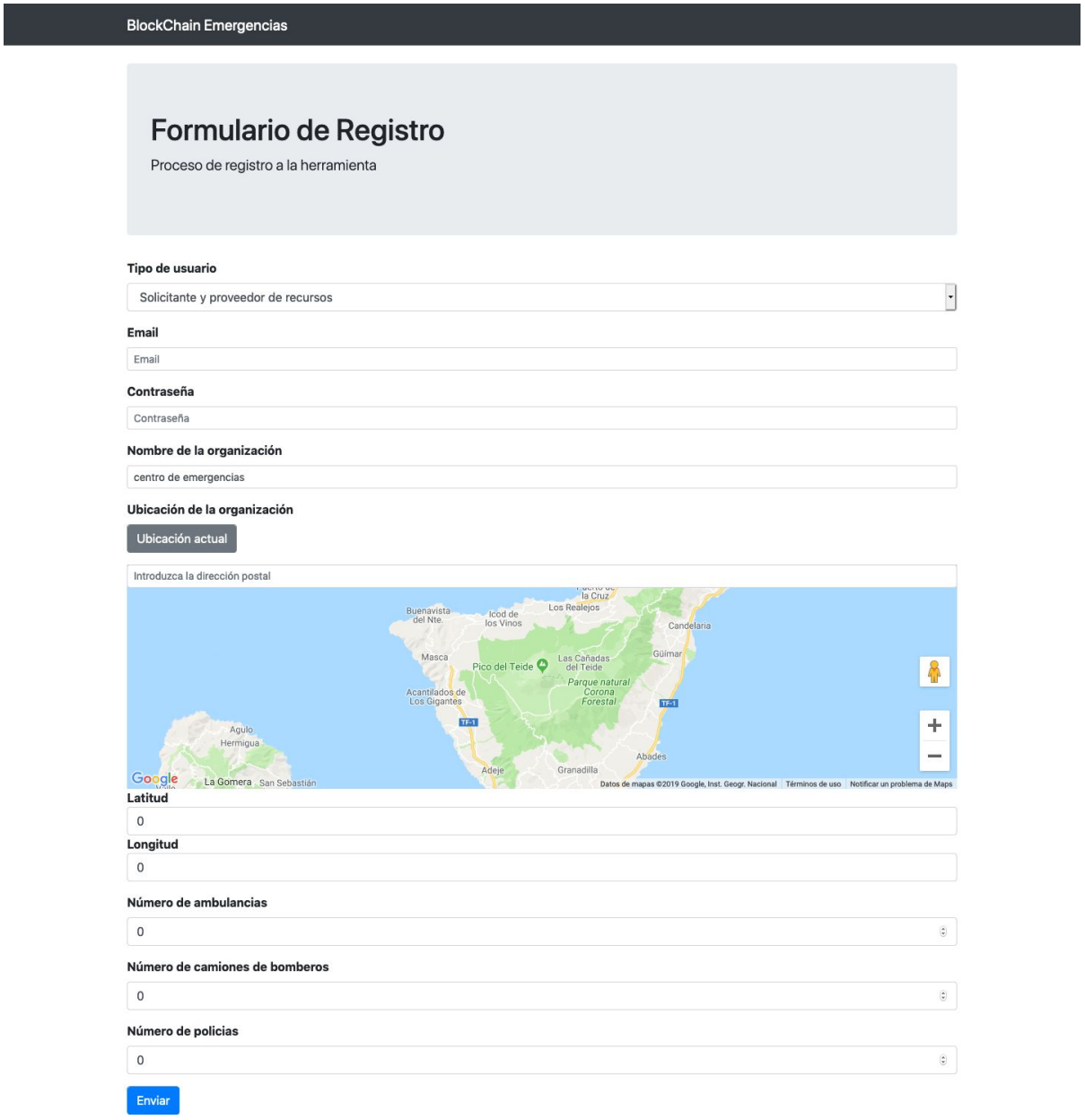


Figura 3.6: Formulario de registro del usuario

3.3.3 Solicitud y abastecimiento de recursos

En caso de que se desee registrar una nueva emergencia ocurrida, el usuario deberá de cumplimentar el formulario para emergencias (ver Figura 3.7) compuesto por los siguientes campos:

- Tipo de Emergencia (incendio, accidente de tráfico, homicidio, etc.).
- Tipo de alertante (propio afectado, relacionado con el afectado o externo).
- Datos del alertante
- Datos del afectado (en caso de que el alertante no coincida con el afectado).
- Ubicación de la emergencia, las formas para especificarla son las mismas que en el formulario de registro del usuario.
- Número de recursos necesarios.
- Organizaciones encargadas de satisfacer los recursos demandados.

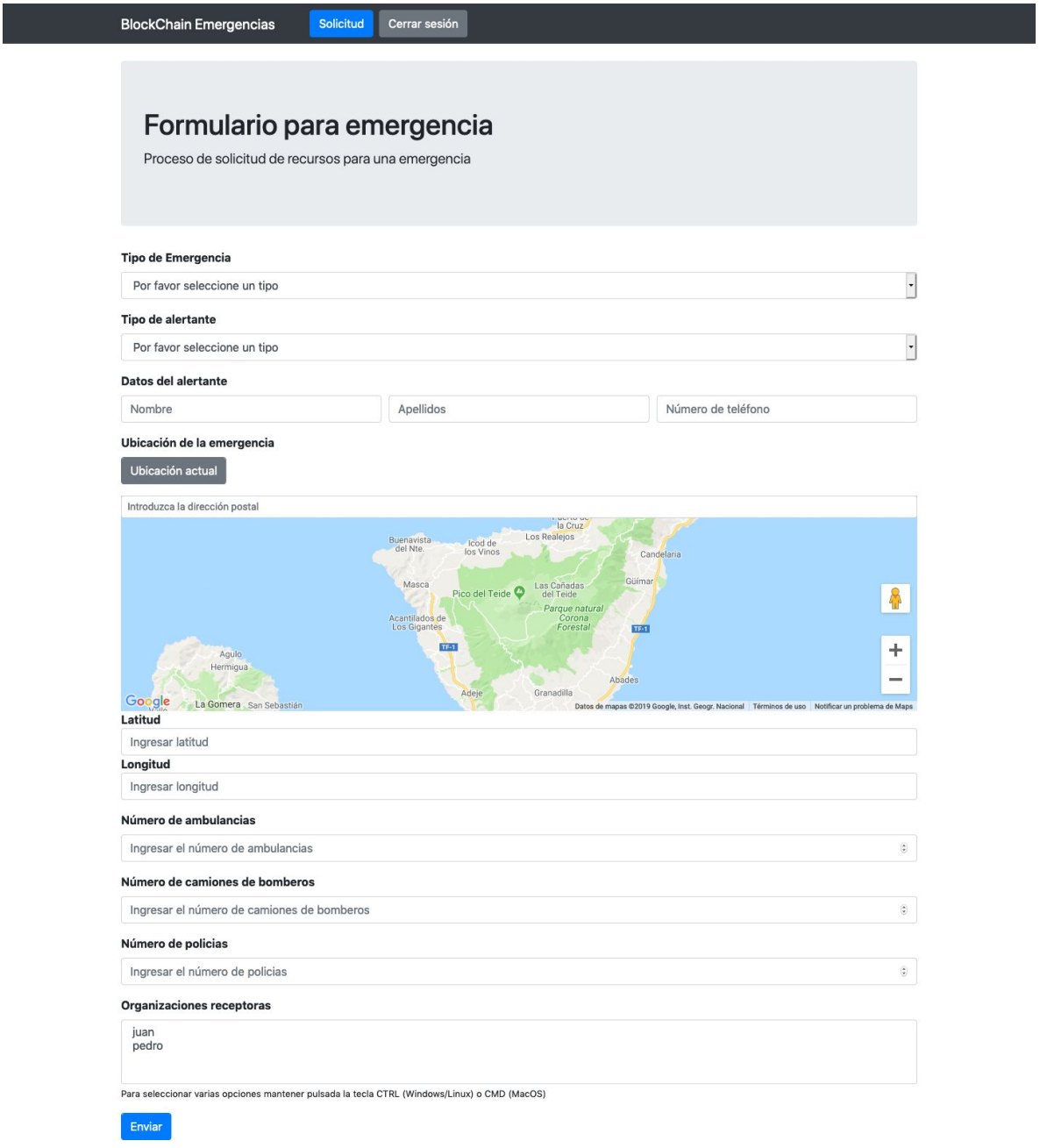


Figura 3.7: Formulario para notificar una emergencia

Desde el punto de vista del envío de recursos se pondrá a disposición del individuo un

panel de gestión de recursos, como el mostrado en la figura 3.8. En el mismo aparecerán los recursos disponibles a enviar y por otro lado, las distintas emergencias asignadas a satisfacer.

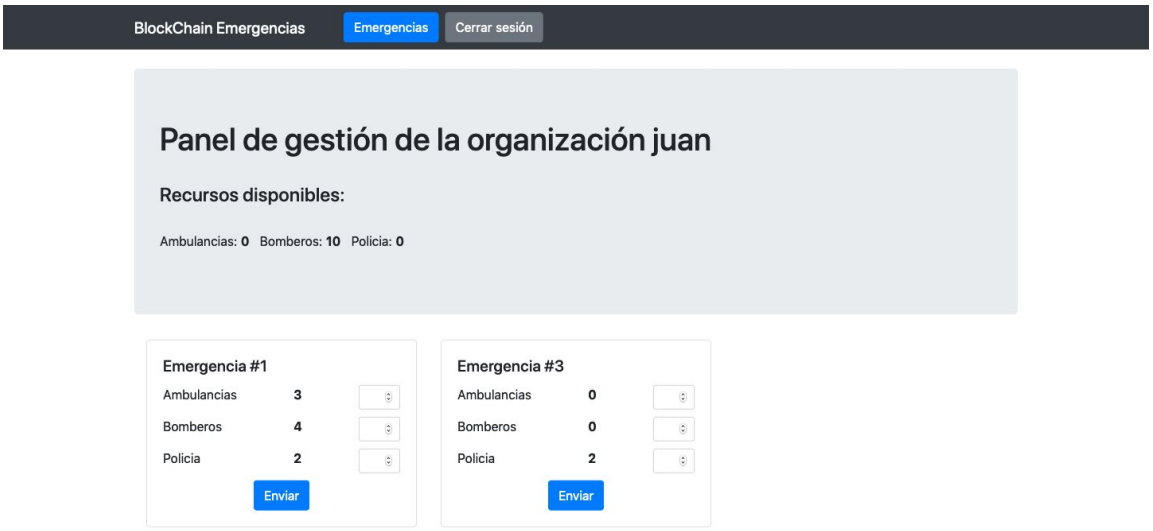


Figura 3.8: Panel de gestión de recursos

3.3.4 Panel de administración

Debido a que se ha contemplado la posibilidad de un usuario administrador, este dispondrá de un panel de administración (ver Figura 3.9) que aportará una visión de todas las emergencias y de sus respectivos recursos a satisfacer. Además se le brinda un filtro de visualización, con el cual podrá decidir si quiere ver sólo las emergencias pendientes de satisfacer, las que ya lo han sido o ambas.

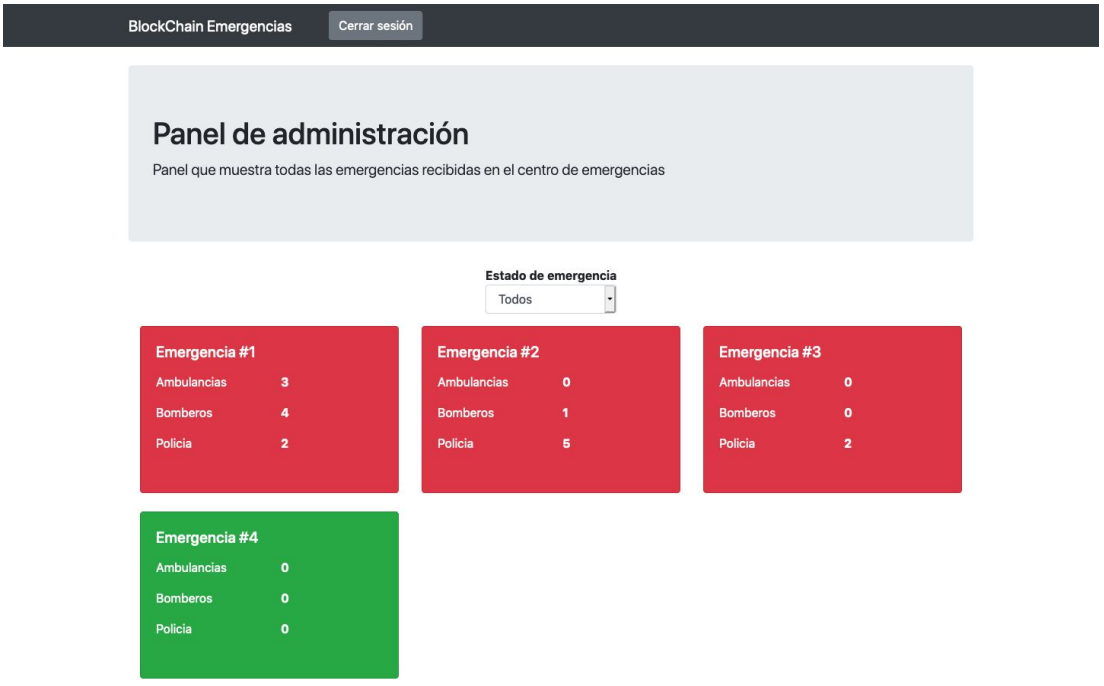


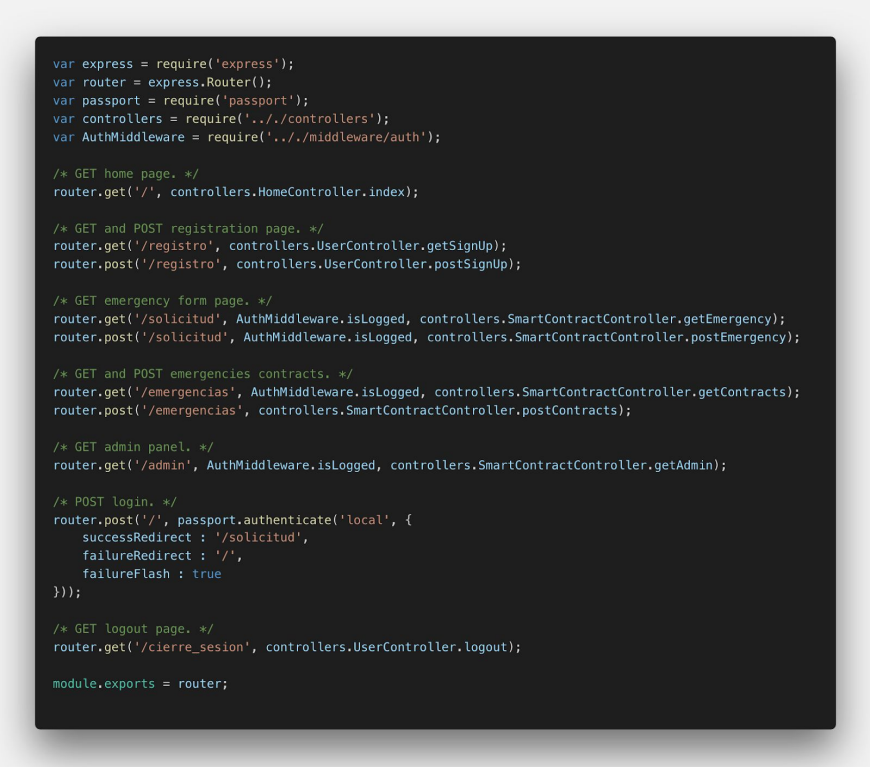
Figura 3.9: Panel de administración

3.4 Combinación de sistemas

Con el objetivo de enlazar el sistema web de la aplicación con la Blockchain desarrollada, se ha optado por el uso de múltiples formularios, para la creación de varias instancias del contrato inteligente que representarán las emergencias en el sistema, el envío de recursos a las mismas y la obtención de información acerca del estado de estas.

Todos los formularios hacen llegar la información cumplimentada en ellos mediante el uso de un método POST, pues el direccionamiento de los mismos ha sido configurado haciendo uso del método Router del framework Express [22], dicha configuración puede observarse en la figura 3.10.

Para la obtención de datos sobre la emergencia se ha empleado el mismo procedimiento de los formularios, pero en esta ocasión haciendo uso del método GET.



```
var express = require('express');
var router = express.Router();
var passport = require('passport');
var controllers = require('.././controllers');
var AuthMiddleware = require('.././middleware/auth');

/* GET home page. */
router.get('/', controllers.HomeController.index);

/* GET and POST registration page. */
router.get('/registro', controllers.UserController.getSignUp);
router.post('/registro', controllers.UserController.postSignUp);

/* GET emergency form page. */
router.get('/solicitud', AuthMiddleware.isLogged, controllers.SmartContractController.getEmergency);
router.post('/solicitud', AuthMiddleware.isLogged, controllers.SmartContractController.postEmergency);

/* GET and POST emergencies contracts. */
router.get('/emergencias', AuthMiddleware.isLogged, controllers.SmartContractController.getContracts);
router.post('/emergencias', controllers.SmartContractController.postContracts);

/* GET admin panel. */
router.get('/admin', AuthMiddleware.isLogged, controllers.SmartContractController.getAdmin);

/* POST login. */
router.post('/', passport.authenticate('local', {
    successRedirect : '/solicitud',
    failureRedirect : '/',
    failureFlash : true
}));

/* GET logout page. */
router.get('/cierre_sesion', controllers.UserController.logout);

module.exports = router;
```

Figura 3.10: Configuración de direccionamiento

3.4.1 Creación de emergencias

Una vez el usuario recopila los datos de la emergencia y los envía, gracias a la configuración de la figura 3.10, se crea una nueva instancia del contrato inteligente, estableciendo organizaciones receptoras y recursos solicitados.

La función mostrada en la figura 3.11 recopila los datos de la emergencia necesarios para el contrato inteligente, los cuales posteriormente son proporcionados a una función javascript desarrollada, la cual se muestra en la figura 3.12, para que con dichos datos se cree la nueva instancia del contrato inteligente.

```
postEmergency : function(req, res, next)
{
    var resources = {
        ambulances : req.body.ambulances,
        firefighters : req.body.firefighters,
        police : req.body.police,
    };
    req.flash('info', 'Se ha enviado correctamente la emergencia');
    smart_contract.set_resources(resources,req.user,req.body.organizations);
    return res.redirect('/solicitud');
},
```

Figura 3.11: Recopilación de información de la emergencia

```
set_resources : function (resources,user,organizations)
{
    const contract_creation = EmergencyContract.new({from: web3.eth.accounts[user.id-1], data: contract_json.bytecode, gas: 4712388, gasPrice: 100000000000});
    const contract_address = web3.eth.getTransactionReceipt(contract_creation.transactionHash).contractAddress
    var contractInstance = EmergencyContract.at(contract_address);
    let ambulances = resources.ambulances;
    let firefighters = resources.firefighters;
    let police = resources.police;
    contractInstance.set_resources(ambulances,firefighters,police,{from: web3.eth.accounts[user.id-1]});
    if(organizations.constructor === Array)
    {
        organizations.forEach(function(i) {
            contractInstance.set_organization(i,{from: web3.eth.accounts[user.id-1]});
        });
    }
    else
    {
        contractInstance.set_organization(organizations,{from: web3.eth.accounts[user.id-1]});
    }
    var contract = { address : contract_address };
    var config = require('../../database/config');
    var db = mysql.createConnection(config);
    db.connect();
    db.query('INSERT INTO contracts SET ?', contract, function(err, rows, fields){
        if (err) throw err;
        db.end();
    });
}
```

Figura 3.12: Creación de instancia del contrato inteligente

3.4.2 Envío de recursos a emergencias

En el panel de gestión de recursos, la visualización de las emergencias asignadas se consigue mediante el uso de un método GET sobre /emergencias. El enrutamiento del mismo se muestra en la figura 3.10.

Cuando se especifican los recursos a enviar a una emergencia, se está realizando un método POST sobre el formulario correspondiente, reduciendo así la cantidad de recursos disponibles por parte del gestor y los requeridos por la emergencia. Para ello se ha hecho uso de las funciones send_resources y check_request en ese orden, definidas en la estructura del contrato y ejecutadas como se puede observar en la figura 3.13.

```
    postContracts : function(req, res, next)
    {
        for(var i=1;i<4;i++){
            if(req.body['resource_' + i]=='')
                req.body['resource_' + i]='0';
        };
        var config = require('../database/config');
        var db = mysql.createConnection(config);
        db.connect();
        db.query('SELECT address FROM contracts WHERE id=?', req.body.contract_id, function(err, rows, fields){
            if (err) throw err;
            var contractInstance = EmergencyContract.at(rows[0].address);
            contractInstance.send_resources(req.body.resource_1,req.body.resource_2,req.body.resource_3,{from: web3.eth.accounts[req.body.user_id-1]});
            contractInstance.check_request({from: web3.eth.accounts[req.body.user_id-1]});
        });
        db.query('UPDATE users SET ambulances=?, firefighters=?, police=? WHERE id=?', [req.body.user_ambulances-req.body.resource_1, req.body.
user_firefighters-req.body.resource_2, req.body.user_police-req.body.resource_3, req.body.user_id], function(err, rows, fields){
            if (err) throw err;
        });
        req.flash('info', 'Se han enviado correctamente los recursos');
        return res.redirect('/emergencias');
    },
```

Figura 3.13: Envío de recursos

3.4.3 Panel de administración

Como en los dos casos anteriores, la visualización de las emergencias se consigue mediante un direccionamiento definido en base a un GET (/admin), se obtienen la cantidad de recursos pendientes de satisfacer por cada emergencia y su estado, mostrando el procedimiento realizado en la figura 3.14.

```
    getAdmin : function(req, res, next)
    {
        if(req.user.user_type == "Admin")
        {
            var config = require('../database/config');
            var db = mysql.createConnection(config);
            db.connect();
            db.query('SELECT * FROM contracts', function(err, rows, fields){
                if (err) throw err;
                var emergencies = [];
                var i;
                rows.forEach(function(i) {
                    var contractInstance = EmergencyContract.at(i.address);
                    emergencies.push({id: i.id, state: contractInstance.state(), ambulances: contractInstance.ambulances(),
 firefighters: contractInstance.firefighters(), police: contractInstance.police()});
                });
                return res.render('admin_panel', {
                    emergencies: emergencies,
                    message: req.flash('info'),
                    isAuthenticated : req.isAuthenticated(),
                    user : req.user,
                    title: 'Panel de administración'
                });
            });
        }
        else
        {
            return res.redirect('/');
        }
    }
```

Figura 3.14: Visualización de emergencias en el panel de administración

3.5 Seguridad

3.5.1 Autentificación de usuarios

Para garantizar una seguridad robusta en lo que respecta a la autentificación del usuario en la aplicación web se hace uso de Passport [23], un middleware de autenticación muy popular entre los desarrolladores NodeJS. Se escogió por la facilidad de alcance en lo que respecta a documentación y ayuda sobre su funcionamiento.

Con el objetivo de no almacenar las contraseñas en la base de datos, se ha hecho uso de una librería para NodeJS denominado Bcrypt [24], la cual añade la función hash bcrypt destinada a su uso sobre contraseñas [25] y funciones de comparación, que posteriormente son usadas para comprobar la contraseña introducida por el usuario con la registrada en la base de datos.

3.5.2 Control de acceso en el sistema web

En las acciones a realizar por cada direccionamiento dentro de la aplicación, se comprueba en todo momento el tipo de usuario autentificado para así verificar si tiene el rol correspondiente para acceder a el servicio deseado, en caso contrario se le redirige a la página que se considere más apropiada dependiendo de la situación.

Un ejemplo de dicha comprobación ocurre si el usuario autentificado no es de rol administrador, pues automáticamente es redirigido a la página principal en caso de que intente acceder al panel de administración.

3.5.3 Control de acceso en contratos inteligentes

Existen una serie de métodos de la estructura del contrato inteligente, que no resulta comprensible que sean usados por un usuario distinto del creador, como por ejemplo set_organization, ya que es el empleado para establecer las organizaciones receptoras del mismo. Para evitar que un usuario distinto al creador haga uso de estas funciones, se ha hecho uso de un modificador que se puede ver en la figura 3.15.

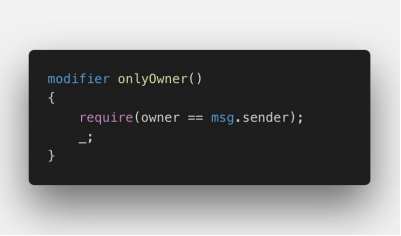


Figura 3.15: Método de tipo modificador en el contrato inteligente

En dicho modificador se establece una variable con la dirección de cartera del usuario creador del contrato, para posteriormente ser empleado en las funciones que deseamos que solo sean usadas por el creador, como en el ejemplo de la figura 3.16.

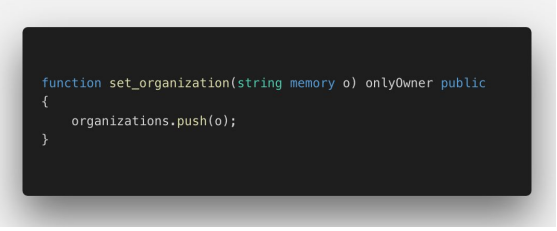


Figura 3.16: Función para establecer organización receptora

Capítulo 4 Pruebas realizadas

4.1 Frontend

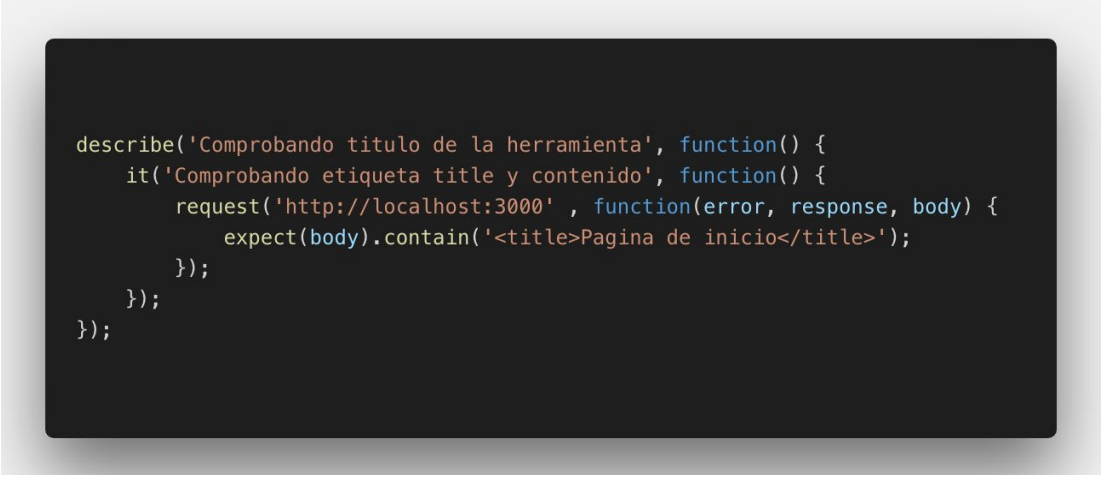
En cada una de las páginas que conforman la aplicación web hemos empleado una filosofía de desarrollo basado en pruebas (TDD), en la cual, en cada una de ellas, se ha verificado que correctamente el código de estado HTTP retornado es el correcto, como a continuación podemos ver en la figura 4.1, dichas pruebas han sido desarrolladas bajo la libreria Chai [26].

```
describe('Comprobando código de estado HTTP 200', function() {
    request('http://localhost:3000' , function(error, response, body) {
        expect(response.statusCode).to.equal(200);
    });
});

describe('Comprobando código de estado HTTP 404', function() {
    request('http://localhost:3000/about' , function(error, response, body) {
        expect(response.statusCode).to.equal(404);
    });
});
```

Figura 4.1: Comprobando códigos de estado HTTP sobre la página de inicio

Posteriormente, se comprueban aspectos como el título de la página (ver Figura 4.2), el contenido del menú, estructura del formulario en caso de tener que estar en la página y finalmente las dependencias javascript y css correspondientes (ver Figura 4.3).



```
describe('Comprobando titulo de la herramienta', function() {
    it('Comprobando etiqueta title y contenido', function() {
        request('http://localhost:3000' , function(error, response, body) {
            expect(body).contain('<title>Pagina de inicio</title>');
        });
    });
});
```

Figura 4.2: Comprobando título de la página principal

```
describe('Comprobando dependencias', function() {
    describe('Javascript', function() {
        it('Comprobando jquery', function() {
            request('http://localhost:3000' , function(error, response, body) {
                expect(body).contain('<script src="js/jquery.min.js"></script>');
            });
        });
        it('Comprobando bootstrap', function() {
            request('http://localhost:3000' , function(error, response, body) {
                expect(body).contain('<script src="js/bootstrap.min.js"></script>');
            });
        });
        it('Comprobando popper', function() {
            request('http://localhost:3000' , function(error, response, body) {
                expect(body).contain('<script src="js/umd/popper.min.js"></script>');
            });
        });
        it('Comprobando loading_button.min.js', function() {
            request('http://localhost:3000' , function(error, response, body) {
                expect(body).contain('<script src="js/loading_button.min.js"></script>');
            });
        });
    })
    describe('CSS', function() {
        it('Comprobando bootstrap', function() {
            request('http://localhost:3000' , function(error, response, body) {
                expect(body).contain('<link rel="stylesheet" href="css/bootstrap.min.css">');
            });
        });
    })
});
```

Figura 4.3: Comprobando dependencias de la página principal

4.2 Contratos inteligentes

Mediante el uso de Truffle, se han realizado distintas pruebas sobre la estructura del contrato, comprobando aspectos como el funcionamiento de las distintas funciones que lo conforman, revisando si realizan las acciones que se pretende que hagan (ver Figura 4.4) y que el acceso a las mismas sea el adecuado (ver Figura 4.5), dicha forma de realizar las pruebas, mantiene la filosofía de desarrollo basado en pruebas (TDD).

```
it('Inicializando contrato especificando cantidad de recursos', async () => {
  emergencyContractInstance.set_resources(3,4,5);
  let ambulances = await emergencyContractInstance.ambulances();
  let firefighters = await emergencyContractInstance.firefighters();
  let police = await emergencyContractInstance.police();
  assert.equal(ambulances.valueOf(), 3, 'Numero de ambulancias no es correcto');
  assert.equal(firefighters.valueOf(), 4, 'Numero de bomberos no es correcto');
  assert.equal(police.valueOf(), 5, 'Numero de policias no es correcto');
});
```

Figura 4.4: Comprobando asignación de recursos

```
it('Comprobando restricción creador de contrato', async () => {
  try {
    let result = await emergencyContractInstance.set_resources.call({from: account});
    assert.equal(result.toString(), owner);
  } catch (e) {
    console.log(`${account} no es el creador del contrato`);
  }
});
```

Figura 4.5: Comprobando restricción a solo creador de contrato

4.3 Automatización

Con el objetivo de automatizar la realización de las distintas pruebas desarrolladas para la aplicación web, se ha hecho uso del servicio de integración continua TravisCI, para ello se ha procedido por una parte a definir un script NodeJS en el fichero package.json para ejecutar las pruebas relativas al frontend haciendo uso del framework MochaJS [27], dicho script se ha denominado test y se puede visualizar a continuación en la figura 4.6.

```
"scripts": {
  "start": "node ./bin/www",
  "test": "mocha --exit test"
},
```

Figura 4.6: Script NodeJS para ejecutar pruebas frontend

El script NodeJS creado es usado en el fichero de configuración de TravisCI (ver Figura 4.7), para indicarle como ejecutar las pruebas del frontend y así se puedan realizar de forma automática. En lo que respecta a las pruebas sobre la estructura del contrato inteligente, se han indicado a TravisCI, mediante el comando correspondiente definido para este cometido por parte de Truffle, el cual es “truffle test”, las acciones de testing automático, en ambos casos, se ha indicado a TravisCI, en el mismo fichero de configuración, que han de ser realizadas únicamente cuando se perciben cambios en la rama master del repositorio git.

```
language: node_js
node_js:
- '11'

sudo: enabled
branches:
  only:
  - master
cache: yarn
before_install:
- npm install -g npm@latest
- npm install -g ganache-cli truffle
install:
- yarn install
script:
- yarn start &
- yarn test
- ganache-cli -p 7545 &
- cd blockchain-project
- truffle test
```

Figura 4.7: Fichero de configuración TravisCI

Capítulo 5 Presupuesto

Para aportar una visión aproximada de los costes que puede suponer el desarrollo e implementación de esta aplicación web, a continuación se procede a analizar este aspecto en detalle, teniendo en cuenta que se desee usar un blockchain privado.

5.1 Personal

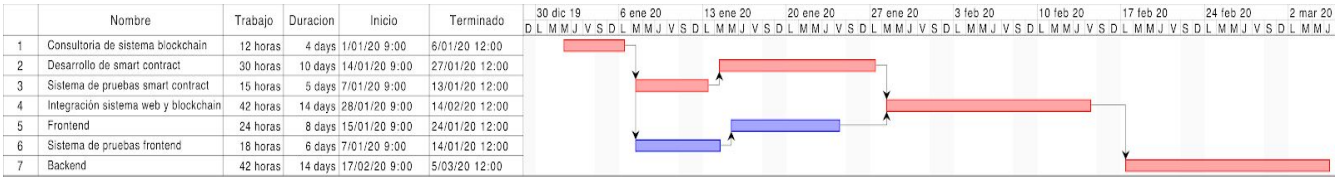
Actividad a realizar	**Horas**	**Coste por hora***	**Subtotal**
Consultoria de sistema blockchain	12	19€	228€
Desarrollo de smart contract	30	19€	570€
Sistema de pruebas smart contract	15	19€	285€
Integración sistema web y blockchain	42	19€	798€
Frontend	24	16€	384€
Sistema de pruebas frontend	18	16€	288€
Backend	42	16€	672€
Total	183	-	3225€

Tabla 5.1: Tabla resumen del presupuesto de personal

*Importes extraídos del salario bruto medio en España [28], de un desarrollador web (16€) y de un experto en ciberseguridad (19€), con una jornada laboral de 40 horas semanales.

5.2 Planificación de ejecución

Con el objetivo de aportar mayor comprensión sobre la planificación de actividades a realizar, se adjunta un diagrama de Gantt (ver Figura 5.1), el cual ha sido creado gracias al programa ProjectLibre [29].



	Nombre	Trabajo	Duracion	Inicio	Terminado
1	Consultoria de sistema blockchain	12 horas	4 days	1/01/20 9:00	6/01/20 12:00
2	Desarrollo de smart contract	30 horas	10 days	14/01/20 9:00	27/01/20 12:00
3	Sistema de pruebas smart contract	15 horas	5 days	7/01/20 9:00	13/01/20 12:00
4	Integración sistema web y blockchain	42 horas	14 days	28/01/20 9:00	14/02/20 12:00
5	Frontend	24 horas	8 days	15/01/20 9:00	24/01/20 12:00
6	Sistema de pruebas frontend	18 horas	6 days	7/01/20 9:00	14/01/20 12:00
7	Backend	42 horas	14 days	17/02/20 9:00	5/03/20 12:00

Figura 5.1: Diagrama de Gantt

5.3 Componentes

Recurso	Especificaciones	Precio Mensual*
Servidor web	Linux, 8GB RAM, 4 vCores y 20GB SSD	22€
Servidor de BBDD	Linux, 16GB RAM, 4 vCores y 1TB SSD	133€
Despliegue Blockchain	Linux, 64GB RAM, 16 vCores y 100GB SSD	130€
Dominio web	Extensión de dominio .es	0,84€
Total	-	285,84€

Tabla 5.2: Tabla resumen del presupuesto de componentes

*Importes extraídos de Clouding.io [30] en relación a servidores y de Godaddy [31] con respecto al dominio.

5.4 Coste total

Atendiendo a los importes calculados en los apartados anteriores, entre personal y componentes, el primer año de la aplicación supondría un coste total de 6655,08€.

Sin embargo, hay que tener en cuenta, que los costes derivados por los componentes empleados, se han de abonar de forma mensual siempre y cuando se desee mantener la aplicación web operativa, por tanto, tras el primer año, el coste anual sería de 3430,08€ en lo que respecta a esta parte.

Otro aspecto que es necesario tener en cuenta para calcular el coste total, es el mantenimiento, el cual suele rondar un importe del 30% con respecto al precio abonado a los desarrolladores, siendo en este caso el importe del mantenimiento de 968€, teniendo que realizar su pago de forma anual.

El coste total de forma anual, tras el primer año, teniendo en cuenta componentes y el mantenimiento de la aplicación constaría de 4398,08€.

Capítulo 6 Conclusiones y trabajos futuros

Alcanzado este capítulo, a continuación se presenta una serie de conclusiones obtenidas a lo largo del progreso de desarrollo de este trabajo fin de grado, además de una serie de posibles incorporaciones que se podrían realizar a futuro con el fin de seguir trabajando en la puesta en marcha de este trabajo en un entorno real.

6.1 Conclusiones

La aplicación web desarrollada mediante el uso de Blockchain y diversas herramientas, cuenta con las siguientes características:

- Las interacciones con el sistema son registradas en un sistema de información inmutable, consiguiendo así, evitar modificaciones sobre las mismas.
- Ofrece transparencia ante los usuarios que conforman la red, de toda acción realizada en la plataforma.
- Se garantiza una mayor seguridad del sistema al no encontrarse centralizada en una base de datos de única localización.
- Puede usarse en cualquier dispositivo y sistema operativo, siempre y cuando disponga de conexión a internet, debido a que se trata de una aplicación web.
- Permite el conocimiento de los recursos disponibles por cada organización en tiempo real.
- Es una aplicación única y de fácil integración con otras mediante el uso de APIs para la obtención de información.

Gracias al uso de contratos inteligentes, se ha conseguido aportar de autonomía a la la aplicación, pues sin necesidad de interacción por parte de ningún usuario, son los contratos los encargados de informar en todo momento de los recursos satisfechos o no en cada una de las emergencias introducidas en el sistema.

Debido a que la naturaleza de esta plataforma, partiendo de su desarrollo, es de una dapp, es decir de una aplicación descentralizada, se consigue aportar redundancia al sistema sin necesidad de coste adicional.

Además, gracias a que está planteado como un sistema distribuido, la escalabilidad resulta ser más económica de garantizar y de fácil implementación, ya que cada nodo nuevo que conforme la red, aporta su potencia computacional al sistema.

Este trabajo fin de grado ha sido y será presentado en forma de contribución en los congresos: VII Jornadas De Tecnología Y Nuevas Emergencias organizadas por el 112 Canarias y TLP Innova 2019 organizado por la asociación Innova 7.

6.2 Trabajos futuros

La aplicación web va destinada a un centro de emergencias, independientemente de la comunidad autónoma o país al que pertenezca. Sin embargo, por cercanía geográfica, se ha requerido la colaboración por parte del 112 Canarias para poder realizar una plataforma más fiel a la realidad que día a día afronta este centro de emergencias.

Siguiendo con esta idea en mente hemos considerado, que en caso de disponer de más tiempo para el desarrollo de este proyecto. Las siguientes mejoras serían las más adecuadas para una mejor integración a la estructura y forma de trabajo que integra el centro de emergencias:

- La potestad de seleccionar la cantidad de recursos necesarios para una emergencia no debería radicar en un operador de demandas, sino más bien en un individuo con este cometido, como el coordinador multisectorial. Por tanto, se debería de crear un rol de usuario con este cometido.
- Se pueden contemplar muchos más tipos de emergencia en la aplicación. Teniendo en cuenta los que son contemplados a día de hoy por parte del software usado en el centro de emergencias por parte de los operadores de demanda.
- Parte de la obtención de la localización de la emergencia, debería de recabarse mediante una API proveniente del software que gestiona la localización de los repetidores más cercanos a la llamada recibida.
- Para garantizar mayor seguridad en la autenticación de usuarios, lo adecuado sería que estuviera basada en el uso de blockchain, sin necesidad de un base de datos MySQL para este objetivo.
- Antes de entrar en la fase de despliegue de la aplicación web, sería recomendable probarla por parte de los distintos integrantes del centro de emergencia, desde el operador de demandas hasta un gestor de recursos de los distintos departamentos (sanitario, seguridad y salvamento), garantizando así que sea usable y agradable para los usuarios a los que va destinada.

Capítulo 7 Conclusions and future works

Reached this chapter, we will present below a series of conclusions obtained throughout the development progress of this final degree project. Also a series of possible changes or incorporations that could be made in the future in order to continue working in the incorporation of this work in a real environment are presented.

7.1 Conclusions

The application through the use of Blockchain and various tools has the following characteristics:

- The interactions with the system are recorded in an immutable information system, avoiding modifications to them.
- It offers transparency to the users that are in the network, and all actions carried out on the platform.
- The system security is guaranteed, because it is not centralized in a single location database.
- It can be used in any device and operative system, whenever there is internet connection because it is a web application.
- Knowledge of the resources available for each organization in real time.
- Unique application and easy integration with others using APIs to obtain information.

The use of smart contracts in the application allows to provide autonomy to it, without the necessity of interaction by any user. Therefore, they are responsible for reporting at all times of the resources satisfied and not in each of the emergencies introduced in the system.

Because the nature of this platform, based on its development, is a dapp (decentralized application), it provides redundancy to the system without the need for additional costs.

In addition, thanks to the fact that it is raised as a distributed system, scalability is cheaper to guarantee, and it is easier to implement, since each new node that conforms the network, contributes its computational power to the system.

This final degree project has been and will be presented as a contribution to the congresses: VII Technology and New Emergencies Journeys organized by the 112 of the Canary Islands and TLP Innova 2019 organized by the association Innova 7.

7.2 Future works

The application is intended for an emergency center, regardless of the autonomous community or country to which it belongs. However, due to geographical proximity, collaboration has been requested to the 112 of the Canary Islands, to be able to make a more realistic platform in collaboration with the people on this emergency center.

Continuing with this idea in mind, we have considered that in case of having more time for the development of this project, the following improvements that are going to be exposed would be the more appropriate for a better integration to the structure and form of work that integrate the emergency centers:

- The power to select the amount of resources needed for an emergency should not lie in a claims operator, but rather a staff with this role, such as the multisectoral coordinator. Therefore, a user role should be created with this task.
- The types of emergency contemplated in the application should be much more. Taking into account those that are contemplated today by the software used in the emergency center by demand operators.
- Part of obtaining the location of the emergency should be collected from the software of the API, that manages the location of the repeaters closest to the received call.
- The authentication of users to ensure greater security. It would be appropriate to use blockchain without the need of a MySQL database as an intermediary for this purpose.
- Before entering the deployment phase of the application, it would be advisable to give the application testing by the different members of the emergency center, from the claims operator to a resource manager of the different departments (health, safety and rescue), thus ensuring that it is usable and pleasant for the users to whom it’s intended.

Bibliografía

[1] Nelson Rodriguez. Historia de la tecnología Blockchain: Guía definitiva. 101blockchains. Dec 2018.

[2] Satoshi Nakamoto. Bitcoin: Un Sistema de Efectivo Electrónico Usuario-a-Usuario. Bitcoin.org.

[3] Hyperledger. https://www.hyperledger.org/. Último acceso 2019-05-29.

[4] Ethereum. https://www.ethereum.org/. Último acceso 2019-05-29.

[5] Linda Xie. A beginner's guide to Ethereum. Medium. Feb 2017.

[6] ¿Qué es el hash?. Bit2Me Academy. Jan 2019.

[7] Qué es el Árbol de Merkle. Bit2Me Academy. May 2019.

[8] ¿Qué es la clave pública?. Bit2Me Academy. Feb 2019.

[9] Qué es Blockchain (Introducción). Bit2Me Academy. Apr 2019.

[10] Qué es un bloque dentro de la blockchain. Bit2Me Academy. Oct 2018.

[11] Qué es Prueba de trabajo / Proof of Work (PoW). Bit2Me Academy. Mar 2019.

[12] NodeJS. https://nodejs.org/. Último acceso 2019-05-18.

[13] Joanna Staromiejska. 8 Examples of Node.js Development in Enterprise Products. Monterail. Oct 2018.

[14] MySQL. https://www.mysql.com/. Último acceso 2019-05-18.

[15] Jason Wong. The 6 Most Common Blockchain Programming Languages. Verypossible. Aug 2018.

[16] Solidity Documentation. https://solidity.readthedocs.io/. Último acceso 2019-05-18.

[17] Truffle Suite. https://truffleframework.com/. Último acceso 2019-05-18.

[18] Boostrap. https://getbootstrap.com/. Último acceso 2019-05-18.

[19] PugJS. https://pugjs.org/. Último acceso 2019-05-20.

[20] Web3JS Documentation. https://web3js.readthedocs.io/. Último acceso 2019-05-20.

[21] TravisCI. https://travis-ci.org/. Último acceso 2019-05-20.

[22] Express. https://expressjs.com/. Último acceso 2019-05-22.

[23] Passport. http://www.passportjs.org/. Último acceso 2019-05-22.

[24] Bcrypt Package. https://www.npmjs.com/package/bcrypt/. Último acceso 2019-05-22.

[25] Bcrypt Information. https://en.wikipedia.org/wiki/Bcrypt/. Último acceso 2019-05-22.

[26] ChaiJS. https://www.chaijs.com/. Último acceso 2019-05-22.

[27] MochaJS. https://mochajs.org/. Último acceso 2019-05-22.

[28] Montse Mateos. Cuál es el sueldo de los profesionales IT. Expansión.com. May 2018.

[29] ProjectLibre. https://www.projectlibre.com/. Último acceso 2019-05-26.

[30] Clouding.io. https://clouding.io/. Último acceso 2019-05-26.

[31] Godaddy. https://godaddy.com/. Último acceso 2019-05-26.