GRADO EN INGENIERÍA TELEMÁTICA

TRABAJO FIN DE GRADO

IMPLEMENTACIÓN DE UNA SOLUCION DISTRIBUIDA IOT MEDIANTE TECNOLOGIA BLOCKCHAIN

Autor: José María Moyano Suárez

Director: David Contreras Bárcena

Madrid

Julio 2019

Declaro, bajo mi responsabilidad, que el Proyecto presentado con el título

"Implementación de una solución distribuida IoT mediante tecnología BlockChain"

en la ETS de Ingeniería - ICAI de la Universidad Pontificia Comillas en el

curso académico 2018/19 es de mi autoría, original e inédito y

no ha sido presentado con anterioridad a otros efectos.

El Proyecto no es plagio de otro, ni total ni parcialmente y la información que ha sido

tomada de otros documentos está debidamente referenciada.

Fdo.: José María Moyano Suárez Fecha: 11/ 7/ 2019

Autorizada la entrega del proyecto

EL DIRECTOR DEL PROYECTO

Fdo.: David Contreras Bárcena Fecha: 11/ 7/ 2019

AUTORIZACIÓN PARA LA DIGITALIZACIÓN, DEPÓSITO Y DIVULGACIÓN EN RED DE PROYECTOS FIN DE GRADO, FIN DE MÁSTER, TESINAS O MEMORIAS DE BACHILLERATO

1º. Declaración de la autoría y acreditación de la misma.

El autor D. Jose María Moyano Suarez DECLARA ser el titular de los derechos de propiedad intelectual de la obra "IMPLEMENTACIÓN DE UNA SOLUCION DISTRIBUIDA IOT MEDIANTE TECNOLOGIA BLOCKCHAIN" que ésta es una obra original, y que ostenta la condición de autor en el sentido que otorga la Ley de Propiedad Intelectual.

2º. Objeto y fines de la cesión.

Con el fin de dar la máxima difusión a la obra citada a través del Repositorio institucional de la Universidad, el autor **CEDE** a la Universidad Pontificia Comillas, de forma gratuita y no exclusiva, por el máximo plazo legal y con ámbito universal, los derechos de digitalización, de archivo, de reproducción, de distribución y de comunicación pública, incluido el derecho de puesta a disposición electrónica, tal y como se describen en la Ley de Propiedad Intelectual. El derecho de transformación se cede a los únicos efectos de lo dispuesto en la letra a) del apartado siguiente.

3º. Condiciones de la cesión y acceso

Sin perjuicio de la titularidad de la obra, que sigue correspondiendo a su autor, la cesión de derechos contemplada en esta licencia habilita para:

a) Transformarla con el fin de adaptarla a cualquier tecnología que permita incorporarla a internet y hacerla accesible; incorporar metadatos para realizar el registro de la obra e incorporar "marcas de agua" o cualquier otro sistema de seguridad o de protección.
b) Reproducirla en un soporte digital para su incorporación a una base de datos electrónica, incluyendo el derecho de reproducir y almacenar la obra en servidores, a los efectos de garantizar su seguridad, conservación y preservar el formato.
c) Comunicarla, por defecto, a través de un archivo institucional abierto, accesible de modo libre y gratuito a través de internet.
d) Cualquier otra forma de acceso (restringido, embargado, cerrado) deberá solicitarse expresamente y obedecer a causas justificadas.
e) Asignar por defecto a estos trabajos una licencia Creative Commons.
f) Asignar por defecto a estos trabajos un HANDLE (URL *persistente).*

4º. Derechos del autor.

El autor, en tanto que titular de una obra tiene derecho a:

a) Que la Universidad identifique claramente su nombre como autor de la misma
b) Comunicar y dar publicidad a la obra en la versión que ceda y en otras posteriores a través de cualquier medio.
c) Solicitar la retirada de la obra del repositorio por causa justificada.
d) Recibir notificación fehaciente de cualquier reclamación que puedan formular terceras personas en relación con la obra y, en particular, de reclamaciones relativas a los derechos de propiedad intelectual sobre ella.

5º. Deberes del autor.

El autor se compromete a:

a) Garantizar que el compromiso que adquiere mediante el presente escrito no infringe ningún derecho de terceros, ya sean de propiedad industrial, intelectual o cualquier otro.
b) Garantizar que el contenido de las obras no atenta contra los derechos al honor, a la intimidad y a la imagen de terceros.
c) Asumir toda reclamación o responsabilidad, incluyendo las indemnizaciones por daños, que pudieran ejercitarse contra la Universidad por terceros que vieran infringidos sus derechos e intereses a causa de la cesión.

d) Asumir la responsabilidad en el caso de que las instituciones fueran condenadas por infracción de derechos derivada de las obras objeto de la cesión.

6º. Fines y funcionamiento del Repositorio Institucional.

La obra se pondrá a disposición de los usuarios para que hagan de ella un uso justo y respetuoso con los derechos del autor, según lo permitido por la legislación aplicable, y con fines de estudio, investigación, o cualquier otro fin lícito. Con dicha finalidad, la Universidad asume los siguientes deberes y se reserva las siguientes facultades:

- La Universidad informará a los usuarios del archivo sobre los usos permitidos, y no garantiza ni asume responsabilidad alguna por otras formas en que los usuarios hagan un uso posterior de las obras no conforme con la legislación vigente. El uso posterior, más allá de la copia privada, requerirá que se cite la fuente y se reconozca la autoría, que no se obtenga beneficio comercial, y que no se realicen obras derivadas.
- La Universidad no revisará el contenido de las obras, que en todo caso permanecerá bajo la responsabilidad exclusive del autor y no estará obligada a ejercitar acciones legales en nombre del autor en el supuesto de infracciones a derechos de propiedad intelectual derivados del depósito y archivo de las obras. El autor renuncia a cualquier reclamación frente a la Universidad por las formas no ajustadas a la legislación vigente en que los usuarios hagan uso de las obras.
- La Universidad adoptará las medidas necesarias para la preservación de la obra en un futuro.
- La Universidad se reserva la facultad de retirar la obra, previa notificación al autor, en supuestos suficientemente justificados, o en caso de reclamaciones de terceros.

Madrid, a 12 de Julio de 2019

ACEPTA

Fdo José María Moyano Suárez

Motivos para solicitar el acceso restringido, cerrado o embargado del trabajo en el Repositorio Institucional:

GRADO EN INGENIERÍA TELEMÁTICA

TRABAJO FIN DE GRADO

IMPLEMENTACIÓN DE UNA SOLUCION DISTRIBUIDA IOT MEDIANTE TECNOLOGIA BLOCKCHAIN

Autor: José María Moyano Suárez

Director: David Contreras Bárcena

Madrid

Julio 2019

Agradecimientos

Agradecer a mi tutor, David Contreras Bárcena por proponerme este proyecto y guiarme durante su ejecución.

A mis compañeros de carrera, me han aportado durante estos años motivación y apoyo.

A mis padres, por enseñarme el valor del esfuerzo y del trabajo, trasmitirme numerosas inquietudes y ser una fuente de conocimiento.

A mi hermana, por ser una luchadora y darme fuerzas cada día.

Al ser humano, por su apetito insaciable de seguir explorando y creando el mundo tecnológico, el cual me apasiona.

A todos, muchísimas gracias.

IMPLEMENTACIÓN DE UNA SOLUCION DISTRIBUIDA IOT MEDIANTE TECNOLOGIA BLOCKCHAIN

Autor: Moyano Suarez, Jose Maria
Director: Contreras Bárcena, David
Entidad Colaboradora: ICAI – Universidad Pontificia Comillas

RESUMEN DEL PROYECTO

En la era en la que vivimos muchas de las tecnologías tienden a converger. Las soluciones distribuidas ayudan a la confianza y trasparencia en las comunicaciones, y más en entornos destinados a escalar, como es el caso del IoT. En este proyecto se ha estudiado como el internet de las cosas(IoT) y la tecnología BlockChain, gracias al uso de contratos inteligentes (Smart Contracts), pueden darse la mano para aumentar la seguridad en el tratamiento de la información y el acceso requerido por entornos IoT, buscando el aumento de la confianza del usuario con este tipo de soluciones.

Palabras clave: IoT, BlockChain, Smart Contracts, Seguridad, Confianza, Trazabilidad sensores IoT

1. Introducción

Internet de las cosas (IoT) es una de las nuevas tecnologías emergentes más prometedoras en nuestros días. Esta tecnología se basa en la idea de que cualquier elemento, gracias al hardware y al software, es capaz de conectarse a la red IoT, por ejemplo, vehículos, electrodomésticos, dispositivos mecánicos, o simplemente objetos tales como calzado, muebles, maletas, dispositivos de medición, biosensores, o cualquier objeto que nos podamos imaginar. Sin embargo, muchas implantaciones IoT presentan problemas de seguridad como aparece en el IoT Security Market Report [], destacando las necesidades de mejora en áreas como la Autenticación/ Autorización, el Control de Accesos o la Encriptación de Datos .

Por otra parte, Blockchain ha demostrado poseer un nivel de seguridad muy evolucionado e inviolable hasta el momento, soportando la extensión de sistemas tan sensibles como las Cryptomonedas a través de Internet, que requieren un nivel extremo de seguridad y que pueden cubrir las necesidades de mejora de los Sistemas IoT.

Aprovechando ambas tecnologías, este proyecto se marca como objetivo demostrar que es viable la utilización de Smart Contracts de Blockchain para securizar de forma efectiva los conjuntos de datos obtenidos por los sensores IoT y su intercambio con los Sistemas de Gestión y control. De esta forma, se podrían eliminar los problemas de seguridad que están apareciendo en las implantaciones de Sistemas IoT.

2. Definición del Proyecto

Para la correcta realización y desarrollo del proyecto se definieron una serie de objetivos a cumplir.

El primer objetivo consistió en la realización de un estudio de las tecnologías a utilizar. Se ha realizado un análisis del estado actual del IoT y de BlockChain, para la detección de los problemas de seguridad que presentan los Sistemas IoT y encontrar cómo podría una cadena de bloques mitigar o solucionar los distintos problemas detectados.

Una vez cumplido el objetivo de estudio de las tecnologías, el siguiente objetivo planteado fue el desarrollo de un caso de uso el cual cumpliera con las motivaciones planteadas, para así resolver o mitigar los problemas detectados que se identificaron en el objetivo anterior.

Para la consecución del caso de uso y desarrollo del sistema planteado en su totalidad se identificaron varias fases de desarrollo e implementación:

- Desarrollo de nuevas funcionalidades en el Gateway IoT para la comunicación con la Rest-API.
- Desarrollo de una Rest-API, su objetivo es el almacenamiento temporal de los datos generados por el Gateway IoT.
- Desarrollo y despliegue de un contrato inteligente o Smart Contract, para la gestión distribuida de los datos de los sensores.
- Despliegue del Smart Contract sobre una red BlockChain, la cual permita el uso de las tecnologías seleccionadas.
- Desarrollo de un servidor el web, el cual permita al usuario interaccionar con dispositivos IoT de una forma segura y sencilla, realizando transacciones con el Smart Contract para la gestión de datos del dispositivo.

3. Descripción del Sistema

El caso de uso desarrollado quiere implantar una solución para la detección de dispositivos en un lugar determinado, por ejemplo, una entrada de un edificio, pudiéndose así generar un registro de trazabilidad del dispositivo, para así su posterior reporte si se produce una pérdida o sustracción.

Para la consecución del caso de uso planteado se han desarrollado dos módulos diferenciados: una Plataforma IoT y una Aplicación distribuida.

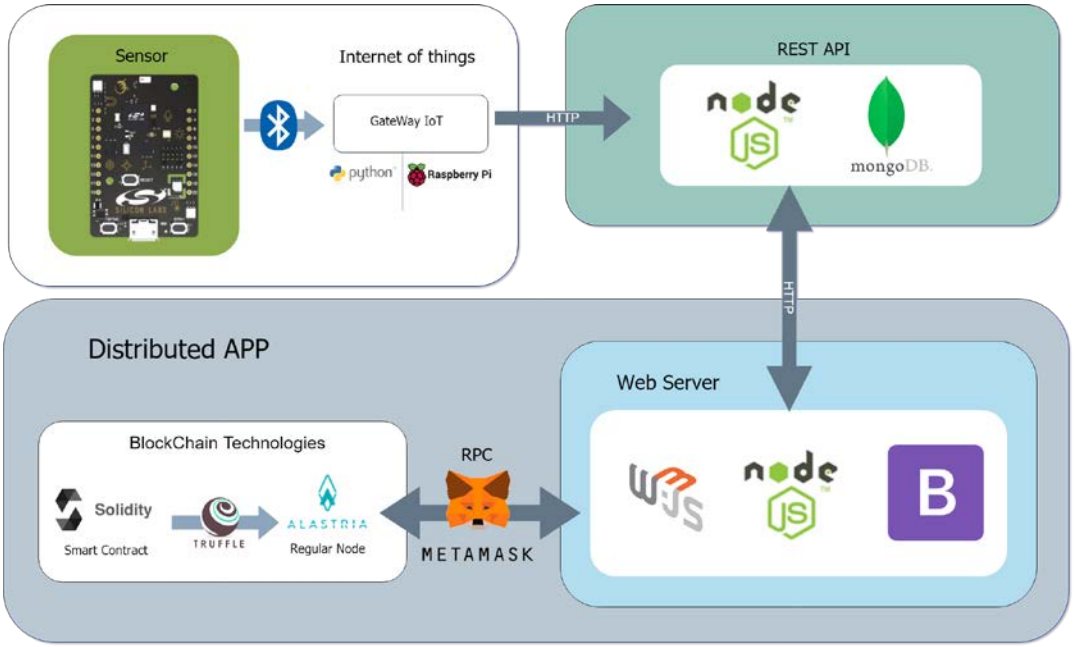


Figura 1. Arquitectura del sistema desarrollado

Como se puede observar en la Figura 1, la plataforma IoT desarrollada, es la encargada de la detección un sensor Thunderboard, a través de un módulo Bluetooth. Una vez esta ha detectado la información del sensor, se envía a la REST-API, utilizando el protocolo HTTP, guardando en esta la información del sensor detectado.

Una vez el servidor web necesita la información de la REST API, estos módulos se comunican mediante peticiones HTTP, para así obtener la información del dispositivo deseado.

El funcionamiento de la aplicación distribuida implementa la utilización de un servidor web, el cual aloja la aplicación desarrollada, la cual está basada en la gestión de la información de dispositivos IoT, implementando un registro de trazabilidad. Para una gestión de la información generada confiable y segura, se implementa el uso de los Smart Contracts, siendo desplegados gracias al uso de la tecnología Truffle sobre la red Alastria. Este contrato es llamado por el usuario desde el Servidor Web y gracias a la librería Web3 y a la herramienta Metamask, mediante el protocolo JSON RPC, se realiza una conexión con el contrato, para la ejecución de transacciones sobre el mismo, implantando así una forma segura y confiable de almacenar datos en una cadena de bloques, siempre que el usuario de su consentimiento.

4. Resultados

El Sistema desarrollado cumple con los objetivos planteados para la correcta consecución del proyecto. A través del despliegue de una aplicación web de un gestor de dispositivos, el cual permite la creación, actualización y reporte de perdida de uno o varios dispositivos, pudiendo realizar una trazabilidad de los dispositivos detectados, en un lugar determinado. En la figura 2, se puede ver como un dispositivo es creado

dentro del gestor, para luego la posterior actualización de este, utilizando la herramienta Metamask para la confirmación de estas transacciones realizadas.

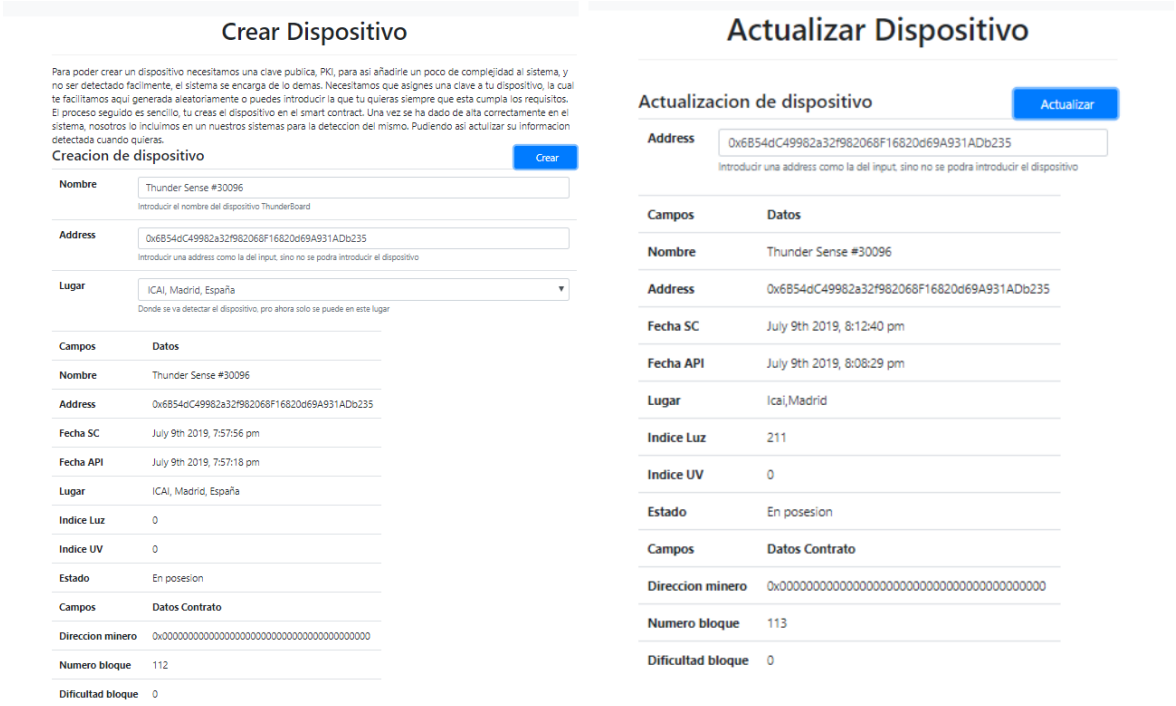


Figura 2. Creación y Actualización de un dispositivo de forma distribuida

La aplicación desarrollada cumple exactamente con el planteamiento realizado en el caso de uso, permitiendo observar las ultimas conexiones del dispositivo en el lugar indicado, y la correcta actualización de datos detectados por el sensor. Pudiendo así saber el estado actual del dispositivo.

5. Conclusiones

Como hemos planteado anteriormente, muchas de las soluciones IoT no son totalmente seguras. Por eso el objetivo del sistema desarrollado se ha centrado en resolver los problemas de seguridad detectados en IoT, como son la Autenticación, la Autorización y el Control de Accesos para una correcta gestión y protección de la información.

En este proyecto se ha comprobado que el uso de Blockchain, con sus Smart Contracts, mejora la seguridad de las soluciones IoT, ya que se puede utilizar como un sistema confiable en el cual implementar diferentes casos de uso. Más concretamente, se ha definido y desarrollado un sistema para la trazabilidad de los sensores IoT y el almacenamiento y gestión segura de los datos.

En este sentido se ha demostrado que el concepto de contrato inteligente se puede utilizar como base de datos distribuida para guardar y gestionar los datos generados por dispositivos IoT. Estos contratos al ser desplegados sobre una red distribuida como

BlockChain, son inmutables por lo que la información guardada en el mismo queda totalmente protegida de riesgos de fuga o modificación.

De esta forma el sistema de autenticación para la ejecución de transacciones dentro del contrato queda cubierto, ya que en un contrato solo las partes definidas en él pueden beneficiarse del mismo, y por tanto en un contrato inteligente, solo pueden acceder los intermediarios definidos en este, cubriendo así accesos no autorizados al mismo.

En resumen, se ha comprobado que el Blockchain y sus Smart Contracts son una excelente opción para mejorar sensiblemente el nivel de seguridad de las implantaciones de Sistemas IoT actuales.

6. Referencias

[1] IoT Analytics “IoT Security Market”, 2017.

https://iiot-world.com/reports/an-overview-of-the-iot-security-market-report-2017-2022/

IMPLEMENTATION OF A DISTRIBUTED IOT SOLUTION BY BLOCKCHAIN TECHNOLOGY

Author: Moyano Suarez, Jose Maria
Supervisor: Contreras Bárcena, David
Collaborating Entity: ICAI – Universidad Pontificia Comillas

ABSTRACT

In the era we are living in, many of the technologies tend to converge. Blockchain as a distributed solution helps to build trust and transparency in communications, and ir is even more important in environments destined to scale, such as the IoT. In this project we have studied how the Internet of Things (IoT) and BlockChain technology, thanks to the use of Smart Contracts, can come together to increase security in the information processing and the access required by IoT environments, seeking to increase the user confidence in this type of solutions.

Keywords: IoT, BlockChain, Smart Contracts, Trust, Security, Traceability

1. Introduction

Internet of Things (IoT) is one of the most promising new emerging technologies nowadays. This technology is based on the idea that any element, thanks to hardware and software, is able to connect to the network IoT, for example, vehicles, appliances, mechanical devices, or simply objects such as footwear, furniture, suitcases, measurement devices, biosensors, or any object that we can imagine. However, many IoT deployments present security problems as it appears in the IoT Security Market Report [1], highlighting the needs for improvement in areas such as Authentication / Authorization, Access Control or Data Encryption

On the other hand, Blockchain has demonstrated to have a level of security highly evolved and inviolable up to now, supporting the extension of systems as sensitive as the Cryptocurrencies through Internet, that require an extreme level of security and that can cover the needs of improvement of the IoT Systems.

Taking advantage of both technologies, this project aims to demonstrate the feasibility of using Blockchain Smart Contracts to effectively secure the datasets obtained by the IoT sensors and their exchange with the Management and Control Systems. In this way, the security problems that are appearing in the IoT Systems implementations could be eliminated.

2. Objectives

For the correct realization and development of the project, a series of objectives were defined.

The first objective was to carry out a study of the technologies to be used. An analysis of the current state of the IoT and of BlockChain has been completed to detect the security problems presented by the IoT Systems and to find out how a chain of blocks could mitigate or solve the different problems found.

Once the objective of studying the technologies was fulfilled, the following objective was the development of a use case which would comply with the stated motivations, in order to solve or mitigate the problems detected that were identified in the previous objective.

To achieve the case of use and development of the system as a whole, several phases of development and implementation were identified:

- Development of new functionalities in the IoT Gateway to communicate with the Rest-API.
- Development of a Rest-API with the objective of temporary storing of the IoT Gateway generated data.
- Development and deploy of a Smart Contract to manage the distributed data of the sensors.
- Deploy of the Smart Contract in a Blockchain Network allowing the use of the selected technologies.
- Development of a Web Server that allows the user to interact with IoT devices in a secure and easy way, making transactions with the Smart Contract to manage the data of the device.

3. System Description

The use case developed intends to implant a solution for the detection of devices in a certain place, for example an entrance of a building, being able to generate traceability record of the device, for its subsequent report if there is a loss or subtraction.

In order to achieve the proposed use case, two differentiated modules have been developed: an IoT Platform and a Distributed Application.

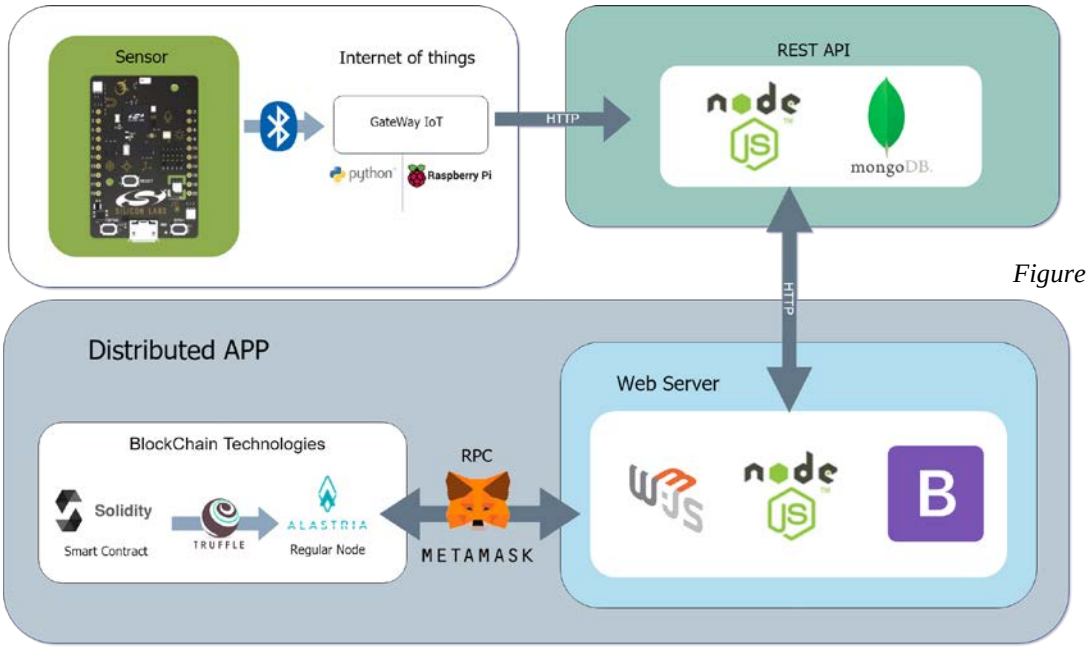


Figure 1 – Architecture of the developed System

As can be seen in Figure 1, the developed IoT platform is responsible for the detection of a Thunderboard sensor, through a Bluetooth module. Once it has detected the sensor information, it is sent to the REST-API, using the HTTP protocol, saving in this the detected sensor information.

Once the web server needs the information of the REST API, these modules communicate through HTTP requests, in order to obtain the information of the desired device.

The operation of the distributed application implements the use of a web server, which hosts the developed application, which is based on the management of IoT device information, offering a traceability record. For the management of reliable and secure generated information, the use of Smart Contracts is implemented, being deployed thanks to the use of Truffle technology over the Alastria network. This contract is called by the user from the Web Server and thanks to the Web3 library and the Metamask tool, through the JSON RPC protocol, a connection is made to the contract, for the execution of transactions on it, thus implanting a secure and reliable way to store data in a chain of blocks, as long as the user's give it consent.

4. Results

The developed system complies with the objectives set for the correct achievement of the project. Through the deployment of a web application of a device manager, which allows the creation, updating and reporting of loss of one or several devices, being able to perform a traceability of the detected devices, in a specific place. In Figure 2, you can see how a device is created within the manager, and then update it later, using the Metamask tool to confirm these transactions.

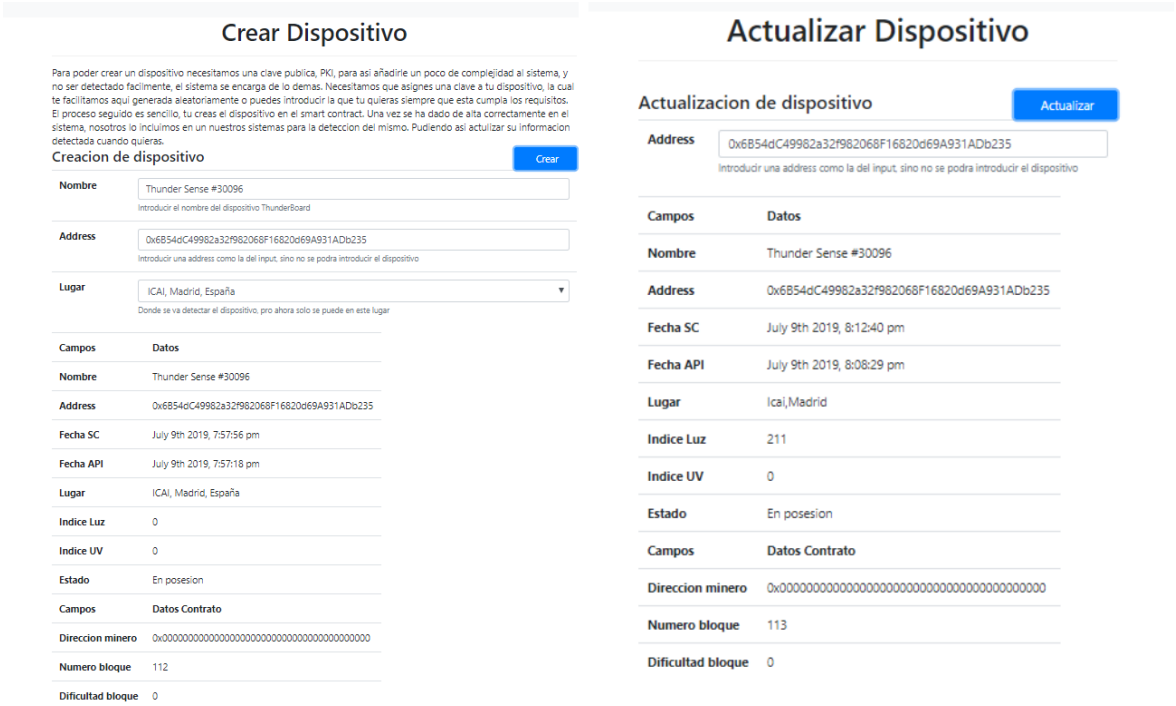


Figure 2 - Creation and update of a device in a distributed way.

The developed application complies exactly with the approach taken in the use case, allowing to observe the last connections of the device in the indicated place, and the correct update of data detected by the sensor. Being able to know the current state of the device.

5. Conclusions

As we have stated previously, many of the IoT solutions are not totally secure. That is why the objective of the developed system has been focused on solving the security problems detected in IoT, such as Authentication, Authorization and Access Control for a correct management and protection of information.

In this project it has been proven that the use of Blockchain, with its Smart Contracts, improves the security of IoT solutions, since it can be used as a reliable system in which to implement different use cases. More specifically, a system for the traceability of IoT sensors and the safe storage and management of data has been defined and developed.

In this sense, it has been demonstrated that the concept of intelligent contract can be used as a distributed database to store and manage the data generated by IoT devices. These contracts, when deployed over a distributed network such as BlockChain, are immutable, so the information stored in it is totally protected against risks of leakage or modification.

In this way the authentication system for the execution of transactions within the contract is covered, since in a contract only the parties defined in it can benefit from it, and therefore in an intelligent contract, only the intermediaries defined in this contract can access. , thus covering unauthorized access to it.

In summary, it has been proven that the Blockchain and its Smart Contracts are an excellent option to significantly improve the security level of the implementations of current IoT systems.

6. References

[1] IoT Analytics “IoT Security Market”, 2017.

https://iiot-world.com/reports/an-overview-of-the-iot-security-market-report-2017-2022/

Índice de la memoria

Índice de figuras

Índice de tablas

Capítulo 1. INTRODUCCIÓN

Internet de las cosas (IoT) es una de las nuevas tecnologías emergentes más prometedoras en nuestros días. Esta tecnología se basa en la idea de que cualquier elemento, gracias al hardware y al software, es capaz de conectarse a la red, por ejemplo, vehículos, electrodomésticos, dispositivos mecánicos, o simplemente objetos tales como calzado, muebles, maletas, dispositivos de medición, biosensores, o cualquier objeto que nos podamos imaginar, se pueden conectar al Internet de las Cosas (IoT). Estos elementos al estar conectados a la red transmiten y reciben información en tiempo real, permitiendo al ser humano automatizar muchos actos cotidianos, o incluso proporcionarnos información que antes nunca habíamos pensado que la tendríamos a nuestro alcance.

El ser humano en general no es consciente que sus datos son de gran valor, además muchas veces no se da cuenta que él mismo está generando información muy de mucho interés para la ciencia o la sociedad. La forma de aumentar la seguridad de esos datos es una tarea muy complicada y costosa, además de vital importancia para muchos entornos de usuarios, en los que debemos restringir la visualización y utilización de los datos. Muchas veces no se almacenan, gestionan o procesan los datos de una forma segura, confiable y transparente.

Esa cantidad ingente de datos necesita de muchas y variadas tecnologías para paliar y solucionar diversos problemas que aparecen, ya que el mundo tecnológico siempre está en desarrollo, afrontando nuevos retos. El Internet de las Cosas necesita un entorno en el cual la seguridad sea un punto clave, donde la confianza del usuario debería ser un punto principal en el desarrollo de este tipo de implantaciones. El objetivo es buscar entornos donde pueda desplegarse una solución IoT de una manera segura, para así poder así procesar los datos que se generan, ganando la confianza dicha anteriormente, para que así solo se tengan que procesar una vez esos datos, ahorrando el procesamiento de cantidades ingentes de datos, los cuales amenazan estos los requisitos de seguridad y confianza.

Para encontrar una solución a este problema, hoy en día existen tendencias tecnológicas que están revolucionando el mercado, industria y el entorno diario de las personas. Según un artículo publicado por Forbes technology Council, consejo compuesto de once expertos que explican y enumeran las tendencias tecnológicas más importantes en el sector, los puntos más destacados son: Incremento en la automatización, vuelta de BlockChain, mejora de la colaboración entre el humano y la Inteligencia artificial, expansión de los dispositivos conectados, mejora en ciberseguridad usando Machine learning e Inteligencia artificial, soluciones al rechazo (BackLash) tecnológico y convergencia tecnológica.[1] El análisis realizado es muy relista e interesante, destacando cuatro puntos clave que impulsaran la realización de este proyecto, siendo básicos para la consecución del mismo. Son los siguientes: Incremento en la automatización, vuelta de BlockChain, soluciones al rechazo BackLash tecnológico, y convergencia tecnológica.

Una solución, es la automatización de estos para poder así asegurar los requisitos de procesamiento y confianza. La automatización generó la industria, como la conocemos hoy, pero se busca una industria conectada, en la cual todos los sucesos que ocurran estén automatizados y los procesos realizados, sean almacenados de una manera confiable.

El punto principal que me llamó la atención es el uso de BlockChain como tecnología. Esta revolucionó la forma de entender como almacenar y construir sistemas totalmente distribuidos, confiables y escalables, siendo creada gracias a la elaboración de una Criptomoneda, el BitCoin. Como todas las nuevas tecnologías, siempre hay un primer rechazo por parte de la comunidad tecnológica. Un informe publicado por Deloitte apunta que esta tecnología está dejando atrás las barreras que se le impusieron desde un principio. El factor clave se debe al progreso de esta tecnología en cinco áreas diferentes. Esto ha supuesto un gran cambio a la hora de entender de verdad, lo que esta tecnología nos puede ofrecer. []

Cinco áreas de progreso en BlockChain

- Incremento rendimiento, procesamiento y almacenaje.
- Mejora en los estándares e interoperabilidad.

- Reducción de complejidad y coste.
- Soporte regulatorio.
- Consorcios de empresas y universidades (Alastria).

Como comentaba el artículo de Forbes, vamos a presenciar el mayor cambio tecnológico en una generación. Muchas tecnologías son de mentalidad abierta, por lo que el desarrollo de éstas nunca se sabe hasta dónde puede llegar. Para el desarrollo e integración de estas tecnologías, hay que tener un objetivo claro y conciso orientado a analizar, desarrollar y solucionar los problemas y necesidades detectadas.

El Internet de las cosas está destinado a explorar nuevas soluciones que resuelvan los problemas planteados y así intentar crear arquitecturas seguras y confiables. Por lo que se ha decidido implementar el uso de la tecnología BlockChain, en este caso una red BlockChain creada por un consorcio de Universidades y empresas, para poder así ayudarse de la comunidad de usuarios para el desarrollo del proyecto. La Universidad está especialmente implicada en proyectos sobre las cadenas de realizando ambiciosos proyectos en años anteriores. Además, en el año 2018 se implantó en la Universidad el primer nodo Blockchain universitario de la red Alastria (en el clúster que posee la universidad), de la cual es socio fundador. En este proyecto se quiere incluir el uso de la red Alastria para la consecución de los objetivos del proyecto, entorno a la tecnología BlockChain. La utilización de esta red inspirará nuevos casos de uso a desarrollar en la Universidad, para ir así mejorando su configuración e incrementando el interés de los alumnos por esta tecnología.

La principal idea del presente proyecto es la implementación de una red distribuida sobre un entorno IoT para validar y analizar, si gracias al uso BlockChain y otras tecnologías, se pueden resolver algunos de los actuales problemas del internet de las cosas, y desarrollando en este proyecto las cuatro ideas de desarrollo tecnológico mencionadas anteriormente.

En este proyecto se quiere implementar una solución distribuida, en la cual con la ayuda de un dispositivo IoT, mediante una conexión por BLE (se han realizados proyectos en la

universidad con este tipo de sensores) se pueda realizar una solución segura y confiable que el usuario pueda entender y manejar de una forma sencilla.

IoT y BlockChain son dos tecnologías que están encaminadas a darse la mano y colaborar entre sí, existe un gran potencial en ellas y se están desarrollando numerosos casos de uso, para ver si son compatibles. Este proyecto se centrará en ver cómo se pueden unir estas dos tecnologías, desarrollando una solución a baja escala, es decir, en cómo integrando una BlockChain podemos transmitir la confianza y seguridad necesaria para entornos distribuidos IoT.

Capítulo 2. DESCRIPCIÓN DE LAS TECNOLOGÍAS

2.1 PLATAFORMA IOT

2.1.1 GATEWAY IOT

El Gateway IoT (Este Gateway fue desarrollado en otro Trabajo Fin de Grado, y cedido por mi tutor para su uso en este proyecto) está implementado en una Raspberry Pi 3 Model B [3], con sistema operativo Linux Raspbian. Este emite una red WLAN con ssid "IoTGW". En esta red el gateway sirve una aplicación web (Figura 3) en la dirección "192.168.42.1:5050/scripts" desde la cual se activa el gateway ejecutando el script 4, llamado "On premise BlockChain".

Figura 3. Aplicación gateway e IOT

Una vez activado el Gateway, la Raspberry Pi 3 a través de un módulo Bluetooth, empieza recoger los datos de los sensores BLE cercanos y los envía mediante vía HTPP a una Rest-API desarrollada y explicada, en este proyecto. Todo el módulo creado para la detección y envío de peticiones ha sido desarrollado en Python 2.7.

2.1.2 SENSOR THUNDERBOARD

El sistema Gateway IoT, permite el uso de diferentes modelos de sensores. En este proyecto se ha trabajado con el sensor Thunderboard Sense de Silicon Labs [4].

Thunderboard™ de Silicon Labs:

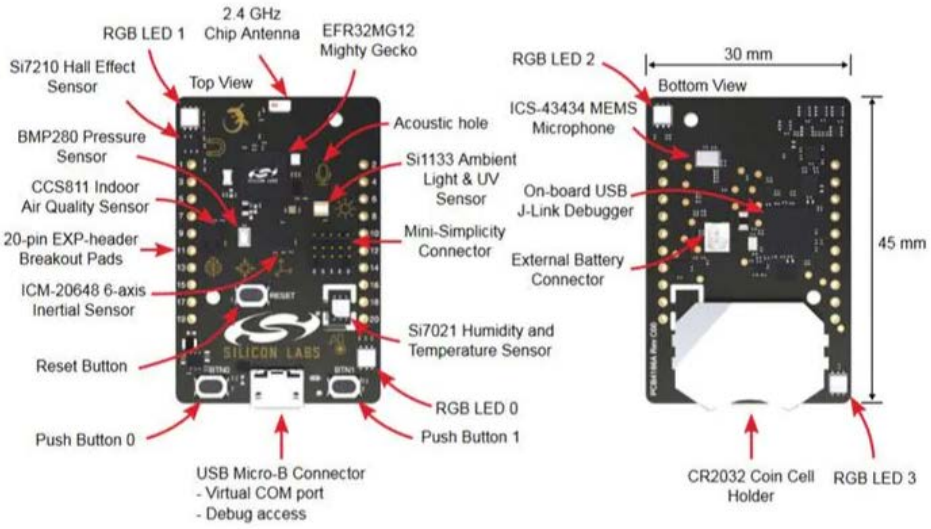


Figura 4. Sensor Thunderboard de Silicon Labs

Los sensores utilizados han sido, sensor que mide la luz solar y Índice UV(Ultravioleta). Estos kits de sensores envían cada 3 segundos las medidas al Gateway IoT a través de BLE (Bluetooth Low Energy) [4].

2.2 PLATAFORMA BLOCKCHAIN

2.2.1 BLOCKCHAIN

2.2.1.1 Introducción

Blockchain, o "Cadena de Bloques" en castellano, es la Tecnología que nació para soportar la primera Criptomoneda llamada "Bitcoin" y lanzada en 2009, aunque desde un año antes había aparecido ya en artículos y publicaciones técnicas. Para atender los requerimientos básicos de una Criptomoneda – no requerir la intervención de una Autoridad única confiable y eludir el control de Gobiernos o Bancos Centrales – el Blockchain está fundamentado en una base de datos distribuida que contiene todas las transacciones realizadas de esa moneda, replicada en todos los nodos que operan, accesible a todos los usuarios y con unos mecanismos de control descentralizados que impiden cualquier tipo de fraude, como gastar más de una vez un Bitcoin o la aparición de varios propietarios reclamando un mismo Bitcoin.

Gracias a esa arquitectura, Blockchain aporta todas las funcionalidades básicas requeridas por una Criptomoneda para poder operar con garantías en un mercado electrónico global como:

1. Disponibilidad: al estar replicada la cadena en todos los nodos, la indisponibilidad simultánea de todos ellos es casi imposible.

2. Inmutabilidad: los mecanismos de protección y vínculo criptográfico de cada bloque con el anterior y el siguiente en la cadena conjuntamente con la mencionada réplica en todos los nodos impiden además que pueda pasar desapercibida cualquier posible manipulación malintencionada en transacciones realizadas.

3. Comprobación de transacciones: cualquier usuario tiene acceso a la cadena y por tanto puede realizar con éxito las comprobaciones necesarias para comprobar la autenticidad de cualquier transacción.

4. Garantía de Verificación de las Nuevas Transacciones: el proceso de verificación de nuevas transacciones, su agrupamiento y la generación de un nuevo bloque en múltiples nodos, llamado "minado", y el "Mecanismo de Consenso" por el que se elige finalmente el bloque que se incorpora a la cadena son suficientemente complejos y exigentes para garantizar la autenticidad del bloque y su incorporación a la cadena.

Todas estas funcionalidades de la Tecnología Blockchain hacen que sea un Sistema confiable sin las figuras de una Autoridad única confiable o intermediarios y eso está extendiendo su aplicación no sólo a Criptomonedas como Ether, Litecoin, Dash, Monero … sino a un amplio abanico de otros Casos de Negocio como Registros de la Propiedad, Cadenas de Suministro, Compra-Venta de activos, Negociación de Contratos y un largo etcétera.

2.2.1.2 Bases de Funcionamiento

Como ya hemos adelantado, el Blockchain se basa en establecer una cadena de bloques enlazados que se replica en los ordenadores de todos los actores y usuarios. Cada bloque se compone de una serie de transacciones, cabeceras que lo vinculan a los bloques anteriores y posteriores en la cadena, un sello de tiempo y protecciones a través de mecanismos de seguridad a nivel de transacción y de bloque.

Más concretamente se utilizan mecanismos de seguridad de tipo "Hash" criptográfico que calculan un "Código Hash", también llamado Código Resumen, que representa a un conjunto de datos – que puede ser una transacción o un bloque en el caso de Blockchain - y que idealmente cambia con cualquier tipo de modificación que podamos introducir en el mismo por pequeña que ésta sea. De forma que si tenemos 2 conjuntos de datos y calculamos sus respectivos Códigos Hash podemos decir que si los Códigos son iguales entonces los conjuntos de datos originales también lo son. De forma complementaria, también podemos decir que si tenemos una transacción y su Hash originales grabados dentro de un bloque de Blockchain y para verificar su autenticidad procedemos a calcular

de nuevo su Hash y coincide con el original podemos concluir que la transacción original no ha sido manipulada ni modificada.

En el siguiente diagrama podemos ver la representación de las transacciones que conforman un bloque de Blockchain y los restantes componentes y vínculos que incluye[5]:

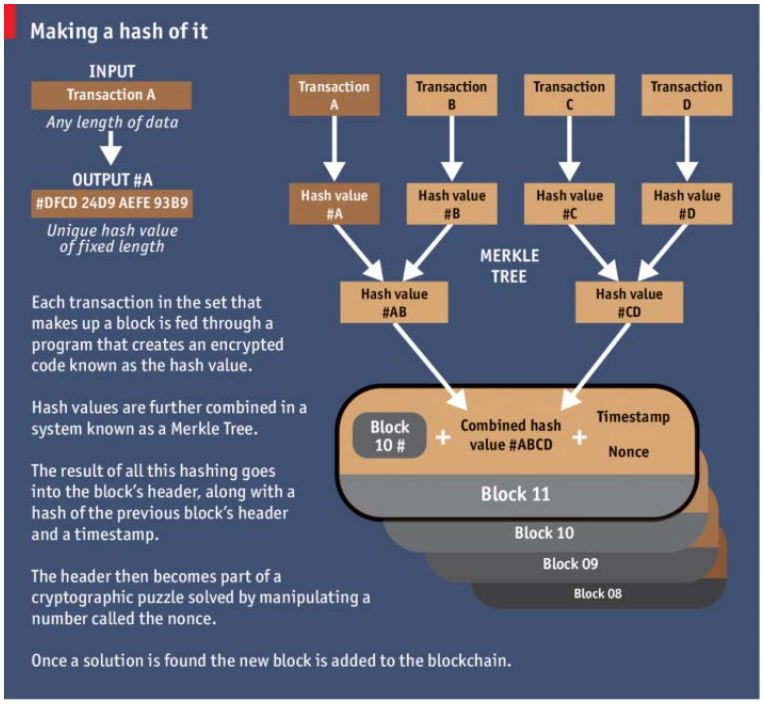


Figura 5. Estructura de un Bloque de Blockchain. (Fuente: The Economist, octubre 2015)

De esta forma tan elegante técnicamente se construye una cadena de bloques inmutable que está lista para ser distribuida en todos los nodos y que constituye una de las principales claves de Blockchain. Las otras dos claves son el Proceso de Generación de un nuevo bloque, llamado Minado, y el Mecanismo de Consenso por el que se selecciona el bloque definitivo que se incorporará a la cadena entre los propuestos por distintos nodos "mineros".

2.2.1.3 Proceso de Minado y Mecanismo de Consenso

Cuando se quiere realizar una transacción, por ejemplo, Antonio quiere traspasar 2 Bitcoins de su cuenta a la cuenta de Carmen en pago de bienes o servicios recibidos, se envía esta propuesta de transacción a todos los nodos de la red. A continuación, los nodos de la red verifican la propuesta de transacción, es decir comprueban que en la cuenta de Antonio hay más de 2 Bitcoins y que la cuenta de Carmen está lista para recibirlos analizando la cadena de Blockchain. Si la verificación es correcta, nodos especializados llamados mineros agruparán esta transacción con otras de la misma índole también verificadas en un nuevo bloque de Blockchain, calcularán el Código Hash de cada transacción y generarán la cabecera del bloque que contiene el Hash del bloque anterior de la cadena. El proceso continúa tomando la cabecera como la base de un problema matemático de prueba y error muy exigente que se debe resolver y que usa también la función Hash. Los nodos mineros deben analizar trillones de diferentes posibilidades hasta encontrar la respuesta. Una vez que varios nodos mineros propongan soluciones, otros nodos comprueban que son correctas y entre los bloques que superen la prueba se selecciona el definitivo mediante un Mecanismo de Consenso. Una vez seleccionado el bloque se recompensa al nodo que ha minado el bloque con unos Bitcoins, se calcula su Código Hash y ese será el Código de Identificación del Bloque en la cadena de Blockchain, insertándolo en la misma. A partir de ese momento, la nueva cadena de Blockchain es difundida a todos los nodos y la transacción entre Antonio y Carmen y todas las demás del bloque son confirmadas.

Como acabamos de ver, el rol que juega el Mecanismo o también llamado Algoritmo de Consenso es realmente muy importante ya que es el encargado de seleccionar el bloque alcanzando un acuerdo y evitando que algunos nodos dominen o manipulen la red. Para ello el Algoritmo más utilizado es el de la Prueba de Trabajo (PoW) donde se requiere que se demuestre haber realizado un proceso muy “costoso” en términos de computación, como es encontrar una solución al problema matemático planteado a partir de la cabecera del bloque, y que limita sensiblemente el número de nodos que participan. Este algoritmo es considerado un algoritmo firme y sencillo, aunque no es el más rápido.

Existen otros Algoritmos de Consenso como el de Prueba de Participación (PoS), Prueba de Participación Delegada (DPoS), Prueba de Importancia (PoI), Tolerancia a Firmas Bizantinas (BFT), Acuerdo Bizantino Federado (FBA) … que tienen distintas características y que en función de ellas pueden adaptarse mejor al Caso de Negocio.

2.2.2 SMART CONTRACTS

Los contratos inteligentes en inglés, *Smart Contracts*, son códigos ejecutables los cuales funcionan igual que un contrato en la vida real. Se definen reglas y acuerdos, en los que se establecen ciertas condiciones. Una vez se cumplen estas, determinadas acciones son realizadas para así el cumplimiento de este. Estos contratos son redactados en lenguajes de programación, por lo que son escuetos en su forma de entendimiento. Gracias a esa cualidad, no son interpretables, sino que se ciñen a cumplir lo acordado.

Para darle un uso real a este tipo de contratos, se despliegan sobre redes BlockChain. Al poder desplegarse sobre este tipo de redes, no hace falta un intermediario entre los dos puntos donde se realiza el acuerdo, sino que la red, gracias a las cualidades que posee, hace que estos se replican en cada nodo, siendo así inmutables. Pero la mejor cualidad, es que solo los usuarios participantes en el contrato pueden leerlo, ganando así una confianza extra.

Nuevas implementaciones se están desarrollando para el mundo de los Smart Contracts, en concreto, la que interesa en este proyecto, es la de un contrato inteligente como una base de datos segura, ya que podemos guardar cualquier tipo de información que queramos, la cual será segura e inmutable.

2.2.3 JSON RPC

Este protocolo está basado en dos tecnologías, JSON y RPC.

RPC son las siglas de llamada a procedimiento remoto, en ingles *Remote Procedure Call*, el cual sirve para llamar a procedimientos remotos y así establecer una conexión con el

sistema deseado. Una vez la conexión es establecida es posible la ejecución de código remoto, siendo la conexión segura.

JSON es un formato de texto sencillo para el intercambio de datos.

En los entornos BlockChain se utiliza JSON RPC para la conexión sobre la red BlockChain deseada, este se tiene que habilitar sobre la red donde se quiera trabajar.

2.2.4 WEB3

Librería que ofrece JavaScript, diseñada para la conexión e interacción con la cadena de bloques Ethereum. El uso de esta permite la interacción con las cuentas de la red deseada, la realización de transacciones e interacción con los contratos inteligentes que hayamos desplegado en la cadena de bloques. Utiliza HTTP o IPC como métodos de conexión.

2.2.5 SOLIDITY

Es un lenguaje de programación desarrollado para la elaboración de Contratos Inteligentes que se ejecutan en una Máquina Virtual Ethereum. Destaca por ser orientado a objetos y de alto nivel. Solidity se ha visto influenciado por lenguajes de programación como C++, Python y JavaScript. [6]

2.2.6 GANACHE

Ganache es una aplicación desarrollada por el consorcio de Truffle, dando soporte a la misma. En términos generales su función es la de desplegar una BlockChain en Ethereum, de manera muy rápida y sencilla (no es necesario desplegar ningún cliente Ethereum), poseyendo todas las cualidades que nos ofrece una arquitectura BlockChain. Entre sus funcionalidades se encuentra la personalización de la red a tu gusto, para así poder probar diferentes casuistas. Ofrece el control del estado de todas las cuentas de la red incluidas las direcciones, las claves privadas, el gas utilizado, las transacciones o los balances. Se define como una red Ethereum la cual se puede utilizar para realizar pruebas, ejecución de comandos, e inspeccionar el estado de la misma, mientas controlas el estado de la cadena. []

2.2.7 TRUFFLE

Ambiente de desarrollo cuya función principal es la compilación y despliegue de contratos inteligentes sobre redes Ethereum. Posee una consola la cual nos permite interaccionar con los contratos desplegados, pudiendo probar el correcto funcionamiento de los mismos.

2.2.8 REMIX

Remix herramienta de software libre, que te ayuda a la creación de Contratos Inteligentes en Solidity. Es una herramienta muy cómoda debido a que se pueden crear contratos inteligentes directamente desde el navegador o localmente. Estos contratos se pueden testear, realizar *debugs* y realizar despliegues del contrato sobre diferentes opciones que ofrece la plataforma. Escrito en JavaScript. [7-8]

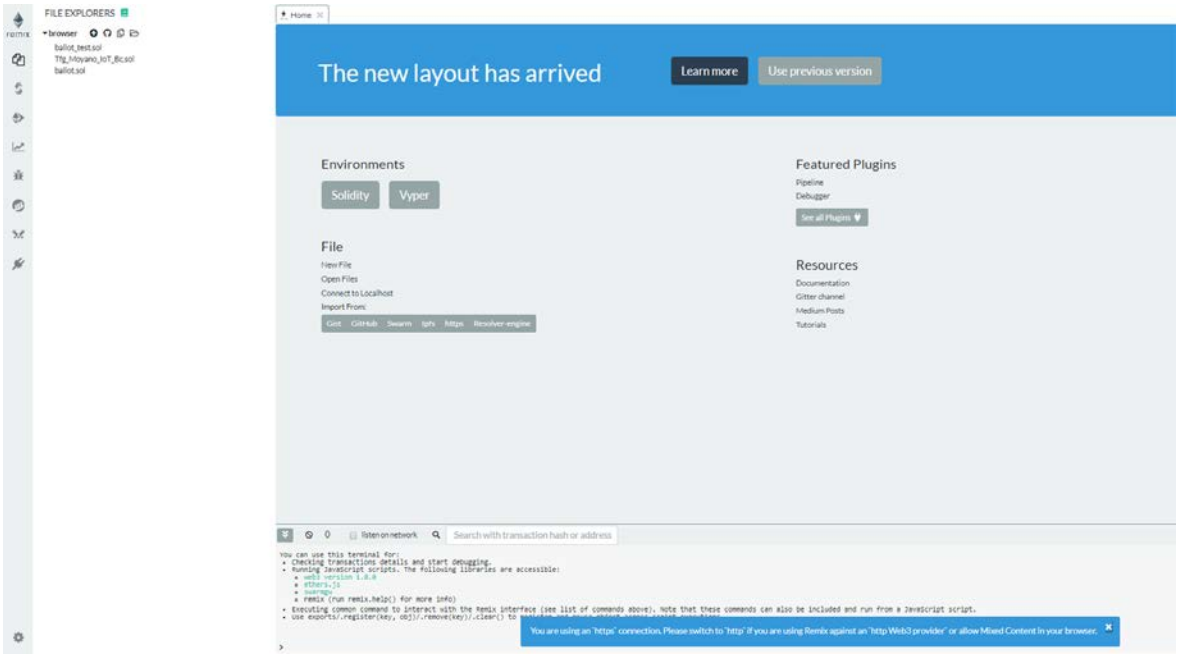


Figura 6. Remix IDE para desarrollo de Smart Contracts

2.2.9 METAMASK

Extensión ofrecida para varios navegadores, la cual ofrece una manera sencilla de gestionar una cartera de cuentas BlockChain, sobre Ethereum. Permite gestionar las cuentas de todo tipo de redes Ethereum, cuando se introduce su clave privada, además de incluir una clave privada para el uso de esta. Gracias a integración de una API web3, permite a Metamask poder conectarse con los interfaces web, y con la BlockChain, siendo el puente entre estos dos puntos. Destaca por la ejecución de transacciones seguras desde el mismo, sin tener que recurrir a la confirmación dentro del sistema.

2.3 OTRAS TECNOLOGÍAS UTILIZADAS EN EL PROYECTO.

2.3.1 NODE JS

NodeJS es un entorno que utiliza de JavaScript que está diseñado para generar aplicaciones web de forma altamente optimizada. Destaca por su escalabilidad como uno de sus objetivos principales. Por sus operaciones asíncronas, un servidor se encarga de ejecutar

diferentes tareas para facilitar la comunicación con los diferentes clientes. La importancia de los eventos, su arquitectura se basa en eventos y gracias a ella es posible generar un tipo de procesamiento asíncrono de operaciones de entrada y salida. Por su gestor de librerías npm (Node Package Manager), gestor que da acceso a un conjunto de librerías muy extenso, son gratuitas y generadas a partir de la colaboración de los usuarios de su comunidad.[]

2.3.2 MONGO DB

Sistema de base de datos NoSQL orientado a documentos de código abierto. En lugar de guardar los datos en tablas, tal y como se hace en las bases de datos relacionales, MongoDB guarda estructuras de datos BSON (una especificación similar a JSON) con un esquema dinámico, haciendo que la integración de los datos en ciertas aplicaciones sea más fácil y rápida.

2.3.3 DOCKER

Docker es un programa de código abierto que permite un sistema operativo y sus dependencias se empaqueten como un contenedor. La virtualización basada en contenedores aísla las aplicaciones entre sí en un sistema operativo (OS) compartido. Este enfoque estandariza la entrega del programa de la aplicación, permitiendo que las aplicaciones se ejecuten en cualquier entorno Linux, ya sea físico o virtual. Dado que comparten el mismo sistema operativo, los contenedores son portátiles entre diferentes distribuciones de Linux, y son significativamente más pequeños que las imágenes de máquinas virtuales (VM).

Capítulo 3. ESTADO DE LA CUESTIÓN

3.1 EL MUNDO DEL INTERNET DE LAS COSAS

El internet de las cosas (IoT), se podría catalogar como una de las nuevas tecnologías más importantes hoy en día. Su repercusión en el ámbito diario humano es muy elevada, pero su evolución avanza a pasos agigantados, impactando en múltiples ámbitos cotidianos e industriales. Este gran avance, posibilita el desarrollo de ciudades, industrias, hogares y espacios sostenibles, incrementando así su eficiencia y sostenibilidad, además de un control y trazabilidad.

Es una tecnología que está en auge, según "International Data Corporation (IDC)" se estima que el gasto en esta tecnología tendrá una tasa de crecimiento anual del 13,6% en el periodo de 2017 a 2022, año en el que llegará a los 1,2 billones de dólares. En la misma línea, Bain & Company vaticina que la inversión en los mercados combinados de IoT llegará a unos 520.000 millones de dólares en 2021 (más del doble de los 235.000 millones de 2017). Otros de los grandes factores por los que IoT está creciendo desmesuradamente es la cantidad de nuevos dispositivos conectados a la red, se estima que en 2020 habrá unos 20.400 millones de dispositivos conectados a la red , de los cuales el 95% de los productos nuevos implementará esta tecnología. En la siguiente figura se puede observar el crecimiento de los dispositivos conectados mundialmente en un periodo de 10 años, es una estimación, pero en diez años se habrá aumentado en 5 veces los dispositivos conectados a la red. [9]

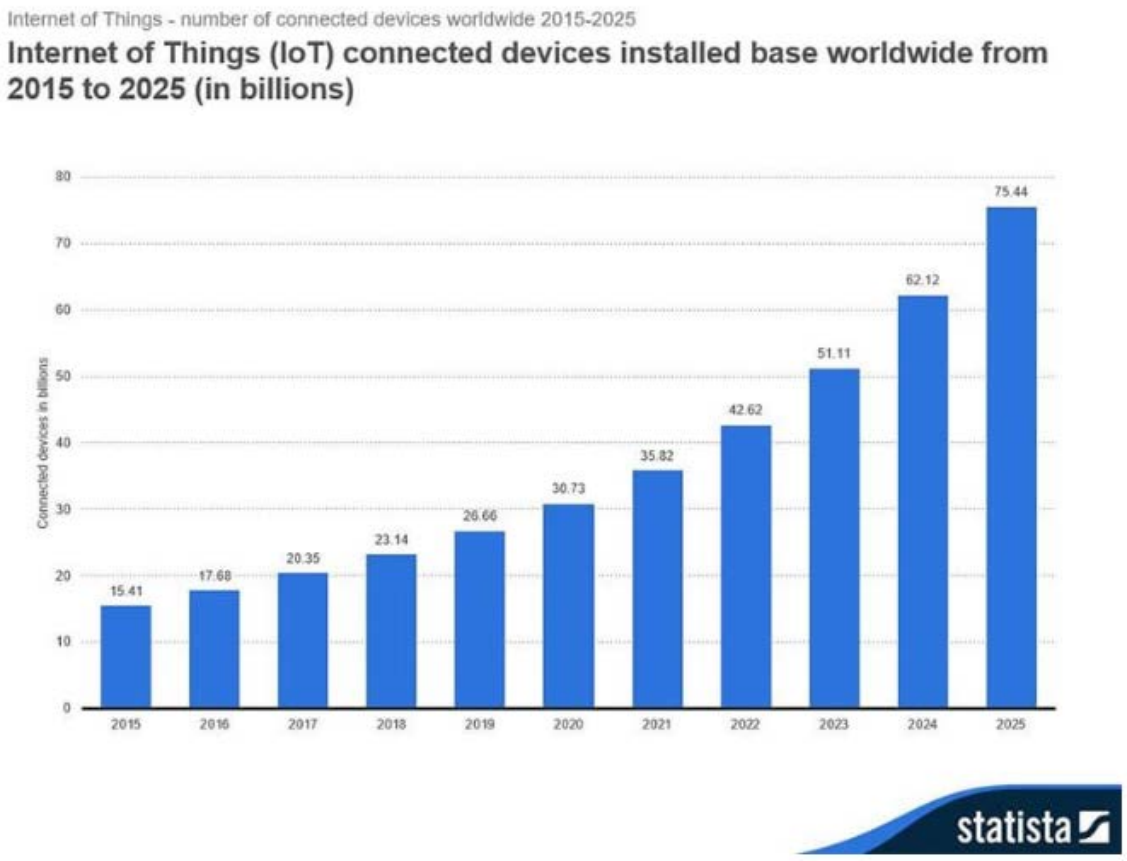


Figura 7. Dispositivos Conectados. (Fuente: Statista)

Este crecimiento tan exponencial de dispositivos conectados conlleva un gran volumen de datos generados, que son de gran valor debido a su capacidad de captación de información relevante. El IoT supone un cambio cuantitativo y cualitativo en el panorama tecnológico. Muchos expertos dicen que el internet de las cosas cambiara nuestra comunicación personal. Supone tal transformación, que nuestra percepción del mundo se verá transformada de forma definitiva. Ya se están viendo los primeros cambios, la siguiente figura representa el crecimiento interanual de tipos de casos de uso de IoT de 2015-2020. Están creciendo más los casos de uso orientados a personas y sus hogares que a las Ciudades Inteligentes o a la industria conectada. Sobre todo, remarcar la pequeña escala, la arquitectura tecnológica que necesitan los hogares o las personas, utilizan muchos menos recursos, por ejemplo, sensores, que hacen que sean opciones con una casuística mucho más fácil y sencilla. [10]

Figura 8. Crecimiento interanual según caso de uso IoT. (Fuente: IDC)

Sin ninguna duda el Internet de las Cosas está revolucionando todos los aspectos, como podemos ver. Está cambiando las totalmente las industrias y sus desafíos son muy ambiciosos.

3.2 SEGURIDAD EN EL INTERNET DE LAS COSAS

IoT Analytics, uno de los mayores proveedores del mercado en Inteligencia empresarial estratégica para industria 4.0 y el Internet de las cosas publicó un informe en 2017 de 294 páginas sobre la Seguridad de IoT. Este informe se elaboró gracias a la investigación de más de 150 compañías que ofrecen soluciones de seguridad IoT, revisó más de 40 proyectos de seguridad en IoT implementados, realizó más de 25 entrevistas específicas a personajes relevantes en la industria y participó en varias conferencias relacionadas con el tema. Siendo éste un informe de gran relevancia en el ámbito de la seguridad en el complejo mundo del Internet de las cosas [11].

El gasto de seguridad en IoT se estima en unos 703 millones de dólares para 2017 y se prevé que el mercado crezca exponencialmente hasta llegar a unos 4,4 billones de dólares americanos (un 44% de crecimiento)

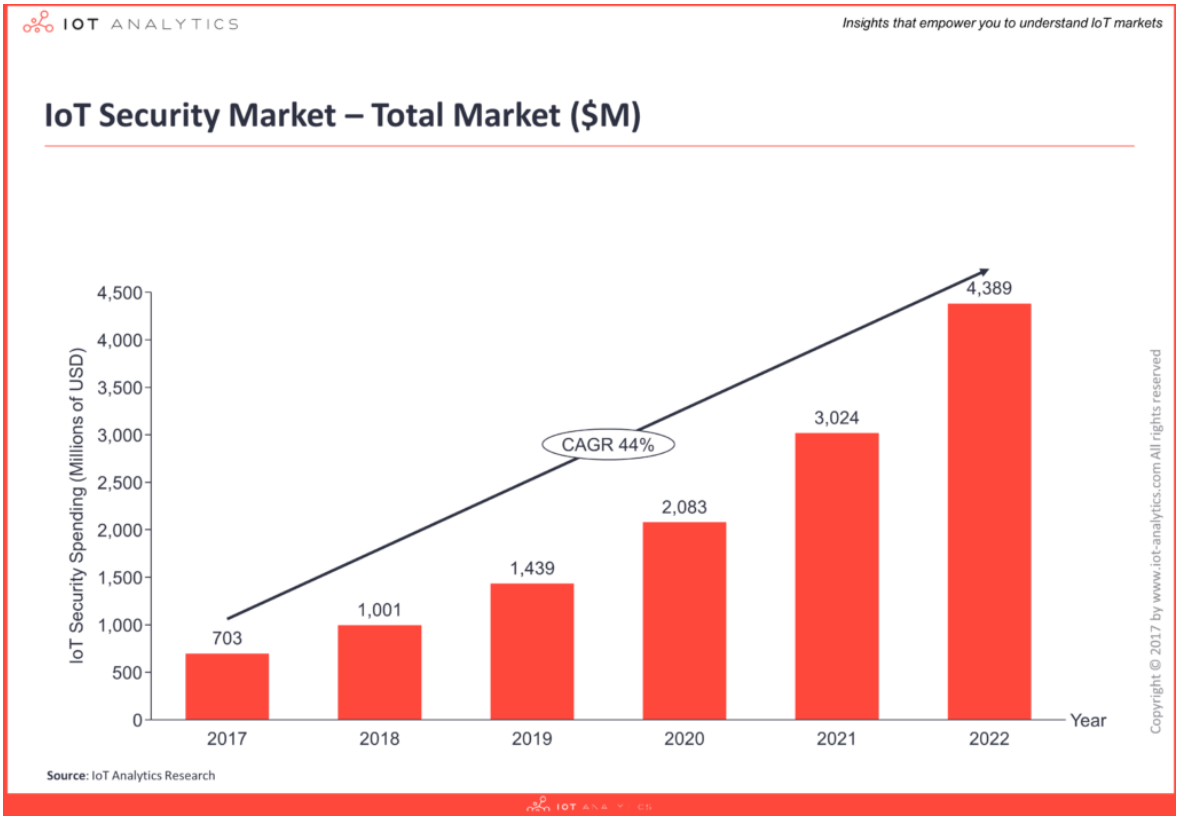


Figura 9. Crecimiento de inversión en seguridad del internet de las cosas. (Fuente: IoT analytics)

Según el informe, en este momento no hay ningún proveedor de seguridad IoT que pueda proporcionar una solución de seguridad completa de principio a fin. En la siguiente figura, se muestra cómo sería una solución completa de extremo a extremo, viéndose cuatro niveles diferentes: componentes de seguridad, los cuales son el hardware, la seguridad en las comunicaciones, seguridad en la nube y el aumento seguridad en la gestión del ciclo de vida.

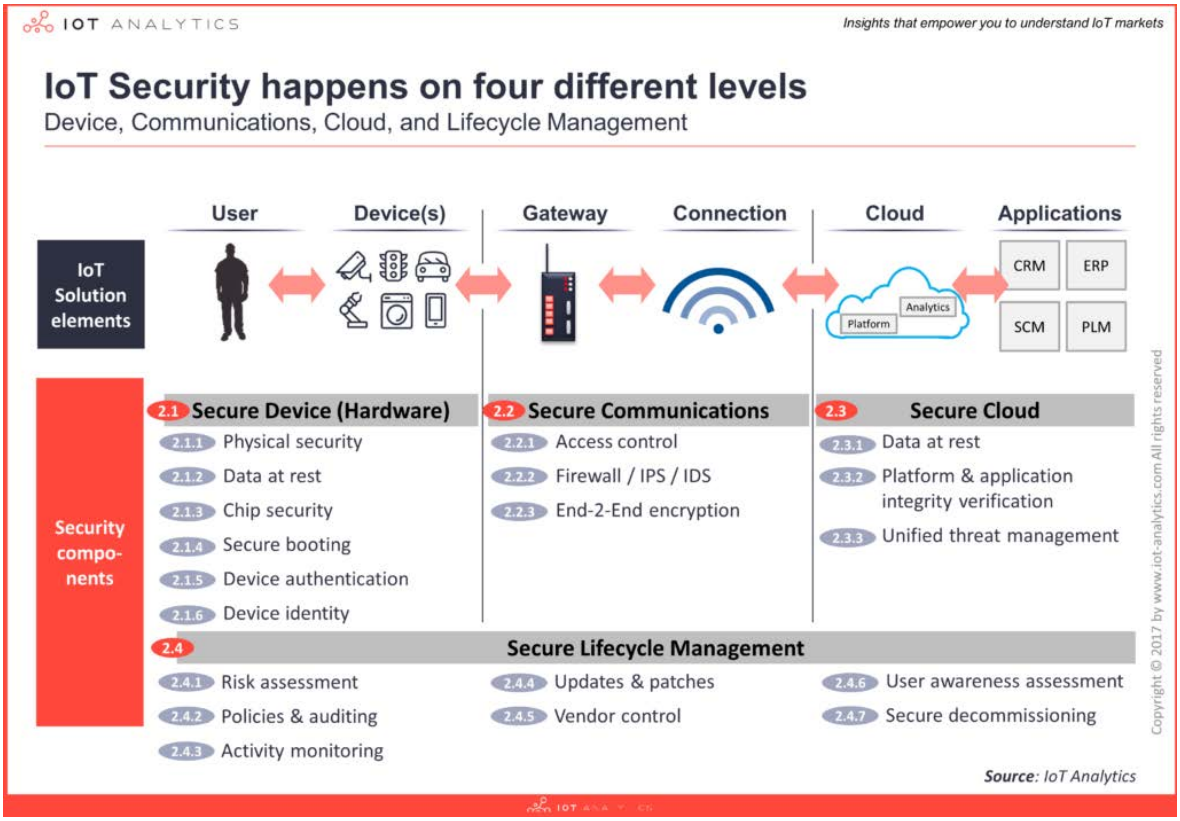


Figura 10. Seguridad en IoT ocurren en 4 niveles diferentes. (Fuente: IoT Analytics)

Una de las cuestiones más importantes es que piensan los expertos en seguridad sobre cuáles son las mayores necesidades que tienen margen de mejora en la seguridad de IoT. Muchos de ellos concuerdan en que la primera es la autorización/autenticación, debido a que es uno de los mayores problemas que deben abordarse. Seguido del control de acceso y el cifrado de datos. Los Firewall, contraseñas, seguridad del servidor y la privacidad no fueron considerados como debilidades.

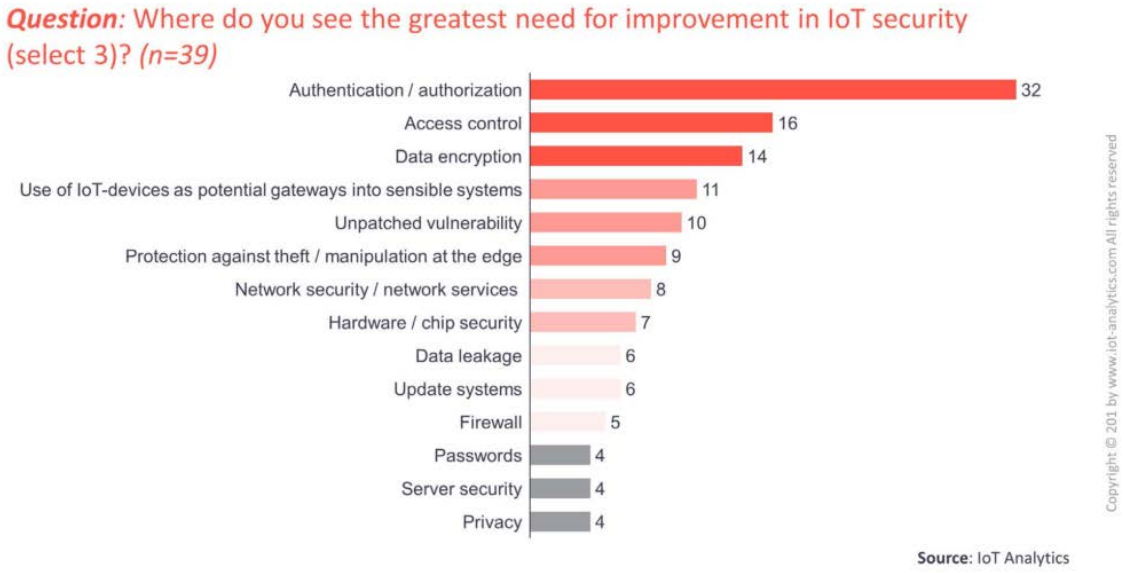


Figura 11. Necesidades de mejora de IoT en Seguridad. (Fuente: IoT Analytics)

Como se ve, la seguridad en el internet de las cosas es un tema que está creciendo y ganando repercusión e inversores anualmente. Siendo un gran tema que tratar, con un enorme foco en su industria.

3.3 TECNOLOGÍAS BLOCKCHAIN PARA EL USO DE SMART CONTRACTS

Actualmente nos podemos encontrar distintos tipos de BlockChain según como queramos orientar el uso de esta. En la siguiente figura se muestran los diferentes tipos de cadena de bloques, pudiendo ser Publica, Privada, Federada o en modo servicio (Cloud). También se muestran las diferentes BlockChain que siguen este tipo de infraestructura, siendo las mencionadas en la figura las principales redes actualmente utilizadas. Todas las redes mencionadas implementan el uso de Smart Contracts. [12]




	Públicos Bitcoin, Ethereum, Litecoin	**Privados** Hyperledger, Corda, Quorum	**Federados** Hyperledger, Corda, Quorum	***Blockchain as a Service*** IBM, Microsoft, Amazon
Cualquiera puede participar	✓	✗	✗	NA
Los participantes actúan, en general, como nodos	✓	✗	✗	NA
Transparencia	✓	≈	≈	NA
Hay un único administrador	✗	✓	✗	NA
Hay más de un administrador	✗	✗	✓	NA
No hay administradores	✓	✗	✗	NA
Ningún participante tiene más derechos que los demás	✓	✗	✗	NA
Se pueden implementar Smart *Contracts*	✓	✓	✓	NA
Existe recompensa por minado de bloques	≈	✗	✗	NA
Soluciona problema de falta de confianza	✓	✗	≈	NA
Seguridad basada en protocolos de consenso	✓	✗	≈	NA
Seguridad basada en funciones *hash*	✓	≈	≈	NA
Provee servicios en la nube	NA	NA	NA	✓

✓ Sí ✗ No ≈ A veces **NA** No Aplica

Figura 12. Diferentes tipos de BlockChain que implementan el uso Smart Contracts

3.4 RED ALASTRIA Y LA UNIVERSIDAD

Durante la consecución del proyecto, la red Alastria ha ido cambiando considerablemente. El 17 de diciembre de 2018 se produjo la incorporación de Montse Guardia, nueva directora general del Consorcio de Alastria. La nueva directora general contará con un responsable financiero, un responsable de Marketing, una oficina de gestión administrativa y soporte y se sumará un responsable de arquitectura y soporte de la plataforma. Su principal objetivo es seguir construyéndose de un modo iterativo durante los próximos años. Está afrontando nuevos retos como ganar más peso a nivel europeo, incrementar el nivel de soporte, ofrecer visibilidad a los miembros del consorcio y el más ambicioso, trabajar en un modelo teórico de identidad, así como en la validación de sus protocolos y usabilidad. [13]

Si accedemos al GitHub que Alastria posee, podremos ver los numerosos proyectos que engloba [14].

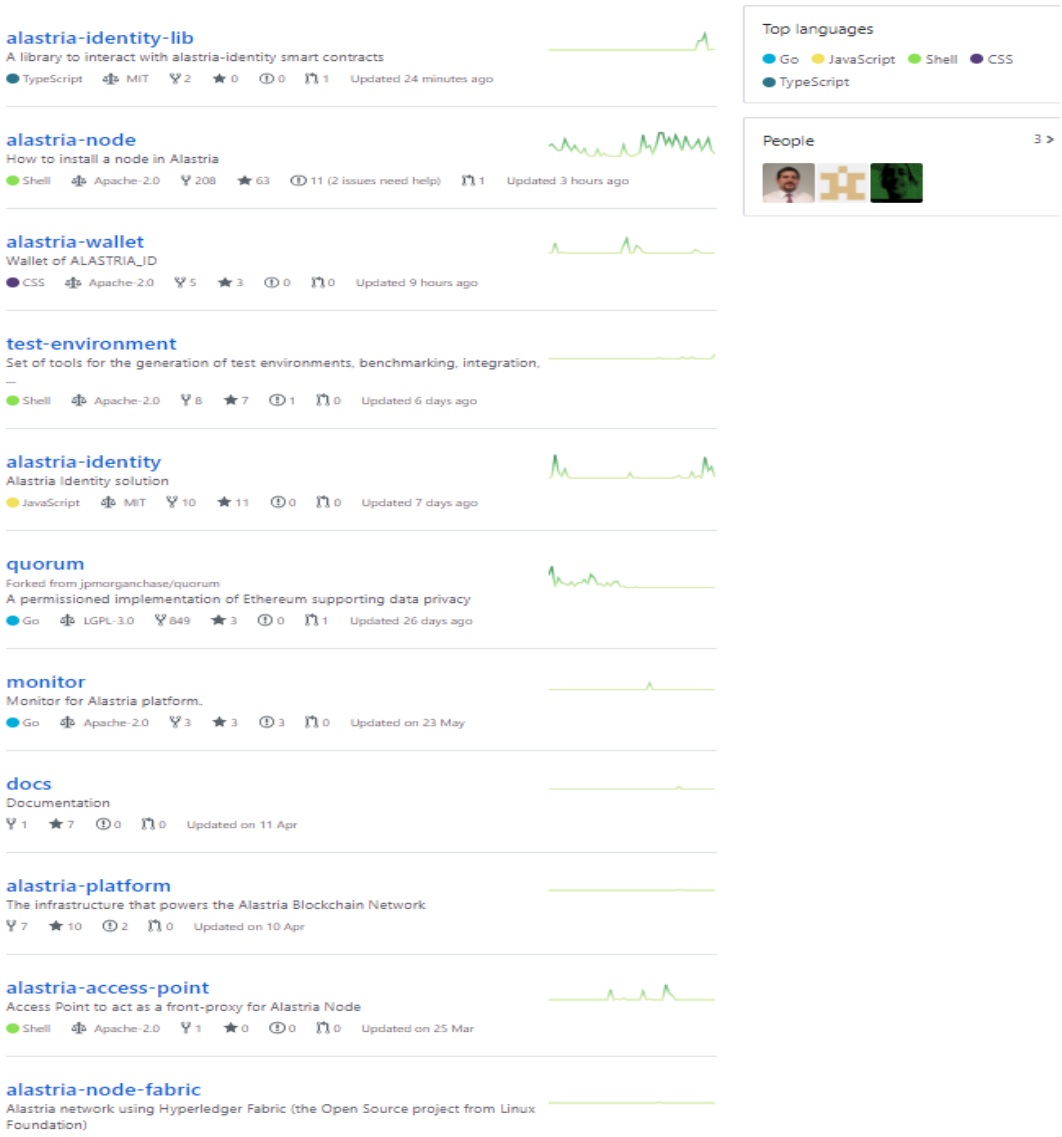


Figura 13. Proyectos actuales de Alastria. (Fuente: Github/Alastria)

Como se puede observar en la figura 13, Alastria cuenta con numerosos proyectos en los cuales están trabajando, como Alastria-Wallet o Alastria-identity, siendo proyectos muy ambiciosos, estando a la orden del día de los objetivos que BlockChain puede implementar. También está probando con nuevas tecnologías BlockChain que no

implementen Quorum, en este es Hyperledger fabric, con el proyecto alastria-node-fabric, BlockChain desarrollada por IBM. Pero el principal uso de Alastria y donde surgió todo, es gracias a la tecnología Quorum, desarrollado por JPMorgan [15]. Se puede encontrar en la carpeta Alastria-node y es el proyecto más importante de Alastria en términos de complejidad y cómputo global de Usuarios que utilizan esta red. Este 2019, la red de pruebas Arrakis, la cual fue la red principal, fue cerrada en mayo de 2019, para seguir con una nueva Test Net llamada T, la cual es la Test net principal.

La universidad inauguró el 13 de junio de 2018 el primer nodo universitario en España de la red Alastria, de la que es socio fundador. Los nodos estuvieron funcionando hasta finales de mayo de 2019, debido al cierre de la red Arrakis (Test net sobre la que se basó Alastria el último año). Actualmente, se han puesto en funcionamiento esos nodos, haciendo una instalación de estos sobre el clúster de la universidad, pero sobre una Test net diferente, llamada T. Los nodos dados de alta han sido, un nodo Validador y un nodo Regular. Véase Anexo A, en el cual se hace una explicación de la instalación de un nodo regular.

3.5 CASOS DE USO

Enfoque dado en los siguientes casos de uso explica como BlockChain al ser un libro de cuentas de distribuido totalmente confiable, permitiéndola creación de reglas para poder monitorizar y controlar, los datos generados por los dispositivos IoT.

Cadena de suministro

Se utiliza la implementación de BlockChain para tener una trazabilidad total de los diferentes ámbitos de una cadena de suministro otorgando más visibilidad, para si una optimización de la cadena. Por ejemplo, saber con exactitud de donde proviene cada alimento en un supermercado, la cadena de frio a la que se ha expuesto un alimento, verificando y pudiendo ver todo el ciclo de vida de este, hasta que es comprado por un consumidor.

Sector de la Automoción

A la hora de fabricar un coche, las diferentes piezas se fabrican en diferentes países y siguen varios procesos de verificación, con la ayuda de BlockChain se puede llevar una trazabilidad de las piezas, ya que cada una proviene de un lugar diferente, pudiendo obtener la información total de su origen y fabricación, para así llevar in sistema trazable y confiable, para la detección de fallos.

Sector Energético

Las redes eléctricas que implementan estas dos tecnologías permiten realizar transacciones peer-to-peer, por lo que, si se detecta un exceso de energía, el exceso sobrante de energía se vende a otro participante de la red BlockChain, quedando todo el pago y la transacción registradas.

Smart Cities and Smart Homes

La seguridad y confianza en la generación masiva de datos se pueden ver protegidas gracias al uso de la tecnología BlockChain. Si todos los integrantes de servicios de este tipo de soluciones IoT, pueden participar en un BlockChain privada , la cual ofrece servicios entre los posibles proveedores de gestión y automatiza las formas de pago dentro del sistema.

Sector Sanidad

Ayuda a que el paciente, diferentes médicos o aseguradoras, puedan visualizar de una manera totalmente confiable los datos del paciente, pudiendo solo acceder a ellos una serie de usuarios autorizados.

Capítulo 4. DEFINICIÓN DEL TRABAJO

4.1 JUSTIFICACIÓN

Como se ha podido comprobar en el apartado anterior, el internet de las cosas se encuentra en un estado actual en el cual repercute en casi todos los ambientes posibles, siendo esta un compañero diario del ser humano diariamente. Mucha de las soluciones implementadas en IoT, están basadas en las personas conectadas, el 17%, siendo esto de gran repercusión ya que los datos generados, incluyen datos personales.

La seguridad de los entornos y donde se almacenan los datos es primordial. Según el informe analizado de la seguridad de las soluciones IoT, la autenticación y la autorización, es el mayor problema detectado, pero también se quiere implementar una forma segura de almacenar los datos, la cual implemente una autorización necesaria, totalmente segura .

La falta de seguridad muchas veces no viene del diseño de las soluciones IoT, sino también de las malas prácticas del usuario. En este proyecto se quiere desarrollar un sistema en el cual, el usuario no aumente los riesgos de seguridad ya que la solución está adaptada a él, es decir, que los métodos y consecución estén enfocados en el ámbito de la fácil comprensión y utilización de la solución IoT para que se pueda mitigar esta vulnerabilidad en su mayor parte.

Las razones explicadas anteriormente son las que han impulsado, por consiguiente, al desarrollo del proyecto realizado, explorándose nuevas maneras de utilización de tecnologías, las cuales puedan mitigar los riesgos de seguridad, y sobre todo el problema de autenticación y almacenamiento. Por lo que se decidió realizar un estudio de la Tecnología BlockChain, para así poder desarrollar un sistema el cual implemente las cualidades anteriormente detectadas.

Muchas empresas y casos de uso están siendo desarrollados, siendo complejos y solucionando muchas de las carencias que posee el Internet de las Cosas, pero son todas desarrolladas por empresas. Este proyecto está enfocado en el desarrollo de un sistema basado en software libre, buscando un enfoque para que los usuarios obtengan un nuevo punto de vista de cómo la información de los dispositivos que utilizan diariamente, puede ser almacenada y tratada de una forma segura, desarrollando un sistema el cual gestione de una forma transparente y segura la información de sus dispositivos, incluyendo un sistema de identificación basado en unas de las tecnologías más seguras en la actualidad, BlockChain.

4.1.1 BLOCKCHAIN COMO SOLUCIÓN AL PROBLEMA DE LA AUTENTICACIÓN

El uso de la tecnología BlockChain ha sido una revolución en los sistemas distribuidos, ya que su confianza y trasparencia son sus principales características. Al disponer un sistema de gestión de cuentas, utilizando clave publica y clave privada, siendo este muy difícil de vulnerar, aporta la confianza buscada para una segura autenticación.

Actualmente el concepto de criptomoneda y de las BlockChain es de repercusión mundial, y mucha de la población está interesada y comprende estos conceptos, ya que son una manera confiable de ejecución transacciones de manera una manera segura y trasparente. El concepto abordado para una solución IoT con una cadena de bloques es el mismo, en vez de realizar una transacción con criptodivisas, se realiza una transacción de información generada por el dispositivo a un entorno seguro de gestión de datos que incluye disponibilidad, confianza y unas grandes medidas de seguridad, implementando un sistema de autenticación totalmente seguro en términos de ciberseguridad.

4.1.2 USO DE CONTRATOS INTELIGENTES COMO UN GESTOR DE ALMACENAMIENTO SEGURO

Los contratos inteligentes han evolucionado de muchas maneras posibles, en este proyecto se va a utilizar el concepto de contrato inteligente como una base de datos distribuida para guardar y gestionar los datos generados por dispositivos IoT.

Estos contratos al ser desplegados sobre una red distribuida como BlockChain, son inmutables por lo que la información guardada en el mismo queda totalmente protegida de riesgos de fuga o modificación.

Al beneficiarse de las cualidades que poseen las cadenas de bloques actualmente, el sistema de autenticación para la ejecución de transacciones dentro del contrato queda cubierta, ya que, en un contrato, solo las partes definidas en él pueden beneficiarse del mismo, por lo que en un contrato inteligente, solo pueden acceder los intermediarios definidos en este, cubriendo así accesos no autorizados al mismo.

4.1.3 ELECCIÓN DE USO DE ALASTRIA

Alastria, se basa en gran parte en la arquitectura una red basada en Quorum, la cual está basada Ethereum. Por lo que Alastria implementa la tecnología de Ethereum. Esta se encuentra en un gran punto de madurez, en términos de implementación e interacción con contratos inteligentes, por lo cual es una red que ofrece los servicios y herramientas necesarias para la consecución de este proyecto.

Además, el año pasado la universidad se convirtió en la primera en desplegar primer nodo BlockChain universitario de España, desplegando un nodo validador y un regular sobre la red Alastria en concreto, sobre la Test Net Arrakis. Por lo que mi tutor del proyecto me propuso el uso de esta red para la realización del proyecto.

4.2 OBJETIVOS

En este proyecto está estructurado en la consecución de una serie de objetivos planteados. Estos objetivos sirven para poder mantener un seguimiento del proyecto, para poder evaluar si se están completando estos y hacer un análisis de si el proyecto se está realizando adecuadamente.

Objetivo 1. Realización de un análisis y estudio las tecnologías principales, IoT y BlockChain. Ver cuáles son los principales problemas que alberga IoT. Aportación de soluciones gracias al uso de la tecnología BlockChain.

Objetivo 2. Desarrollo de un caso de uso, el cual cumpla las motivaciones planteadas e implemente una solución a los problemas detectados, identificados en el objetivo anterior.

Objetivo 3. Desarrollo un sistema el cual cumpla el caso de uso descrito. Para el desarrollo del sistema se han seguido los siguientes apartados:

- Desarrollo de nuevas funcionalidades en el Gateway IoT para la comunicación con la Rest-API.
- Desarrollo de una Rest-API, su objetivo es el almacenamiento temporal de los datos generados por el Gateway IOT.
- Desarrollo y despliegue de un contrato inteligente, para la gestión distribuida de los datos de los sensores.
- Despliegue del Smart Contract sobre una red BlockChain, la cual permita el uso de las tecnologías a utilizar. Despliegue del contrato sobre la red Alastria.
- Desarrollo de un servidor el web, el cual permita al usuario interaccionar con dispositivos IoT de una forma segura y sencilla, realizando transacciones con el Smart Contract desarrollado de una forma segura.

Objetivo 4. Configuración y despliegue de la red Alastria, para el despliegue de los contratos inteligentes desarrollados, y una interacción con los mismos.

4.3 METODOLOGÍA Y PLANIFICACIÓN

Para la realización de este proyecto, se está utilizando las metodologías de trabajos agiles, de tal forma que se ha dividido el trabajo en pequeñas partes que se van entregando cada dos/tres semanas. Se ha utilizado la plataforma Trello, es una fabulosa herramienta para la organización de tareas. Es ideal para la coordinación de equipos de trabajo y se basa en la metodología Kanban, la cual propone un sistema de uso colaborativo. Se han implementado ciclos de scrums en semanas durante el transcurso del proyecto.

Se ha trabajo sobre el proyecto llamado BlockChain-IoT y se encuentra estructurado en cinco partes:

- Información general: campo en el cual están incluidas la documentación de tecnologías a utilizar, webs de utilidad, documentación, programación de reuniones.
- Tareas por realizar: En este punto se identificaban los puntos por realizar.
- En proceso: tareas que están en pleno desarrollo.
- Realizadas: tareas realizadas las cuales han sido testeadas.
- Memoria: Refleja el estado de la Memoria del proyecto.

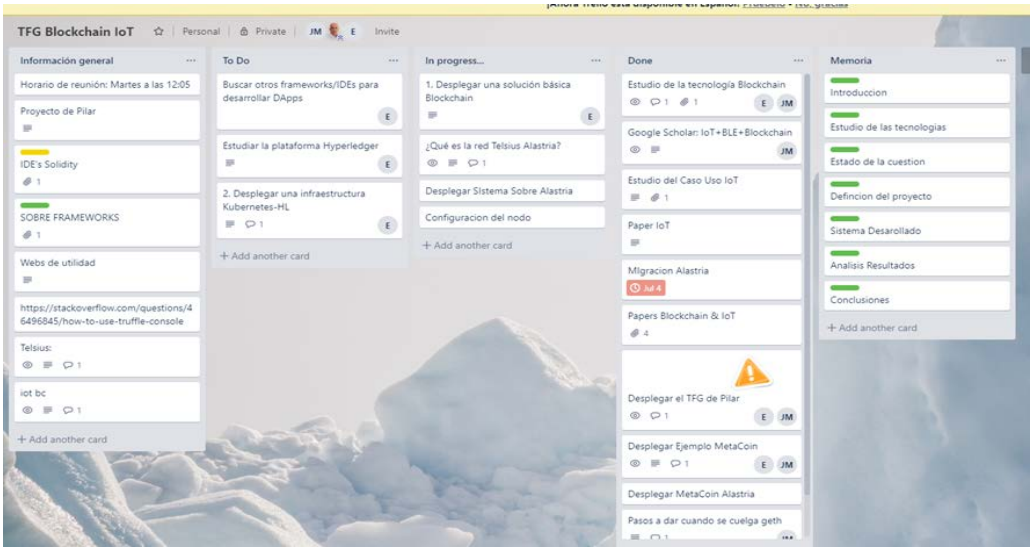


Figura 14. Proyecto Trello BlockChain-IoT

Se ha trabajado también con un diagrama de GANT, figura 15, el cual ha servido para poder planificar las tareas principales a realizar.

ACTIVIDAD	INICIO	FINAL	Octubre	Noviembre	Diciembre	Enero	Febrero	Marzo	Abril	Mayo	Junio	Julio
Estudio de tecnologías	01/10/2018	15/01/2019										
Anexo B	15/01/2019	15/02/2019										
Presentacion Seguimiento	26/03/2019	26/03/2019										
Objetivo 1	15/11/2018	28/02/2019										
Objetivo 2	14/12/2018	15/02/2019										
Objetivo 3	03/03/2019	30/06/2019										
Objetivo 4	15/01/2019	12/07/2019										
Redaccion Memoria	15/04/2019	12/07/2019										
Defensa del proyecto	17/07/2019	17/07/2019										
Duración del proyecto	01/10/2018	17/07/2019										

Figura 15. Proyecto Trello BlockChain-IoT

4.4 ESTIMACIÓN ECONÓMICA

En este apartado se realiza un modelo económico de cuánto costaría realizar el desarrollo del Proyecto. Para poder realizar este modelo tenemos que separar los recursos necesitados en dos partes, la humana y el material.

Recursos humanos que se necesitan para la consecución del proyecto. Se han identificado un perfil necesario para la optimo desarrollo del proyecto. El perfil sería el de un programador full-stack, el cual tenga conocimientos de las tecnologías a utilizar. Aun así, ya que son tecnologías diferentes, se le podría impartir una formación online de los conocimientos posiblemente necesitados. No se necesita de alguien más ya que toda la arquitectura ya está diseñada, solo sería la realización del código.

El enfoque de los recursos materiales se basa equipo necesario, tanto hardware como software para la correcta y funcional realización del proyecto. En este proyecto nos hemos ayudado de herramientas código libre, por lo que no se ha requerido inversión en software. Al ser un proyecto para ser realizado en el clúster de la universidad, donde se encuentra

implementado Alastria, no se incluyen los gastos del mismo, ni de los recursos ya que todo ha sido provisto por la universidad.

Recursos Humanos	**Justificacion**	**Cantidad de Horas**	**Precio**	**Coste**
Programador full-stack	Se incluyen horas de aprendizaje del programador, ya que al aprender tambien esta realizando el proyecto Se incluyen 20 horas mas de posible formacion	140	50 €	7.000 €
Formacion	Posible formacion en tecnologiasque el programador no sepa utilizar.	20	100 €	100 €
				7.100 €
Recursos Materiales	**Justificacion**		**Precio**	**Coste**
Hardware				
Dispositivo ThunderBoard	Sensor facilitado al alumno por la universidad		20 €	0 €
GateWay IoT	Programado en una RaspBerry Pi 3 , la cual se facilito al alumno		35 €	0 €
Ordenador 8 GB de RAM	Ordenador necesario para la consecucion del proyecto, minimo de 8GB y IntelCore I5		700 €	0 €
Cluster	El Cluster utulizado es el de la Universidad		60.000 €	0 €
Software				
Herramientas OpenSource				0 €
				0 €
TOTAL				7.100 €

Tabla 1. Estimación de costes para el proyecto

El total estimado es de 7100 euros como se puede ver en la tabla 1, ya que los recursos materiales no han supuesto ningún coste. En cambio, los recursos humanos sí que supondrían un coste ya que habría que desarrollar el caso de uso. En términos económicos sale muy asequible montar esta infraestructura ya que solo habría que emplear recursos en el desarrollo, siendo este muy asequible.

Capítulo 5. MODELO DESARROLLADO

En este capítulo es donde se va a desarrollar el caso de uso para la consecución de los objetivos 2 y 3 . Este caso de uso ha sido elegido con una clara orientación al entorno del usuario, centrándose en una forma segura y confiable de la gestión de los dispositivos IoT, gracias a la ayuda de los contratos inteligentes.

5.1 ANÁLISIS DEL SISTEMA

El caso de uso desarrollado quiere implantar una solución para la detección de dispositivos en un lugar determinado, por ejemplo, una entrada de un edificio, pudiéndose así generar un registro de trazabilidad del dispositivo, pudiendo saber cuándo este ha entrado o salido del mismo.

Por ejemplo, pongamos que un usuario cuando va a un lugar el cual posee una entrada principal, pongamos de ejemplo una universidad. Implementando el caso de uso desarrollado se puede tener un registro de las entradas y salidas, del dispositivo en ese lugar, para así saber información de entrada y salida de uno de sus dispositivos u objetos (este implemente el uso de IoT) que siempre lleva consigo mismo, por ejemplo, un wearable.

Si en la puerta principal del edificio se instalara un detector de dispositivos IoT en la entrada del edificio, el cual detectara todos los sensores que pasan por ahí ,se dispondría de un sistema real el cual informa que su dispositivo, ha pasado por ese lugar. Cuando un dispositivo se encuentre en el radio de detección , el detector establece una conexión con el mismo , obteniendo la información de este. Una vez obtenida la información del dispositivo, el usuario puede realizar una consulta de la información generada, pudiendo saber si su dispositivo ha pasado por ese lugar, pudiendo sacar ciertas conclusiones, como por ejemplo si el dispositivo ha abandonado el edificio. Analizado la información de la conexión, se puede saber hora de la última conexión entre el detector y el dispositivo.

Además, si se implementa el uso de sensores lumínicos, o atmosféricos, se podrían obtener datos como el índice de luz , sabiendo si el dispositivo sale del recinto al aire libre, o tapado , ya que no recibiría luz, para así obtener más información, por ejemplo, en caso de robo o perdida.

Para poder desarrollar este caso de uso para que cumpla las justificaciones del proyecto, la información de los sensores detectados se debería guardar en un entorno seguro, el cual implemente varias medidas se seguridad y confianza para que el sistema sea totalmente trasparente, y así el usuario confié en él. Esta idea se puede implementar utilizando el uso de la tecnología BlockChain, en concreto con el uso de los contratos inteligentes, siendo estos una manera segura de gestionar la información de los dispositivos. En estos contratos, se puede definir los intermediarios que participan en el, en este caso el usuario sería el único intermediario , solucionando así el problema de autenticación/autorización planteado en la justificación. Además de poderse utilizar como sistemas de gestión de información, implementado así una manera segura, confiable e inmutable de gestionar los datos generados por los dispositivos, en este caso se guardaría la última conexión del dispositivo con el detector instalado, ya que estos no poseen gran capacidad de almacenamiento, si se quiere desarrollar un gestor rápido y efectivo.

5.2 ARQUITECTURA DEL SISTEMA

Para el caso de uso planteado en el apartado anterior, se tiene que analizar que diseñar tres módulos los cuales son requeridos para así la realización de la arquitectura del sistema. Se han detectado las siguientes:

1. Plataforma IoT, la cual implemente un detector de dispositivos. Buscar una manera de que este sistema guarde la información temporalmente hasta que el usuario requiera de su uso.
2. Una aplicación distribuida la cual implemente el uso de contratos inteligentes, como forma de gestionar la información. Además de incluir de un sistema el cual sea seguro en términos de autenticación y desarrollo del sistema.

Con el estudio de las tecnologías previamente realizado, se han desarrollado los siguientes módulos para el correcto funcionamiento del sistema. Siendo este el tercer objetivo a cumplir.

5.2.1 PLATAFORMA IOT

Para la realización de la plataforma IoT se han detectado dos necesidades, un detector de dispositivos, y un sistema el cual guarde la información detectada.

Detección de los dispositivos:

Utilización de un gateway IoT, el cual ha sido desarrollado en la realización de otro proyecto de fin de grado. Utilización de un sensor Thunderboard Sense como dispositivo a detectar.

Sistema que guarde la información detectada:

Se ha desarrollado una Rest-API, la cual ha sido diseñada para almacenar la información requerida para la consecución de este caso de uso, se explica el desarrollo en el apartado de diseño.

5.2.2 APLICACIÓN DISTRIBUIDA

Para la realización de esta aplicación, se han detectado dos módulos a desarrollar.

Aplicación web

Utilización de un servidor web el cual implemente, un desarrollo front-end como back-end.

Tecnologías BlockChain

Despliegue de un contrato inteligente sobre una red de pruebas Ganache, para utilizarlo como gestor de la aplicación.

5.2.3 ARQUITECTURA DESARROLLADA

En este apartado se explica la arquitectura del sistema desarrollado en la siguiente figura:

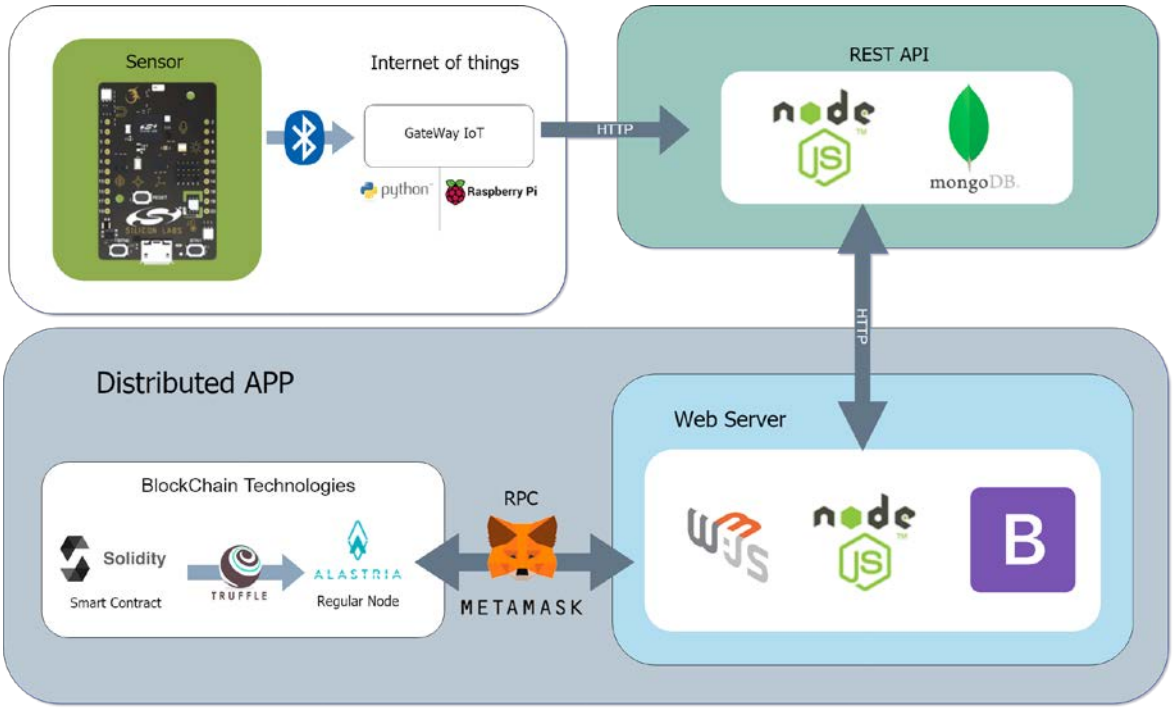


Figura 16. Diagrama de la arquitectura del sistema

Se va a explicar como la arquitectura desarrollada influye en la consecución del caso de uso implementado.

Como se puede observar en la Figura 16, la plataforma IoT desarrollada, es la encargada de la detección un sensor Thunderboard, a través de un módulo Bluetooth. Una vez esta ha detectado la información del sensor, se envía a la REST-API, utilizando el protocolo HTTP, guardando en esta la información del sensor detectado.

Una vez el servidor web necesita la información de la REST API, estos módulos se comunican mediante peticiones HTTP, para así obtener la información del dispositivo deseado.

El funcionamiento de la aplicación distribuida implementa la utilización de un servidor web, el cual aloja la aplicación desarrollada, la cual es un gestor de dispositivos el cual implementa un registro de trazabilidad. Para una confiable y segura gestión de estos, se implementa el uso de los Smart Contracts, siendo desplegados gracias al uso de la tecnología Truffle sobre una red BlockChain, en este caso se ha utilizado Ganache(red de pruebas). Este contrato es llamado desde el Servidor Web y gracias a la librería Web3 y a la herramienta Metamask, mediante el protocolo JSON RPC, se realiza una conexión con el contrato, para la ejecución de transacciones sobre el mismo, implantando una forma segura y confiable de almacenar datos y para el usuario en una cadena de bloques, siempre que el usuario de su consentimiento.

5.3 DISEÑO DE LA APLICACIÓN WEB

En este apartado se va a definir el funcionamiento de la aplicación web desarrollada.

5.3.1 INTERFAZ

Diagrama de navegabilidad desarrollado muestra cómo se puede navegar por la aplicación desarrollada, siendo esta de uso sencillo e interfaz amigable.

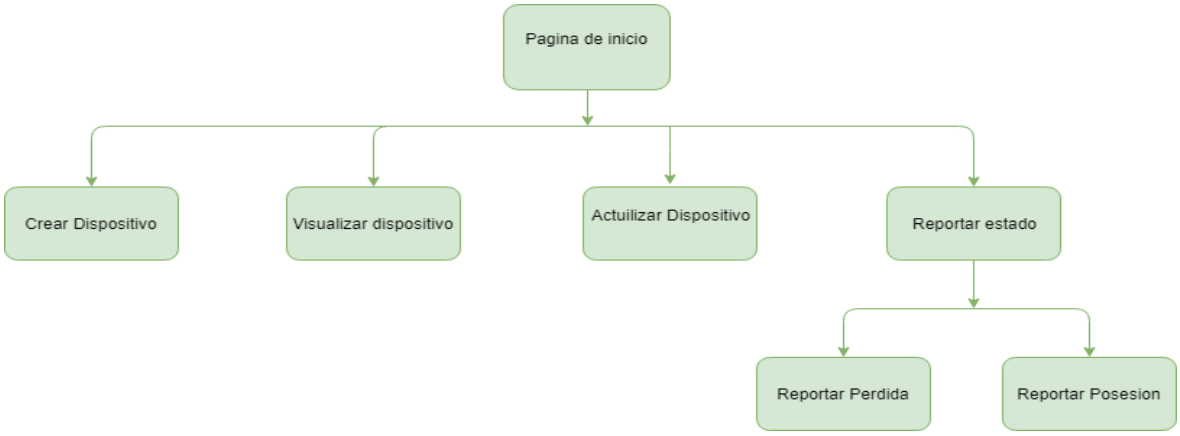


Figura 17. Diagrama de navegabilidad de la aplicación desarrollada

5.3.2 COMPONENTES

1. Casos de Uso

El usuario dentro de la aplicación web puede realizar las siguientes funcionalidades:

- Creación de un dispositivo, al ejecutar una transacción sobre el contrato inteligente incluye el uso de Metamask.
- Actualización de un dispositivo, al ejecutar una transacción sobre el contrato inteligente incluye el uso de Metamask.
- Visualización de un dispositivo.
- Reporte de pérdida o posesión de un dispositivo, al ejecutar una transacción sobre el contrato inteligente incluye el uso de Metamask.

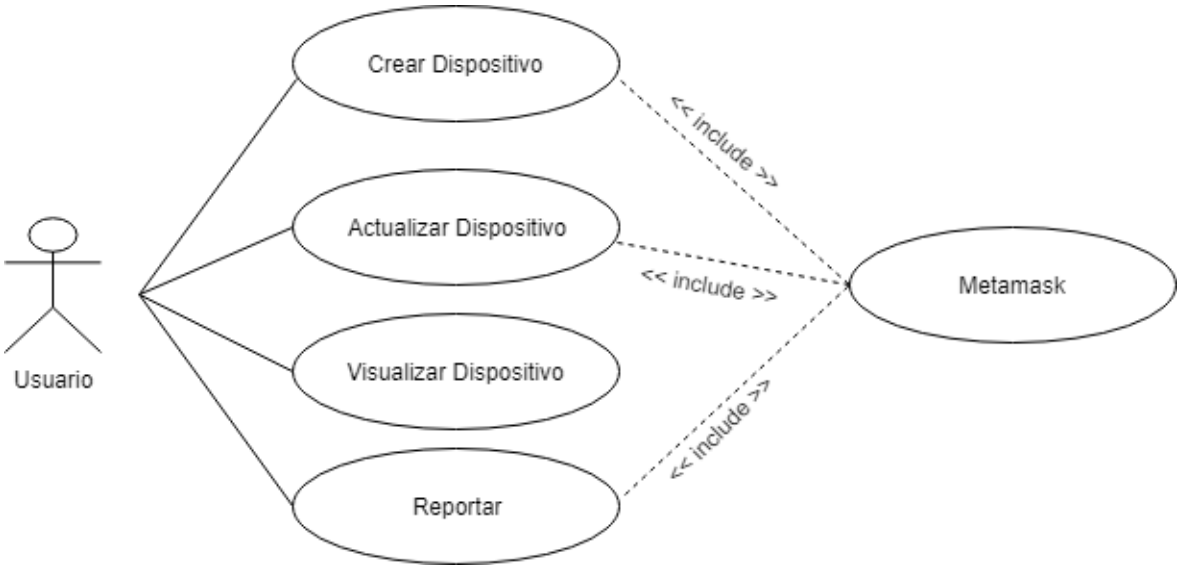


Figura 18. Caso de uso implementado en la aplicación web

2. Diagramas de Secuencia

Se han realizado cuatro diagramas de secuencia, debido a que el caso de uso implementa cuatro posibles funcionalidades dentro de la aplicación web.

Para la creación de un dispositivo se ha diseñado el siguiente diagrama de secuencia, para poder definir las acciones a realizar.

1. Si pulsamos el botón de la creación de un dispositivo, se realiza la detección de la cuenta con la que se va a ejecutar la transacción.
2. Una vez se obtiene esta cuenta, se llama a la función crear dispositivo
3. Cuando se ha verificado que el dispositivo se ha creado correctamente sin error, se dispone a hacer una petición de información del mismo dispositivo, para ver la información creada, y la posterior creación de este en la Rest-API.
4. Una vez se obtiene esta información se realizan dos peticiones HTTP, POST, las cuales dan de alta e insertan los datos en la Rest-API.

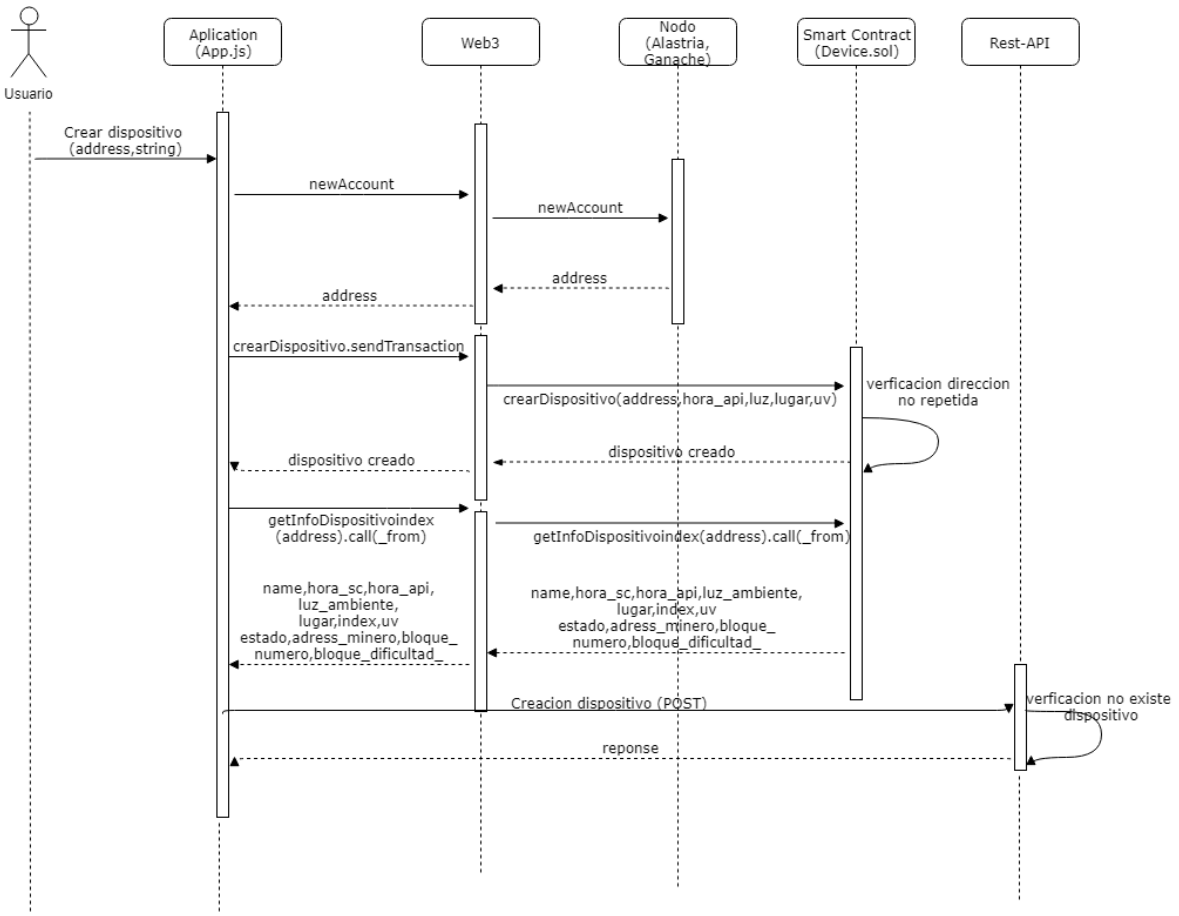


Figura 19. Diagrama de secuencia de la creación de un dispositivo en el sistema

Para la actualización de un dispositivo se ha diseñado el siguiente diagrama de secuencia, para poder definir las acciones a realizar.

1. Si pulsamos el botón de la actualización de un dispositivo, se realiza la detección de la cuenta con la que se va a ejecutar la transacción.
2. Una vez se obtiene esta cuenta, se realiza una petición HTTP,GET a la Rest-API, para la obtención de los datos la última de la conexión del dispositivo con el gateway.
3. Una vez recibida la información, se procede a la actualización del dispositivo, verificando que este no se encuentra en estado de perdida, ya que de ser así no se podría actualizar.
4. Cuando se ha verificado que el dispositivo se ha actualizado correctamente sin error, se dispone a hacer una petición de información del mismo dispositivo, para poder ver la información actualizada en la aplicación web.

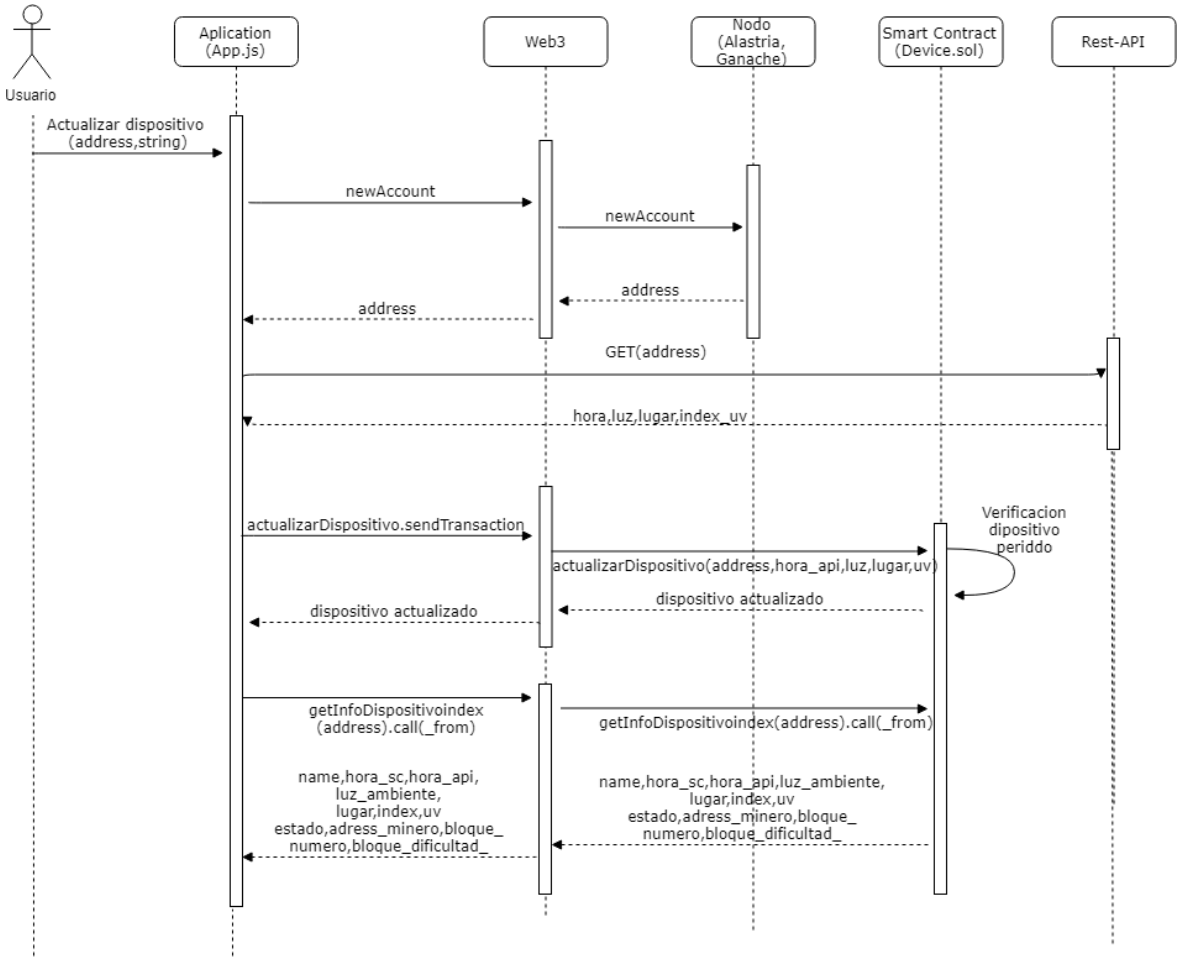


Figura 20. Diagrama de secuencia de la actualización de un dispositivo en el sistema

Para la visualización de la información de un dispositivo se ha diseñado el siguiente diagrama de secuencia, para poder definir las acciones a realizar.

1. Si pulsamos el botón de la visualización para obtención de información de un dispositivo, se realiza la detección de la cuenta.
2. Una vez se obtiene esta cuenta, el sistema se dispone a realizar una petición de información, para la visualización del dispositivo deseado en la aplicación.

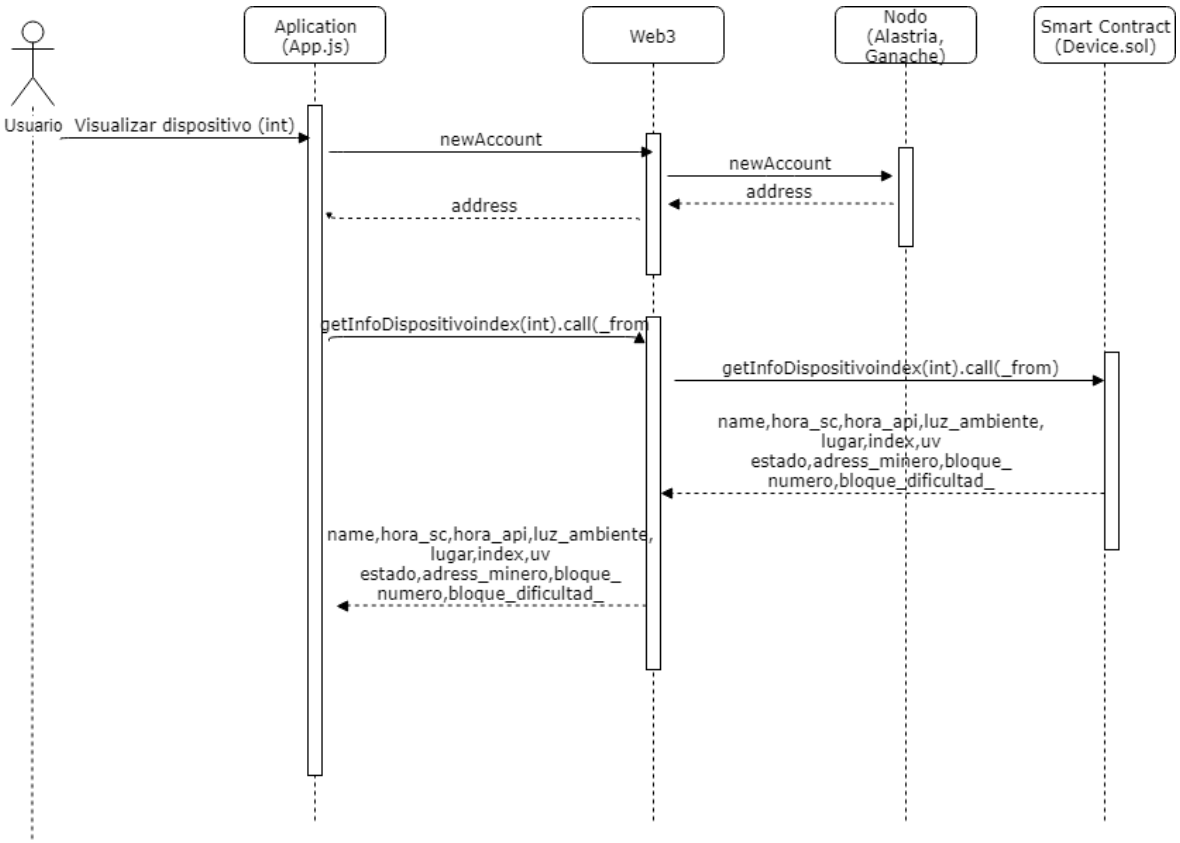


Figura 21. Diagrama de secuencia de la visualización de un dispositivo en el sistema

Para el reporte de pérdida o posesión de un dispositivo se ha diseñado el siguiente diagrama de secuencia, para poder definir las acciones a realizar.

1. Si pulsamos el botón de reporte dispositivo, se realiza la detección de la cuenta con la que se va a ejecutar la transacción.
2. Una vez se obtiene esta cuenta, se llama las funciones o reportar para así cambiar el estado del dispositivo, verificando que este no se encontraba en ese.
3. Cuando se ha verificado que el reporte se ha realizado sin error, se dispone informa al usuario del resultado,

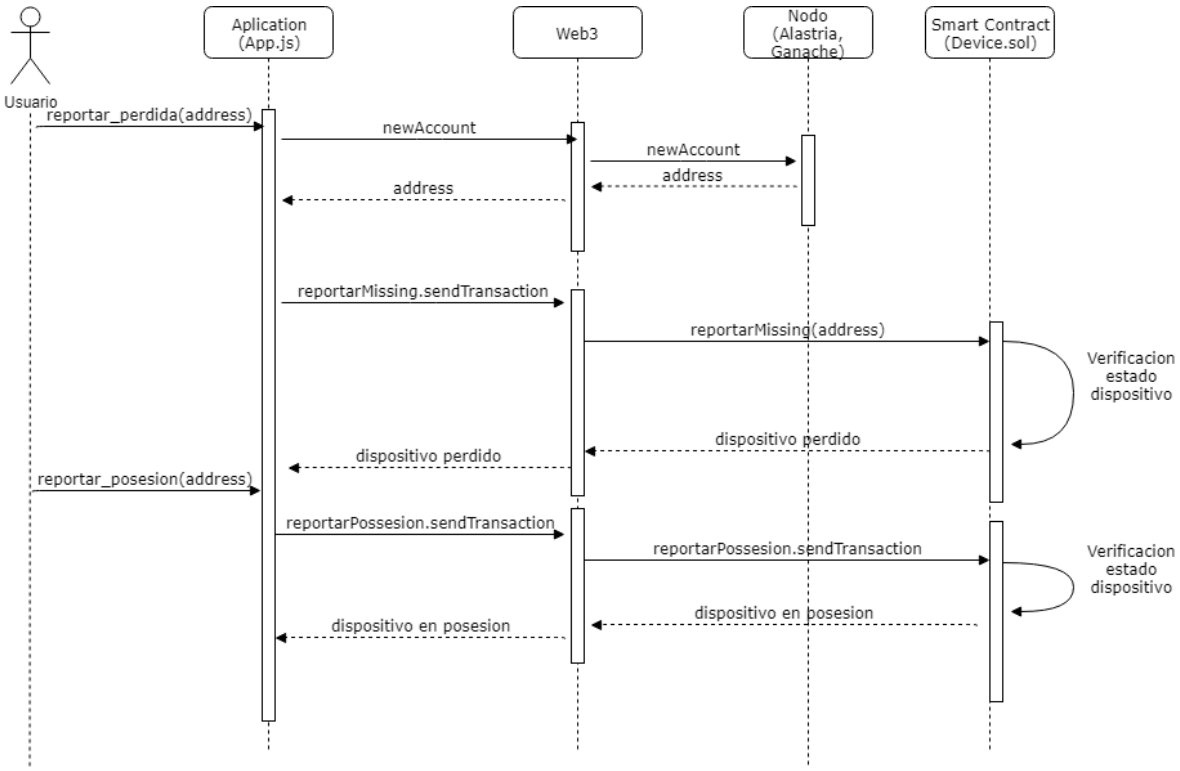


Figura 22. Diagrama de secuencia del reporte de pérdida o posesión de un dispositivo en el sistema

5.3.3 CONTEXTO

MODELO DE LA ARQUITECTURA

En esta aplicación web el front-end es quien interacciona con todos los módulos, para el correcto funcionamiento de la aplicación. Para la conexión con el servidor web se utilizan peticiones HTTP, GET/POST. Para conexión con la cadena de bloques y contrato inteligente, se utiliza el módulo web3, para la conexión, y Metamask, para la ejecución de las transacciones. Para la conexión con la Rest-API también se utilizan peticiones HTTP, GET/POST. Véase en la figura 23, el modelo desarrollado para el funcionamiento de nuestra aplicación web.

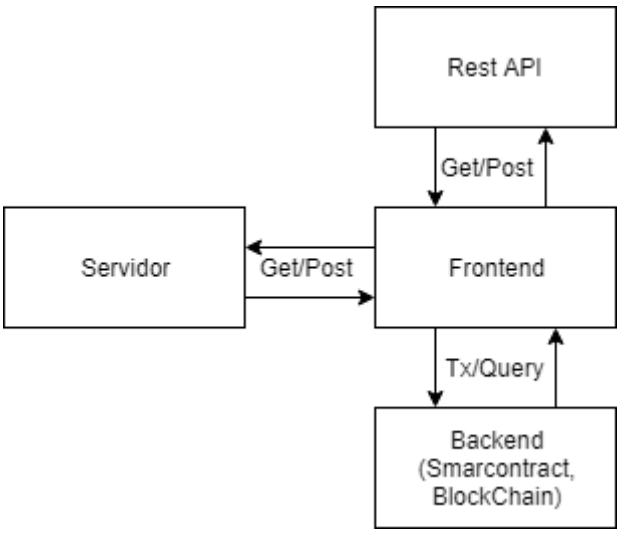


Figura 23. Modelo de Arquitectura de la aplicación web

DIAGRAMA FRONT-END

Se muestra a continuación, en la ilustración 18, el diagrama de clases del front-end. Como se puede observar toda la lógica se encuentra en el mismo script “app.js”. Esta decisión se ha tomado para poder reutilizar variables, para que no sea necesario instanciar más de una vez el contrato y para reducir la posibilidad de errores.

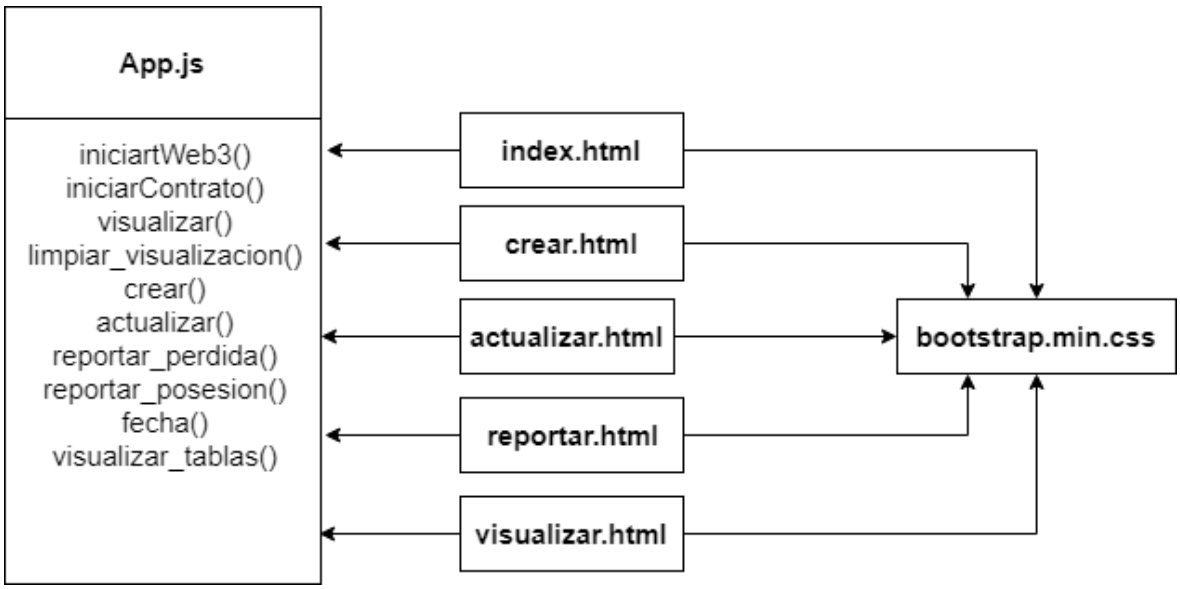


Figura 24. Diagrama front-end de la aplicación

DIAGRAMA-BACKEND

En la figura 25, podemos observar que el contrato inteligente desarrollado se utiliza como el back-end de la aplicación web.

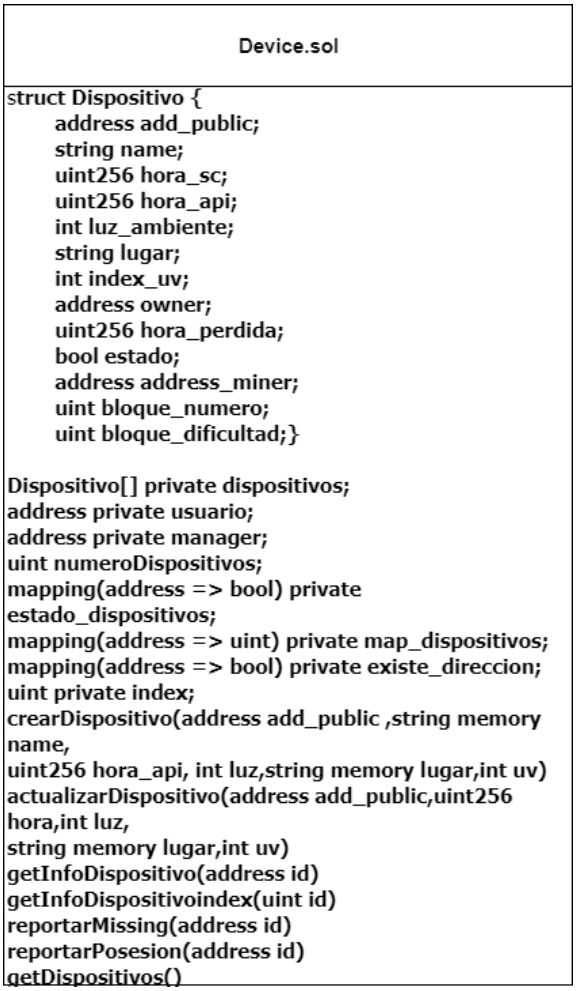


Figura 25. Diagrama back-end de la aplicación

5.4 IMPLEMENTACIÓN

En este apartado se diseñarán los componentes de la arquitectura del sistema, haciendo una explicación de las tecnologías utilizadas. Para así mas tarde implementarlos en el sistema.

5.4.1 DESARROLLO DE NUEVAS FUNCIONALIDADES EN EL GATEWAY IOT

Para el diseño de la nueva funcionalidad, se han definido los datos que el GateWay IoT, detecta del sensor ThunderBoard.

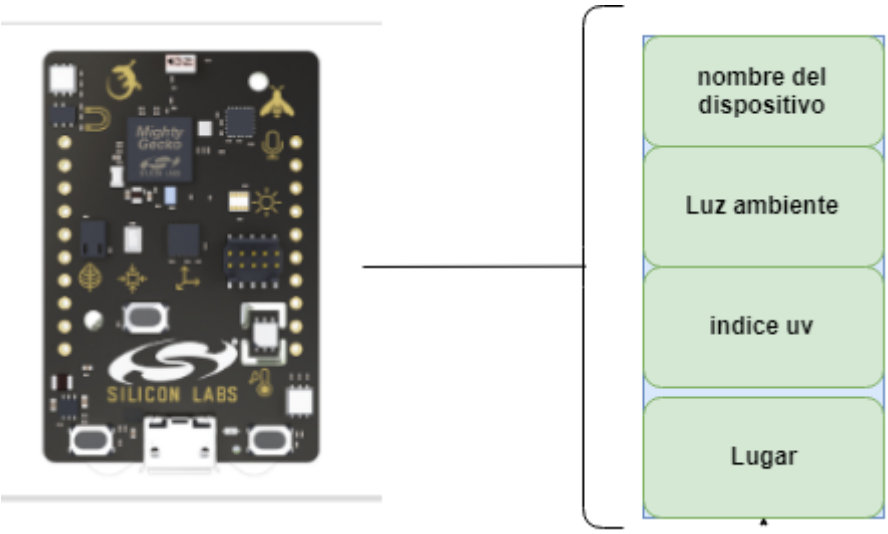


Figura 26. Datos detectados por el Gateway IoT

Para correcta consecución del proyecto, se replicó uno de los ficheros desarrollados, y se modificó para que el gateway para que incluyera un módulo de peticiones HTPP.

Código ejecutado cuando se detecta un sensor Thunderboard. Una vez este es detectado se obtienen los datos que se quieren introducir en la Rest-Api. Una vez estos datos son adquiridos de forma correcta, se realiza una petición HTTP, PATCH, la cual actualiza la información del sensor.

```
lastmeasure=data['temperature']
```

```
            print("Sensor detectado")
            print("Nombre: " + str(data['name']))
            print("DevID: " + data['devId'])
            print("timestamp: " + str(data['timestamp']))
            print("Temperatura: " + str(lastmeasure))
            print("luz: " + str(data['ambientLight']))
            print("index uv: " + str(data['uvIndex']))
            print(chemipaquete)

patch("http://192.168.42.26:3000/Sensors",data['name'],data['ambientLight'],data[
'uvIndex'])
```

Función desarrollada para la realización de la petición HTTP, PATCH.

```
def patch(url,name,light,indexUv):
    body = {
     "devs": name,
     "light": light,
     "place": "Icai,Madrid",
     "index_uv": indexUv,}
    y=json.dumps(body)
    headers = {'Content-type': 'application/json', 'Accept': 'text/plain'}
    r = requests.patch(url,data=y,headers=headers)
    print(r.status_code)
    print("body request:"  + r.request.body)
    print("response: " + r.text)
```

El desarrollo del código ha sido realizado en Python 2.7.

5.4.2 DESARROLLO Y DESPLIEGUE DE UNA REST-API

Para el desarrollo de la Rest-API se ha utilizado Node JS como el servidor y gestor de las rutas desarrolladas y Mongo DB, como base de datos donde se guarda la información del dispositivo.

Las funciones utilizadas durante el proyecto son las siguientes, véase en la figura 27.Se puede observar los distintos tipos de petición HTTP desarrollados, sus rutas para la realización de la petición, una explicación de estas y si se requiere, y si la petición espera la recepción de valores, para la correcta realización de la misma.

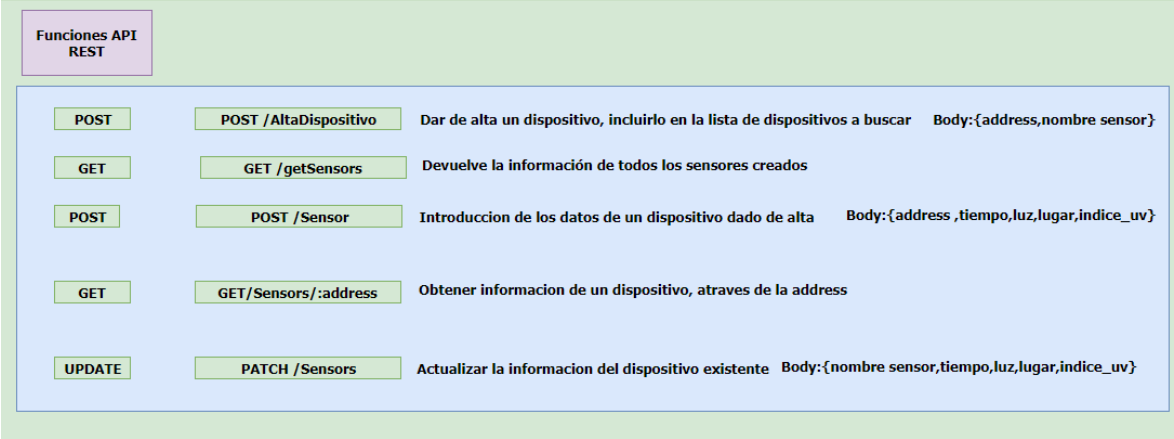


Figura 27. Esquema Funciones Rest-API

Para guardar los datos del dispositivo en la base de datos, se ha realizado un modelo gracias al módulo mongoose, el cual permite desarrollar modelos de objetos para la posterior introducción en MongoDB . El modelo desarrollado, es un sensor en el cual se han definido los campos necesarios para el correcto almacenaje del Sensor, siendo estos, address, tiempo, índice de luz, lugar e índice uv . Por ejemplo, si analizamos el campo address, se ha podido definir que sea un String, que siempre sea requerido cuando se crea un sensor, que la longitud de la cadena sea mayor que 10, y que la address sea única, es decir, que ningún sensor más posea esa misma address.

```
var mongoose = require('mongoose');
var Sensor = mongoose.model('Sensor', {
  address: {
    type: String,
    required: true,
    minlength: 10,
    unique: true
  },
  time: {
    type: Number,
    required:true,
  },
  light: {
    type: Number,
    required: true,
    default: null
  },
  place: {
    type: String,
    required: true,
```

```
  },
  index_uv: {
    type: Number,
    required: true
  }
});
module.exports = {Sensor};
```

Se va a explicar el código y funcionamiento de una función desarrollada dentro de la API , para así entender cómo se guarda la información dentro de la Rest- API. Explicación del a función de actualización del dispositivo, la cual es la que se comunica con el Gateway IoT.

1. Cuando esta función es llamada se verifica que el cuerpo de la petición cumple con la configuración deseada, para así saber que lo datos del sensor se podrán actualizar correctamente.
2. Se crea el modelo del sensor para su actualización en la base de datos
3. Se verifica que el id del sensor existe dentro de la base de datos de no ser así, manda una respuesta de error.
4. Una vez este es verificado, se procede a actualizar la información del sensor,

```
App.patch('/Sensors', (req, res) => {
 //Verificacion de que el cuerpo cumple con los
  var body = _.pick(req.body, ['devs', 'place','light','index_uv']);
  var id = mapaPKI.get(mapaDevs.get(body.devs));
  var sensor = {
    time: new Date().getTime(),
    light: body.light,
    place: body.place,
    index_uv: body.index_uv
  }
//verificacion de que el id se encuentra en el sistema
  if (!ObjectID.isValid(id)) {
    return res.status(404).send();
  }
//actualizacion del sensor
  Sensor.findByIdAndUpdate(id, {$set: sensor}, {new: true}).then((Sensor) => {
    if (!Sensor) {
      return res.status(404).send();
    }
    res.send({Sensor});
  }).catch((e) => {
    res.status(400).send();
  })
});
```

Para la implementación de la Rest-API en el sistema, al haberse utilizado el Node JS, el cual implementa el gestor npm, se ha podido configurar los módulos necesarios para el funcionamiento del sistema y configurar el archivo para correr la Rest-API.

Cuando queramos desplegar bastará con correr el siguiente comando, corriendo la Rest-API en el puerto 3000, del sistema donde se despliegue.

```
npm start
```

5.4.3 DESARROLLO Y DESPLIEGUE DE UN SMART CONTRACT.

Para la creación del Smart Contract se utilizó el lenguaje de programación Solidity .

El objetivo de este contrato es el almacenamiento de los datos de los dispositivos, por lo que se creó una estructura la cual guarda los datos de los dispositivos:

```
struct Dispositivo {
        address add_public;
        string name;
        uint256 hora_sc;
        uint256 ;
        int luz_ambiente;
        string lugar;
        int index_uv;
        address owner;
        bool estado;
        address address_miner;
        uint bloque_numero;
        uint bloque_dificultad;
    }
```

Para que un usuario que utilice este Smart Contracts como gestor de dispositivos, se ha implementado el uso de un Array que permite guardar múltiples estructuras, y las diferentes variables utilizadas, siendo todas privadas para que nadie menos el usuario las pueda visualizar.

```
Dispositivo[] private dispositivos;
address private usuario;
address private manager;
```

```
uint numeroDispositivos;
 //Map que sirve para saber el estado del dispositivo, false en posesion, true en perdida
mapping(address => bool) private estado_dispositivos;
//Map que sirve para saber el index en el array de dispositivos, para una sensor
mapping(address => uint) private map_dispositivos;
//Map que spermite saber si la direccion del dispositivo ya ha sido utilizada
mapping(address => bool) private existe_direccion;
 //Cuenta el numero de dispitivos creados en el contrato
 uint private index;
```

Para la definición del usuario como intermediario en el contrato, ya que el no despliega. En el constructor del contrato se tiene que introducir una address valida de la red BlockChain donde se quiera desplegar:

```
constructor(address user) public {
        manager = msg.sender;
        usuario = user;
        index=0;
    }
```

Para la definición de las reglas del contrato, se ha implantado el uso de una función la cual solo deja que el usuario dado de alta en el contrato pueda llamar a todas las funciones disponibles. Este código se explica siendo

```
modifier restricted_usuario() {
        require(msg.sender == usuario || msg.sender == manager);
        _;
    }
```

Definición de las funciones principales del contrato para la consecución del caso de uso:

1. Función de creación de un dispositivo dentro del contrato.

```
function crearDispositivo(address add_public ,string memory name,uint256
hora_api, int luz,string memory lugar,int uv) public restricted_usuario{

       require(existe_direccion[add_public]==false);

        Dispositivo memory newDispositivo = Dispositivo({
           add_public: add_public,
           name: name,
           hora_sc: now,
           hora_api: hora_api,
           luz_ambiente: luz,
           lugar: lugar,
           index_uv: uv,
           owner: usuario,
           estado: false,
           address_miner: block.coinbase,
           bloque_numero: block.number,
```

```
            bloque_dificultad: block.difficulty
        });

        dispositivos.push(newDispositivo);
        estado_dispositivos[add_public]=false;
        existe_direccion[add_public]=true;
        map_dispositivos[add_public]=index;
        existe_direccion[add_public]==true;
        index++;
    }
```

Como se puede ver implanta la función *restriced usuario*, con la cual solo el usuario dado de alta en el contrato puede ejecutar esta función. Las demás funciones también lo implementan. En la creación del contrato, se verifica que no se ha dado de alta un sensor con el mismo address.

2. Función de actualización de un dispositivo dentro del contrato.

La primera línea de código implementa si un dispositivo se encuentra en perdida, este no se pueda actualizar.

```
function actualizarDispositivo(address add_public,uint256 hora,int luz,string
memory lugar,int uv) public restricted_usuario {
        require(!estado_dispositivos[add_public]);
        Dispositivo storage disp = dispositivos[map_dispositivos[add_public]];
        require(!disp.estado);

        disp.hora_api=hora;
        disp.hora_sc=now;
        disp.luz_ambiente=luz;
        disp.index_uv=uv;
        disp.lugar=lugar;
        disp.address_miner= block.coinbase;
        disp.bloque_numero= block.number;
        disp.bloque_dificultad= block.difficulty;

    }
```

3. Funciones de reporte del estado de un dispositivo dentro del contrato

Se verifica antes de cambiar el estado, que el dispositivo no está en el estado al que lo queremos cambiar.

```
function reportarMissing(address id) public restricted_usuario
```

```
    {
        require(!estado_dispositivos[id]);
        Dispositivo storage disp = dispositivos[map_dispositivos[id]];
        require(!disp.estado);
        disp.estado= true;
        estado_dispositivos[id]= true;
    }
     function reportarPosesion(address id) public restricted_usuario
    {
        require(estado_dispositivos[id]);
        Dispositivo storage disp = dispositivos[map_dispositivos[id]];
        require(disp.estado);
        disp.estado= false;
        estado_dispositivos[id]= false;

    }
```

4. Funciones de visualización de la información del contrato

Al implementar la función restricted usuario, las funciones de petición de información de los dispositivos, solo podrán ser ejecutadas por el usuario.

```
function getInfoDispositivo(address id) public view restricted_usuario returns (
        string memory,uint256,uint256,int,string
memory,int,bool,address,uint,uint){
         Dispositivo storage disp = dispositivos[map_dispositivos[id]];
         return(
             disp.name,
             disp.hora_sc,
             disp.hora_api,
             disp.luz_ambiente,
             disp.lugar,
             disp.index_uv,
             disp.estado,
             disp.address_miner,
            disp.bloque_numero,
            disp.bloque_dificultad);}

function getInfoDispositivoindex(uint id) public view restricted_usuario returns
(
        string memory,uint256,uint256,int,string
memory,int,bool,address,uint,uint,address){
            require(id < 6);
         Dispositivo storage disp = dispositivos[id-1];
         return(
             disp.name,
             disp.hora_sc,
             disp.hora_api,
             disp.luz_ambiente,
             disp.lugar,
             disp.index_uv,
             disp.estado,
```

```
             disp.address_miner,
            disp.bloque_numero,
            disp.bloque_dificultad,
            disp.add_public
         );
    }
```

Para el despliegue de un contrato inteligente utilizaremos las tecnologías Truffle. Para poder realizar migraciones sobre una red, se de configurar el fichero truffle.js, configurando el fichero para el despliegue de la red elegida. Para el correcto despliegue del contrato en la se deben configurar los siguientes os pensar en el host, el puerto, el ID de la red, el precio del Gas.

```
module.exports = {
  // Uncommenting the defaults below
  networks: {
    development: {
      host: "172.24.128.77",
      port: 7545,
      network_id: "5777"
    },
  alastria: {
        host: "130.206.64.6",
        port: 22000,
        network_id:"82584648528",
        gasPrice:0,
        gas:2000000
    },

    test: {
      host: "127.0.0.1",
      port: 8545,
      network_id: "*"
    }
  }

};
```

Para ejecutar la compilación del contrato y su migración, se utilizan los siguientes comandos

```
Truffle compile
Truffle migrate -network <red_a_elegir>
```

5.4.4 DESARROLLO Y DESPLIEGUE DEL SERVIDOR WEB

Para el correcto lanzamiento de la aplicación web, se ha creado un servidor web el cual ha despliega la misma sobre el puerto 8000 del sistema deseado.

Para la configuración de este servidor se ha implementado el modulo express, para la definición de las rutas gestionadas por este. A cada una de las rutas se le asigna una request y una response, cuando se accede una ruta definida en el servidor, este responde con un fichero HTML de las paginas que se han desarrollado, por ejemplo, si hago, /visualizar, es servidor mandara al browser a “Visualizar.html”,

```
const express = require('express');
const path = require('path');
const app = express()
app.use(express.static(__dirname + '/public'));
app.get('/', function(req, res) {
	res.sendFile(path.join(__dirname + '/public/html/index.html'));
});
app.get('/visualizar', function(req, res) {
	res.sendFile(path.join(__dirname + '/public/html/visualizar.html'));
});
app.get('/crear', function(req, res) {
	res.sendFile(path.join(__dirname + '/public/html/crear.html'));
});
app.get('/actualizar', function(req, res) {
	res.sendFile(path.join(__dirname + '/public/html/actualizar.html'));
});
app.get('/reportar', function(req, res) {
	res.sendFile(path.join(__dirname + '/public/html/reportar.html'));
});
const PORT = process.env.PORT || 8000;
app.listen(8000, function() {
	console.log('Listening on port ' + PORT + '...');
});
```

Para cuando se quiera correr el servidor, se debe ejecutar el siguiente comando

```
npm start
```

Este servidor se desplegará sobre el puerto 8000, pudiendo acceder a él, por ejemplo,

Introducir en Google Chrome ‘localhost:8000’.

Capítulo 6. ANÁLISIS DE RESULTADOS

6.1 GESTOR DE DISPOSITIVOS

Se va a explicar el proceso seguido para el despliegue del contrato creado, para así poder utilizar el sistema desarrollado.

Para el despliegue del contrato, se va a utilizar la red de pruebas Ganache, la cual nos la proporciona el consorcio de Truffle. Esta red es muy cómoda ya que al poseer una interfaz gráfica.

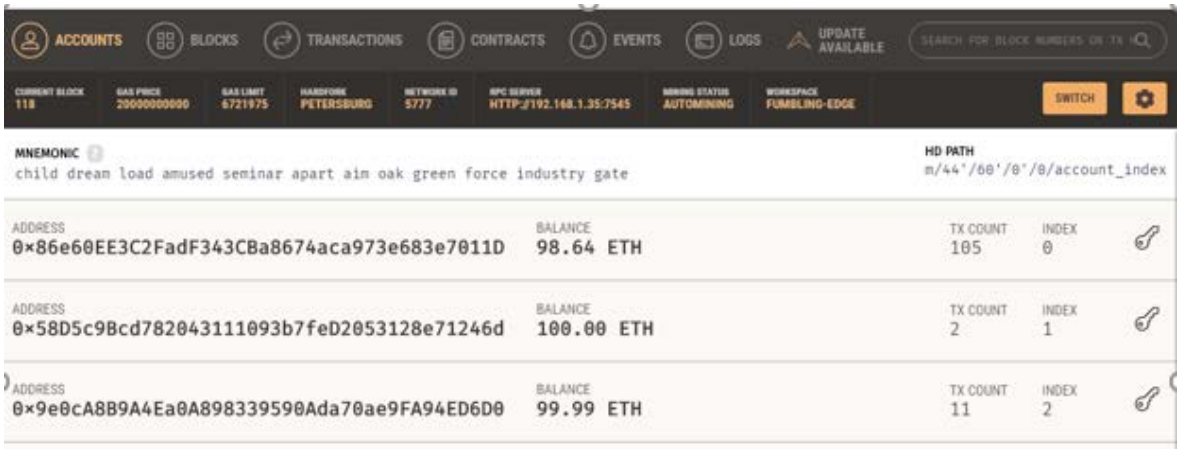


Figura 28. Inicialización de Ganache

Se procede a realizar la migración del contrato desplegado sobre ganache, utilizando la herramienta Truffle, la cual ofrece la compilación y la migración del contrato:

```
osboxes@osboxes:~/Desktop/Dispositivos$ ls
contracts  migrations  test  truffle-config.js
osboxes@osboxes:~/Desktop/Dispositivos$ truffle compile
Compiling ./contracts/Device.sol...
Compiling ./contracts/Migrations.sol...
Writing artifacts to ./build/contracts

osboxes@osboxes:~/Desktop/Dispositivos$ ls
build  contracts  migrations  test  truffle-config.js
osboxes@osboxes:~/Desktop/Dispositivos$ truffle migrate --network development
Using network 'development'.

Running migration: 1_initial_migration.js
  Deploying Migrations...
  ... 0xe1262b26b896db1d5f8ca8ed51c635864fe3a68f252daf85ad2ffeb2b2d67c85
  Migrations: 0xf1d1d4f9740d058c86c7c95affea16036a45fad4
Saving artifacts...
Running migration: 2_deploy_contracts.js
  Deploying Device...
  ... 0x6bf5eaf6563ac711ee1a5f6b9b1dab26e01ad8dbc896dfae9f9237ed9725fc09
  Device: 0x1c1eb254882c094ecc867adacc77369e260fce34
Saving artifacts...
osboxes@osboxes:~/Desktop/Dispositivos$
```

Figura 29. Despliegue de un contrato utilizando Truffle

El contrato desplegado solo puede interaccionar con la cuenta de usuario definida en su despliegue, la cual ha sido "0x9e0ca8b9a4ea0a898339590ada70ae9fa94ed6d0", y la address del contrato es "0x1C1EB254882C094Ecc867adaCC77369E260FCE34".

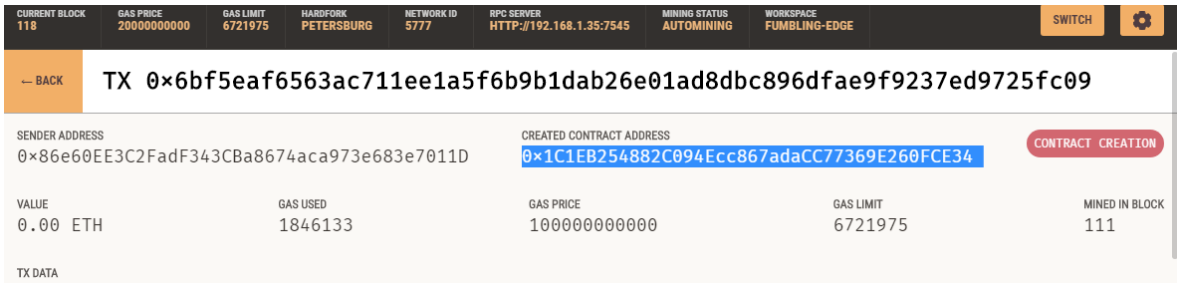


Figura 30. Creación del contrato dentro de ganache

Para el análisis de resultados del gestor de dispositivos se van a evaluar las tres funciones de utilizan la ejecución de transacciones con Metamask, las cuales son la creación, actualización y reporte de dispositivos.

Las transacciones de Metamask permiten aceptar o rechazar las transacciones generadas al invocar funciones que modifican información dentro del Smart Contract. En estas se

realiza una verificación del usuario y de los datos introducidos, los cuales no pueden ser duplicados, para la correcta ejecución de la transacción. Véase en el Anexo B la configuración de cuentas en Metamask.

6.1.1 CREACIÓN DE UN DISPOSITIVO

En este apartado se va a dar de alta un dispositivo en el gestor. Para poder dar de alta un dispositivo hacen falta la introducción de tres campos los cuales son: nombre del dispositivo, dirección que se le quiera asignar al dispositivo y el lugar donde el Gateway detectaría el dispositivo. Los datos introducidos han sido el nombre del sensor , Thunder Sense #30096,el cual es el nombre del sensor real utilizado para la realización del proyecto ,y la dirección que le hemos asignado ha sido "0x6B54dC49982a32f982068F16820d69A931ADb235", utiliza el formato de una address, dirección, ya que para este tipo de direcciones hay miles de combinaciones posibles .

Figura 31. Creación de un dispositivo en el gestor

Una vez se pulsa el botón de crear, el servidor web llama a la función del contrato inteligente, la cual crea el dispositivo. Esta acción genera una transacción de la cuenta del usuario a la dirección donde se encuentra el contrato desplegado.

En la siguiente figura se puede observar cómo aparece una ventana de Metamask, la cual ofrece al usuario confirmar o rechazar la transacción de la introducción de un nuevo dispositivo, en el contrato.

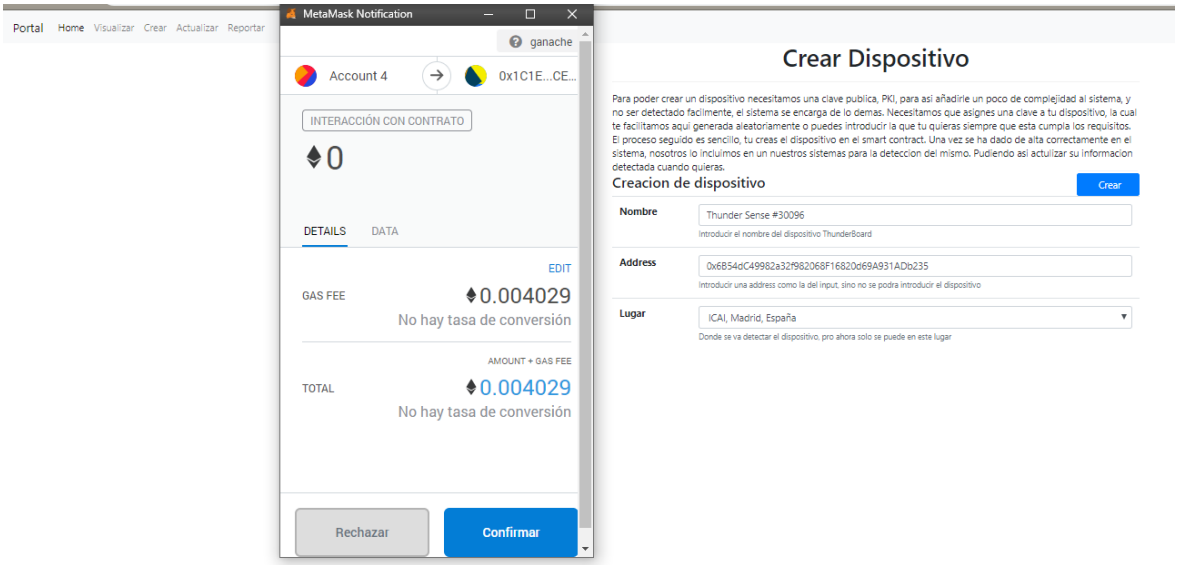


Figura 32. Transacción para la creación de un dispositivo

Debido a que la cuenta seleccionada es la indicada y los datos introducidos son correctos, , además el contrato acaba de ser desplegado, por lo que se encuentra en blanco, no se encuentra ningún dispositivo creado. La transacción ha sido confirmada y dispositivo ha sido creado exitosamente.

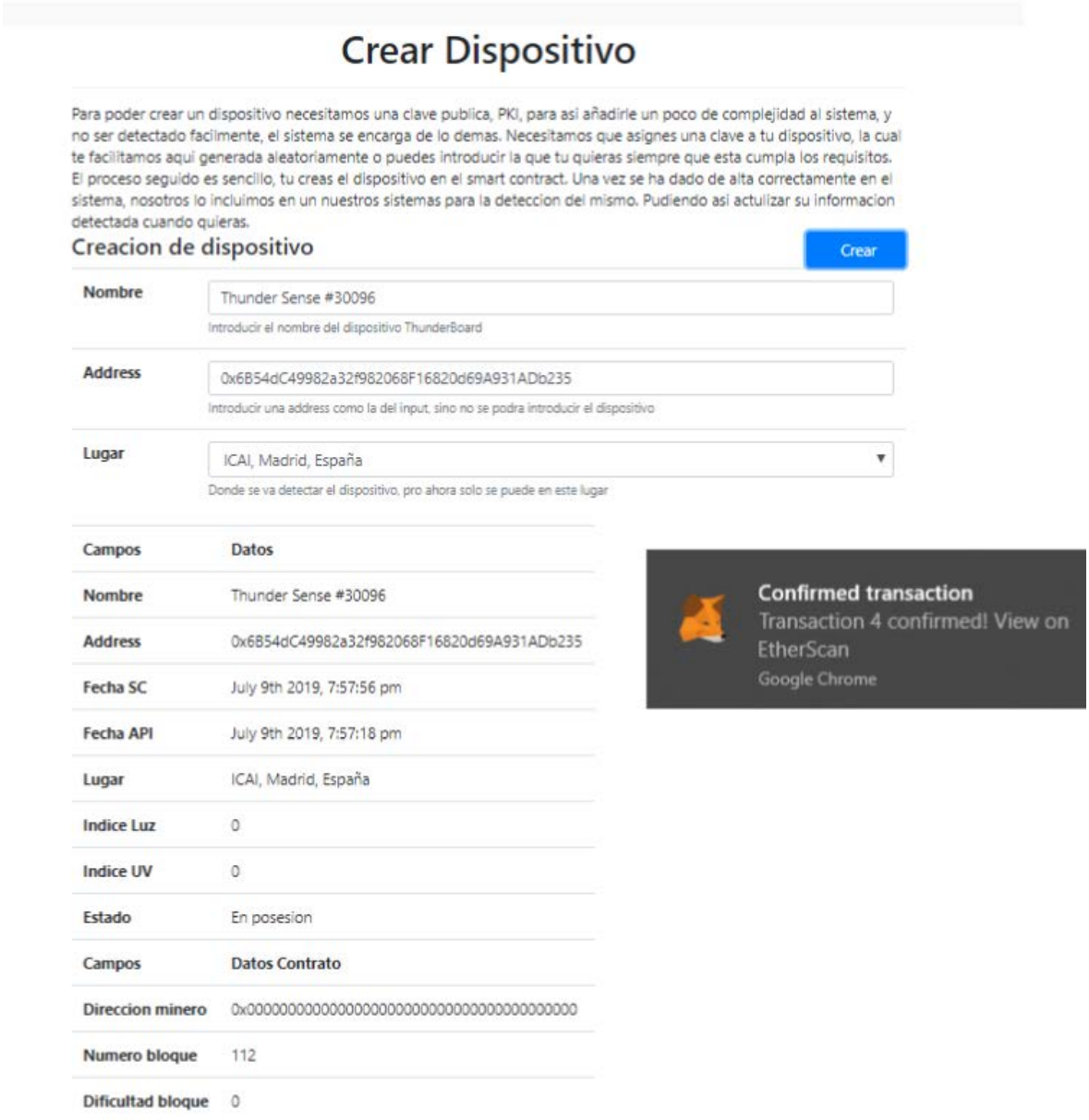


Figura 33. Transacción exitosa de la creación de un dispositivo

Como se puede observar la transacción ha sido realizada exitosamente, creando el dispositivo de manera correcta ya que los campos de *address* del dispositivo, nombre y lugar coindicen, con los datos introducidos anteriormente. Los demás campos también han sido creados exitosamente, vamos a analizar su correcta creación. Las Fechas SC y Fecha API, significan la fecha y hora actual de la creación del dispositivo, tanto en el Smart Contracts como en la Rest-API, estas han sido creadas de forma correcta. Ya que si

miramos la hora de la transacción ejecutada coincide con la fecha del Smart Contract. El índice de luz y el índice uv, se le ha asignado por defecto el valor “0”, ya que estos todavía no han sido detectados, ya que estamos creando la información dentro del contrato. El estado del dispositivo se encuentra en posesión, ya que acaba de ser creado. Los siguientes datos, son los campos de la ejecución de la transacción, informando al usuario quien ha minado la transacción, el número de bloque donde dentro de la cadena de bloques y la dificultad de minado de este, enumerados por orden de arriba a abajo. Al estar el contrato desplegado sobre ganache, no hay ningún minero que incluya la transacción, sino que el mismo sistema lo hace por sí solo, ya que este es un entorno de pruebas.

Véase en la figura, la transacción realizada dentro de la cadena de bloques.

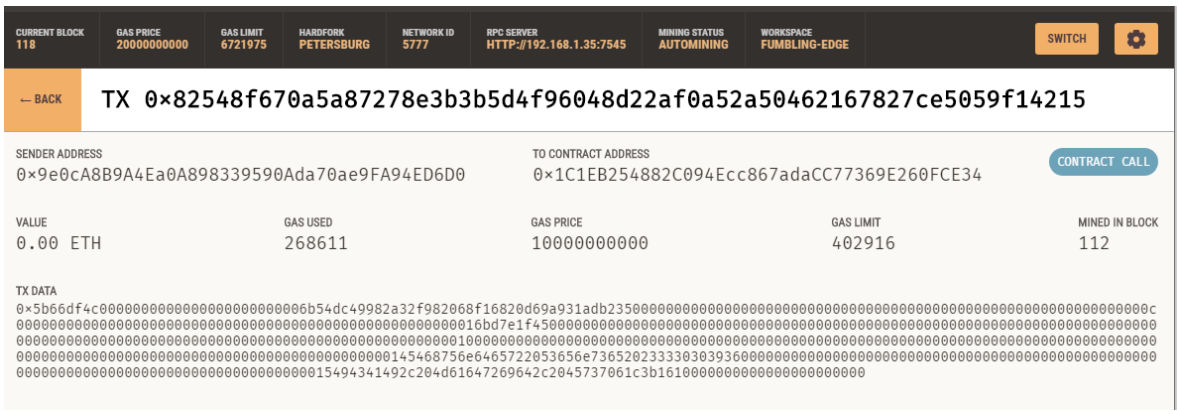


Figura 34. Transacción ejecutada en ganache para la creación del dispositivo

El dispositivo ha sido creado exitosamente dentro del contrato, pero también ha sido dado de alta en la REST-API, véase en la figura35 la correcta creación del dispositivo, ya que se han ejecutado dos peticiones HTTP POST. La primera ha sido la creación del dispositivo dentro del sistema de detección de la API, enviando los valores de address y el nombre, para que en los siguientes métodos se permita trabajar con la address asignada al sensor. La siguiente petición, ha sido un post de la introducción de los datos creados en el contrato para así crear el objeto sensor, el cual para ser creado necesita la recepción de la address,

fecha actual para el Rest-Api, el lugar donde se quiere detectar el dispositivo, y los índices de luz y uv.

Figura 35. Correcta creación del dispositivo en la Rest-API

Analizado todo el proceso de creación del dispositivo, el alta del dispositivo en el gestor se ha completado con éxito, pudiendo visualizar y comprobar que la correcta creación, ya que, al estar el contrato vacío, sería el primer dispositivo creado, véase en la siguiente figura. Pudiendo el usuario volver a crear otro dispositivo diferente al anterior, siempre que sea con otra address y otro sensor.

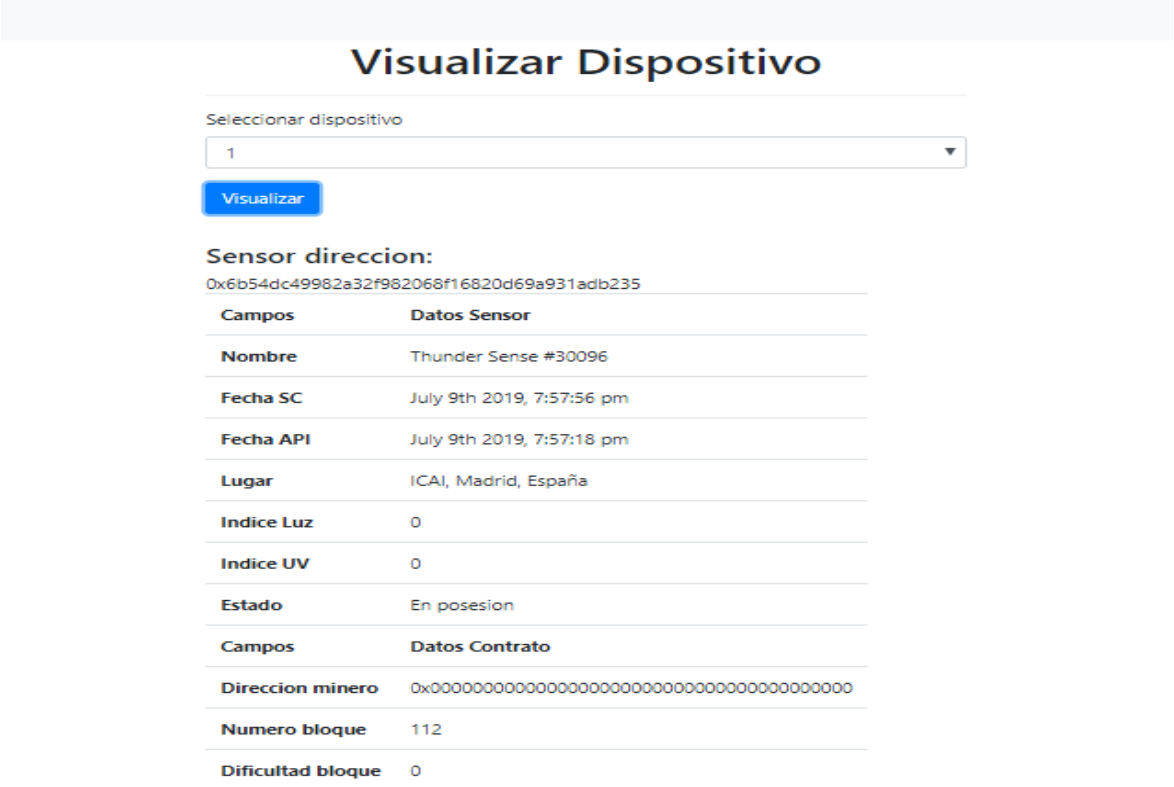


Figura 36. Visualización de los datos del dispositivo creado

6.1.2 ACTUALIZACIÓN DE UN DISPOSITIVO

En este apartado se analiza la actualización de información del dispositivo creado en el apartado anterior. Este apartado es el más importante ya que es el que de verdad genera un gran valor al gestor desarrollado, siendo el método el cual nos permite ver la última información que ha sido detectada al sensor en cuestión, pudiendo ver si este ha salido del edificio, cuando, en qué estado, y como.

Para la actualización de información de un dispositivo, el gateway IoT tiene que actualizar la información de este, véase en la siguiente figura. Tras la exitosa actualización del dispositivo, se verifica la actualización de la información de este en la REST-API, véase en la siguiente figura. Los datos del sensor que han sido actualizados han sido, el índice de luz

detectado por el sensor y el índice uv, para el sensor Thunder Sense #30096, creado en el gestor anteriormente.

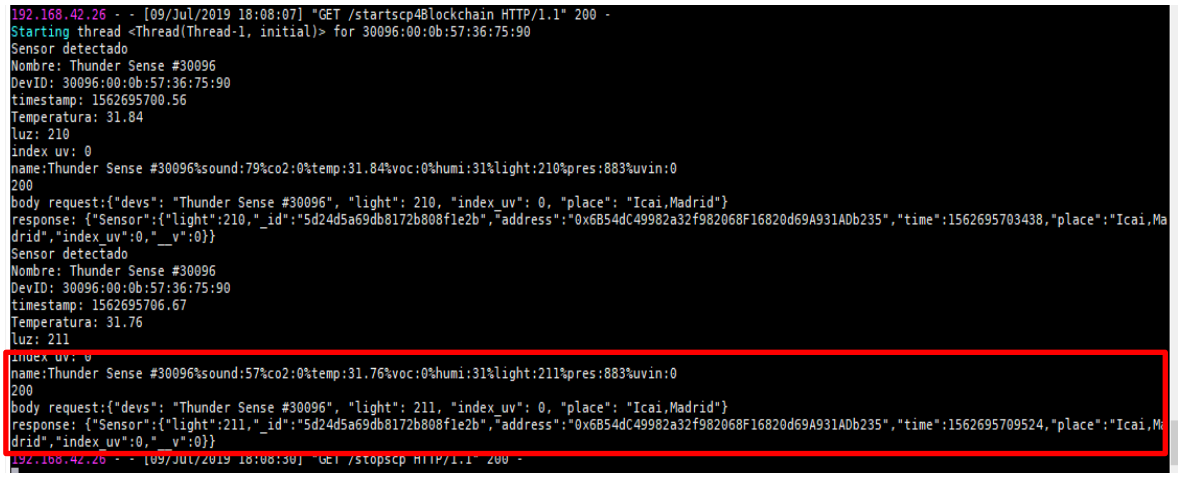


Figura 37. Ultima conexión detectada por el gateway

Figura 38. Actualización en la Rest-API del dispositivo detectado

Una vez se han actualizado los datos del sensor creado, se dispone a la actualización de estos en el Smart Contract, utilizando el sistema de actualización desarrollado en el gestor de dispositivos.

Para la correcta actualización del sensor Thunder Sense #30096, se debe introducir la dirección asociada al dispositivo, la cual es 0x6B54dC49982a32f982068F16820d69A931ADb235.

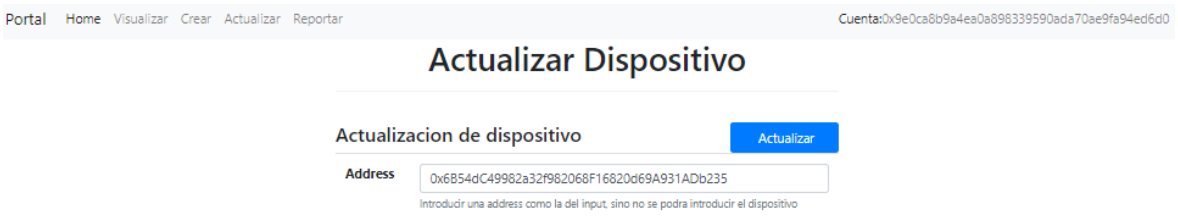


Figura 39. Actualización de un dispositivo en el gestor

Una vez se pulsa el botón de actualizar, el servidor web realiza una petición sobre la Rest-Api, una petición GET con la identificación de la address, si no se encuentra el dispositivo en esta, la transacción nunca se llegaría a realizar, ya que antes daría un error la función desarrollada de actualización de los dispositivos ,la cual devuelve la información actualizada del dispositivo. Una vez completada esta petición, el servidor web llama a la función del contrato inteligente, la cual realiza la transacción para la actualización de información asociada al dispositivo, introduciendo en esta los datos obtenidos por la petición anterior, la cual queda pendiente de confirmación.

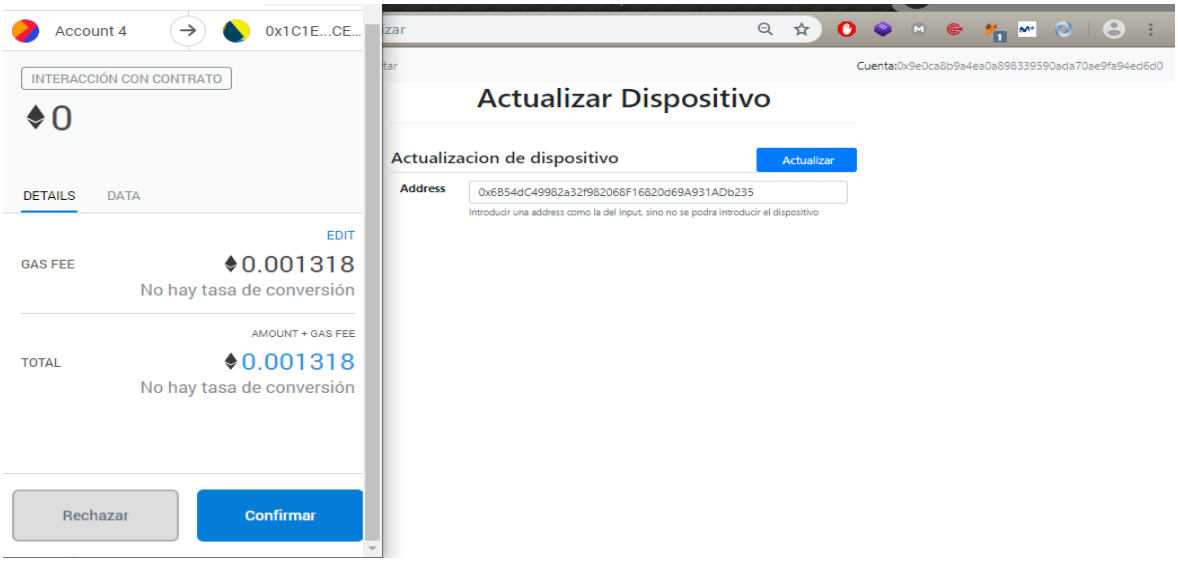


Figura 40. Transacción para la creación de un dispositivo

Confirmamos en Metamask la actualización del dispositivo, y visualizamos que los datos obtenidos por el Gateway IoT, posteriormente actualizados en la Rest-Api, son los mismos que los que se han incluido en el contrato Inteligente. Siendo estos los mismos que en la figura 38, pudiendo verificar la correcta actualización del dispositivo. En la siguiente figura se puede verificar la correcta ejecución de la actualización ya que la fecha de la fecha de la actualización del dispositivo en la Rest-API, es anterior a la fecha de actualización del dispositivo en el contrato.

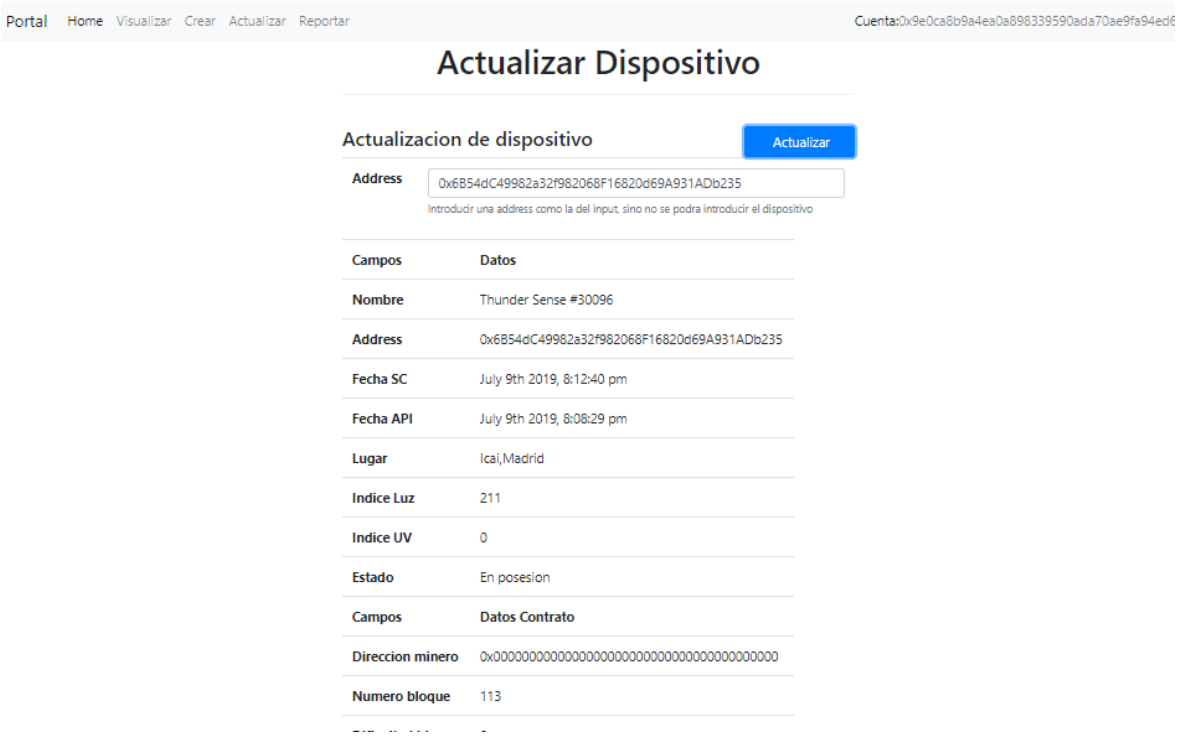


Figura 41. Transacción exitosa de la creación de un dispositivo

Confirmamos que la transacción ha sido realizada con éxito en la cadena de bloques, véase en la siguiente figura.

Figura 42. Transacción ejecutada en ganache para la actualización del dispositivo

6.1.3 REPORTAR ESTADO DE UN DISPOSITIVO

En este apartidado se analiza la realización de un reporte de pérdida del dispositivo.

Para poder realizar un reporte de perdida en el gestor, debemos introducir la dirección asociada al dispositivo. Hacemos click en el botón de perdida, el cual llamara a la función del contrato la cual se encarga de esta funcionalidad. Se genera una transacción a confirmar para la realización del reporte de la perdida.

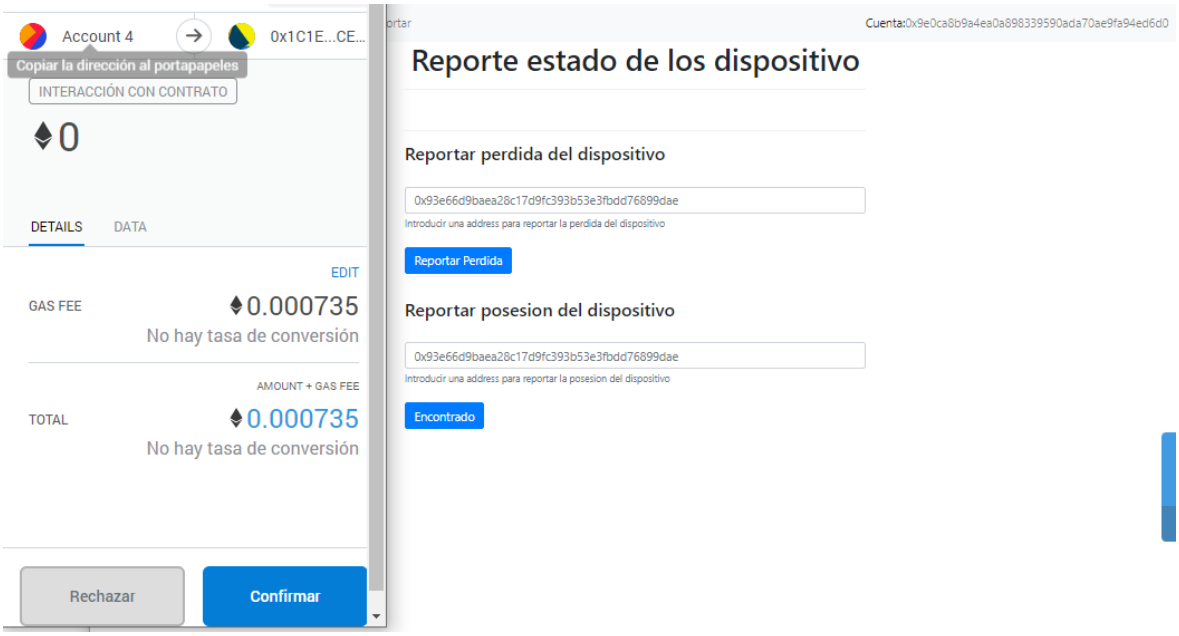


Figura 43. Reporte de la pérdida del dispositivo

Una vez se reporta la pérdida del dispositivo, el gestor está diseñado para que cuando esto pase, la función de actualización de la información quede obsoleta. El motivo es que cuando se reporta una perdida se quiere saber cuál fue su última conexión, ya que este podría haber salido del edificio, por lo que quedaría registrada de forma totalmente trasparente y confiable e inmutable, la hora en la cual el dispositivo salió del edificio. Esta información al proveer de una cadena de bloques es totalmente trasparente, por lo que se podrían tomar medidas para la búsqueda del dispositivo.

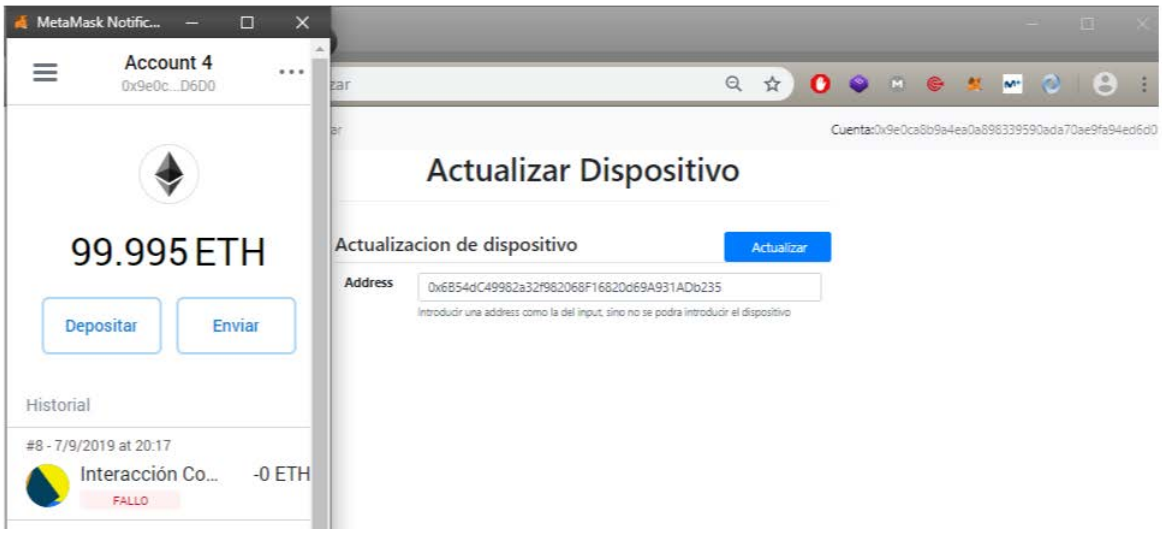


Figura 44. Fallo en la actualización del dispositivo por reporte perdida

El gestor también ofrece el reporte de la nueva posesión del dispositivo, para que así si encontramos este, podamos seguir actualizando la información de nuestros dispositivos.

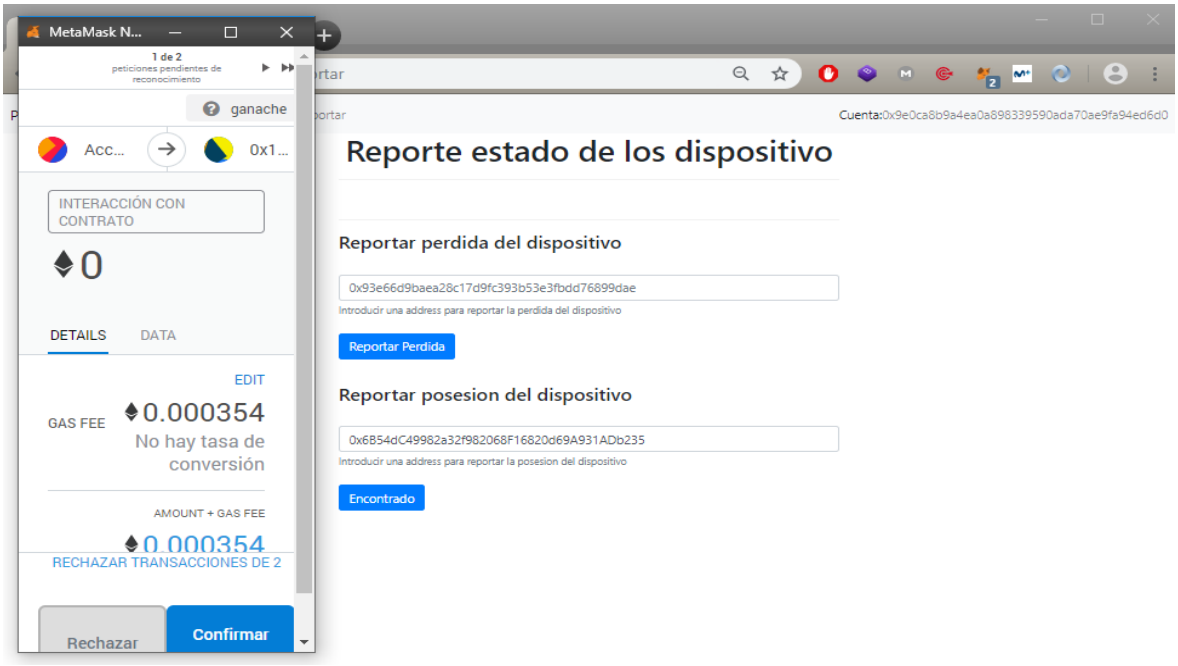


Figura 45. Reporte de la nueva posesión del dispositivo

6.2 DESPLIEGUE DEL SMART CONTRACT EN LA RED ALASTRIA

El objetivo del despliegue del contrato en la red Alastria no se ha podido cumplir, debido al cambio de Test Net de Arrakis a Telsius. Esto ha supuesto la aparición de numerosos problemas, ya que la red utiliza otras tecnologías diferentes. El mayor error encontrado fue que el clúster donde se alojaban lo nodos soportaba un sistema operativo CentOS, y los requisitos de despliegue de un nodo sobre Telsius solo acepta el sistema operativo Ubuntu 16.04. A la hora de la instalación se consiguió desplegar una imagen de Ubuntu 16.04, pero esta daba errores en la instalación de Docker.

Finalmente se consiguió resolver el problema, véase el Anexo A, pudiéndose así desplegar el nodo regular en sobre Telsius[16]:

Figura 46. Despliegue de un nodo validador sobre Alastria (Fuente: Monitor Telsius Alastria)

Aunque el nodo fuera desplegado, la complejidad de configuración de la red alastria es notable, por lo que queda pendiente en trabajos futuros.

Capítulo 7. CONCLUSIONES Y TRABAJOS FUTUROS

7.1 CONCLUSIONES

Muchas de las soluciones IoT no son totalmente seguras, ya que abarcan muchas casuísticas, como se hablaba en el reporte de seguridad de IoT. El sistema desarrollado se ha centrado en mejorar uno de los grandes problemas detectados en IoT, el cual es la autenticación y la autorización, cumpliendo con las motivaciones planteadas en el caso de uso. Sobre todo, el uso de los contratos inteligentes es lo que hace que la tecnología BlockChain este escalando e implantando numerosos casos de uso en muchas de las principales industrias, ya que con estos se pueden implementar cualquier casuística que se pueda programar, y así tener sistemas totalmente seguros y confiables.

Se ha podido observar que el uso de una cadena de bloques mejora la confianza de las soluciones IoT, ya que se pueden utilizar como un sistema confiable en el cual se puede realizar diferentes casuísticas, como la implementación de pagos entre servicios, trazabilidad de los sensores, almacenamiento seguro de los datos.

Durante el desarrollo del sistema desarrollado en el proyecto, se ha detectado la falta de un módulo de identificación segura para el dispositivo IoT, ya que, si se implementara el uso de PKI, esto facilitaría la autenticación y autorización de la información por las dos partes, ya que directamente se podría implementar una dirección publica y una privada para el cifrado en las comunicaciones desde el dispositivo a la cadena de bloques. El módulo pki se instalaría en el sensor y se integraría con los demás sistemas.

Hay que destacar la madurez del lenguaje de programación de contratos inteligentes Solidity, ya que se ha podido implementar todas las funcionalidades requeridas en el proyecto, además habiendo una muy buena documentación y buenas herramientas para su compilación y despliegue.

Hay que destacar la compleja configuración e implementación de la red Alastria, no pudiéndose desplegar la aplicación sobre esta red, ya que le falta madurez y una configuración y un desarrollo más sencillo, mejorando con respecto al año anterior.

7.2 TRABAJOS FUTUROS

No se ha cumplido con todos los objetivos establecidos, ya que no se ha podido implementar el uso de la red alastria al sistema desarrollado. Para trabajo futuros, se realizará la correcta configuración de esta, ya que la instalación del nodo llevo varias semanas solucionar el problema de la instalación de Docker. Si la configuración de alastria es realizada con éxito, tiene mucho potencial ya que es una red mixta, la cual implementa transacciones públicas y privadas, y además cuenta con una gran comunidad, formada por las empresas más importantes del sector tecnológico español.

Aun que se ha cumplido con los objetivos del proyecto, me gustaría destacar el uso de dispositivos que implementan un módulo PKI, para así su implantación en la infraestructura, ya que esto hace que la forma de detección del dispositivo cambie totalmente, consumiendo así menos recursos ya que el detector solo detectaría los dispositivos los cuales conoce su clave privada, pudiendo así generar una conexión totalmente segura.

Cabe destacar que se ha tenido que desarrollar una Rest-API, para el desarrollo del proyecto, pero hay soluciones BlockChain que ya implementan el uso de estas para la captación de información, como Hyperledger Fabric, además de que también implementa el desarrollo de contratos inteligentes. Se recomienda el uso de esta red en proyectos futuros ya que también está basada en BlockChain con IoT ya que simplifica mucho el proceso de desarrollo.

Del caso de uso desarrollado se podría crear una aplicación a nivel real, en la cual el desarrollo del sistema, pero a niveles más avanzados, pudiendo ofrecer varios lugares

donde poder detectar tus dispositivos, y así generar un sistema de perdida de dispositivos el cual funcione a nivel de ciudad o de comunidad.

Capítulo 8. BIBLIOGRAFÍA

[1] Forbes.com , Top Tech Trends In 2019: 11 Experts Detail What You Need To Watch , https://www.forbes.com/sites/forbestechcouncil/2018/12/20/top-tech-trends-in-2019-11-experts-detail-what-you-need-to-watch/#11eb116e5ae8

[2] Deloitte, Blockchain and the five vectors of progress: https://www2.deloitte.com/insights/us/en/focus/signals-for-strategists/value-of-blockchain-applications-interoperability.html

[3] Rasberry Pi 3: https://www.raspberrypi.org/products/raspberry-pi-3-model-b/

[4] Thunderboard sense BLE sensor: https://www.silabs.com/products/development-tools/thunderboard/thunderboard-sense-kit

[5] The Economist:

Blockchains the great chain of being sure about things" artículo Publicado en The Economist en su edición del 31 de Octubre de 2015

[6] Solidity GitHub, https://github.com/ethereum/solidity

[7] Remix, Github : https://github.com/ethereum/remix-ide,

[8] Remix IDE: https://remix.ethereum.org/#optimize=false&evmVersion=null

[9] IWO: Las grandes estadísticas del Internet de las Cosas (IoT) https://www.iotworldonline.es/las-grandes-estadisticas-del-internet-de-las-cosas-iot/

[10] Mas que negocios: La importancia del Internet de las Cosas para los negocios

https://www.masquenegocio.com/2017/10/31/internet-de-las-cosas-negocios/

[11] IoT Analytics "IoT Security Market", 2017.

https://iiot-world.com/reports/an-overview-of-the-iot-security-market-report-2017-2022/

[12] Figura Diferentes tipos de BlockChain que implementan Smart Contracts,

https://blogs.iadb.org/conocimiento-abierto/es/tipos-de-blockchain/

[13] Alastria refuerza su estructura para afrontar con garantías los retos de 2019. https://medium.com/@alastria_es/alastria-refuerza-su-estructura-para-afrontar-con-garant%C3%ADas-los-retos-de-2019-6be0bb0c85ac

[14] GitHub de Alastria, https://github.com/alastria

[15] Quorum, JPmorgan , https://www.goquorum.com/

[16] Monitor Telsius Alastria, http://netstats.telsius.alastria.io/

ANEXO A – PUESTA EN MARCHA DE UN NODO REGULAR SOBRE ALASTRIA TELSIUS

En este anexo se explica el proceso de configuración de un nodo regular de Alastria, ya que la antigua Test Net de pruebas Arrakis ha quedado obsoleta, por consiguiente, los nodos instalados en 2018 han dejado de funcionar. Ahora la nueva Test Net principal se llama T(Telsius), y es sobre donde se ha desplegado el nodo regular. Solo se realiza la instalación del nodo regular debido a que es un tipo de nodo el cual permite el despliegue de contratos inteligentes

Para la configuración del nodo regular de Alastria se ha seguido la siguiente instalación oficial de un nodo regular provista por Alastria:

- https://medium.com/babel-go2chain/c%C3%B3mo-poner-en-marcha-un-nodo-regular-en-la-red-telsius-de-alastria-876d9dcf7ccb

Requerimientos para la puesta en marcha

Para la instalación de un nodo regular en el clúster de la universidad , se deben cumplir los siguientes requerimientos,

Sistema operativo: El sistema donde se despliegue el nodo debe ser Ubuntu 16.04.

Docker: Necesitamos tener instalado Docker en la máquina donde se va desplegar el nodo, ya que se necesita para el despliegue del nodo .

Puertos: Dentro del sistema donde se va desplegar la red, se debe permitir el tráfico a los siguientes puertos:

- 21000/TCP y UDP: Permite conectarse y sincronizarse a los nodos entre sí. Esto se hará incluyendo el "enode", IP y puerto 21000 en los ficheros: boot-nodes.json, regular-nodes.json o validator-nodes.json dependiendo del tipo de nodo que sea.
- 80/TCP y 443/TCP: Puertos en los que escucha el Access Point que filtra algunas familias de la API Geth y trabaja por defecto con listas blancas.
- 9000/TCP: Este puerto será opcional. Es el encargado de las transacciones privadas de Constellation. Si en nuestro caso no queremos realizar transacciones privadas con otros socios no es necesario que habilitemos el puerto.
- 8443/TCP: Este puerto será opcional. Exposición de la API REST del monitor del nodo. Para poder acceder a este API, es necesario disponer de un certificado digital instalado en el navegador. El monitor vendrá instalado por defecto en las imágenes de Docker de los nodos regulares, pero no será obligatorio tenerlo operativo.
- **Hardware:**

Hardware	minimum	desired
CPU's:	2	4
Memory:	4 Gb	8 Gb
Hasrd Disc:	100 Gb	1000 Gb

Requisitos de Hardware mínimos y deseados para un nodo regular.

Configuración del sistema y puesta en marcha del nodo deseado

En este apartado se va a explicar el proceso llevado para el despliegue de un nodo regular Alastria, sobre el clúster de la universidad.

Para una correcta puesta en marcha se debe comprobar si el sistema donde se va a configurar el nodo cumple con los requerimientos anteriormente mencionados, el cual cumplía con todos ellos menos con la instalación de docker dentro de Ubuntu 16.04

Solución al problema de Instalación de docker sobre una imagen Ubuntu 16.04

A la hora de realizar la instalación Docker sobre el contendor de Ubuntu 16.04, este dio un error el cual no dejaba realizar la instalación del nodo.

Problema detectado:

```
docker: Cannot connect to the Docker daemon at unix:///var/run/docker.sock. Is
the docker daemon running?.

See 'docker run --help'.
```

Para la solución de este error, se modifico el limite de recursos empleados por Docker dentro del contendor Ubuntu 16.04. Solucionando así el problema y pudiendo correr Docker dentro del sistema operativo.

Véase la información detallada de la instalación de Docker sobre un contendor Ubuntu 16.04

- https://www.digitalocean.com/community/tutorials/como-instalar-y-usar-docker-en-ubuntu-16-04-es

Una vez solucionado el problema , ejecutamos e instalamos docker sobre Ubuntu 16.04:

```
sudo docker run -it --ulimit nproc=1248576:1248576 --privileged=true --name
icaitelsius02 -p 2222:22 -p 8443:8443 -p 21000:21000 -p 22000:22000 -p 9001:9000
-p 21000:21000/udp ubuntu:16.04
apt-get update;apt-get install sudo;sudo apt-key adv --keyserver
hkp://p80.pool.sks-keyservers.net:80 --recv-keys
58118E89F3A912897C070ADBF76221572C52609D;sudo apt-add-repository 'deb
https://apt.dockerproject.org/repo ubuntu-xenial main';sudo apt-get update;sudo
```

```
apt-get install -y software-properties-common; sudo apt-get install apt-
transport-https;sudo apt-get update;apt-cache policy docker-engine;sudo apt-get

instaló -y docker-engine
```

Una vez instalado Docker sobre Ubuntu 16, accedemos al contenedor para la instalación del nodo.

```
service dbus start
service docker start
root@2cec73523833:/# cd
root@2cec73523833:~# pwd
/root
```

Por lo que podemos ver que docker ya funciona correctamente dentro del contenedor Ubuntu 16.04.

Puesta en marcha del nodo regular

Ya cumpliendo todos los requerimientos anteriormente dichos, se procedió a la puesta en marcha del nodo regular siguiendo la guía de instalación oficial provista por Alastria.

Pasos a seguir:

1. Descargamos el repositorio con el siguiente comando:

```
git clone -b testnet2 https://github.com/alastria/alastria-node
```

2. Navegamos hasta la carpeta donde esta el script para lanzar la instalación del nodo:

```
cd alastria-node/docker/general
```

3. Ejecutamos el script init dentro de esta ruta:

```
./init.sh
```

4. Respondemos a todas las preguntas que nos vaya haciendo el script:

- Nombre de la compañía
- Número de Cus de la máquina
- Número de Ram de la máquina
- Número de secuencia de nodos en la red, empezando por 00 si es primer nodo e incrementando en un en caso de tener más (00, 01, 02 …)
- Deseas lanzar el monitor (Y/N)
- Deseas habilitar constellation para las transacciones privadas (Y/N)

  Confirmamos si la información es correcta, introduciendo 1 en caso afirmativo o 2 en caso negativo.

Ejecución realizada para la instalación del nodo:

```
root@2cec73523833:/# cd
root@2cec73523833:~# pwd
/root
git clone -b testnet2 https://github.com/alastria/alastria-node
cd alastria-node/docker/general
root@d59c86fa3437:~/telsius/alastria-node/docker/general# ./init.sh
```
```
Write company name:
Comillas
Number of CPUs:
20
Number of RAM:
128
Sequential starting at 00:
01
1) Yes
2) No
Do you want to install the monitor?
Press 1 (Yes) or 2 (No) => 1
1) Yes
2) No
Do you want to enable the constellation?
Press 1 (Yes) or 2 (No) =>
Are you sure that these data are correct?
Node Type => general
```

```
Node Name => REG_Comillas_Telsius_20_128_01
Press 1 (Yes) or 2 (No) =>
##Start process of instalation
Starting nginx nginx
...done.
nginx: [emerg] bind() to 0.0.0.0:80 failed (98: Address already in use)
.
.
nginx: [emerg] bind() to [::]:443 failed (98: Address already in use)
nginx: [emerg] still could not bind()
[*] Starting Constellation node
[*] constellation node at 9000 is still starting. Awaiting 5 seconds.
[*] constellation node at 9000 is still starting. Awaiting 5 seconds.
[*] constellation node at 9000 is now up.
[*] resuming start procedure
Relinking permissioning file
[*] Starting quorum node
[*] Monitor enabled. Starting monitor...
INFO [06-27|08:32:27.983] Maximum peer count                       ETH=25 LES=0
total=25
INFO [06-27|08:32:27.984] Starting peer-to-peer node
instance=Geth/REG_Comillas_Telsius_20_128_01/v1.8.12-stable/linux-amd64/go1.9.5
INFO [06-27|08:32:27.984] Allocated cache and file handles
database=/root/alastria/data/geth/chaindata cache=768 handles=1024
INFO [06-27|08:32:29.308] Initialised chain configuration
config="{ChainID: 83584648538 Homestead: 2 DAO: <nil> DAOSupport: false EIP150: 3
```

Verificación de la correcta instalación del nodo regular:

```
root@88bb714e3a16:~/alastria-node/docker/general# bg
[1]+ ./init.sh &
root@88bb714e3a16:~/alastria-node/docker/general# ps
  PID TTY          TIME CMD
  1 pts/0    00:00:00 bash
5854 pts/0    00:00:00 init.sh
5869 pts/0    00:00:00 docker
22712 pts/0    00:00:00 ps
root@88bb714e3a16:~/alastria-node/docker/general# docker ps
CONTAINER ID        IMAGE                             COMMAND
CREATED             STATUS              PORTS
NAMES
f5bac556df65        alastria/alastria-node-general   "entrypoint.sh /bi..."   10
minutes ago      Up 10 minutes       0.0.0.0:80->80/tcp, 0.0.0.0:443->443/tcp,
0.0.0.0:8443->8443/tcp, 0.0.0.0:9000->9000/tcp, 0.0.0.0:21000->21000/tcp,
0.0.0.0:21000->21000/udp, 22000/tcp   REG_Comillas_Telsius_20_128_01
root@88bb714e3a16:~/alastria-node/docker/general# WARN [07-03|14:55:32.975]
Failed to decode stats server message    err="json: cannot unmarshal string into
Go value of type map[string][]interface {}"
WARN [07-03|14:56:11.771] Full stats report failed                 err="write tcp
172.18.0.2:34852->54.229.130.27:80: use of closed network connection"
```

```
docker container ls
CONTAINER ID        IMAGE                                COMMAND
CREATED             STATUS              PORTS
NAMES
f5bac556df65        alastria/alastria-node-general   "entrypoint.sh /bi..."   11
minutes ago      Up 11 minutes       0.0.0.0:80->80/tcp, 0.0.0.0:443->443/tcp,
0.0.0.0:8443->8443/tcp, 0.0.0.0:9000->9000/tcp, 0.0.0.0:21000->21000/tcp,
0.0.0.0:21000->21000/udp, 22000/tcp   REG_Comillas_Telsius_20_128_01
root@88bb714e3a16:~/alastria-node/docker/general# WARN [07-03|14:56:32.975]
Failed to decode stats server message    err="json: cannot unmarshal string into
Go value of type map[string][]interface {}"
root@88bb714e3a16:~/alastria-node/docker/general# f5bacWARN [07-03|14:56:42.000]
Full stats report failed                 err="write tcp 172.18.0.2:37312-
>54.229.130.27:80: use of closed network connection"
root@88bb714e3a16:~/alastria-node/docker/general# docker exec -it WARN [07-
03|14:57:02.976] Failed to decode stats server message    err="json: cannot
unmarshal string into Go value of type map[string][]interface {}"
f5bac556df65 /bin/bash
root@f5bac556df65:~/alastria-node# WARN [07-03|14:57:12.228] Full stats report
failed                 err="write tcp 172.18.0.2:39780->54.229.130.27:80: use of
closed network connection"
```

El nodo fue puesto en marcha satisfactoriamente, pero para la sincronización con la red se siguieron una serie de pasos.

1. Obtención de Información necesaria para la sincronización, siendo esta la información de nuestro nodo regular desplegado. Contiene las crac

```
Universidad Pontificia de Comillas
David Contrera Barcena
Self Hosted (20,128,1T)
--cat ~/alastria/data/constellation/keystore/node.pub
lA9xy5Ubt/L9n/qs99XsWQm75EgdhW+fZXulTlvLZ0s=
-- git diff /root/alastria-node/data/constellation-nodes.json
##Dirección de nuestro nod regular desplegado
enode://b518a8fefa23383d1b9ab1228bb28c0250bca2d0d218179a228a58ea5a97cd29ccc1e2a55
a002b8d2ec095c4e96595a1109cfc51ec7db5260562435553bcb1ef@130.206.64.6:21000?discpo
rt=0
```

2. Enviar un pul request al repositorio GitHub alastria-node a su rama “testnet2”. Pasos seguidos para la consecución de este paso:

   - Iniciamos sesión en https://github.com

   - Navegamos a https://github.com/alastria/alastria-node/tree/testnet2

- Pulsamos sobre el botón Fork y nos creará una copia del repositorio actual alastria node en nuestro usuario, ej.: https://github.com/go2chain/alastria-node

- En este repositorio, navegamos a la rama testnet2 y después al fichero data/regular-nodes.json y con el icono lápiz, agregamos la modificación al final de la lista y fijándonos en la estructura de los demás nodos. Una vez realizados los cambios en el fichero, deberemos utilizar el botón Commit changes.
- Navegando al fichero data/constellation-nodes.json agregamos la modificación al final de la lista dentro de los corchetes, teniendo cuidado de NO añadir la "," al final. Una vez realizados los cambios en el fichero, deberemos utilizar el botón Commit changes.
- Navegando al fichero DIRECTORY_REGULAR. añadimos en la última línea del fichero la información:

```
Universidad Pontificia de Comillas
David Contrera Barcena
Self Hosted (20,128,1T)
lA9xy5Ubt/L9n/qs99XsWQm75EgdhW+fZXulTlvLZ0s=
enode://b518a8fefa23383d1b9ab1228bb28c0250bca2d0d218179a228a58ea5a97cd29ccc1e2a55
a002b8d2ec095c4e96595a1109cfc51ec7db5260562435553bcb1ef@130.206.64.6:21000?discpo
rt=0
```

- Utilizamos el botón Commit changes como en el resto de los casos.

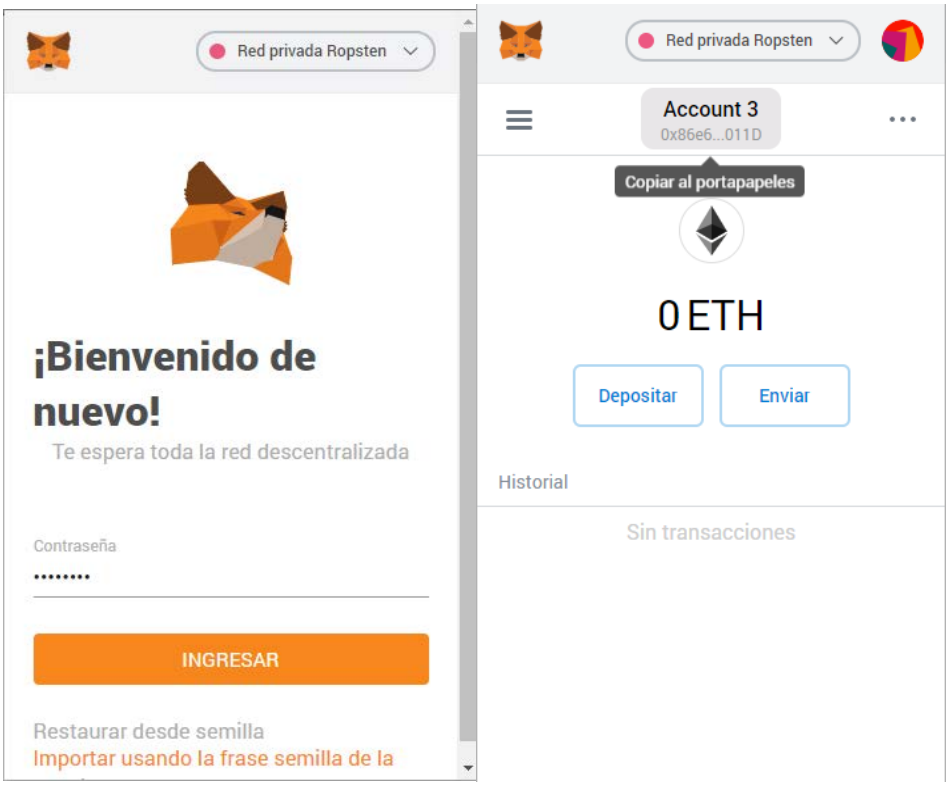
- Navegamos a la pestaña Pull request y hacemos clic sobre el botón New Pull Request. Utilizando el botón Create pull request se crea en el repositorio alastria-node el correspondiente pull request que el equipo de plataforma deberá aceptar e integrar.

Todos estos pasos se realizaron con éxito.

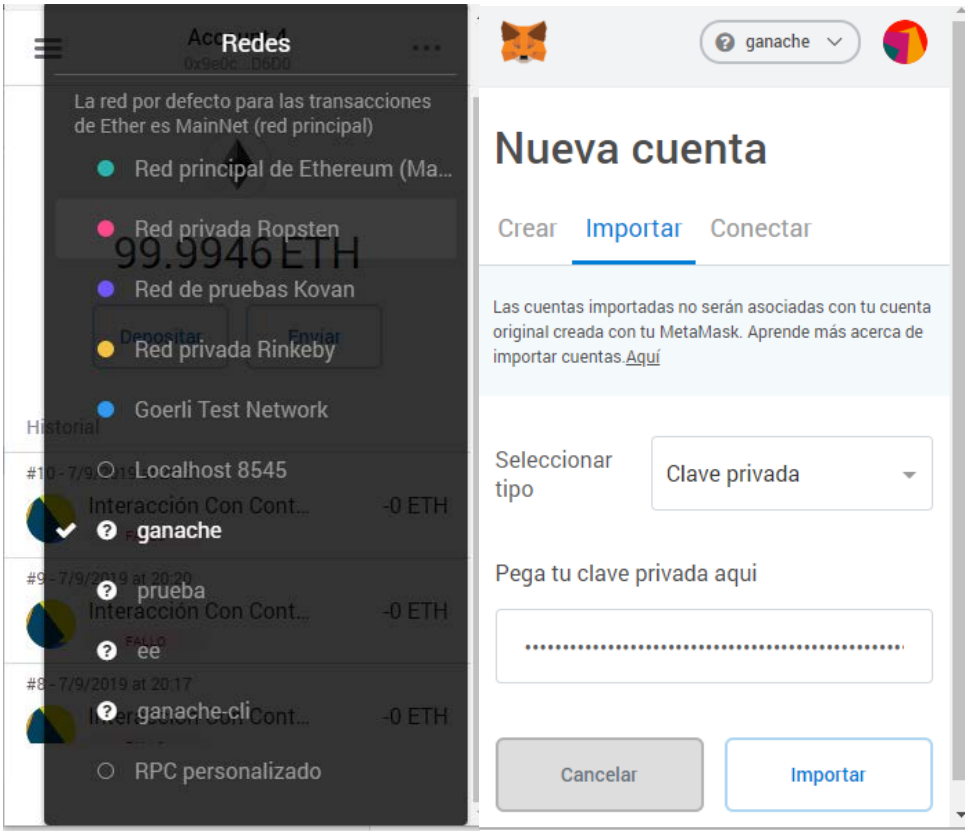
ANEXO B – CONFIGURACIÓN DE CUENTAS METAMASK Y GANACHE

En este anexo se explica cómo conectar las cuentas provistas por ganache, sobre la herramienta Metamask para así poder realizar transacciones con estas cuentas.

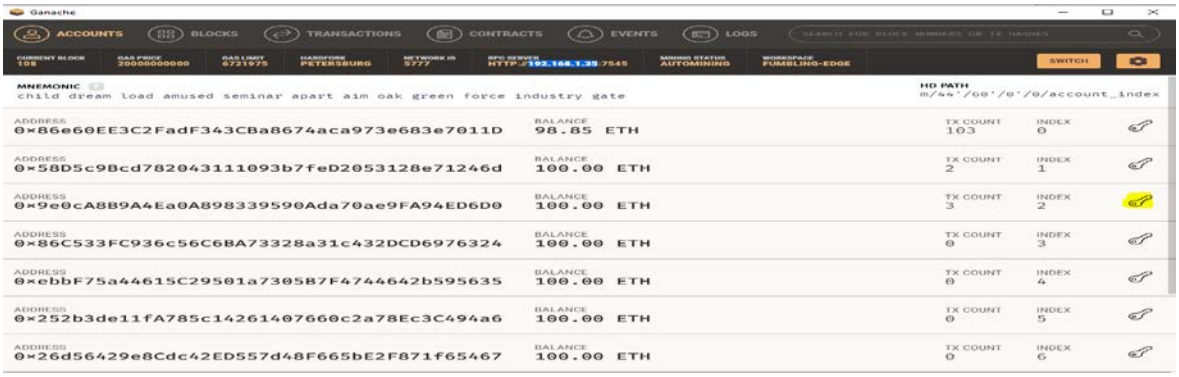
Accedemos a Metamask, introducimos la cuenta creada para la utilización de la herramienta

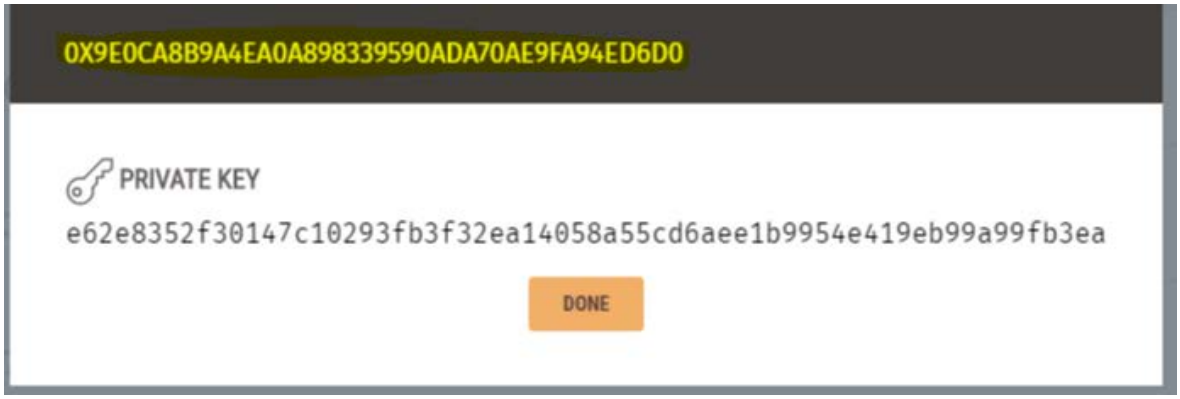


Ahora nos vamos al menú de arriba para la elección de una red, y elegimos la red ganache, y le damos a importar cuenta, introduciendo una clave privada



Obtención de la clave privada





Importamos la cuenta y veremos el saldo de la cuenta y la dirección justo en la parte de arriba