



COMILLAS

UNIVERSIDAD PONTIFICIA

ICAI

GRADO EN INGENIERÍA EN TECNOLOGÍAS DE TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

BLOCKCHAIN, SMART CONTRACTS Y DAPP:
INMUEBLES TURÍSTICOS, PRÉSTAMOS Y
SEGUROS

Autor: Teresa Rodríguez Benito

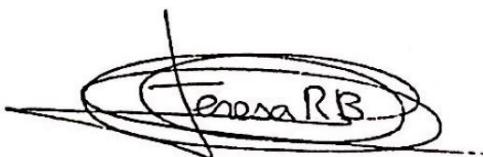
Director: Atilano Fernández-Pacheco Sánchez-Migallón

Director: José Luis Gahete Díaz

Madrid

Declaro, bajo mi responsabilidad, que el Proyecto presentado con el título
“Blockchain, Smart Contracts y Dapp: Inmuebles Turísticos, Préstamos y Seguros”
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.

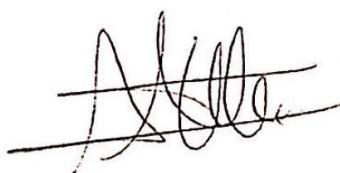


Fdo.: Teresa Rodríguez Benito

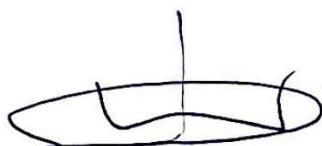
Fecha: 11/Junio / 2019

Autorizada la entrega del proyecto

LOS DIRECTORES DEL PROYECTO



Fdo.: Atilano Fernández-Pacheco Sánchez-Migallón Fecha: 11/ Junio / 2019



Fdo.: José Luis Gahete Díaz

Fecha: 11/ Junio / 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. Teresa Rodríguez Benito DECLARA ser el titular de los derechos de propiedad intelectual de la obra: “Blockchain, Smart Contracts y Dapp: inmuebles turísticos, préstamos y seguros”, 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 exclusiva 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 11 de Junio de 2019

ACEPTA



Fdo: Teresa Rodríguez Benito

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

--



COMILLAS

UNIVERSIDAD PONTIFICIA

ICAI

GRADO EN INGENIERÍA EN TECNOLOGÍAS DE TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

BLOCKCHAIN, SMART CONTRACTS Y DAPP:
INMUEBLES TURÍSTICOS, PRÉSTAMOS Y
SEGUROS

Autor: Teresa Rodríguez Benito

Director: Atilano Fernández-Pacheco Sánchez-Migallón

Director: José Luis Gahete Díaz

Madrid

"Para ser irremplazable, uno debe siempre ser diferente"

-Coco Chanel

BLOCKCHAIN, SMART CONTRACTS Y DAPP: INMUEBLES TURÍSTICOS, PRÉSTAMOS Y SEGUROS

Autor: Rodríguez Benito, Teresa.

Director: Fernández-Pacheco Sánchez-Migallón, Atilano.

Director: Gahete Díaz, José Luis.

Entidad Colaboradora: ICAI – Universidad Pontificia Comillas

RESUMEN DEL PROYECTO

Con el fin de explotar la revolucionaria tecnología *Blockchain*, se plantea la creación de una plataforma descentralizada para aprovechar un nicho de mercado que está creciendo: los alojamientos turísticos. En el alcance se integran compradores y vendedores, así como, la interacción de este mercado con prestamistas y aseguradoras, posibilitando un sistema que reduce los costes de transacción y que es seguro, confiable, inmutable y transparente.

Palabras clave: Blockchain, Smart Contracts, Dapp, Ethereum, Rinkeby, Inmuebles, Seguros, Préstamos

1. Introducción

En la actualidad, el turismo es uno de los sectores primordiales en España, siendo el país con más influencia de este sector en su Producto Interior Bruto. [1] Además, se está produciendo un cambio de conducta en los turistas respecto a su alojamiento, albergándose cada vez una mayor proporción en apartamentos turísticos, como los ofrecidos por *Airbnb*. [2] Con el fin de mantener el gran aporte económico que genera el turismo, y adaptando la nueva actitud de los consumidores, se propone la introducción de la disruptiva tecnología *Blockchain*.

Blockchain es la tecnología que se encuentra detrás de las criptomonedas, como la popular *Bitcoin*. Permite eliminar el actor mediador de la transacción, puesto que es descentralizado; cada nodo de la red intercambia información con el resto y almacena una copia de los datos permutados. Debido a los algoritmos de consenso en uso y la criptografía utilizada, junto con la descentralización de la información, permiten mantener un sistema inmutable. Además, posibilita una reducción del coste de la transacción y una operativa disponible sin interrupción, a diferencia del sistema tradicional bancario. Asimismo, la integración con las cláusulas contractuales de *Blockchain*, *Smart Contracts*, se permite optimizar los procesos, puesto que se ejecutan al cumplir los requisitos diseñados de forma automática.

Por lo tanto, en este proyecto se integran los *Smart Contracts* y la tecnología *Blockchain* en una *Dapp*, para permitir una experiencia del consumidor agradable al disponer de una interfaz amigable, junto a la posibilidad de conectarse a través de una *Wallet HD*, con la que, registrándose con una única contraseña, permite almacenar todas las cuentas de cualquier red de forma segura y confiable.

2. Definición del proyecto

El objetivo principal del proyecto es utilizar las ventajas comentadas de la tecnología de *Blockchain* para gestionar el nicho de mercado de los inmuebles turísticos en España entre compradores-vendedores y arrendatarios-arrendadores, así como la posibilidad de obtener un préstamo y contratar una póliza de seguro para el inmueble deseado. Todo

ello se basa en una aplicación descentralizada: *Dapp*, que registre de forma automática las transacciones realizadas si se cumplen las condiciones necesarias.

Para poder llevar a cabo el objetivo fundamental, en primer lugar se profundiza y documenta la tecnología *Blockchain* con su aplicación de *Smart Contracts*, con vistas a ser desplegado en la red de pruebas *Rinkeby*. Más tarde, se estudia y compara las plataformas para la aplicación de *Smart Contracts* en una *Dapp*, con el objeto de diseñar, desarrollar y testear el caso de uso en un nodo local. Por último, se realiza un despliegue en la red *Rinkeby*, para así analizar las transacciones en una red pública.

3. Descripción del sistema

Con el objeto de solucionar la gestión de inmuebles, préstamos y seguros, se desarrolla una plataforma *Blockchain* de forma distribuida.

El comprador publica en *Blockchain* un anuncio de un inmueble que desea vender a través de la *Dapp*. Por otra parte, la aseguradora puede publicar una oferta de póliza de seguro referente al inmueble visto en la *Dapp*. Con relación al comprador, si le interesa un inmueble, tendrá la opción de comprarlo sin necesidad de préstamo ni seguro, aunque si lo desea, podrá aceptar una oferta de las aseguradoras, así como publicar la necesidad de un préstamo con las condiciones que le interesan. En cuanto al prestamista, si le resulta ventajosa la oferta del prestatario, la puede aceptar para intercambiar *Ethers*. La siguiente figura muestran los diferentes actores de la aplicación junto al sistema descentralizado.

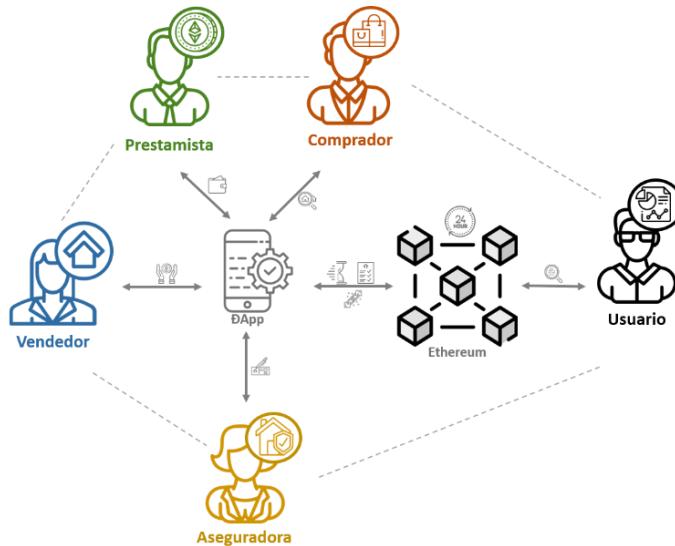


Ilustración 1: Esquema del sistema

Para resolverlo, se utiliza el framework de *Truffle* para fijar la estructura del proyecto, compilar y migrar los *Smart Contracts* desarrollados a través de *Atom*. Tras implementar los *Smart Contracts*, se necesita un nodo local de *Ethereum*, *Ganache*, para ejecutarlos y testearlos. Después, se realiza el despliegue y migración a través de un nodo remoto, *Infura*, en la *testnet Rinkeby*.

Por otro lado, en *Atom* se programa la parte del front: *HTML*, *CSS* y *JavaScript*. A través de *JavaScript*, junto a la librería *Web3.js*, el plugin de *Metamask* y el servidor de desarrollo *lite-server* de *Node.js*, se posibilita la unión de las partes tal como se muestra.

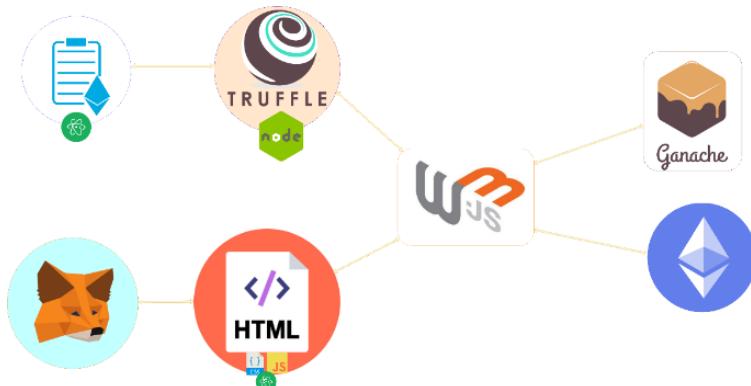


Ilustración 2: Arquitectura Dapp

4. Resultados

Se ha posibilitado utilizar *Blockchain* de forma transparente para el usuario a través de la *Dapp*. En ella, el comprador puede anunciar sus inmuebles y ser comprados por otros usuarios. Además, pueden solicitarse préstamos y contratar pólizas de seguro. También, se puede visualizar un registro histórico de las transacciones realizadas en la plataforma. Se muestra un ejemplo de su utilización.



Ilustración 3: Dapp

Al realizarse el despliegue en *Rinkeby*, se puede observar que en los resultados se ha reducido los costes de las transacciones, se han evitado intermediarios, se ha almacenado en varios nodos la información, y se ha elevado la seguridad del sistema respecto a una aplicación tradicional.

5. Conclusiones

En esta iniciativa se observa cómo la gestión de inmuebles, préstamos y seguros puede reducir sus costes de tramitación basándose en la tecnología. Este proyecto supone una plataforma novedosa y única que posibilita interactuar a los compradores, vendedores, prestamistas y aseguradoras a través de *Blockchain*. A parte de la implementación y el despliegue, el proyecto es un encuentro de información sobre las características de esta tecnología, sus mejores herramientas y sus aportaciones. Además, el análisis y diseño del caso de uso en detalle ha fomentado su aprendizaje y el uso de la disruptiva tecnología *Blockchain*, que reafirma la confianza y fiabilidad en las partes.

El presente proyecto sirve como punto de partida para implementar algunos de los trabajos futuros posibles indicados sobre la plataforma y su gestión. Con el fin de facilitar el progreso del proyecto, se ha entregados varios anexos para que cualquier persona sin experiencia y conocimientos de *Blockchain* pueda disponer de la instalación y funcionamiento del proyecto paso a paso para poder mejorarlo. Además de las contribuciones indicadas, el trabajo colabora con la sociedad aportando la información actualizada y contrastada para que las empresas españolas sigan invirtiendo en la tecnología como motor de mejora de los procesos.

6. Referencias

- [1] [En línea]. Available: <https://cepmenews.es/espana-pais-mundo-donde-turismo-aporta-mas-al-pib-segun-la-ocde/> [Último acceso: 2 Junio 2019].
- [2] [En línea]. Available: <https://es.statista.com/estadisticas/802182/ingresos-anuales-de-airbnb-en-espana/>. [Último acceso: 31 Mayo 2019].

BLOCKCHAIN, SMART CONTRACTS AND DAPP: TOURISM PROPERTIES, LOANS AND INSURANCES

Author: Rodríguez Benito, Teresa.

Supervisor: Fernández-Pacheco Sánchez-Migallón, Atilano.

Supervisor: Gahete Díaz, José Luis.

Collaborating Entity: ICAI – Universidad Pontificia Comillas

ABSTRACT

In order to exploit the revolutionary *Blockchain* technology, the creation of a decentralized platform is presented to take advantage of a market niche that is growing: tourism apartments. The scope includes buyers and sellers, as well as the interaction of lenders and insurers at this market, making possible a system that reduces transaction costs and is safe, reliable, immutable and transparent.

Keywords: Blockchain, Smart Contracts, Dapp, Ethereum, Rinkeby, Real Estate, Insurances, Loans

1. Introduction

Nowadays, tourism is one of the most important sectors in Spain, being the country with the biggest influence of this sector in its Gross Domestic Product. [1] In addition, there is a change in the behavior of tourists regarding their accommodation, with a growing proportion of tourist apartments, such as those offered by Airbnb. [2] In order to maintain the great economic contribution generated by tourism, as well as adapting the new attitude of consumers, it is proposed the introduction of the disruptive *Blockchain* technology.

Blockchain is the technology behind cryptocurrencies, like the popular *Bitcoin*. It allows eliminating the mediating actor of the transaction, since it is decentralized; each node of the network exchanges information with the rest and stores a copy of the permuted data. Due to the consensus algorithms and the cryptography used, along with the decentralization of information, it allows maintaining an immutable system. In addition, it enables a reduction in the cost of the transaction and an operation available without interruption, unlike the traditional banking system. Furthermore, the integration with the contractual clauses of *Blockchain*, *Smart Contracts*, it allows optimizing the processes, since they are executed automatically when the configured requirements are met.

Therefore, in this project the *Smart Contracts* and the *Blockchain* technology are integrated in a *Dapp*, to allow a pleasant consumer experience by having a friendly interface, together with the possibility of connecting through an *HD Wallet*, with which registering with only one password, it allows to store all the accounts of any network in a safe and reliable way.

2. Project definition

The main objective of the project is to use the mentioned advantages of the *Blockchain* technology to manage the market niche of tourist properties in Spain between buyers-sellers and tenants-lessors, as well as the possibility of obtaining a loan and contracting an insurance policy for the desired property. All this is based on a decentralized

application: *Dapp*, which automatically records transactions made if the registered conditions are met.

In order to carry out the fundamental objective, *Blockchain* technology is first deepened and documented with its *Smart Contracts* application, in order to be deployed in the *Rinkeby* test network. Later, the platforms for the application of *Smart Contracts* in a *Dapp* are studied and compared, in order to design, develop and test the use case in a local node. Finally, a deployment is made in the *Rinkeby* network, in order to analyze transactions in a public network.

3. System description

In order to solve the management of real estate, loans and insurance, a *Blockchain* platform is developed in a distributed way.

The buyer publishes in *Blockchain* an advertisement for a property that he wishes to sell through the *Dapp*. On the other hand, the insurer can publish an offer of insurance policy regarding the property seen in the *Dapp*. In relation to the buyer, if he is interested in a property, he will have the option of buying it without the need for a loan or insurance, although if he wishes, he may accept an offer from the insurers, as well as publish the need for a loan with the conditions in which he is interested. As for the lender, if the borrower's offer is advantageous, he can accept it in order to exchange *Ethers*. The following figure shows the different actors of the application as well as the decentralized system.

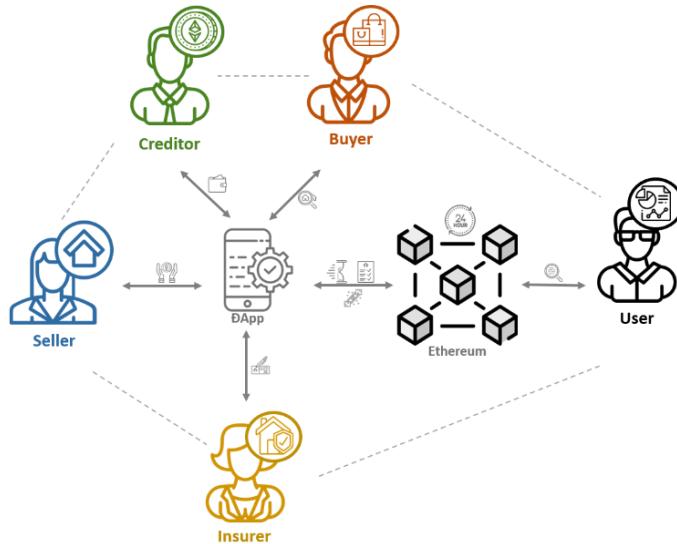


Illustration 1: System scheme

To solve the problem, *Truffle* framework is used to set the structure of the project, compile and migrate the *Smart Contracts* developed through *Atom*. After implementing *Smart Contracts*, a local *Ethereum* node, *Ganache*, is needed to run and test them. Then, the deployment and migration is performed through a remote node, *Infura*, in the *Rinkeby* testnet.

On the other hand, *Atom* is used to programme the front part: *HTML*, *CSS* and *JavaScript*. Through *JavaScript*, together with the *Web3.js* library, the *Metamask* plugin and the *Node.js* *lite-server* development server, it is possible to join the different parts as shown below.

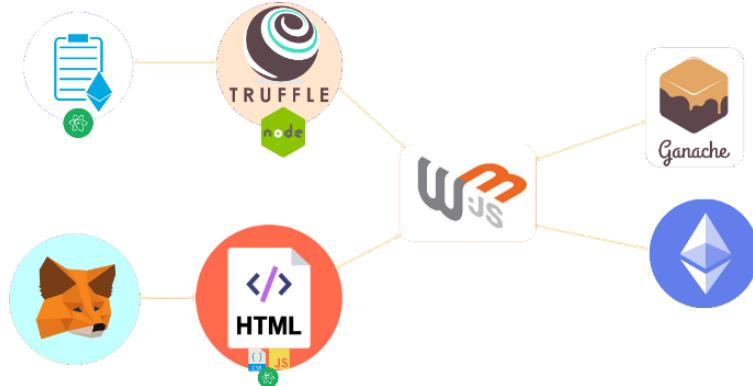


Illustration 2: Dapp Architecture

4. Results

It has been possible to use *Blockchain* transparently for the user through the *Dapp*. In the app, the buyer can advertise their properties and be purchased by other users. In addition, loans and insurance policies can be requested. Also, it is possible to view the historical record of the transactions made in the platform. An example of the *Dapp* developed is shown.



Illustration 3: Dapp

When deployed in *Rinkeby*, it can be observed that as a result, the transaction costs have been reduced, intermediaries have been avoided, the information has been stored in several nodes, and the security of the system has been raised in comparison with a traditional app.

5. Conclusions

This initiative shows how the management of real estate, loans and insurance can reduce their processing costs based on this technology. This project supposes a new and unique platform that makes it possible to interact with buyers, sellers, lenders and insurers through *Blockchain*. Apart from the implementation and deployment, the project is a repository of information about the characteristics of this technology, its best tools and its contributions. In addition, the analysis and design of the use case in detail has fostered the learning and the use of the disruptive *Blockchain* technology, which reaffirms the trust and reliability of the parts.

The present project serves as a starting point to implement some of the possible future works indicated about the platform and its management. In order to facilitate the progress of the project, several annexes have been delivered so that anyone without experience and knowledge of *Blockchain* can make the installation and operation of the project step by step in order to improve it. In addition to the contributions indicated, the work collaborates with society by providing up-to-date and verified information so that Spanish companies continue investing in technology as an engine for improving processes.

6. References

- [1] [En línea]. Available: <https://cepymenews.es/espana-pais-mundo-donde-turismo-aporta-mas-al-pib-segun-la-ocde/> [Último acceso: 2 Junio 2019].
- [2] [En línea]. Available: <https://es.statista.com/estadisticas/802182/ingresos-anuales-de-airbnb-en-espana/>. [Último acceso: 31 Mayo 2019].

Índice de la Memoria

Capítulo 1. Introducción	13
1.1 Motivación del Proyecto	15
1.2 Solución Propuesta.....	17
Capítulo 2. Descripción de las Tecnologías.....	21
2.1 Ethereum	21
2.2 Solidity	22
2.3 Truffle.....	23
2.4 Ganache	24
2.5 Entorno de Desarrollo	26
2.5.1 Remix IDE	26
2.5.2 Visual Studio Solidity	27
2.5.3 Editor Atom	28
2.5.4 Comparativa	29
2.6 Metamask	30
2.7 Web3.js.....	31
2.8 EtherScan	32
2.9 Node.js.....	33
2.10 Rinkeby	34
2.11 Infura	35
Capítulo 3. Estado de la Cuestión	37
3.1 Blockchain Respecto al Caso de Uso en España.....	38
3.2 Blockchain Respecto a la Información Manejada en el Caso de Uso	42
3.3 Blockchain y la Universidad Pontificia Comillas ICADE-ICAI.....	45
Capítulo 4. Definición del Trabajo	47
4.1 Justificación.....	47
4.2 Objetivos	49
4.3 Metodología.....	50
4.4 Planificación.....	52
4.5 Estimación Económica	54

Capítulo 5. Blockchain.....	57
5.1 Descripción General	57
5.2 Almacenamiento en Blockchain.....	59
5.3 Algoritmos de Consenso de las Redes Blockchain	65
5.4 Versiones de Blockchain	67
5.4.1 Criptomonedas.....	67
5.4.2 Smart Contracts.....	69
5.4.3 Dapp	71
5.5 Ico.....	73
5.6 Ventajas de Blockchain	74
5.7 Limitaciones de la Tecnología Blockchain	76
Capítulo 6. Análisis del Sistema.....	79
6.1 Casos de Uso	79
6.1.1 Caso de Uso Común	79
6.1.2 Caso de Uso Vendedor	80
6.1.3 Caso de Uso Comprador	80
6.1.4 Caso de Uso Prestamista.....	82
6.1.5 Caso de Uso Aseguradora.....	82
6.2 Diagramas de Secuencia.....	83
6.2.1 Diagrama de Secuencia Compra.....	83
6.2.2 Diagrama de Secuencia Alquiler.....	87
6.3 Diagrama de Actividad.....	89
6.3.1 Diagrama de Actividad Compra.....	89
6.3.2 Diagrama de Actividad Alquiler.....	92
Capítulo 7. Diseño	95
7.1 Funcionalidades Comunes.....	95
7.2 Funcionalidades de Vendedor	96
7.3 Funcionalidad de Comprador	96
7.4 Funcionalidad de Prestamista	98
7.5 Funcionalidad de Aseguradora	98
7.6 Funcionalidad del Sistema.....	99
7.7 Funcionalidad usuario con Privilegios	99

7.8 Navegabilidad en la Dapp	100
Capítulo 8. Arquitectura.....	103
Capítulo 9. Desarrollo de la Dapp.....	109
9.1 Smart Contracts	109
9.2 Interfaz de Usuario	113
9.2.1 Página de Bienvenida.....	113
9.2.2 Cuenta	114
9.2.3 Publicar	115
9.2.4 Anuncios	118
9.2.5 Préstamos	119
9.2.6 Seguros	120
9.2.7 Registro Histórico	121
Capítulo 10. Despliegue en Rinkeby y Análisis de las Transacciones.....	125
10.1 Despliegue de los Smart Contracts.....	128
10.2 Flujo de Transacciones	133
10.3 Transacciones Internas de los Smart Contracts	153
Capítulo 11. Alcance Comercial	157
Capítulo 12. Conclusiones y Trabajos Futuros.....	161
12.1 Conclusiones	161
12.2 Trabajos Futuros.....	165
Capítulo 13. Bibliografía.....	171
A. Anexo A: Guía de Instalación para el Usuario	181
A.1 Atom	181
A.2 Ganache	183
A.3 Node.js.....	186
A.4 Truffle.....	191
A.5 Estructura y Ficheros en el Proyecto	192
A.6 Git.....	195
A.7 Metamask	202

A.8 Infura Rinkeby.....	205
B. Anexo B: Guía de Funcionamiento para el Usuario	215
B.1 Ejecución en Local a Través de Ganache.....	215
B.2 Ejecución en Rinkeby.....	216

Índice de Ilustraciones

Ilustración 1: Valoración del turismo en España 2017 por WEF	13
Ilustración 2: Gasto total de los turistas extranjeros que visitaron España en 2018, según el motivo principal del viaje (millones de euros) [3]	14
Ilustración 3: Evolución anual de los ingresos declarados por Airbnb en España entre 2012 y 2017 (en millones de euros) [5].....	16
Ilustración 4: Evolución anual del número de turistas internacionales que alquilaron una vivienda en España de 2000 a 2018 (en millones) [6].....	16
Ilustración 5: Logo Ethereum	21
Ilustración 6: Logo Solidity [12]	22
Ilustración 7: Logo Truffle [13]	24
Ilustración 8: Logo Ganache [14].....	25
Ilustración 9: Aplicación Ganache	25
Ilustración 10: Remix IDE [17].....	27
Ilustración 11: Visual Studio Solidity [19].....	28
Ilustración 12: Atom Editor	29
Ilustración 13: Logo Metamask [21]	31
Ilustración 14: Logo Web3.js	32
Ilustración 15: Logo Etherscan [23]	32
Ilustración 16: Transacción Etherscan [24]	33
Ilustración 17: Logo Node.js [26]	33
Ilustración 18: Logo Lite-server [27]	33
Ilustración 19: Rinkeby estado de la red [29].....	35
Ilustración 20: Logo Infura [30]	36
Ilustración 21: Información en descubierto [55]	42
Ilustración 22: Identidad digital Blockchain [57].....	44
Ilustración 23: Logo Alastria [61]	45
Ilustración 24: Kanban utilizado a través de Trello.....	51
Ilustración 25: Diagrama de Gantt del proyecto.....	52

Ilustración 26: Nodo universitario Alastria ICAI [67]	58
Ilustración 27: Hash, la huella dactilar digital [68]	61
Ilustración 28: Se cambia información del bloque, hash no comienza por 0000 [68]	62
Ilustración 29: Tras minar el bloque, hash comienza por 0000 [68]	62
Ilustración 30: Cada bloque tiene hash anterior [68].....	63
Ilustración 31: Minar el bloque y los posteriores	64
Ilustración 32: Diferentes tipos de algoritmos de consenso [69].....	66
Ilustración 33: Valor Bitcoin durante los últimos tres años [70].....	68
Ilustración 34: Comparación App y Dapp	71
Ilustración 35: Conjunto de Dapps [76]	72
Ilustración 36: Caso de uso común.....	79
Ilustración 37: Caso de uso vendedor	80
Ilustración 38: Caso de uso comprador	81
Ilustración 39: Caso de uso prestamista	82
Ilustración 40: Caso de uso aseguradora	83
Ilustración 41: Diagrama de secuencia compra (I)	85
Ilustración 42: Diagrama de secuencia compra (II).....	86
Ilustración 43: Diagrama de secuencia alquiler.....	88
Ilustración 44: Diagrama de actividad compra.....	91
Ilustración 45: Diagrama de actividad alquiler.....	93
Ilustración 46: Navegabilidad en la Dapp	100
Ilustración 47: Arquitectura Dapp	103
Ilustración 48: Arquitectura Dapp con tecnologías	103
Ilustración 49: Estructura del proyecto- Truffle	104
Ilustración 50: Metamask interacción Dapp	107
Ilustración 51: Página de bienvenida.....	114
Ilustración 52: Cuenta y saldo sin Metamask	114
Ilustración 53: Cuenta y saldo con Metamask.....	115
Ilustración 54: Pestaña Publicar	115
Ilustración 55: Publicación anuncio	116

Ilustración 56: Solicitud préstamo	117
Ilustración 57: Publicación póliza de seguro	117
Ilustración 58: Pestaña Anuncios	118
Ilustración 59: Comprar inmueble	118
Ilustración 60: Pestaña Préstamos	120
Ilustración 61: Prestar dinero.....	120
Ilustración 62: Pestaña Seguros.....	121
Ilustración 63: Póliza de seguro	121
Ilustración 64: Pestaña Registro Histórico	122
Ilustración 65: Registro histórico anuncios	123
Ilustración 66: Registro histórico préstamos	123
Ilustración 67: Registro histórico seguros	124
Ilustración 68. Cuentas existentes en Wallet	127
Ilustración 69. Transacción Rinkeby: generación Propiedad.sol	128
Ilustración 70. Bloque generado en la red Rinkeby.....	129
Ilustración 71. Transacción Rinkeby: generación Prestamista.sol	130
Ilustración 72. Transacción Rinkeby: generación Aseguradora.sol	131
Ilustración 73. Despliegue desde consola en la red Rinkeby	133
Ilustración 74: Puesta en venta del primer inmueble.....	134
Ilustración 75: Transacción Rinkeby: Venta primer inmueble.....	135
Ilustración 76: Transacción Rinkeby: Log de eventos venta primer inmueble	136
Ilustración 77: Cuenta de vendedor antes de realizar anuncio	136
Ilustración 78: Cuenta de vendedor después de realizar anuncio	137
Ilustración 79: Dirección de vendedor en Etherscan	137
Ilustración 80: Transacción Rinkeby: Alquiler segundo inmueble	138
Ilustración 81: Transacción Rinkeby: Seguro primer inmueble	139
Ilustración 82: Transacción Rinkeby: Log de eventos seguro primer inmueble	140
Ilustración 83: Transacción Rinkeby: Préstamo primer inmueble	141
Ilustración 84: Transacción Rinkeby: Log de eventos préstamo primer inmueble	141
Ilustración 85. Notificación de MetaMask previa a transacción	142

Ilustración 86: Transacción Rinkeby: Aceptación préstamo primer inmueble	143
Ilustración 87: Transacción Rinkeby: Interna aceptación préstamo primer inmueble	143
Ilustración 88. Contratación de póliza de seguro desde el Front.....	144
Ilustración 89: Transacción Rinkeby: Contratación seguro primer inmueble	145
Ilustración 90: Transacción Rinkeby: Interna contratación seguro primer inmueble	145
Ilustración 91. Notificación de MetaMask a la hora de contratar el seguro.....	146
Ilustración 92: Transacción Rinkeby: Compra primer inmueble	147
Ilustración 93: Transacción Rinkeby: Interna compra primer inmueble	147
Ilustración 94: Transacción Rinkeby: Log de eventos compra primer inmueble.....	148
Ilustración 95: Transacción Rinkeby: Alquiler tercer inmueble	149
Ilustración 96: Transacción Rinkeby: Alquilado tercer inmueble.....	150
Ilustración 97: Transacción Rinkeby: Interna alquilado tercer inmueble.....	150
Ilustración 98: Transacción Rinkeby: Alquilado segundo inmueble	151
Ilustración 99: Transacción Rinkeby: Interna alquilado segundo inmueble	152
Ilustración 100: Transacción Rinkeby: interna del vendedor.....	154
Ilustración 101: Transacción Rinkeby: interna del comprador	154
Ilustración 102: Transacción Rinkeby: interna del prestamista.....	155
Ilustración 103: Transacción Rinkeby: interna del asegurador	155
Ilustración 104: Transacción Rinkeby: Cuenta Smart Contract Propiedad.sol	156
Ilustración 105: Smart Contract Escrow [88]	166
Ilustración 106: Stages de la normativa IFRS9 [89]	167
Ilustración 107: IPFS [90]	168
Ilustración 108: Smart Door [91]	169

Índice de Tablas

Tabla 1. Comparativa entre entornos de desarrollo	30
Tabla 2: Análisis dedicado a cada tarea.....	55
Tabla 3: Relación entre tiempo y coste	55
Tabla 4. Flujo de transacciones	125
Tabla 5. Relación entre perfil y cuenta.....	126
Tabla 6. Hash de las transacciones que han generado Smart Contracts	132
Tabla 7. Cuentas sobre las que se han desplegado los Smart Contracts.....	132
Tabla 8: Relación transacciones hash en el flujo Rinkeby	153
Tabla 9: Relación tarea principal, objetivo y grado de cumplimiento.....	165

Índice de Fragmentos de Código

Código 1: Fragmento versión Solidity.....	109
Código 2: Fragmento import	109
Código 3: Fragmento Propiedad.sol estructura	110
Código 4: Fragmento Propiedad.sol mapping	110
Código 5: Fragmento Asegurador.sol evento	110
Código 6: Fragmento desactivación Smart Contract	111
Código 7: Fragmento Propiedad.sol vender Anuncio	111
Código 8: Fragmento Prestamos.sol obtener número de prestámos.....	112
Código 9: Fragmento Prestamos.sol aceptación préstamo	112
Código 10: fragmento Asegurado.sol seguros publicados sin asegurado	112
Código 11: fragmento Propiedad.sol Obtener todas los anuncios.....	113
Código 12: Fragmento Javascript Web3.js [82]	119
Código 13. Detalle despliegue en Rinkeby de truffle.js	126

Capítulo 1. INTRODUCCIÓN

En España, el turismo tuvo un primer impulso de la mano de los romanos con los centros termales. Más tarde, a finales del siglo XIX, únicamente viajaban los aristocráticos a las aguas termales y zonas costeras como terapia, destacando las playas del norte: *Sardinero* y la *Concha*, mientras que en los siguientes años empezó a destacar la zona del Mediterráneo, y, además, este turismo pasó a ser una fuente de ocio. Debido al período de guerra civil española, hasta los años cincuenta no se empezó a impulsar fuertemente la industria turística: transporte, restauración, hoteles... [1] hasta llegar a nuestros días.

Actualmente, el turismo es uno de los principales motores de la economía española. Según el estudio sobre competitividad turística elaborada por *World Economic Forum*, España recibió, en el año 2017, la friolera cifra de 68.521.255 de turistas internacionales, los cuales, de media, han generado 824,1\$. Esto ha permitido clasificar a España en primera posición, frente a otros 136 países, destacando por su infraestructura turística, como se aprecia en la Ilustración 1. [2]

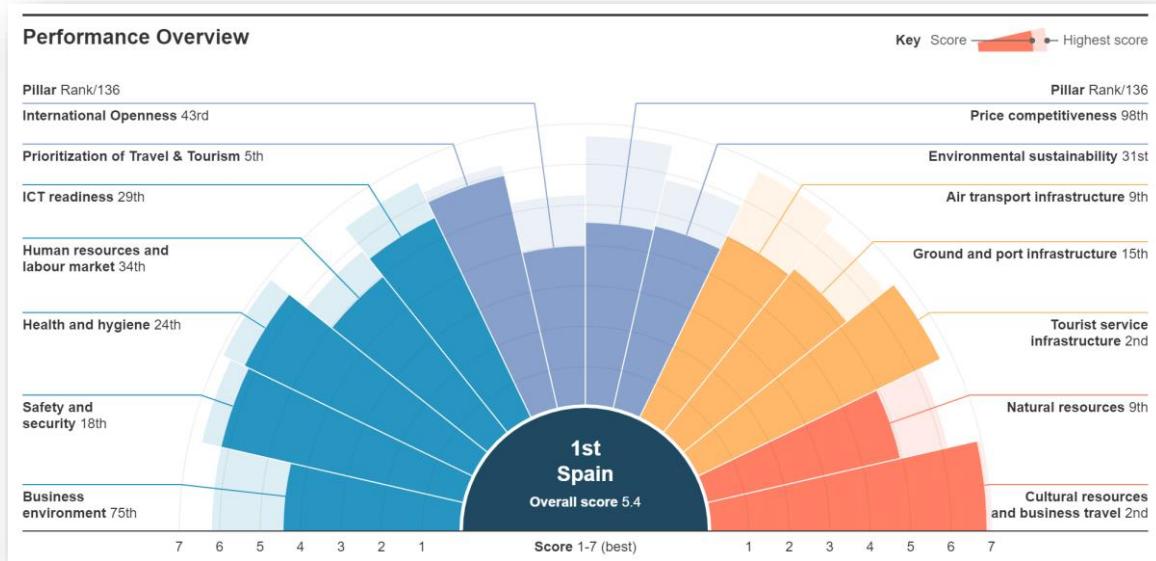


Ilustración 1: Valoración del turismo en España 2017 por WEF

Esta magnífica industria genera gran cantidad de servicios adjuntos. El turismo atrae a personas que están dispuestas a pagar más que un residente habitual puesto que la mayoría de los viajes de extranjero son por ocio, tal como se muestra en la gráfica de Ilustración 2. Para clarificar este hecho, nos situamos en la piel de un turista extranjero al viajar a otro país. Se necesita comprar los billetes de avión, necesita un taxi, Cabify o Uber para desplazarse del aeropuerto a su hotel, necesita reservar una habitación, realizará un gasto de restauración de esos días, con alta probabilidad comprará entradas a museos, concierto, o parques de atracciones, y realizará, seguramente, algunas compras de ocio.

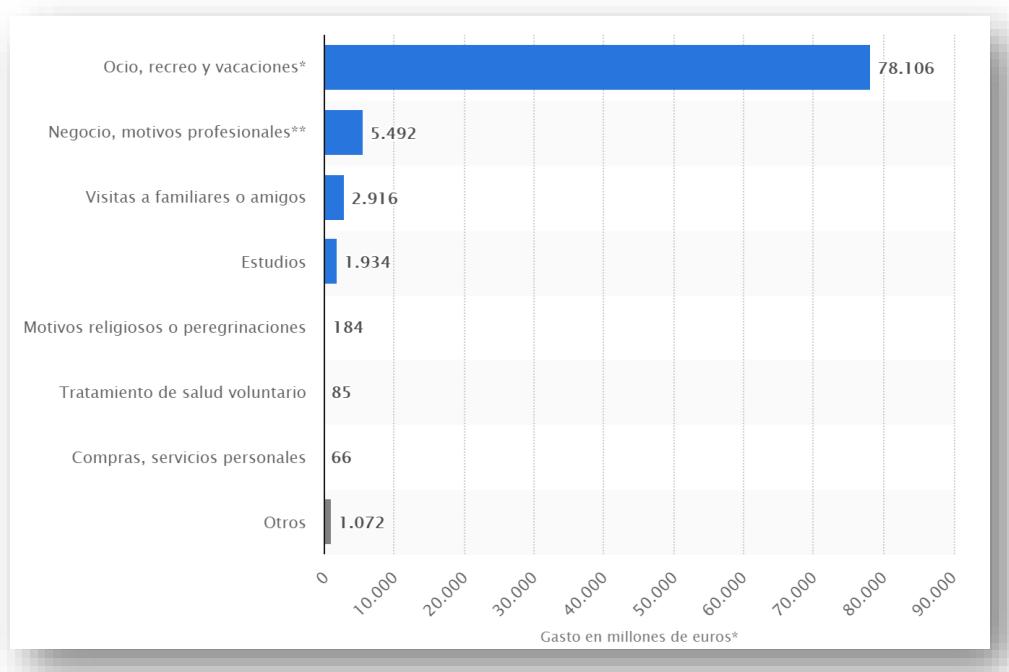


Ilustración 2: Gasto total de los turistas extranjeros que visitaron España en 2018, según el motivo principal del viaje (millones de euros) [3]

En 2018, la afluencia de turistas internacionales ha supuesto un aumento del 0,9% respecto al 2017. Este crecimiento ha supuesto aproximadamente 90.000 millones de euros más a la economía española, es decir, un 3,1% de ingresos turístico respecto al año 2017. Además, de la importancia de los ingresos del sector, el turismo permite impulsar y generar altas tasas de empleo, colaborando a la robustez de la economía española [4]

La afluencia de turistas produce efectos positivos como la activación y estimulación del comercio, del espíritu empresarial, de la creación de empleo, de generación de infraestructura nueva para uso turístico y el aumento de la recaudación de impuestos.

Además de la gran ayuda económica y generación de empleos que produce en España el turismo, la experiencia viajera ayuda a entender las diferencias de culturas, hechos históricos, estilo de vida y ser más respetuosos, tanto para el viajero como para la persona que le atiende para que disfrute de su estancia. En resumen, el turismo produce una fuente de bienestar en España.

1.1 MOTIVACIÓN DEL PROYECTO

Para mantener el liderazgo en el ranking, es necesario incluir diversos cambios junto a los nuevos avances tecnológicos para poder acelerar y optimizar los procesos, de tal forma que permita la consolidación de la rentabilidad del sector.

Actualmente, los alojamientos turísticos se encuentran liderados por el sector hotelero, aunque está comenzando un incremento moderado en España de los alojamientos realizados en viviendas propias, prestadas o alquiladas, tal como se demuestra con los ingresos de la plataforma *Airbnb* en la Ilustración 3. *Airbnb* es una plataforma que se encarga de enlazar a turistas que desean encontrar alojamiento a los anfitriones que poseen alojamientos, es decir, se trata de un alquiler temporal de un inmueble.

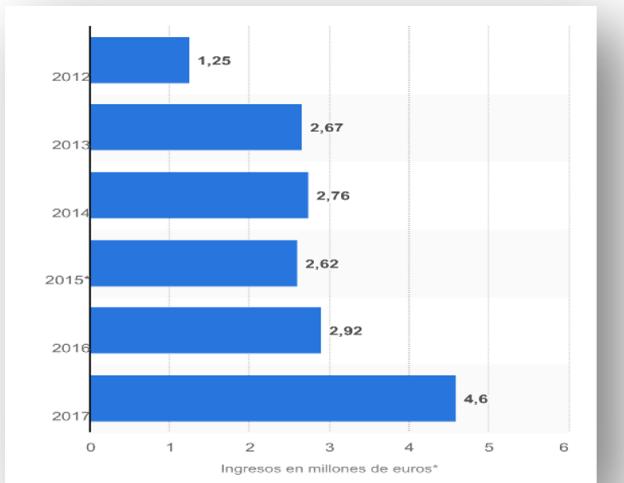


Ilustración 3: Evolución anual de los ingresos declarados por Airbnb en España entre 2012 y 2017 (en millones de euros) [5]

El segmento de los turistas internacionales son lo que más interesan en el turismo español debido a los beneficios que aportan. Este sector está dispuesto a pagar cada vez más por el alquiler de un alojamiento temporal en España tal como se muestra en la Ilustración 4.

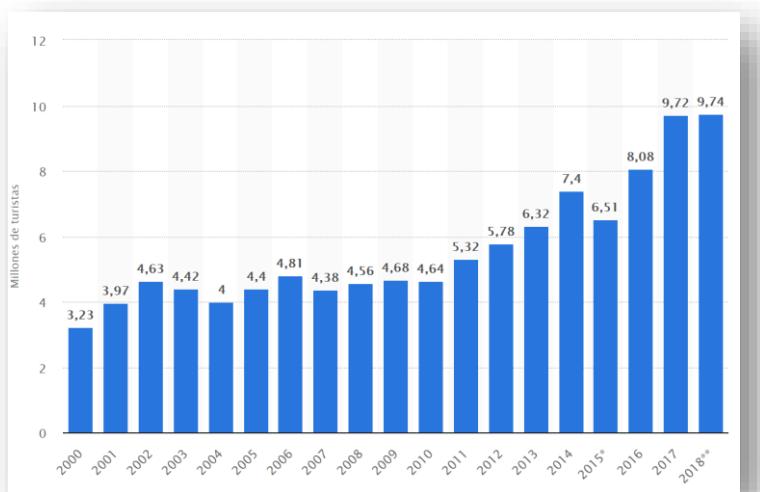


Ilustración 4: Evolución anual del número de turistas internacionales que alquilaron una vivienda en España de 2000 a 2018 (en millones) [6]

Con el fin de aprovechar este nicho de mercado, se replantea dar un nuevo enfoque a las viviendas compartidas que supusieron un boom en los años 1980 y 1990 en España. Las viviendas aprovechas por turnos con fin turístico suponían un reparto de las cuotas como el IBI, comunidad, derramas de esa vivienda que únicamente se aprovechaba durante una o dos semanas al año. Sin embargo, estas viviendas compartidas han creado grandes discusiones como la fecha del disfrute, cuotas impagadas, contratos con perpetuidad, estafas... [7]

Es momento de plantearse: ¿cómo hacer un sistema transparente? ¿Cómo reflejar todas las transacciones de pago realizadas por la vivienda? ¿Se podría reflejar cuando el copropietario desea su período vacacional? ¿Se podría eliminar la empresa gestora? ¿Se podría limitar la duración del contrato?

Además, las organizaciones encargadas de certificar nuestra información, como el derecho de propiedad de un inmueble, almacenan información muy valiosa y confidencial, lo que puede suponer un riesgo, al dejar que un tercero sea propietario de información personal. En muchas ocasiones, estas organizaciones disponen de más información de la que necesitan para verificar la identidad, pudiendo regalar información extra a las entidades que solicitan información. ¿Deberíamos descentralizar esta información y únicamente permitir que el usuario sea el que decida a quién enseñarle qué información? ¿Podría existir un entidad descentralizada encargada de llevar a cabo la identidad digital?

Actualmente, la tecnología ha ayudado y fomentado al hombre en su día a día. Es por esta razón, que debemos recurrir a la tecnología para intentar solucionar ese mundo tan oscuro de los derechos de aprovechamiento por turnos de bienes inmuebles permitiéndonos mejorar todas las cuestiones planteadas previamente.

1.2 SOLUCIÓN PROPUESTA

Tal como se ha mencionado, existen diferentes opciones para disfrutar de un apartamento en régimen turístico. Sin embargo, aparecen distintos problemas; la inexistencia de una plataforma con transparencia donde se especifiquen los requisitos de

forma clara, como la gestión de suministros, cuotas de mantenimiento, acceso de servicios, o la duración del contrato, o la inexistencia de la posibilidad de compartir una vivienda sin la necesidad de tener una gestión económica tan elevada y de ser parte propietaria. Con el fin de solucionar los inconvenientes mencionados, se propone la realización de un sistema descentralizado, en el que se realice un aprovechamiento por turnos de inmuebles por distintos copropietarios a través de la realización de una aplicación descentralizada (*Dapp*) que ejecute *Smart Contracts*, basado en la emergente tecnología *Blockchain* en una red de pruebas de *Ethereum: Rinkeby*.

Al realizarlo a través de *Smart Contracts*, cláusulas contractuales ejecutadas automáticamente, se resuelven los problemas mencionados, y también se deposita confianza en el usuario, se produce una reducción de costes al disminuir los intermediarios y la automatización de pagos. Además, la tecnología *Blokchain* sobre la que se basa, permite verificar y validar la identidad del usuario para evitar suplantación de identidad, validar y verificar que no se ha alterado la información a través de algoritmos de consenso dispuestos en cada uno de los nodos de la red. Esta tecnología se basa en criptografía por lo que reduce el número de ataques y vulnerabilidades a la información almacenada. También, la *Dapp* permitirá al usuario una mayor comodidad y sencillez al poder almacenar sus claves criptográficas en una *Wallet*, así como confiar en la detección de errores y mejoras de la aplicación a través de un consenso al ser un sistema transparente de código *Open Source*, evitando estafas al estar controlada por muchos ojos.

Asimismo, si el comprador necesita un préstamo, la plataforma dispondrá de una sección para indicar qué condiciones e intereses está dispuesto a aceptar para que el prestamista acepte la entrega de la cantidad con la condición de que el prestatario la devuelva, cumpliendo las condiciones. Al realizarlo a través de esta majestuosa tecnología, proporciona una gran transparencia, inmutabilidad y rapidez al compararlo con el proceso de una hipoteca, así como la reducción de los gastos que conlleva.

También, para mejorar la eficiencia en los seguros de hogar, se realizarán las pólizas a través de *Smart Contracts* simplificando el flujo entre asegurador y asegurado, quedando registrados todos los posibles fraudes para poder evitarlos. El asegurador optimiza su intercambio de información y automatizar las pólizas en función de los

parámetros de caracterización del perfil de riesgo reduciendo las pérdidas por fraude. El asegurado tiene un notable mejoría en la experiencia del cliente al ser un proceso más simplificado y transparente

Actualmente, *Blockchain* tiene muchas puertas por abrir, al igual que sucedió en su momento con *Internet*, *Amazon* o *AirBnB*. A pesar de ciertos detractores acerca de la estabilidad y especulación de sus monedas, en los últimos días se ha visto como *Bitcoin* ha vuelto a resurgir después de la fuerte caída que tuvo, suponiendo este hecho un gran impulso para la tecnología. [8] Por ello, se pretende modificar el enfoque de los alojamientos turísticos, seguros y bancos a través de esta tecnología descentralizada.

En este documento, se presenta la información necesaria para comprender la seguridad, inmutabilidad y transparencia de la tecnología *Blockchain* junto a las herramientas que se han utilizado para implementarla. Además, se muestra el desarrollo del caso de uso clave planteado para esta tecnología junto a los posibles avances y mejoras de la emergente tecnología.

Capítulo 2. DESCRIPCIÓN DE LAS TECNOLOGÍAS

En este apartado se describen las tecnologías que han sido utilizadas a lo largo del desarrollo del proyecto; ya haya sido durante la fase de comparativa de las distintas tecnologías, o bien en la construcción de la plataforma descentralizada que se ha apoyado en el uso de *Smart Contracts*. Posteriormente, en el Capítulo 8. se observará la conexión e integración de las tecnologías utilizadas en el proyecto.

2.1 ETHEREUM

La idea de la plataforma de *Ethereum* fue concebida por Vitalik Buterin, un programador con nacionalidad ruso-canadiense. Sin embargo, el proyecto actual ha sido cofundado por Vitalik Buterin, Mihai Alisie, Anthony Di Lorio, Charles Hoskinson, Dr. Gavin Wood y Joseph Lubin. Buterin publicó el documento amarillo de Ethereum (donde los desarrolladores utilizan de referencia) y Mihai Alisie fundó ConsenSys, una compañía centrada en aplicaciones descentralizadas que resuelve problemas reales para organizaciones. [9]

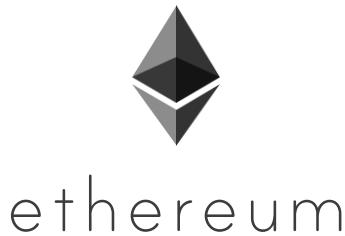


Ilustración 5: Logo Ethereum

Ethereum surgió para una funcionalidad similar a *Bitcoin*, aunque para solucionar algo más complejo; permitir realizar una plataforma descentralizada donde se ejecutan *Smart Contracts*. Ethereum es la plataforma y *Ether*, *ETH* es su criptomoneda.

Las redes de *Ethereum* son creadas para transferir dinero y guardar información, y permiten crear interesantes aplicaciones. Hay muchas redes de *Ethereum*, algunas solo sirven para hacer código de prueba y transacciones de pruebas. De hecho, cada uno puede

crear su red privada de *Ethereum* en el ordenador y ser restringida o abrirla a más personas.

Se ha utilizado *Ethereum* en vez de *Bitcoin* porque la velocidad de las transacciones de *Ethers* es mucho mayor que la de *Bitcoins*, los *Smart Contracts* se encuentran integrados en la plataforma a diferencia de *Bitcoin* que precisa un software externo. También, cabe destacar que a diferencia de *Bitcoin* no existen límites de *Ethers* y la recompensa por minería es mayor. [10] El proceso de minado en *Ethereum* al principio trabajaba a través de *Proof of Work* pero al tener un gran coste, la comunidad de *Ethereum* quiere explotar *Proof of Stake* al mejorar económicamente el proceso.

Las redes *Ethereum* están formadas por nodos, es decir, una máquina con conexión a *Ethereum* y software. Cada nodo funciona en un cliente *Ethereum* y cada uno tiene la copia de la información de la *Blockchain*. En *Ethereum*, el poder reside en cada uno de esos nodos pues es lo que permite no depender de un agente externo ya que la información se almacena encriptada de forma inmutable y segura. Además, *Ethereum* tiene un código open source y los cambios de mejoras se deben llevar con el consenso de la mayoría se conocen como EIP: *Ethereum Improvement Proposal*.

2.2 SOLIDITY

Los *Smart Contracts* se desarrollan en el lenguaje de programación llamado *Solidity* en *Ethereum*. Es un lenguaje que se ha desarrollado en exclusiva para ejecutar los *Smart Contracts*. En el caso de otras plataformas se utilizan lenguajes de programación distintos como *C++*, *Golang*, *Ivy-lan* o *Javascript*, entre otros. [11] Muchas veces se compara con el lenguaje *Javascript* al tener una sintaxis similar, aunque difieren en otros aspectos como que *Solidity* está fuertemente tipado.



Ilustración 6: Logo Solidity [12]

Solidity está en constante desarrollo y avance, es por ello que se puede especificar la versión que se utiliza en la parte inicial de cada fichero, para utilizar el compilador acorde a esto. Al no ser *Solidity* lo que se ejecuta en la red *Ethereum*, necesitamos un compilador. El compilador de *Solidity* lanza dos ficheros:

- ABI: Application Binary Interface, que se usa para trabajar con Javascript. Es similar a la API de las páginas web, es la parte que se utiliza para interactuar la Dapp con los Smart Contracts. Es similar a un interfaz para conectar el código de bytes real con Javascript.
- El código de bytes real: lo que se implementa en la red Ethereum

Solidity tiene la ventaja de proporcionar un medio confiable y seguro entre ambas partes. También, permite realizar herencia a distintos niveles. Otras función muy importante a destacar es que permite comunicarse con funciones seguras a través de *ABI*. Además, facilita la creación de bucles para los cuales se ha implementado un límite de *Gas* que se puede consumir en una transacción para evitar que sean infinitos.

La desventaja de este lenguaje radica en que no se puede modificar el contrato que se realizó puesto que se basa en la inmutabilidad. Otro inconveniente es el constante cambio en el que se encuentra actualmente, puesto que está en constante crecimiento y mejoría, por lo que sus bibliotecas no están consolidadas.

2.3 TRUFFLE

Truffle es un *framework* muy habitual en cualquier proyecto profesional y está muy documentado. Es una herramienta de línea de comandos que se utiliza para facilitar el desarrollo y ejecución de *Smart Contracts*. *Truffle* permite crear una estructura de proyecto: archivos y directorios, con unos simples pasos. También, facilita la configuración y conexión con la redes *Blockchain*.



TRUFFLE

Ilustración 7: Logo Truffle [13]

Dentro del entorno de Truffle de trabajo se encuentra:

- Ganache, que permite la ejecución de pruebas de forma local, tal como se explica en el siguiente apartado.
- Drizzle, que facilita la interfaz de usuario en la creación de una Dapp. Se trata de un conjunto de bibliotecas sobre las que se encargan de sincronizar y actualizar la información del *Smart Contract*.
- Truffle Teams, que posibilita supervisar el estado de la *Blockchain*. Visualizar el número de bloques creados, la cantidad de *Gas* utilizada, ejecutar test de manera automática, o vigilar eventos son algunas de sus funcionalidades.

Truffle aún se encuentra en desarrollo por lo que algunas funciones sobre *Truffle* no funcionan bien, por lo que es importante mantener la última versión.

2.4 GANACHE

Las redes están hechas por uno o más nodos. Un nodo es un máquina que corre un cliente *Ethereum*. Un nodo lo puede crear cualquiera, lo único que hay que hacer es encender el ordenador, descargar el software del cliente y conectarlo a la red *Ethereum*.

Ganache es un simulador del *Blockchain* de forma local, por ello no se tiene que pagar y se puede probar rápidamente. Por lo tanto, *Ganache* permite simular un nodo de *Ethereum*, aunque sin descargarte toda la *Blockchain* existente, a diferencia de *Geth* (cliente *Ethereum*), que es más lento y necesita bastantes recursos para su ejecución.



Ilustración 8: Logo Ganache [14]

Cada transacción hecha en *Ganache* se guarda por defecto en la memoria. En el momento que reseteas, se pierde todo en *Ganache*. Se está modificando esto para que en *Ganache 2* exista un *workspace* sobre el que trabajar. [15]

Ganache pertenece al *framework Truffle*. Existe la opción de utilizar *Ganache-cli*, que se utiliza a través del editor y la ventana de comando para ejecutar, compilar, probar y desplegar *Smart Contracts*, siendo un inconveniente su interfaz. Por ello, se ha trabajado de manera visual con *Ganache* tal como aparece en la Ilustración 9.

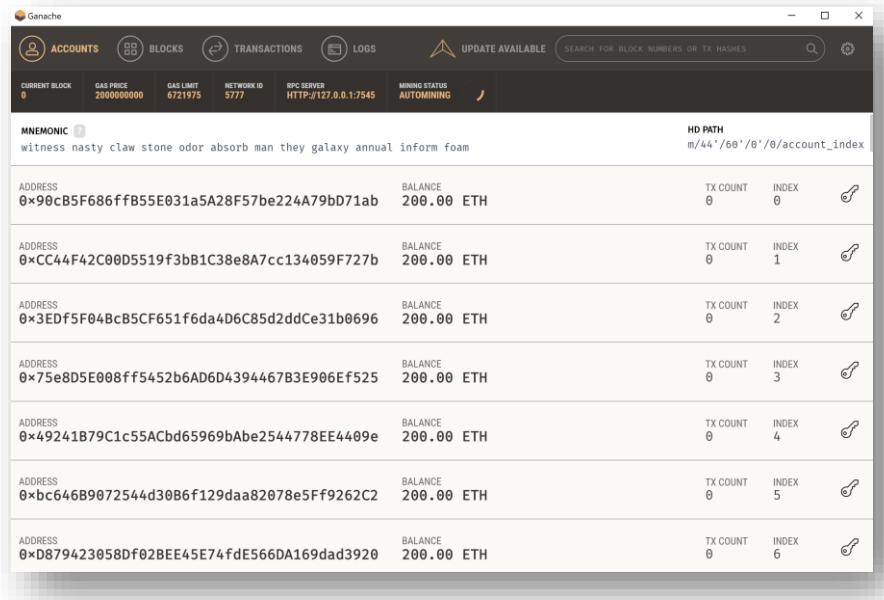


Ilustración 9: Aplicación Ganache

Ganache al ser una simulación de *Blockchain* en local, permite la creación de diferente número de cuentas, ajustar el puerto local, indicar el saldo de la cuenta, ver las diferentes transacciones realizadas, así como importar las diferentes cuentas a *Metamask*

a través de su clave privada. Por tanto, permite inspeccionar el funcionamiento de *Blockchain* y un control avanzado sin ningún coste para la fase de pruebas realizada antes de la implantación.

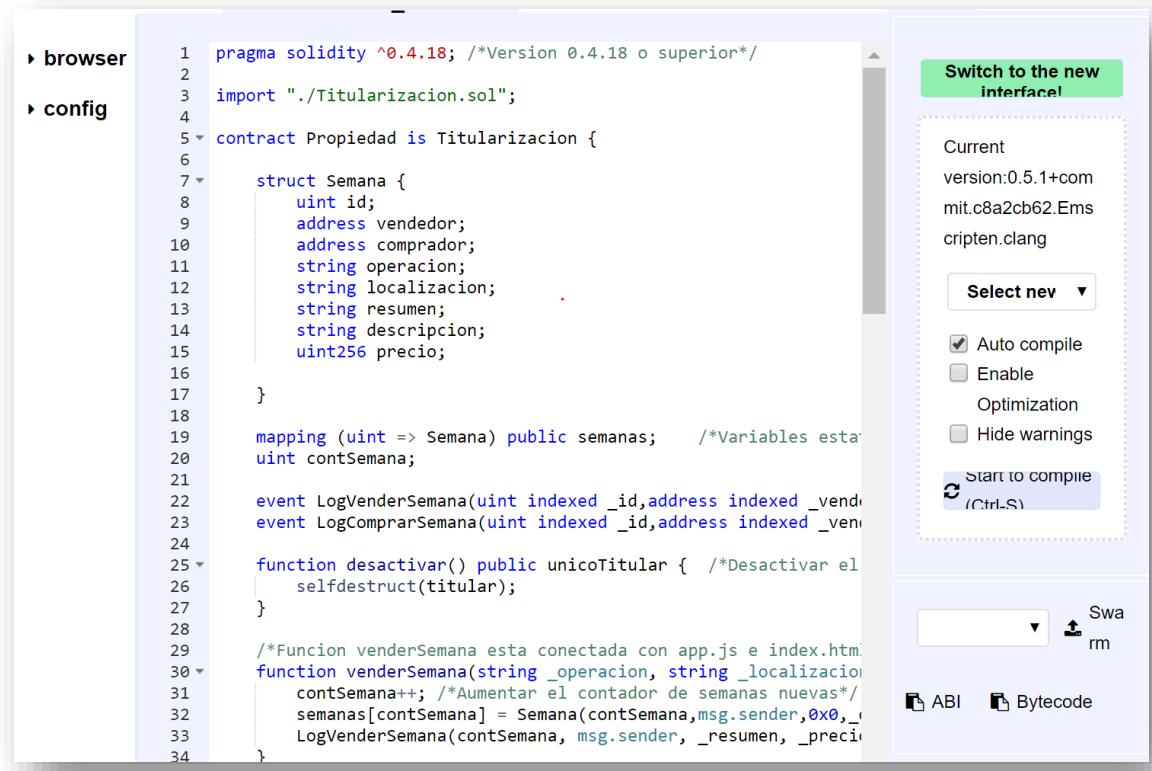
Ganache tiene la desventaja de que no es exactamente igual que la red principal *Ethereum* en el tema de la minería. En *Ganache* no hay mineros por lo que el rendimiento y rapidez de la red puede variar. El segundo inconveniente es que el límite de *Gas* de la red principal *Ethereum* puede ser modificado por los mineros, mientras que en *Ganache* no ocurre.

2.5 ENTORNO DE DESARROLLO

Se presenta un análisis de los entornos de desarrollos posibles a emplear para seleccionar los recursos más apropiados:

2.5.1 REMIX IDE

Es un editor de texto con compilador realizado en el navegador que permite crear *Smart Contracts* de *Ethereum* en *Solidity*. Las ventajas que posee es que es muy sencillo de utilizar y posee una buena interfaz, como puede observarse en la Ilustración 10. Los problemas que presenta es que al funcionar sólo en el navegador, no testea correctamente los tiempos de transacciones. [16] Además, a la hora de testear los *Smart Contracts*, es necesario realizar diversos pasos, lo que supone incomodidad en un gran proyecto. A la hora de realizar una *Dapp*, supone una traba añadida, de manera que *Remix IDE* es un buen sistema como inicio a la programación de *Smart Contracts*, pero no es recomendable para realizar grandes desarrollos.



```

1 pragma solidity ^0.4.18; /*Version 0.4.18 o superior*/
2
3 import "./Titularizacion.sol";
4
5 contract Propiedad is Titularizacion {
6
7     struct Semana {
8         uint id;
9         address vendedor;
10        address comprador;
11        string operacion;
12        string localizacion;
13        string resumen;
14        string descripcion;
15        uint256 precio;
16
17    }
18
19    mapping (uint => Semana) public semanas; /*Variables estaticas*/
20    uint contSemana;
21
22    event LogVenderSemana(uint indexed _id,address indexed _vendedor);
23    event LogComprarSemana(uint indexed _id,address indexed _vendedor);
24
25    function desactivar() public unicoTitular { /*Desactivar el contrato*/
26        selfdestruct(titular);
27    }
28
29    /*Funcion venderSemana esta conectada con app.js e index.html*/
30    function venderSemana(string _operacion, string _localizacion) {
31        contSemana++; /*Aumentar el contador de semanas nuevas*/
32        semanas[contSemana] = Semana(contSemana,msg.sender,0x0,_operacion,_localizacion);
33        LogVenderSemana(contSemana, msg.sender, _resumen, _precio);
34    }

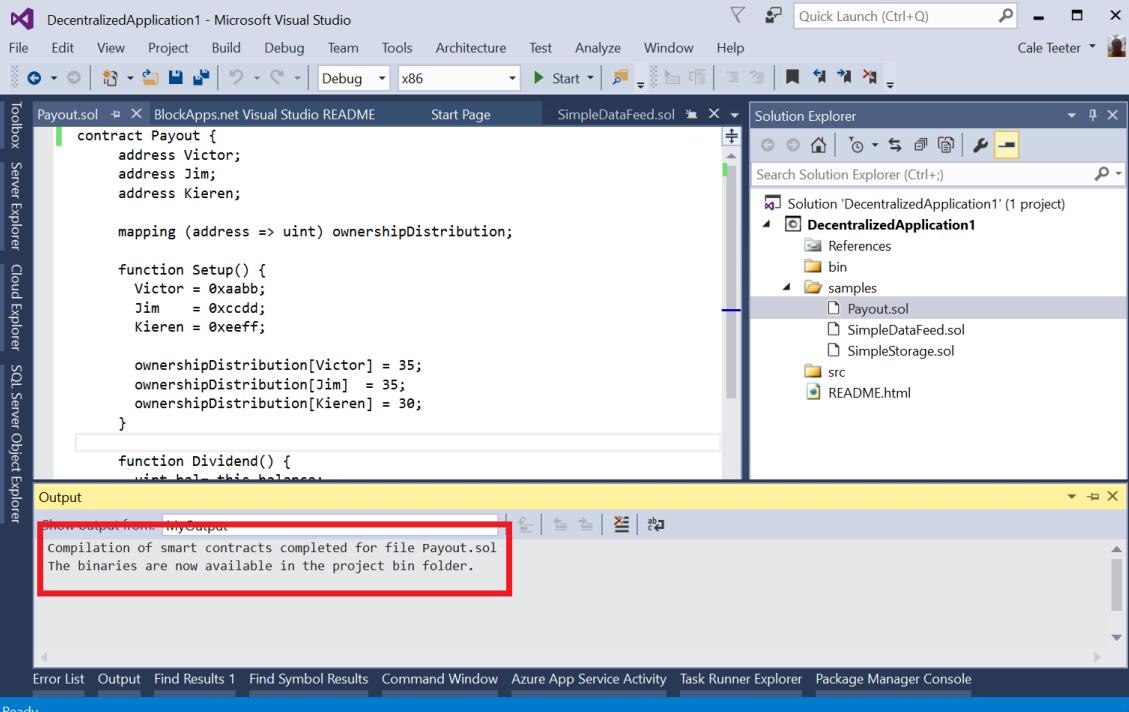
```

The screenshot shows the Remix IDE interface. On the left, there's a sidebar with two sections: 'browser' and 'config'. The main area displays a Solidity smart contract named 'Propiedad' which inherits from 'Titularizacion'. The contract defines a 'Semana' struct with fields for id, vendedor, comprador, operacion, localizacion, resumen, descripcion, and precio. It also contains a mapping of uint to Semana named 'semanas' and a variable 'contSemana' for tracking the number of weeks. Two events are defined: 'LogVenderSemana' and 'LogComprarSemana', both taking an indexed uint and an indexed address as parameters. A function 'desactivar()' is also present. On the right side, there's a configuration panel with tabs for 'Current' (version 0.5.1+com mit.c8a2cb62.Ems cripten clang), 'Select new', 'Auto compile' (checked), 'Enable Optimization', 'Hide warnings', and a button 'Start to compile (Ctrl+S)'. Below these are buttons for ABI and Bytecode.

Ilustración 10: Remix IDE [17]

2.5.2 VISUAL STUDIO SOLIDITY

Es un entorno de desarrollo integrado que actualmente soporta *Solidity*, lo que permite desarrollar más fácilmente a través de diversas herramientas y desplegarlo en la nube de *Microsoft Azure*. Las ventajas que presenta es que se puede desarrollar en una red de pruebas o en la red principal de *Ethereum* y la facilidad añadida de diversas herramientas. Aunque su interfaz no es tan simple como la de *Remix IDE*, también es buena, como puede verse en la Ilustración 11. El gran problema que presenta actualmente es que está en pleno desarrollo y genera muchos problemas de extensiones. [18]



```

contract Payout {
    address Victor;
    address Jim;
    address Kieren;

    mapping (address => uint) ownershipDistribution;

    function Setup() {
        Victor = 0xaabb;
        Jim   = 0xccdd;
        Kieren = 0xeeff;

        ownershipDistribution[Victor] = 35;
        ownershipDistribution[Jim]   = 35;
        ownershipDistribution[Kieren] = 30;
    }

    function Dividend() {
        uint bal = this.balance;
    }
}

```

Show output from MyOutput
Compilation of smart contracts completed for file Payout.sol
The binaries are now available in the project bin folder.

Ilustración 11: Visual Studio Solidity [19]

2.5.3 EDITOR ATOM

Atom es un editor de texto editable que permite tener una visualización de los diferentes paquetes, tal como se muestra en la Ilustración 12, y, unido al plugin *Atom Solidity Linter*, remarca el lenguaje *Solidity*. [20] Además, *Atom* se vincula fácilmente al framework *Truffle*, lo que permite ahorrar mucho tiempo al programador. Su interfaz permite visualizar el proyecto de manera cómoda. Asimismo, posee una gran cantidad de paquetes que se instalan fácilmente junto a gran cantidad de información. También, se puede personalizar diferentes paneles de visualización.

The screenshot shows the Atom IDE interface with the following details:

- Project Path:** C:\Users\teres\Documents\propiedad_pruебaT\propiedad
- File Menu:** File Edit View Selection Find Packages Help
- Truffle Project Structure:**
 - Project
 - propiedad
 - build
 - contracts
 - Asegurador.sol
 - Migrations.sol
 - Prestamista.sol
 - Propiedad.sol**
 - Titularizacion.sol
 - migrations
 - node_modules
 - src
 - css
 - js
 - app.js
 - truffle-contract.js
 - media
 - index.html
 - bs-config.json
 - debug.log
 - package.json
 - truffle.js
- Open Files:** truffle.js, index.html, app.js, Propiedad.sol
- Code Editor Content (Propiedad.sol):**

```
1 pragma solidity ^0.4.18; /*Version 0.4.18 o superior*/
2
3 import "./Titularizacion.sol";
4
5 contract Propiedad is Titularizacion {
6
7     struct Semana {
8         uint id;
9         address vendedor;
10        address comprador;
11        string operacion;
12        string localizacion;
13        string resumen;
14        string descripcion;
15        uint256 precio;
16    }
17
18    mapping (uint => Semana) public semanas;      /*Variables estaticas*/
19    uint contSemana;
20
21    event LogVenderSemana(uint indexed _id,address indexed _vendedor,string
22    _resumen,uint256 _precio);   /*Los eventos se relacionan en la app.js*/
23    event LogComprarSemana(uint indexed _id,address indexed _vendedor,address
24    indexed _comprador,string _resumen,uint256 _precio);
25
26    function desactivar() public unicoTitular { /*Desactivar el contrato*/
27        selfdestruct(titular);
28    }
29
30    /*Funcion venderSemana esta conectada con app.js e index.html*/
31    function venderSemana(string _operacion, string _localizacion, string
32    _resumen, string _descripcion, uint256 _precio) public {
33        contSemana++; /*Aumentar el contador de semanas nuevas*/
34        semanas[_contSemana] =
35            Semana(_contSemana,msg.sender,0x0,_operacion,_localizacion,_resumen,_descrip-
36            cion,_precio);/*Guardar la semana segun el formato y secuencia de la
37            estructura*/
38        LogVenderSemana(_contSemana, msg.sender, _resumen, _precio);
39    }
40
41    function getNumeroSemanas() public view returns (uint) { /*Contar numero
42        de semanas*/
43    }
44}
```
- Bottom Status Bar:** contracts\Propiedad.sol 17:6, LF UTF-8 Solidity GitHub Git (0)

Ilustración 12: Atom Editor

2.5.4 COMPARATIVA

Es difícil, tomar la decisión correcta sobre el IDE de desarrollo, puesto que al no estar afianzadas, hay que enfrentarse a los diversos problemas que surgen en el uso de cada uno de ellos.

Como se quiere realizar el despliegue de una *Dapp* en la red de pruebas *Rinkeby*, se descarta la utilización de *Remix IDE* por complicar los pasos.

Respecto al trío *Atom-Truffle-Ganache* frente a *Visual Studio Solidity*, destaca el primero, a pesar de que su utilización es más compleja, permite una gran cantidad de

pruebas requeridas en este tipo de proyectos donde no funcionar como debe implica, muchas veces, un gran gasto económico.

En la tabla que se muestra a continuación, se han valorados distintos aspectos que caracterizan a los entornos de desarrollo. Estas características a valorar son cómo se integran todas las funcionalidades que se pueden llegar a obtener, la facilidad de desarrollo que muestra el entorno y la simplicidad del programa, la curva de aprendizaje a la que hay que enfrentarse, la facilidad y exactitud con la que se pueden analizar las transacciones de la red Ethereum desde la propia interfaz y por último la estabilidad que muestra en el mercado cada uno de estos entornos de desarrollo. Se han evaluado desde Muy Bajo en el peor de los casos hasta Muy Alto.

	<i>Remix IDE</i>	<i>Microsoft Visual Studio</i>	<i>Atom Editor</i>
Integración	Muy Bajo	Alto	Medio
Facilidad de desarrollo	Medio	Alto	Muy Alto
Curva de aprendizaje	Bajo	Medio	Medio
Análisis de transacciones	Medio	Muy Alto	Muy Alto
Estabilidad	Medio	Bajo	Alto

Tabla 1. Comparativa entre entornos de desarrollo

2.6 METAMASK

Metamask es un *plugin* del navegador que facilita interactuar con la red de pruebas sin la necesidad de ser un desarrollador. *Metamask* es una cartera digital donde almacenar los *Ethers* a través de las claves de las cuentas. Se guardan de manera segura, puesto que acumula las claves en el navegador del usuario en vez de servidores remotos. A diferencia

de otro tipo de cuentas, como el email, una cuenta de *Metamask* se utiliza para todas las redes de Ethereum : red principal *Ethereum*, *Rinkeby*, *Ropsten*, *Kovan*, *Goerli*, y también a nivel local.

Metamask permite crear cuentas, importar cuentas a través de la clave privada y conectar con un monedero físico como *Ledger* o *Trezor*. En *Metamask*, al realizar las transacciones se indica el *Gas* que se utiliza para poder decidir sobre aceptarla o rechazarla.

Metamask se utiliza en la *Dapp* para que el usuario sin nociones de programación pueda realizar transacciones (enviar dinero, almacenar datos...) a través de una interfaz sencilla.



Ilustración 13: Logo Metamask [21]

Se ha preferido utilizar *Metamask* en lugar de *Mist Browser* para interactuar en la red *Ethereum* por encontrarse en una fase de desarrollo más avanzada.

2.7 WEB3.JS

A la red *Ethereum* se puede conectar a través de *Metamask*, que está disponible para cualquier usuario, o a través de *Web3.js*, que es recomendable para ser utilizado por programadores. *Web3.js* es un conjunto de bibliotecas que permite conectar las transacciones: enviar y recibir *Ether*, leer y escribir información en *Smart Contracts*, o la creación de *Smart Contracts* entre otros.

En el proyecto se ha utilizado para desarrollar un cliente que se relacione con la red Ethereum. Es decir, leer y escribir datos del servidor web (*npm lite server*), y leer y escribir en la red *Ethereum*. En el caso de enviar ciertos *Ether* a una dirección, primero

se envía al servidor *backend*. Este servidor utiliza la librería *Web3.js* para interactuar con cualquier red *Ethereum*, no solamente la red de *Ethereum* principal. El servidor *backend* utiliza esta librería *Web3.js* para crear lo que se llama el objeto transacción.



Ilustración 14: Logo Web3.js

2.8 ETHERSCAN

La herramienta *EtherScan* posibilita buscar la transacción realizada en la distintas redes de *Ethereum*, facilitando la transparencia, al ser externo a la *Fundación Ethereum*, permitiendo la indexación y la búsqueda de todas las transacciones en *Ethereum*. [22]



Ilustración 15: Logo Etherscan [23]

En Ilustración 16 se muestra arriba del todo la zona de búsqueda, en la que se puede buscar cualquier transacción, cualquier número de bloque, *token* o cualquier *hash*.

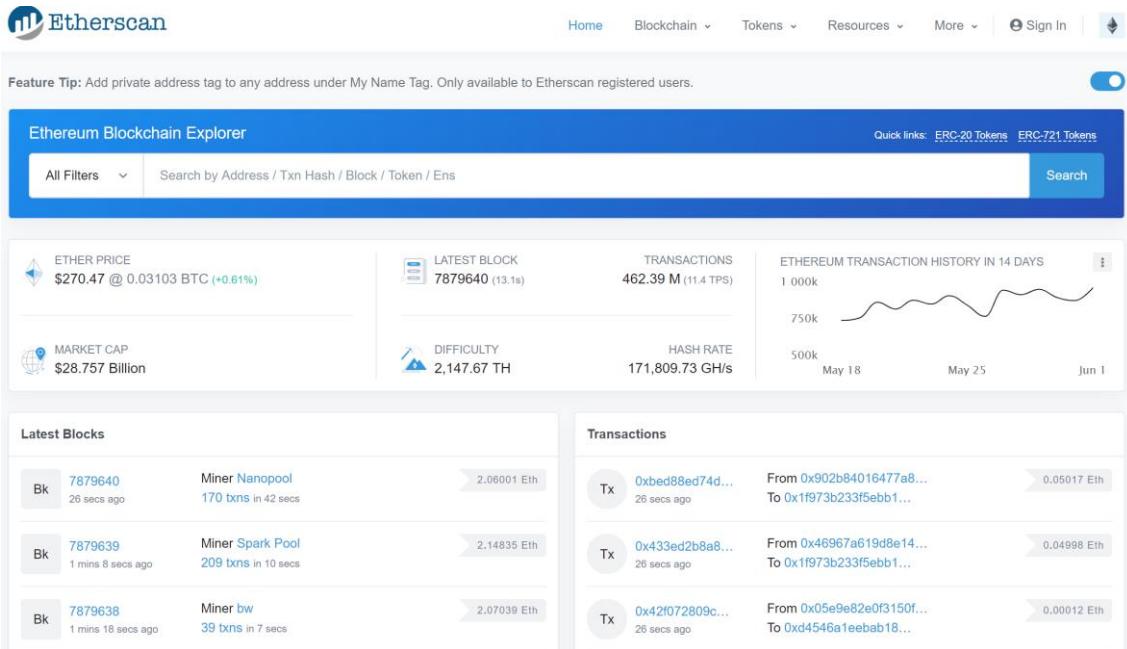


Ilustración 16: Transacción Etherscan [24]

2.9 NODE.JS

Node.js es un lenguaje de servidores que tiene diversas librerías. Es una herramienta de código abierto que posibilita ejecutar *Javascript* en el lado del servidor. Funciona para varias plataformas: *Windows*, *Mac OS x*, o *Linux* entre otros. [25]



Ilustración 17: Logo Node.js [26]



Ilustración 18: Logo Lite-server [27]

Node.js lite-server es un servidor de desarrollo local permite, a la hora de realizar la *Dapp*, que los cambios se actualicen rápidamente respecto a los ficheros *Javascript*, *css* y *html*; es decir, que se recargan automáticamente. Para poder funcionar el servidor, se necesita tener previamente instalado *node.js*.

En el directorio del proyecto, se necesita instalar el servidor, y, tras la instalación, se abrirá automáticamente en el navegador la página *localhost:3000*. Dentro de la configuración del directorio del proyecto, se puede modificar diferentes parámetros, como por ejemplo, el puerto.

2.10 RINKEBY

Las redes *Ropsten*, *Kovan* y *Rinkeby* son todas redes de prueba. Estas redes se utilizan para probar código y obtener *Ether* gratuito para probar los *Smart Contrats*.

La red *Ropsten* ofrece *Proof Of Work*. En cambio, *Kovan* y *Rinkeby* ofrecen *Proof of Address*, que permite que no sean vulnerables a los ataques de spam. *Kovan* solamente soporta *Parity*, mientras que *Rinkeby* soporta *Geth*. Por ello, a pesar de tener un tamaño de almacenamiento menor *Rinkeby* que *Kovan*, ha sido elegida *Rinkeby* para implementar los *Smart Contrats* de prueba, puesto que no es necesario gastar dinero para realizar. [28]

En la Ilustración 19, se observa el estado de la red *Rinkeby*. Además, en su página web se posibilita obtener dinero en *Rinkeby*, a través de la publicación del número de cuenta a través un tweet y con un tiempo limitado para evitar saturar la red.

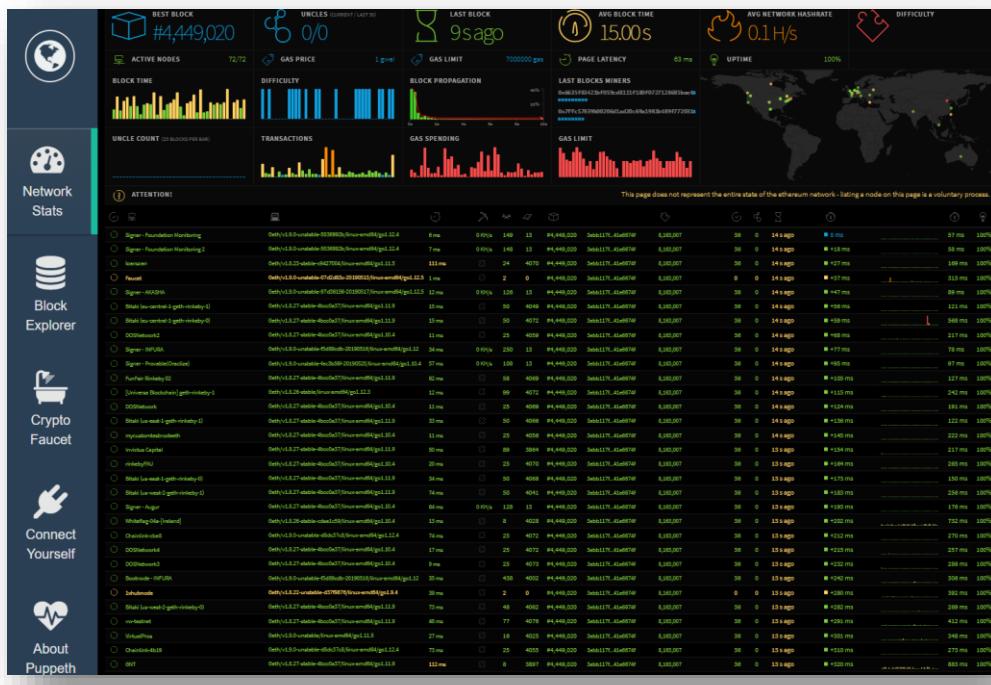


Ilustración 19: Rinkeby estado de la red [29]

2.11 INFURA

En el caso de realizar la implementación de la *Dapp* a nivel local, se ha utilizado *Ganache*, nodo de desarrollo local. Para poder hacer la implementación de la *Dapp* en una red de pruebas como *Rinkeby*, se utilizará *Infura*. Se ha decidido utilizar *Infura* en vez de *Geth*, puesto que *Geth* es un nodo que se instalaría de forma local, pero debe estar perfectamente sincronizado y actualizado para no ocasionar problemas. *Como Infura es un clúster de nodos de Ether, se utiliza un nodo remoto de Infura en el proyecto para la implementación en la red de pruebas Rinkeby, para evitar los problemas de tener desactualizado las versiones con las bases de datos de los nodos si tuviera un nodo local instalado, así como para evitar problemas de capacidad de procesamiento.*

En definitiva, *Infura* tiene una *API* donde *Web3.js* puede interactuar con la infraestructura de *Ethereum* con igual nivel de seguridad que si fuera en un nodo local. Además, la seguridad de *Infura* también reside en que sus claves privadas no son

almacenadas ni administradas por esta infraestructura, si no que se encarga un proveedor de *HD Wallet*.



Ilustración 20: Logo Infura [30]

Capítulo 3. ESTADO DE LA CUESTIÓN

En los años 80, surgieron derechos de aprovechamiento por turno de bienes inmuebles, especialmente de uso turístico. Es decir, disfrutar de forma anual durante un período de ese inmueble durante un número determinado de años, gestionado por una promotora que indica las condiciones de cuota inicial, mensual y mantenimiento. Debido a las altas cuotas de gestión, engaños, fraudes por falta de regulación, imposibilidad de asistencia en el período otorgado, entre otros, han hecho que sus servicios hayan disminuido. [31]

Las viviendas de aprovechamiento por turnos de bienes se encuentran a cargo de una promotora o gestora. Esto supone unos elevados costes junto al mantenimiento anual de los mismos. Estas viviendas no pertenecen a los “copropietarios”, sino a la empresa gestora que se encuentra detrás de esto. [32] [33]

Actualmente, existen diferentes plataformas de software centralizado dedicadas a la oferta de alojamientos de alquiler turístico con impuestos de coste de gestión y sin opción de pertenencia a la vivienda, como *Airbnb*, fundada en 2007, o *HomeAway*, fundada en 2004, lo que supone un beneficio para el propietario del inmueble y la plataforma. [34] La plataforma más utilizada en España es *Airbnb*. *Airbnb* ingresa el 3% de comisión de cada reserva de los anfitriones, y en torno al 8% de los huéspedes. Los turistas suelen elegir *Airbnb* frente a un hotel por tener distintas prestaciones como cocina, piscina privada, lavadora... y por lo general un precio inferior al hotel. Los anfitriones prefieren alquilarlo de este modo que un alquiler anual para poder disfrutar ellos también de la vivienda y obtener una rentabilidad de alquiler mucho mayor.

Respecto a la tecnología, se ha de mencionar en orden cronológico la manera de realizar contratos. En primer lugar, los contratos no iban unidos a ninguna tecnología y se redactaban en lenguaje humano. Más tarde, surgen los contratos ricardianos, en los años 90. Son acuerdos fundamentados en un lenguaje informático vinculado a un lenguaje humano, permitiendo ser auditables por ambos, puesto que se ocupa de registrar las acciones e intenciones sin concernirle su ejecución. [35]

Más tarde, de la mano de Nick Szabo, se definió el concepto de *Smart Contract* en 1995 aunque la tecnología *Blockchain* surgió catorce años después. Los *Smart Contract* son cláusulas contractuales programadas para ejecutarse automáticamente que no pueden ser detenidas. [36] El nombre otorgado ha recibido retoques de su pensador, Nick Szabo y de Vitalik Buterin, autor de la primera *Blockchain* con *Smart Contracts*.

*“Smart contract” is a very useful concept & phrase. “Smart” as in
“smart phone” (shorthand for computerized phone), “contract”
meaning it does some important things we previously relied on
contracts to do for our deals, especially controlling assets &
incentivizing performance.*

*That said, if your dApp/persistent script doesn't control assets or,
short of invoking traditional law, incentivize performance, it's not a
smart contract and you should call it something else.*

*Worrying about whether a smart contract is “legally enforceable”
reflects a profound misunderstanding. The main relation of smart
contract to traditional courts is that smart contracts control burden of
lawsuit. If “possession is 90% of the law”, then a good smart
contracts may be “99% of the law. – Nick Szabo [37]*

*To be clear, at this point I quite regret adopting the term “smart
contracts”. I should have called them something more boring and
technical, perhaps something like “persistent scripts” – Vitalik
Buterin [38]*

3.1 BLOCKCHAIN RESPECTO AL CASO DE USO EN ESPAÑA

Los sectores manejados en el caso de uso implementado en este proyecto corresponden al sector inmobiliario, al sector de la banca y al sector de los seguros. Por ello, se hará un repaso acerca de la tecnología *Blockchain* que existe en cada uno de ellos en España.

En España, están surgiendo diferentes soluciones utilizando *Blockchain* acerca de los inmuebles:

- *ProntoPiso* es una agencia inmobiliaria online que garantiza la venta de la vivienda en España en un período determinado, adelantando parte del importe de la transacción al propietario. [39] En octubre de 2018, *ProntoPiso*, con la tecnología proporcionada por *JLL*, comienza a realizar las tasaciones de viviendas a través de *Blockchain*, ahorrando costes y tiempo, simplificando el proceso de verificación y garantizando la seguridad de los datos. [40]
- En noviembre de 2018, la empresa española dedicada al negocio inmobiliario, *Metrovacesa*, ha lanzado sobre *Blockchain*, en primer lugar, la contratación de los suministros de la vivienda sobre *Alastria*. *Alastria* es una “red *Blockchain/DLT* semipública, independiente, permisionada y neutral, diseñada para ser conforme con la regulación existente, que permite a los asociados experimentar estas tecnologías en un entorno cooperativo” [41] Cabe destacar que el primer nodo universitario de *Alastria* se realizó en la Universidad Pontificia Comillas ICAI-ICADE. [42]
- El segundo lanzamiento de *Metrovacesa* ha sido dedicado a los inversores internacionales en vivienda sobre la red *Ethereum*, con el fin de agilizar el proceso de inversión únicamente a un sencillo paso, en vez de los tres pasos anteriores; oficina internacional, oficina española y promotora. Para ello se ha ayudado de la plataforma de inversión *Brickex* y de la consultora *Grant Thornton*. El proceso consiste en *tokenizar* los inmuebles, es decir, asignar un valor aceptado por una comunidad, y reflejarlos en una cuenta de *Metrovacesa* en *Blockchain*. A través de *Smart Contracts*, liberan los *tokens* cuando se cumplen las requisitos prefijadas en la compra. En caso que no finalice la venta, los *tokens* regresan a la cuenta de *Metrovacesa*. [43]
- El 19 de Marzo de 2019 se presentó en el *Demo Day*, en el campus de *Google* en Madrid la *start up: Real Fund*. Esta empresa se encarga de *tokenizar* los activos

inmobiliarios con el fin de invertir de forma sencilla en el tradicional sector inmobiliario. Además, cabe destacar que en el *Demo Day* se llevó a cabo la votación de los proyectos a través de *Blockchain*. Sus creadores José García Caballero y Juan Manuel Garrido han comentado la falta de marcos jurídicos completos para los proyectos de *Blockchain* en España. [44]

El sector de la banca en España se encuentra entusiasmado descubriendo y profundizando en posibles soluciones utilizando *Blockchain*. A continuación, se muestran datos relevantes sucedidos.

- En marzo de 2018, el Banco Santander y CaixaBank apuestan en un modelo de negocio que utiliza la tecnología *Blockchain* donde se necesita gran confianza, como es el comercio exterior y los avales internacionales. Además, BBVA comenta que *Blockchain* permitirá reducir costes y aumentar la productividad de procesos. Por otro lado, Bankia no deposita confianza en las criptomonedas por su gran riesgo y volatilidad. [45]
- En abril de 2018, BBVA junto a Indra realizaron el primer préstamo en *Blockchain* dentro de una organización, en este caso fue dentro de la entidad BBVA. [46] En este mismo mes, CaixaBank comenzó sus primeras transacciones en *Blockchain* de comercio exterior dentro de su consorcio *Batavia*. [47]
- En julio de 2018, el Banco Santander comenzó a descubrir *Blockchain* enfocado al mercado de valores, donde cambian *trading* de valores e instrumentos financieros a través de la creación de un equipo de Banca de Inversión Digital. [48] La entidad apuesta en este modelo de negocio que utiliza la tecnología *Blockchain* para aquellos que necesita gran confianza como es el comercio exterior y los avales internacionales. [45]
- En noviembre de 2018, BBVA formó parte del encuentro realizado para el lanzamiento de la *Asociación Internacional para Aplicaciones Blockchain* organizado por la Comisión Europea. El objetivo del encuentro es promover la

tecnología y facilitar la interoperabilidad entre las diferentes redes y entornos de desarrollo. [49]

- En febrero de 2019, el Grupo Santander, entidad crediticia más grande española, comenzó a focalizarse en reducir sus costes y evolucionar a un mejor aprovechamiento de la tecnología a través de un acuerdo con la entidad IBM que se apoyara en la tecnología *Blockchain*. El acuerdo está pactado en 700 millones de dólares. [50]

En España, están empezando a brotar situaciones de interés acerca de las aseguradoras con la tecnología *Blockchain*:

- A finales de 2017, Axa presentó un modelo en pruebas de un seguro para los vuelos retrasados en la red de *Ethereum*. El *Smart Contract* se encuentra conectado a varias bases de datos aéreas para verificar y validar la información de los vuelos. [51]
- En junio de 2018, Mapfre, aseguradora española de las más antigua que opera en España, comenzó a configurar un equipo de expertos en la tecnología *Blockchain* para proteger la seguridad de las operaciones y los activos de los clientes a través de la integración con otras tecnologías y recursos disponibles. [52]
- En noviembre de 2018, Axa, la primera compañía de seguros mundial, se unió a la red *Blockchain Alastria* para disponer de un entorno seguro y transparente. [53]
- En marzo de 2019, Seguros Santalucía ha premiado a cinco startups que aceleran la economía a través de la tecnología *Blockchain*. NodalBlock fue una de las startup premiadas que tiene relación con la protección de la identidad digital para los clientes de las pólizas; se encarga de proteger la identidad digital, emitir certificados y realizar firmas digitales. [54]

3.2 **BLOCKCHAIN RESPECTO A LA INFORMACIÓN MANEJADA EN EL CASO DE USO**

La sociedad cada vez se maneja más por internet, dejando información relevante al descubierto que en algún momento podría causarle algún disgusto. La huella digital cada vez es más completa, desde el *GPS* que tenemos en móvil que indica cuánto tiempo pasamos en cada sitio, qué zonas visitamos, el dinero que gastamos a través de tarjeta bancaria, con quién hablamos por *Whatsapp* o por *LinkedIn*, saber si acudiste a una manifestación, o a un acto en contra de una ideología de tu empresa... una cantidad de información que se obtiene a través de diferentes dispositivos como cámaras, móvil, ordenadores, altavoces, robots o radares de tráfico.



Ilustración 21: Información en descuberto [55]

Esta información no se puede despreciar, su valor y la explotación de esos datos puede ser muy valiosa. Tener la información minuto a minuto de lo que hace cada persona, de las llamadas que recibe, las que realiza, los mensajes entrantes o salientes, cuánto tiempo pasa en supermercado o en el gimnasio, el tiempo que se encuentra conectado a Internet o usando el móvil, cuando va en coche, equivale a tener un diario automático. Esto sería prácticamente una misión imposible sin los medios y dispositivos

electrónicos, puesto que ni el mejor espía sería capaz de recopilar tal cantidad de información.

Actualmente la seguridad y la privacidad van de la mano, puesto que un acceso indebido a los datos se convierte en uno de los mayores riesgos. Un ejemplo reciente sucedió en 2018, con la fuga de información de *Facebook* que afectó a más de 87 millones de personas, cuya información se utilizó para influir en resultados electorales y publicidad focalizada. [56]

Tradicionalmente, los organismos certificadores de información se encuentran en manos de una organización. Es decir, un tercero que se encarga de certificar y validar que el servicio o producto se corresponde con lo que se indica. La dependencia de un tercero para validar supone un coste alto y supone exponer información relevante a una entidad en la que confiar.

La tecnología *Blockchain*, es una herramienta muy potente al basarse en la criptografía permitiendo no exponer a la luz datos relevantes y sensibles. Es decir, un sistema de intercambio de información segura.

Una situación cotidiana sucede cuando, en el seguro de contratación de su casa, le realizan una fotocopia del DNI que contiene información de sus padres, fecha de nacimiento, lugar de nacimiento, o dirección entre otros datos, cuando realmente el seguro no necesitaba toda esa información, o el cliente no quería mostrarle toda la información. En cambio, si se utiliza la tecnología *Blockchain* para que solo la información necesaria pueda ser consultada, y la información privada no pueda ser consultada, se estaría reduciendo el peligro de exposición de información sensible, puesto que la información almacenada no depende un tercero que pueda vendernos en cierto momento, sino que se encuentra descentralizada. Es decir, se limitaría esta información a los organismos que contienen la información confidencial.

En la siguiente imagen, Ilustración 22, se muestra como un *DMV* es la entidad que certifica cierta información, en este caso es el DNI. El *DMV* usa claves vinculadas a su identificador descentralizado en la *Blockchain*, firmando la petición para que sea

inviolable y se sepa que fue emitido por *DMV*. El cliente tiene una *Wallet* para guardar sus solicitudes y puede usar claves vinculadas a un identificador descentralizado que pueda unir a su DNI. Cuando la entidad que necesita verificar, *BAR*, quiere comprobar que es mayor de edad, puede presentar la licencia de DNI. La *BAR* verifica que es cierto y que el *DMV* se la emitió, siendo el usuario el que la muestra. Otra solución, sería la utilización de un *token* para validar que la información ofrecida cumple con los requisitos.

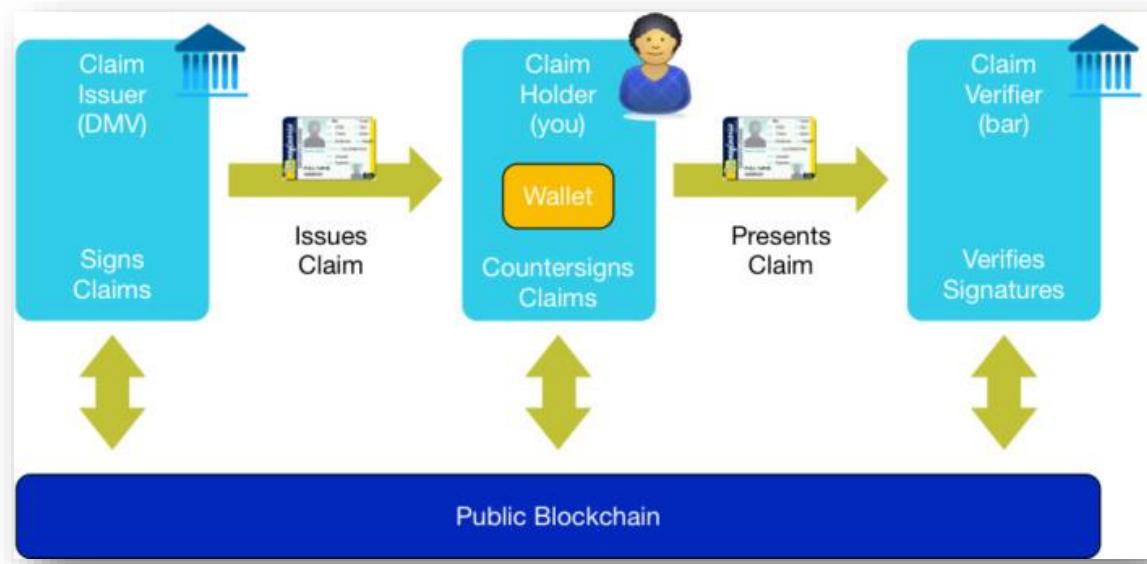


Ilustración 22: Identidad digital Blockchain [57]

En mayo de 2018, nace la aplicación *Blockpass* que posibilita verificar la identidad y proteger la información de los usuarios. Se basa en un sistema para almacenar información descentralizada teniendo los usuarios control sobre su identidad digital, siendo estos los que deciden acerca de su identidad. De este modo, ninguna entidad almacena sus datos confidenciales.

Por tanto, en el momento que un cliente envía datos para verificarse en *Blockpass* a través de plataformas de confianza, se almacena una copia local y se cifra con una contraseña creada por el usuario. Después, *Blockpass* recibe los datos, le realiza un *hash* y los elimina de sus servidores, quedando únicamente disponible para el usuario. De este

modo si se produjera un ataque en los servidores solo serían datos ilegibles. Así, sólo se compartirán con un tercero cuando el usuario lo desee. [58]

3.3 BLOCKCHAIN Y LA UNIVERSIDAD PONTIFICIA COMILLAS ICADE-ICAI

El pasado año 2018, la Universidad Pontificia Comillas se convirtió en la primera universidad en formar parte del consorcio *Alastria*, albergando un nodo de la red. Esta situación animó e impulsó a otras universidades a potenciar la economía española desde la tecnología *Blockchain*. [59]

Alastria es un ecosistema formado por diferentes entidades del sector energético, bancario, educativo y jurídico. El propósito de esta creación es generar un estándar que cumpla el marco jurídico español y europeo en el que puedan operar las diferentes entidades. [60]



Ilustración 23: Logo Alastria [61]

El pasado 4 de junio de 2019, se produjo la primera jornada de *Blockchain-Economía* en la Universidad Pontificia Comillas. En ella se han reunido diferentes ponentes de distintas organizaciones y entidades: Alastria, CNMV, Colegio de Registradores de España, ICADE, Metrovacesa, Bankia, Eurogestión, Everis, Carex Logistics, Vottun, Proteum, ioBuilders, Blockchain Economía y Secretaría General del Tesoro y Financiación Internacional. De los ponentes ha destacado Amanjyot S. Johar, el principal inversor de *Proteum*. [62] *Proteum* es una *startup* de Silicon Valley que se

encarga de potenciar y asesorar a las empresas en la tecnología *Blockchain*. Actualmente, Amanjyot S. Johar se encuentra en búsqueda de iniciativas con cierta viabilidad en España para invertir hasta 20 millones de euros. [63]

Capítulo 4. DEFINICIÓN DEL TRABAJO

4.1 JUSTIFICACIÓN

Actualmente, a causa de la comodidad para obtener un perfil en internet, los casos de suplantación de identidad han crecido exponencialmente. Estas suplantaciones tienen diferentes objetivos; desde obtener información bancaria o personal hasta la realización de acosos. Para realizar la verificación de identidad, se está apelando de un tercero. Este problema lo soluciona la tecnología *Blockchain*, que facilita la verificación del usuario real para poder firmar la transacción evitando las amenazas y vulnerabilidades del segundo factor de autenticación, al utilizarse métodos criptográficos.

A menudo suele surgir problemas acerca de si ha sido *hackeado* o alterada cierta información. Este hecho es solucionado a través de *Blockchain*, porque la información se almacena como una estructura de datos donde no pueden ser alterados o eliminados, puesto que estos bloques se encuentran con un tiempo marcado al registrar en orden cronológico en diferentes nodos de la red. Además, a la hora de almacenar el bloque existen los algoritmos de consenso (*PoW* o *PoS*) para decidir, validar y verificar el bloque con gran confianza, evitando fraudes. El almacenamiento en diferentes nodos de la red permite no depender de terceros, ser descentralizado y poder recuperar los datos si hubieran sido eliminados en algún punto, similar a disponer de varias copias de seguridad.

Además, los algoritmos de validación de bloques *PoW* o *PoS*, conllevan a que hay mineros que validan esta información. Los mineros perciben retribuciones en *tokens* o *criptomonedas* por validar la información lo que fomenta la utilización de la *Dapp* ya que es el medio de compra o intercambio

Debido a la cantidad de ataques en internet y la gran vulnerabilidad, las compañías hacen hincapié en la seguridad de su almacenamiento de información. Es por ello que *Blockchain* aporta protección a la información al ofrecer la criptografía de clave pública y privada para las cuentas junto a los hashes. Es decir, los datos se encuentran encriptados para

todo usuario salvo el propietario. Respecto al almacenamiento de gran información, no está contemplado en el almacenamiento de la cadena de bloques. Es por ello que en el momento que se desea una gran información, se debe vincular un archivo al sistema *IPFS* y desde ahí se hará un *hash* al fichero.

Para evitar que se encubra información, la tecnología *Blockchain* es transparente, se posibilita observar la cadena de origen a fin.

Los *Smart Contracts* permiten no estar pendiente constantemente de una transacción y activarse cuando se cumplen ciertos requerimientos para enviar o recibir información o *Ethers*.

En la actualidad al entrar en una página web de compra, sólo confiamos en el pago a través de ella si se hace a través de una pasarela de pago de un tercero. En una *Dapp* se elimina la pasarela de pago realizándose de forma directa, es decir el usuario puede enviar o recibir *Ether* con un gasto muy bajo de *Gas*, lo que supone que sea prácticamente sin comisión.

Los usuarios cada vez que entran en una página web tienen que hacer el esfuerzo de registrarse o recordar su usuario y contraseña, puesto que dispone de innumerables contraseñas. Esta situación se evita con el uso de una *Dapp*, los usuarios no requieren registrarse debido a que con una única *Wallet* que se vincula a la *Dapp* dispones de tu clave pública y privada, sin necesidad de tener que recordar una cadena enorme. Actualmente existen inicios de sesión a través de ciertas redes sociales, y la diferencia es que una *Dapp* es descentralizada, es decir, si una red social deja de existir, podremos dejar de iniciar sesión con ella.

A diferencia de una aplicación actual, el código de una *Dapp* tiene que ser *Open Source*. El código del desarrollo debe estar disponible para mejorarse si se desea, aunque siempre en consenso debe decidirse si es una mejora oportuna. Esto permite crecer de forma exponencial, en comparación con un código que está disponible para un grupo reducido. Además, debe estar disponible la información del *backend*, es decir, los *Smart Contracts*,

para vigilar que no haya estafas o se robe información facilitando aún más la confianza de los usuarios.

4.2 OBJETIVOS

El objetivo principal es poner en práctica unas cláusulas contractuales que se ejecuten automáticamente, y permitan aprovechar inmuebles por turnos entre distintos copropietarios, asegurando una alta confianza entre los distintos copropietarios, y sin la necesidad de intermediarios. Con el propósito de desempeñar este propósito se deberán realizar los siguientes objetivos:

- i. **Investigar la tecnología *Blockchain* y su aplicación de *Smart Contracts* con vistas a desplegar el caso de uso en la red de pruebas *Rinkeby***

Al tratarse de una tecnología emergente, se estudia el funcionamiento teórico con fin divulgativo y autodidacta para poder realizar su aplicación adecuadamente, para así ser capaz de realizar un despliegue satisfactorio sobre una red de pruebas de *Ethereum: Rinkeby*, donde se explorarán los diferentes intercambios y transacciones.

- ii. **Estudiar y comparar las plataformas para la aplicación de *Smart Contracts* con objeto de diseñar, desarrollar y testear el caso de uso.**

Se estudia las plataformas desarrolladas sobre esta tecnología para poder utilizar la más conveniente al proyecto. Tras fijar la solución, se pone en práctica el desarrollo, a través del lenguaje *Solidity*. Tras obtenerlo, es necesario comprobar su funcionamiento, puesto que cualquier modificación en la red *Blockchain* implica un gasto de dinero, por lo que es muy importante asegurarse de que funciona en el modo que se desea antes de implantarlo en la red principal.

- iii. **Solucionar el problema de gestión de inmuebles con tecnología *Blockchain*.**

Se fijan los requerimientos necesarios para la compartición del inmueble y se realiza un estudio legal de la solución. Se buscará el soporte a estas necesidades en la tecnología *Blockchain*.

4.3 METODOLOGÍA

Para poder cumplir los objetivos propuestos, el trabajo se desarrollará a través de la metodología descrita a continuación.

En primer lugar, en la etapa de investigación de la tecnología, se utilizará el método deductivo, puesto que se partirá del estudio de la tecnología *Blockchain* para centrarse después en los *Smart Contracts* de la red *Ethereum*, tanto en su contenido como en la tecnología. Más adelante, se deduce la forma de aplicar un caso de uso en particular de los *Smart Contracts*. [64]

Por otra parte, al tratarse de un proyecto individual basado en una tecnología emergente, debe considerarse la facilidad para poder adaptarlo a condiciones cambiantes de la tecnología empleando metodologías ágiles. Para ello, se va a utilizar la herramienta *Kanban*, que permite desmenuzar en una serie de pasos el proyecto para lograr el objetivo principal.

Kanban se basa en los siguientes principios:

- i. Flujo de trabajo: visualizar lo que se realiza a diario.
- ii. Limita la cantidad de trabajo que se puede realizar.
- iii. Mejora el flujo: cuando se acaba una tarea, la siguiente tarea más importante que se encuentra atrasada se pone en marcha. [65]

Para poder realizarlo se utiliza una herramienta muy visual y sencilla: *Trello*, tal como se muestra en la Ilustración 24.

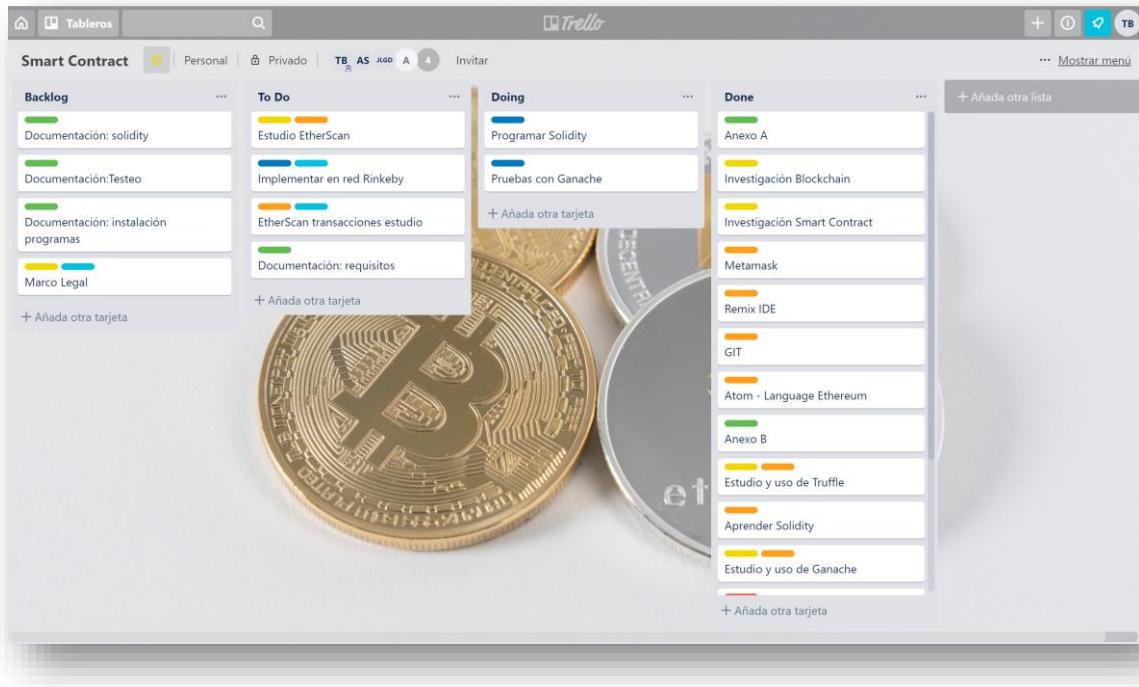


Ilustración 24: Kanban utilizado a través de Trello

Las tarjetas visuales se utilizarán con los siguientes fines:

- **Backlog:** lista de tareas que se deben realizar. Aparecer arriba del todo de la lista significa que se tiene que realizar con prioridad respecto a las que se encuentran en un lugar inferior.
- **To Do:** son las tareas que se mueven de *Backlog* a esta porque se han decidido realizar ahora, ya que se ha determinado que es el momento y se pueden conseguir realizar correctamente.
- **Doing:** tareas sobre las que se está trabajando. Son las tareas que al comenzarse a realizar pasan de encontrarse en *To Do* a esta lista.
- **Done:** son las tareas que ya se han completado. [66]

4.4 PLANIFICACIÓN

Dado que se dispone de un plazo limitado y un alcance relativamente cerrado en la realización del proyecto, se muestra en la Ilustración 25, a modo informativo, el tiempo que se fijó al principio del mismo a la dedicación de las diferentes tareas para conseguir los objetivos marcados, a través de un diagrama de Gantt.

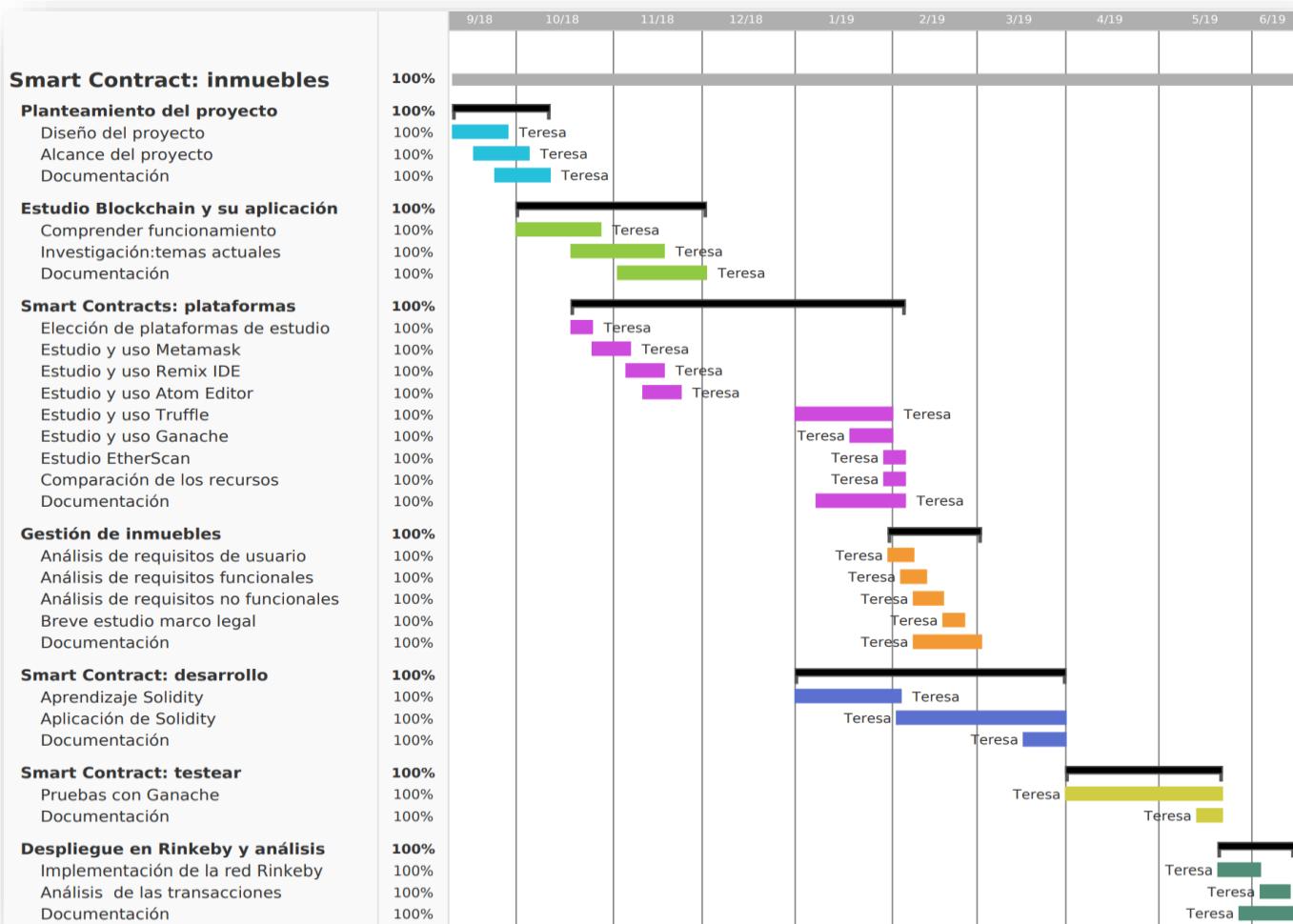


Ilustración 25: Diagrama de Gantt del proyecto

Tras finalizar el proyecto, se ha podido cumplir las tareas marcadas tal como se detalla a continuación:

En primer lugar, se definió el diseño y el alcance del proyecto para poder hacer frente a los costes, recursos, plazos y tareas.

Tras fijarse el alcance, se profundizó en la nueva tecnología de *Blockchain* tal como se detalla en las subtareas del *Kanban*: almacenamiento en *Blockchain*, ventajas de *Blockchain*, algoritmos de consenso de las redes *Blockchain*, *Smart Contracts*, *Tokens*, *Dapp* entre otros.

Más tarde, se estudió y se aplicaron las tecnologías y herramientas de *Blockchain*: *Solidity*, *Truffle*, *Ganache*, *Metamask*, *Web3.js*, *Etherscan*, *Node.js*, *Rinkeby*, *Infura* y el entorno de desarrollo. De forma paralela al estudio, al tener ciertos conocimientos se decidió fijar la funcionalidad y el análisis del sistema detallando en diferentes casos de uso, diagramas de secuencia y diagrama de actividad.

Finalmente, la aplicación se diseñó y desarrolló de forma iterativa en un nodo local a través de *Ganache* para realizar las pruebas oportunas y comprobar su correcto funcionamiento. Además, en esta fase se decidió ampliar el objetivo de la planificación desarrollando una interfaz para poder interactuar cualquier tipo de usuario con los tres *Smart Contracts*. Tras funcionar adecuadamente, se desplegó en la red de pruebas *Rinkeby* de *Ethereum* a través de *Infura* para evitar desagradables problemas con la sincronización y actualización de los nodos.

Durante la realización del proyecto, se han establecido reuniones de seguimiento y encauzamiento del avance del mismo con los directores.

4.5 ESTIMACIÓN ECONÓMICA

En el apartado se presenta la estimación del coste económico del desarrollo del proyecto. En primer lugar cabe destacar que todos el software utilizado ha sido gratuito, por lo que el coste viene dado por el tiempo dedicado al proyecto de cada uno de los miembros del equipo.

A continuación, se presentan detalladas las tareas realizadas por el analista en el proyecto junto a sus horas para poder tener una vista completa del tiempo empleado.

Tarea	Horas
Definición del proyecto	20
Síntesis y documentación de tecnología <i>Blockchain</i>	60
Comparativa técnica entre tecnologías	20
Aprendizaje y aplicación de <i>Solidity</i>	40
Ánálisis y gestión de inmuebles	55
Estudio y uso de <i>Metamask</i>	10
Estudio y uso del entorno de trabajo	32
Estudio y uso de <i>Truffle</i>	9
Estudio y uso de <i>Ganache</i>	12
Estudio y uso de <i>Etherscan</i>	12
Estudio y uso de <i>Javascript</i>	15

Despliegue en <i>Rinkeby</i> y análisis de las transacciones	12
Elaboración de la memoria	60
Revisión de la memoria	20
Elaboración de la presentación	15
Total	392

Tabla 2: Análisis dedicado a cada tarea

Además del analista en el proyecto, se tiene que tener en cuenta el gasto del jefe de proyecto dedicado a las revisiones de documentación y supervisión del trabajo desarrollado. Es por ello que se ha procedido a reflejar en la Tabla 3, el coste de cada uno respecto a sus horas.

Integrante	Horas	Coste/hora
Analista	392	60 €
Jefe de proyecto	30	90 €
Total		26.220 €

Tabla 3: Relación entre tiempo y coste

Como se ha comentado anteriormente, todo el software utilizado ha sido gratuito, y además, no se ha entregado ningún hardware. Por tanto, el único coste que existe en este proyecto son los 26.220 € de personal.

Capítulo 5. BLOCKCHAIN

5.1 DESCRIPCIÓN GENERAL

Blockchain es un registro en todo momento de información. Cada bloque contiene un *timestamp* que posibilita ordenarlos cronológicamente y evitar remplazos puesto que cuando pertenecen a *Blockchain*, es muy difícil modificarlos. *Blockchain* es un sistema descentralizado que no depende de terceros. Los intercambios de información se realizan en el sistema, a través de transacciones como enviar y recibir criptomonedas, o realizar *Smart Contracts* entre otros.

Al ser una tecnología reciente y aún no muy asentada, suele confundirse con el término *Bitcoin*. Cabe destacar que *Blockchain* es la tecnología que se encuentra por debajo de *Bitcoin*. *Blockchain* es imprescindible para poder almacenar o intercambiar *Bitcoins*, es decir, se puede usar *Blockchain* para otras criptomonedas como *Ethereum*, *Ripple*, *Litecoin* o *Dash*.

La información almacenada dentro de cada uno de los bloques que conforman *Blockchain* depende del tipo de *Blockchain*. Existen tres tipos de *Blockchain*:

- **Blockchain pública:** cualquier persona tiene permiso para añadir un bloque a *Blockchain* o verificar un bloque. La utilización de las *Blockchain* públicas es gratuito y se recompensa a los mineros que se encargan de validar la información. Cualquier persona puede ver las transacciones, aunque recordando que estas transacciones aparecen de forma anónima.
- **Blockchain privada:** algunas personas autorizadas por una organización son las que pueden agregar un bloque a *Blockchain* o verificarlo. A pesar de pertenecer a una entidad para favorecer la transparencia, suele permitirse visualizar las transacciones a todos los usuarios. También, podría cerrarse el acceso a visualizar a todo el mundo.
- **Blockchain de consorcio o permisionadas:** surge de la fusión entre pública y privada. La constituyen un grupo de entidades, suelen tener una velocidad mayor al

tener un conjunto de nodos preseleccionados en varias organizaciones que se encargan de realizar el consenso y no encontrarse en una única entidad. Un ejemplo de esta *Blockchain* de consorcio es *Alastria*, formado por importantes compañías, uno de los nodos de *Alastria* se encuentra en la Escuela Técnica Superior de Ingeniería, ICAI. [67]



Ilustración 26: Nodo universitario Alastria ICAI [67]

Al estar refiriéndonos a esta tecnología como almacenamiento de información, se puede caer en el gran error de asemejarlo a un almacenamiento compartido. Por ello, es importante señalar las diferencias entre ambos.

- En el caso de *Blockchain* cabe destacar que su replicación se basa en el sistema *P2P, peer to peer*. Se trata de clientes distribuidos que no necesitan un servidor para realizar el intercambio de información. En el momento que una persona desee pertenecer a la red tiene la opción de obtener la información de *Blockchain* de principio a fin. Es decir, se realiza una replicación completa en cada nodo de la red. En el momento que se quiere añadir un bloque , cada nodo tiene que comprobar que no ha sido modificado y entre todos realizan un consenso acerca de su validez. Cabe señalar que modificar un bloque correctamente supone

modificar todos los bloques de la cadena y volver a realizar el algoritmo de consenso, *PoW* o *PoS*, para cada bloque de la cadena, lo que es prácticamente imposible hacer. En comparación con la base de datos compartida, la replicación se realiza a través de maestro-esclavo, haciendo que los datos del nodo maestro se repliquen en los servidores esclavos.

- Un diferencia muy significativa sucede en que *Blockchain* únicamente puede almacenar información en la cadena, y nunca podrá modificar o eliminar esa información mientras que en la base de datos compartida podrás alterarla.
- A la hora de producirse la validación de bloques es un algoritmo de consenso a nivel de *Blockchain*, por el contrario en la base de datos compartida, se crean reglas de validación, aunque a nivel local.

5.2 ALMACENAMIENTO EN BLOCKCHAIN

En el momento que se desea almacenar en *Blockchain* cierta información, se solicita ejecutar una transacción, ya sea a través de *Smart Contracts*, *Dapps* o criptomonedas. La información de la transacción se distribuye a cada uno de los nodos de la red *peer to peer*. Cada nodo se encarga de verificar y validar la transacción a través de los algoritmos de consenso (*PoW* o *PoS*). Tras verificar el correcto funcionamiento, se agrega el nuevo bloque a *Blockchain* donde no puede ser ya modificado.

Cuando creas una cuenta de *Outlook* y otra de *Gmail*, son cuentas independientes. En *Ethereum*, una cuenta se utiliza en todas las redes. Al crear una cuenta se crean tres partes distintas que constituyen la cuenta en la red *Ethereum*.

- **Dirección de la cuenta:** similar a la dirección de email o el nombre de usuario. Se puede compartir con cualquier persona y permite identificarse con el resto.
- **Clave pública y clave privada:** se combinan para crear una contraseña. Se utilizan para autorizar el envío de fondos de su cuenta a otras cuentas. La clave privada no se debe compartir nunca. Estas claves tienen una longitud muy grande, pues todo se almacena en hexadecimal, por ello se almacenan en *Wallet* digitales.

Al crear una cuenta sirve la misma dirección para todas las redes y sabemos que no se almacena en todas los *Ethers*. Esta información se indica en el *backend* de la aplicación tal como se detalla a continuación: en el *backend* se utiliza la librería *web3* para crear el objeto transacción y este objeto se envía a la red que hayamos indicado en el *backend*. En este proyecto se ha utilizado *Rinkeby*. Tras enviarse, se debe esperar a que el nodo que le ha llegado junte las transacciones que ha recibido, puede ser la que se está enviando y la de otros tres usuarios diferentes, y con todos ellos forme un bloque. La validación del nodo requiere tiempo pues necesita ciertos cálculos para resolver el algoritmo, proceso conocido como minería (*PoW*) o forjado (*PoS*). Es conveniente recordar que la red se comunica con un nodo y este con el resto de nodos. Finalmente, se produce correctamente y se devuelve un mensaje de éxito al navegador.

A continuación, se analizan los parámetros de ese objeto transacción que es donde se indica los fondos que se envían de dinero junto a otras propiedades. Se crea una transacción siempre que haya un intercambio de dinero.

- **Nonce:** es un cálculo que se lleva a cabo a través de los algoritmos de consenso para que de acuerdo a la información del bloque el hash comience por 0000. Es decir, los mineros intenten obtener el valor correcto en la red para obtener una retribución y un nuevo bloque.
- **To:** es la dirección a la cuenta donde se enviará cierta cantidad de *Ether*.
- **Value:** la cantidad de *Ether* que se quiere enviar desde el emisor.
- **gasPrice:** cantidad de *Ether* que el emisor está dispuesto a pagar por unidad de gas para procesar esta transacción.
- **startGas/gasLimit:** unidades de gas que la transacción consume.

- v, r, s : son propiedades criptográficas generados para dar seguridad a la transacción. Los tres valores se usan para generar la dirección de la cuenta de la persona que hace el intento de enviar el dinero. Si tienes la clave privada puedes generar v, r y s; pero si tienes v, r y s **no** puedes generar la clave privada

A través de la aplicación creada por Anders Brownworth, se muestra el funcionamiento de *Blockchain*. [68]. En primer lugar, a primera vista el hash se podría pensar que son números aleatorios. En cambio, es la huella digital de los datos digitales y es única en cada bloque. El hash reconoce tanto el bloque como su contenido. Si escribo en el bloque distinta información, el hash cambia. Por lo tanto, el hash es muy ventajoso cuando se quiere dar cuenta sobre si se han producido cambios; es una de las razones por la que *Blockchain* se considera inmutable.

En la Ilustración 27, la imagen de la izquierda muestra el hash y texto vacío. En la imagen de la derecha, al escribir información, se observa como el hash se modifica.

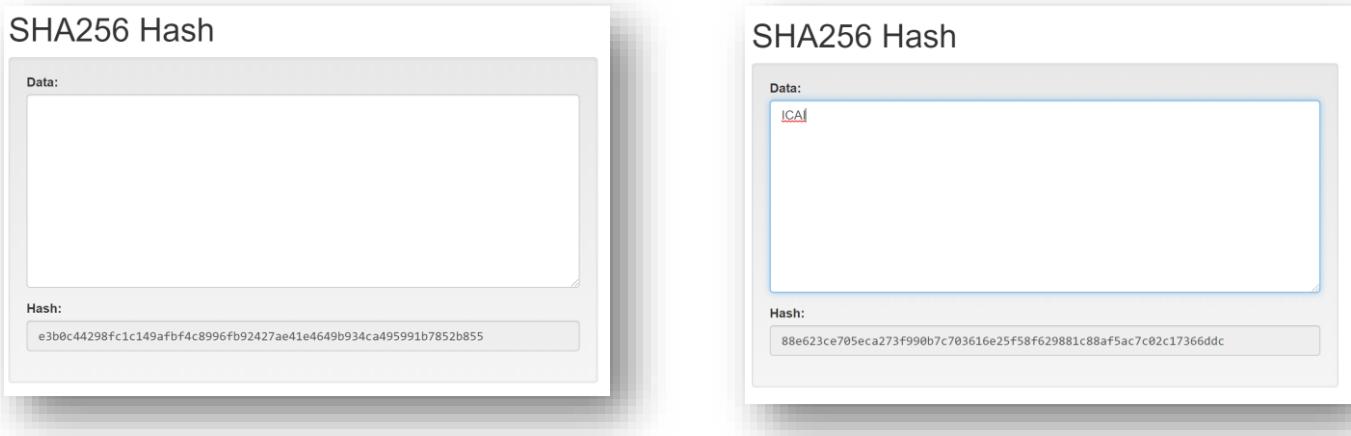


Ilustración 27: Hash, la huella dactilar digital [68]

Al no tratarse de un bloque, sino que únicamente se observa el funcionamiento del hash, no tiene por qué empezar por 0000. En la Ilustración 28 se puede identificar cómo el

bloque que no contiene información aparece firmado, mientras que en la imagen de la derecha se ha incluido información que aún no está firmada. Para ello, habrá que minarlo.

Block

Block:

#	1
---	---

Nonce:

Data:

Hash:

Mine

Block

Block:

#	1
---	---

Nonce:

Data:

Hash:

Mine

Ilustración 28: Se cambia información del bloque, hash no comienza por 0000 [68]

Tal como se aprecia en la Ilustración 29, después de minar el bloque, el hash comienza por 0000 indicando que el bloque está firmado. A través de los algoritmos de consenso se ha configurado un nuevo nonce.

Block

Block:

#	1
---	---

Nonce:

Data:

Hash:

Mine

Ilustración 29: Tras minar el bloque, hash comienza por 0000 [68]

Esta situación del bloque anterior es lo que sucede con cada uno de los bloques de la cadena. A continuación, se muestra un ejemplo de *Blockchain* con tres bloques. El primer bloque también es conocido como el bloque Génesis porque no contiene el hash de ningún bloque previo. El segundo bloque tiene la referencia del hash del bloque primero. Lo mismo sucede con el bloque tercero que tiene el hash del bloque segundo. Es decir, cada bloque tiene el hash del anterior excepto el primer bloque tal como aparece en la Ilustración 30.

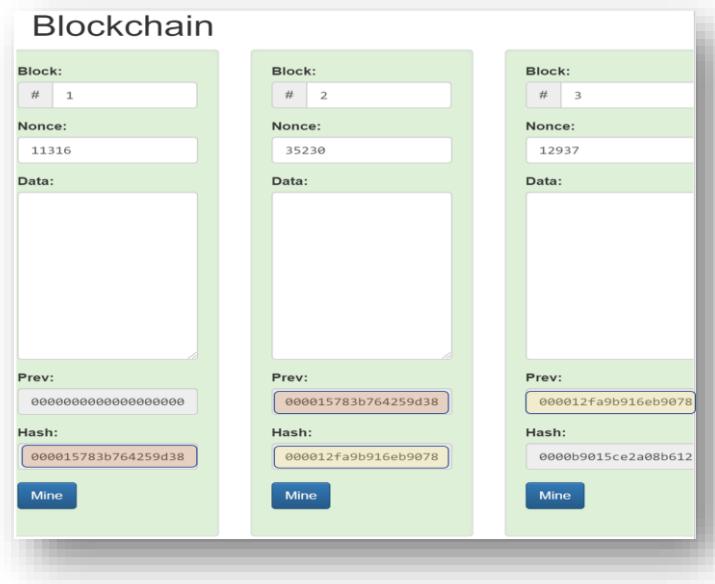
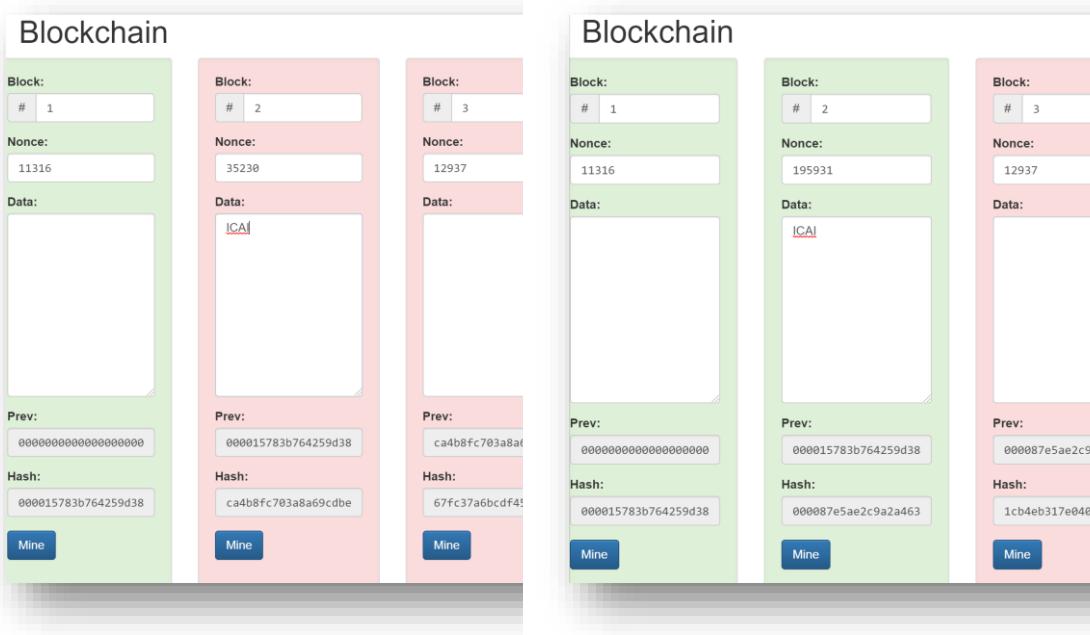


Ilustración 30: Cada bloque tiene hash anterior [68]

Esta situación comentada previamente hace que *Blockchain* sea inmutable debido a que resulta imposible realizar un cambio en ella. Para clarificarlo se plantea un ejemplo. Si un *hacker* modifica la información del bloque segundo, el hash también cambia y debe encontrar el correcto para que esté firmado, es decir minar el bloque dos. Además debería hacerlo para todos los bloques que son posteriores a él. En este caso el bloque tercero ya sería invalido y también debe minarse como se muestra en la Ilustración 31.



Blockchain

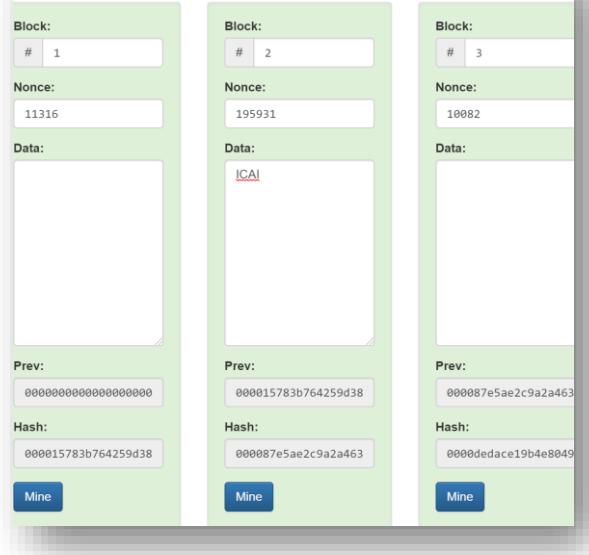


Ilustración 31: Minar el bloque y los posteriores

Cabe destacar que el proceso anterior únicamente no es afectado si se modifica el bloque último de la cadena.

El proceso visto de *Blockchain* anterior es lo que ocurre en un nodo. *Blockchain* es distribuida por lo que esta misma operación sucede en varios nodos de la red, por lo que si se desea modificar parte de la información se debe atacar a diferentes nodos y modificar ese bloque y los posteriores, llevando un tiempo de procesamiento muy elevado y costoso. Es por esta razón que *Blockchain* es immutable.

5.3 ALGORITMOS DE CONSENSO DE LAS REDES BLOCKCHAIN

Los algoritmos de consenso es una herramienta para poder decidir dentro de un grupo basándose en la mayoría. En el caso de *Blockchain* permiten que los nodos distribuidos lleguen a un acuerdo acerca de la validez de cada bloque asegurando la integridad y seguridad de la red.

Existen diversos algoritmos de consenso como *Delegated Proof Of Stake*, *Leased Proof Of Stake*, *Proof of Elapsed Time*, *Práctica de la tolerancia a faltas bizantinas*, *Tolerancia a faltas bizantinas*, *Tolerancia a faltas bizantinas simplificadas*, *Tolerancia a faltas bizantinas delegada*, *Directed Acyclic Graphs*, *Proof of Activity*, *Proof of Importance*, *Proof of Capacity*, *Proof of Burn*, *Proof of Weight*, *Proof of Work* y *Proof of Stake*. [69]

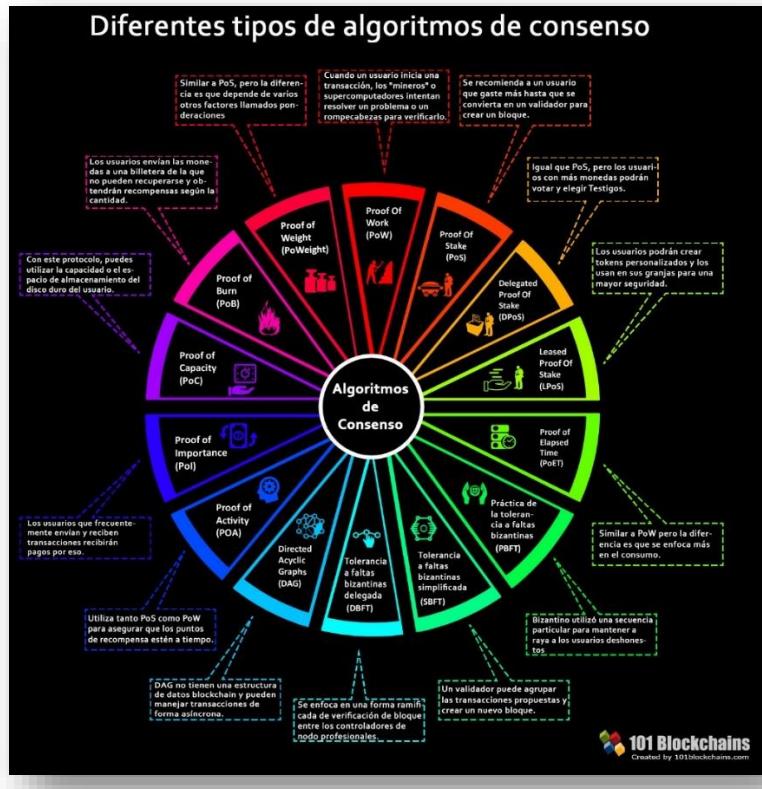


Ilustración 32: Diferentes tipos de algoritmos de consenso [69]

De todos los algoritmos de consenso señalados destacan por su popularidad en *Blockchain*: *Proof of Work* y *Proof of Stake*.

Proof of Work, *PoW*, es una forma de definir el cálculo de una máquina que se realiza al crear un conjunto de transacciones de forma distribuida, confiable y segura en *Blockchain*. Los mineros son los encargados de verificarlo y el primer minero que lo resuelve el algoritmo matemático se le recompensa con criptomonedas. El algoritmo se resolver con fuerza bruta, y por ello requiere una gran capacidad de procesamiento, pues se basa en intentarlo una y otra vez hasta acertar. Cuando se verifican las transacciones, se almacenan en la *Blockchain*, y ese bloque se une al resto de bloques con los métodos criptográficos.

Proof of Work es utilizado por Bitcoin para validar las transacciones y añadir nuevos bloques a la cadena, es decir, minar. En *Bitcoin*, el tiempo aproximado de cálculo de *PoW*

es de diez minutos para cada bloque, por lo que si se modificara, por ejemplo, el bloque tres, tardaría diez minutos cada uno de los bloques posteriores al tres, e incluido este. Este mecanismo produce un efecto muy difícil de manipulación.

Proof of Stake, PoS, requiere que el usuario muestre la propiedad de un cierto número de unidades de criptomoneda. Los participantes se llaman *forgers* (forjadores) en vez de mineros, a los que se encargan de crear el nuevo bloque y validar las transacciones. La forma de elección se basa en la cantidad de participación (*stake*), es decir dependiendo de su riqueza. A diferencia de *PoW*, se refiere a los bloques como forjados en vez de minados. Los *forgers* perciben comisiones por cada transacción. El requisito para ser *forgers* se basa en depositar sus criptomonedas como participación, que en caso de detectar algún fraude pierde sus criptomonedas y puede volver a ser nodo validador. *PoS* es el sistema actualmente utilizado en *Ethereum*.

En resumen, *PoS* se basa en que cuanto mayor número de criptomonedas tengas, mayor probabilidad de recibir recompensa. Sería análogo al caso de un depósito a plazo fijo, en el cual, en función de tu cantidad de dinero, percibes más o menos intereses.

5.4 VERSIONES DE BLOCKCHAIN

5.4.1 CRIPTOMONEDA

La criptomoneda es la *Blockchain* 1.0. Es una forma de intercambio de información digital que usa la criptografía para verificar y controlar las transacciones realizadas. Surgió a raíz de la puesta en marcha de *DLT, Distributed Ledger Technology*. Dentro de las criptomonedas caben destacar las monedas virtuales y los tokens.

5.4.1.1 Monedas Virtuales

Son monedas que no tienen protección del gobierno ni de ningún banco central. El primer ejemplo de las monedas virtuales es *Bitcoin*, que surgió en 2009 por el seudónimo Satoshi Nakamoto. *Bitcoin* se basa en la tecnología *Peer-to-Peer*, y se verifica la

información a través de la red. Las transacciones se realizan sin ningún tercero, puesto que la información se valida a través de algoritmos de consenso. En la actualidad es la principal criptomoneda del mundo.

Actualmente, la emisión de *Bitcoins* y la gestión de las transacciones se llevan a cabo de forma común en la red. Es la criptomoneda dominante del mundo. Se puede comprar *Bitcoin* a través de otras monedas como euros. Se reserva la privacidad al realizar las transacciones. *Bitcoin* se considera como un instrumento financiero volátil de cara a inversión a muy corto plazo, ya que se producen fuertes fluctuaciones de la moneda durante cortos períodos de tiempo. [70]

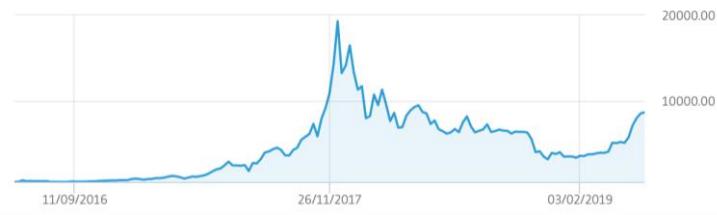


Ilustración 33: Valor Bitcoin durante los últimos tres años [70]

Después de *Bitcoin*, han surgido *Altcoins* o también llamadas monedas alternativas, que son una bifurcación de *Bitcoin*. La utilidad de los *Altcoins* es variada y cada uno tiene un propósito diferente. Cuando surgen *Altcoins*, intentan mejorar la tecnología de *Bitcoin*. Un ejemplo es *Litecoin*, que surgió en 2011 tras reducir el tiempo para validar la información de diez minutos de *Bitcoin* a dos y medio minutos. Actualmente la liquidez de los *Altcoins* es pésima respecto a *Bitcoin*. [71]

5.4.1.2 Token

La palabra *token* hace referencia y representa a un activo. El proceso de creación de *tokens* es más sencillo, puesto que no debe seguir un protocolo. La creación de *tokens* se realiza a través de los *Smart Contracts*. La mayoría de los *tokens* se basan en la plataforma de *Ethereum*.

Los *tokens* tienen diferentes usos, y cada entidad que lo programa puede darle un uso diferente de incentivación al cliente o inversor. También, los *tokens* se facilitan a los usuarios

con el fin de interactuar con sus productos. [72] Los *tokens* se crean y difunden por medio de una ICO, donde los inversores colocan su dinero, se crean diversos *Smart Contracts* con el enfoque empresarial encomendado, y los inversores reciben *tokens* esperando un crecimiento potencial del *token* a lo largo de un tiempo. Es decir, es una promesa a futuro.

Hay *tokens* que simbolizan la posesión de parte de la organización, es similar a tener acciones de la empresa, aunque de un modo mucho más sencillo, puesto que las votaciones de los “accionistas” se pueden llevar a cabo de forma transparente y rápida. [73]

5.4.2 SMART CONTRACTS

Smart Contract es código de programación donde se indica que debe hacer si se cumplen o incumplen ciertas condiciones, es decir, es una cuenta controlada por código. Se conocen como *Blockchain* 2.0. El lenguaje de programación suele ser *Solidity*. Tiene diferentes fines y entre ello está la creación de *tokens*, siendo parte del *backend* de una *Dapp*.

Los *Smart Contracts*, se crearon de la mano de Nick Szabo, en 1994, aunque hasta que no se desarrolló la criptomoneda *Bitcoin*, no empezaron a resurgir, pues la tecnología *Blockchain* posibilitó incluirlos dentro de ella. [74]

Dentro de un *Smart Contract* destacan las siguientes características:

- **Saldo.** Del mismo modo que cuando se tiene una cuenta en *Metamask* con saldo, el *Smart Contract* registra la cantidad de *Ether* de su cuenta.
- **Almacenamiento.** Se pueden guardar datos relacionados con el *Smart Contract*. Se suelen almacenar información relevante para la *Dapp* como números, cadenas o matrices.
- **Código:** contiene todo el código máquina del *Smart Contract*, puesto que el código escrito en *Solidity* se compila hasta llegar al código máquina.

A diferencia de una cuenta externa, es válida para cualquier red, ya sea la red principal de *Ethereum*, *Ropsten*, *Kovan*, *Rinkeby* u otras. Es decir, mantiene la misma clave privada y pública para cualquier red. En el caso de los *Smart Contracts*, la cuenta del contrato se crea para una red específica. Si tengo un *Smart Contract* en *Rinkeby* y quiero acceder a la red principal de *Ethereum*, tengo que crear una cuenta nueva para esta red y desarrollarlo de nuevo, puesto que no hay comunicación entre redes.

Cuando se programa en el ordenador, el código que se programa es código fuente del *Smart Contract* es el código auténtico que indica cómo comportarse el contrato y cómo manejarlo. Al desplegar el código del *Smart Contract*, se crea una instancia del *Smart Contract*, es decir, la cuenta del *Smart Contract*. A través de un código fuente de *Smart Contract* podemos desplegarlo tantas veces como queramos en una o varias redes. Esta relación se asemeja en *Java* a la relación clase (código fuente del *Smart Contract*) con objeto (instancia del contrato). Por ejemplo, para tener un taxi con una matrícula determinada, se necesita crear una instancia de la clase Taxi. Este taxi es similar a la cuenta del *Smart Contract*, que necesita estar en una red concreta de *Ethereum*.

En el caso de vender un inmueble, lleva un proceso con gran documentación, comunicación con ambas partes, y una tercera parte para aportar seguridad a la venta como son las figuras de un agente inmobiliario y un notario. Esta tercera parte implica un gran coste de la operación. En este caso, el *Smart Contract* soluciona el problema, pues únicamente transmitirá los fondos, *Ether*, al comprador si se han cumplido las condiciones impuestas, sirviéndole como garantía al vendedor. Al mismo tiempo, se almacena en *Blockchain*, en los nodos correspondientes, el derecho de posesión, por lo que ambas partes pueden visualizarlo cuando deseen. El coste de ejecución del *Smart Contract* es mucho menor que disponer de un intermediario, incluso aunque la plataforma tuviera cierta comisión. Además, permite que las negociadores se entiendan perfectamente, pues el lenguaje de programación es más preciso en la mayoría de ocasiones que el lenguaje natural. También, al ser un lenguaje abierto, cuantas más personas lo revisen podrán detectar fraudes o realizar mejoras con un consenso.

5.4.3 DAPP

Una *Dapp* es una aplicación descentralizada que depende de los usuarios que la usan. Se reconoce como la Blockchain 3.0. La Dapp utiliza los *Smart Contracts* para interreactuar con *Blockchain*.

La aplicación web utiliza como *frontend*: html, css y javascript. La página web en la parte de *backend* utiliza una API para conectarse a la base de datos. En comparación, la Dapp utiliza el mismo *frontend*, aunque utiliza *Smart Contracts* para conectarse a *Blockchain*. La Dapp puede ser una página web o una aplicación para el smartphone.

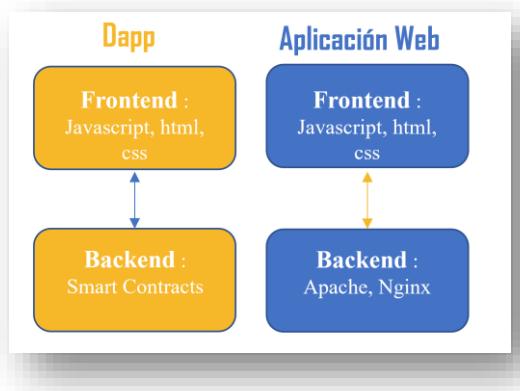


Ilustración 34: Comparación App y Dapp

En función de si utilizan la *Blockchain* de otra Dapp o si tienen su propia *Blockchain*, podemos distinguir tres clases de *Dapps*:

- **Dapp tipo I.** Poseen su propia *Blockchain* independiente. A este grupo pertenecen por ejemplo *Ethereum*, *LiteCoin* y *Dash*. Se pondría comparar con el sistema operativo como *Windows*, *Mac Os* o *Linux*.
- **Dapp tipo II.** Usan la *Blockchain* de una *Dapp* tipo I, no poseen una *Blockchain* independiente. Pueden tener *tokens* o los *tokens* de la *Blockchain* que utilizan.

Un ejemplo es *Raiden Network*. Se basa en la *Blockchain* de *Ethereum*, se encarga de reducir el tiempo de los intercambios de dinero y disminuyendo el coste basándose en no tener que introducir en cada transferencia toda la *Blockchain* de *Ethereum*. Un

símil a este tipo de *Dapp* sería el programa *Word* o *Excel*, que corren sobre el sistema operativo.

- **Dapp tipo III.** Se basan en el protocolo de la *Dapp* tipo II. Pueden tener *tokens* o los *tokens* de la *Blockchain* que utilizan las *Dapp* de tipo II. Respecto a *Raiden Network*, la *Dapp* tipo III es *μRaiden*. Posibilita realizar pequeños pagos a través de pasarelas de pago en una única dirección. La semejanza en esta cuestión es un plugin de *Excel* o *Word*. [75]

En la actualidad existen una cantidad enorme de *Dapps*, en el portal *State of the Dapps*. Se encuentras organizadas por plataforma (*Ethereum*, *EOs*, *GoChain*, *POA*, *Steem*, *xDai* y *Loom*), por categoría/temática y por el estado de la aplicación, ya sea en fase correcta, en riesgo, beta, de prototipo... [76]

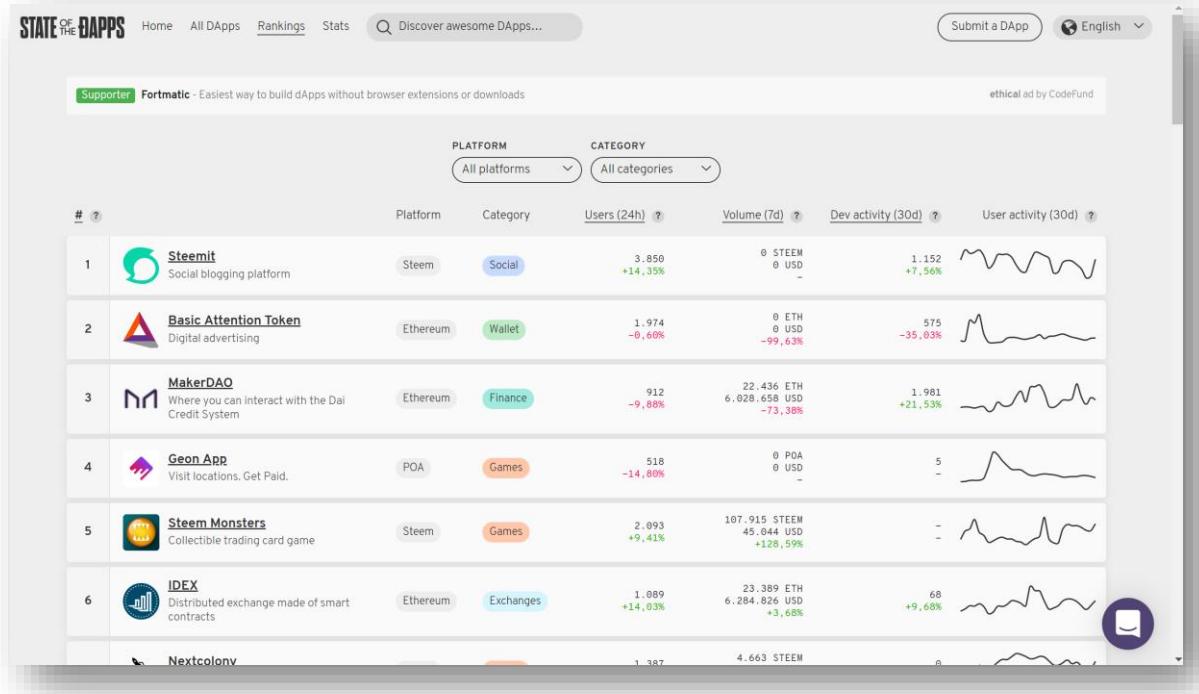


Ilustración 35: Conjunto de Dapps [76]

Cabe destacar la exitosa *Dapp* que congestionó por momentos la red *Ethereum*, *Crypto Kitties*, fue de los primeros juegos en *Blockchain*. Balaji S. Srinivasan, como maestro

en *Blockchain*, lo consideró un gran avance dado que se trata de “comercio internacional de activos digitales (¡no sólo dinero en efectivo!) en Blockchain.” . [75]

5.5 ICO

Crowdfunding es un método de financiación colaborativo de un proyecto basado en recolectar pequeñas cantidades de dinero a través de Internet. La idea surgió a través de pequeñas donaciones de los proyectos *Open Source*. Un caso de uso de *crowdfunding* es una *ICO*, *Initial Coin Offering* para financiar el desarrollo de nuevos protocolos

En una *ICO*, la entidad emite *tokens* sobre *Blockchain* a cambio de dinero. La empresa intercambia los *token* con sus inversores a cambio de criptomonedas con gran liquidez. Este proceso se lleva a través de *Smart Contracts*, que automatizan el proceso cuando se cumplen las cantidades solicitadas entregando los *tokens* a los inversionistas. La plataforma más utilizada es la de *Ethereum* por poder realizar *token* ERC20. ERC20 asegura el intercambio de *token* si se realizan bajo el mismo estándar, es decir usar una librería de funciones que reducen la probabilidad de error. Esto permite también poder interactuar en una *Dapp*.

Actualmente, las *ICO* se están realizando en varias fases. Lo primero de todo es conceptualizar la *ICO*, es decir organizar la documentación necesaria, establecer fechas, indicar las etapas a realizar, las intenciones y objetivos del problema a resolver. Esta documentación permite poder convencer a los futuros inversionistas. Después, es la ronda pre-*ICO*, este periodo suele ser entorno a tres o cuatro meses, en esta fase privada se intenta obtener el máximo dinero posible de los inversionistas. Más tarde, es el momento de preventa, donde se debe llamar la atención de forma pública y potenciar la inversión a través de la incentivación de beneficios para aquellos que también participen en la segunda fase, disminuyendo según se acerca la fecha de la *ICO*. La *ICO* dura cerca de tres meses y es el momento de recolectar el dinero. [77]

Durante los años 2017 y 2018, según *BitMEX*, las nuevas *cripto Startups* consiguieron más de 24 mil millones de dólares. Este hecho garantiza el notable aumento de este método de financiación. [78]

5.6 VENTAJAS DE BLOCKCHAIN

Antes de realizar una implantación de *Blockchain* en cualquier organización, se suelen preguntar cuáles son los beneficios que le aportaría a esa actividad. A continuación, se muestran las principales ventajas de esta tecnología:

- **Transparencia.** *Blockchain* posibilita ser auditado en cualquier momento por los miembros de *Blockchain*. En el caso de una *Blockchain* pública, es visible para todo el mundo, permitiéndose observar la cadena de origen a fin, rehuyendo de ocultar información. A mayores, el código de *Blockchain* es abierto, cualquier usuario con el consenso del resto de la comunidad puede mejorarlo. Esta situación supone que al ser observado el código por diferentes personas, es muy difícil pasar por alto alteraciones en la red, ayudando a crecer también la confianza.
- **Inmutabilidad.** Las transacciones no se pueden modificar, ya que cuando se agrega un bloque a la cadena, no se puede ni eliminar ni modificar. Este hecho se apoya en los métodos criptográficos utilizados durante el almacenamiento: hash y firma digital. Además, las transacciones se almacenan cronológicamente a través de un *timestamp*.
- **Sencillez.** Los clientes deben registrarse continuamente en distintas plataformas, en cambio en el caso de la utilización de la *Dapp*, únicamente requieren registrarse una vez, puesto que al utilizar una *Wallet*, permiten vincularlo a la *Dapp* y tener un único registro para cualquier *Dapp*. Además, la *Wallet* les permite almacenar las claves públicas o privadas, memorizándose únicamente cadenas sencillas.

- **Control del usuario.** Para realizar los cambios, los desarrolladores deben llegar a un consenso y en sus manos está decidir por la red, tal como sucedió en el caso de *Bitcoin* y *Bitcoin Cash*, al no llegar a un consenso con un porcentaje considerable, surgió *Bitcoin Cash* para ofrecer un método más rápido de pago . [79]
- **Descentralización y colaboración.** El principal problema reside en la falta de confianza al realizar una transacción y es por ello que se ha venido utilizando un tercero para realizar intercambio de dinero o información. En cambio, en *Blockchain* no es necesario puesto que se ha creado para que exista confianza entre ambas partes. Cada nodo de la red intercambia la información y cada nodo posee una copia de toda la información. Los mineros o forjadores validan y verifican las transacciones con autonomía sin necesidad de una entidad que lo verifique
- **Reducción del coste de transacción.** A la hora de realizar liquidaciones, intercambio de dinero o pagos, ya sea entre organizaciones o entre amigos, es la forma más rápida y está disponible 24/7 en vez de únicamente cinco días a la semana en horario laboral. Y suponiendo menor coste que en la actualidad, puesto que el gasto de *Gas* es muy bajo, lo que supone que prácticamente no exista comisión.
- **Fiabilidad:** *Blockchain* se encarga de controlar y verificar las identidades para evitar suplantaciones de identidad y registros duplicados. Además, se está avanzado en este asunto a través de la tecnología *DLT*, *Distributed Ledger Technology* junto a *Blockchain*. [80]
- **Seguridad.** *Blockchain* se basa en que cada nodo tenga la misma información. Si existe un ataque, es más difícil derribarlo, puesto que se encuentra almacenada en varios nodos, a diferencia del sistema tradicional de almacenamiento de bases de datos, en el que la información se encuentra en una zona concreta. Aunque tenga alguna copia de seguridad, no posee tal nivel de replicación.

- **Automatización.** Los *Smart Contracts* posibilitan optimizar los procesos al no tener que estar atento de ciertas transacciones. Por ejemplo, los *Smart Contracts* permiten activar la transacción al cumplir determinados requisitos.
- **Prevención de fraude.** En el momento que se involucra a mucha personas, aumenta el riesgo de suceder un fraude. En estas situaciones, *Blockchain* es especialmente útil puesto al ser información compartida entre varios nodos y validada, a través de los algoritmos de consenso, se evitan pérdidas factibles a causa de estafa.

5.7 LIMITACIONES DE LA TECNOLOGÍA BLOCKCHAIN

A pesar de todos las ventajas que posee la tecnología *Blockchain* y ayuda a la sociedad, se deben señalar que “no todo es oro todo lo que reluce”. Por eso a continuación, se muestran tramas acerca de la tecnología:

- **Volatilidad.** Al tratarse de una tecnología muy reciente y aún sin asentar, se refleja en la fluctuación del precio de las criptomonedas. Esta situación provoca detractores acerca de si estamos ante la explosión de una burbuja o si es el rival del oro.
- **Riesgo de error.** Cuando el ser humano se implica en una actividad, surge cierta probabilidad de error. En *Blockchain*, no se puede modificar la información. Por ejemplo, a la hora de revertir una operación, se origina el coste de otra transacción. Esto puede ser especialmente costoso en ocasiones en las que la operativa humana cause numerosos errores.
- **Casos de uso.** No es aplicable a todos los negocios, por lo que, antes de intentar aplicar la tecnología, se debe estudiar detenidamente si interesa ser aplicada.
- **Mercado ilegal.** Al tratarse de transacciones anónimas, se han convertido en un mercado para transacciones ilícitas

- **Personas mayores.** El público al que está dirigido actualmente la tecnología *Blockchain* requiere de ciertos conocimientos técnicos que muchas personas mayores no poseen.

Capítulo 6. ANÁLISIS DEL SISTEMA

Se realiza un análisis íntegro del sistema a realizar. Este estudio sirve para poder examinar y averiguar en detalle las carencias vigentes para realizar correctamente la implantación tal como se ha detallado anteriormente.

6.1 CASOS DE USO

En primer lugar, se comienza el análisis del sistema a través de la determinación del comportamiento del usuario frente a la plataforma. Esto sirve para poder clarificar la utilización de la plataforma.

6.1.1 CASO DE USO COMÚN

Para poder acceder al sistema, su precondición será que cualquier usuario deberá identificarse como vendedor, comprador, prestamista o aseguradora. A cualquier usuario se le mostrará la cantidad de dinero que tiene en su cuenta, así como su número de cuenta. Además, todos los usuarios podrán consultar los anuncios publicados en la plataforma.

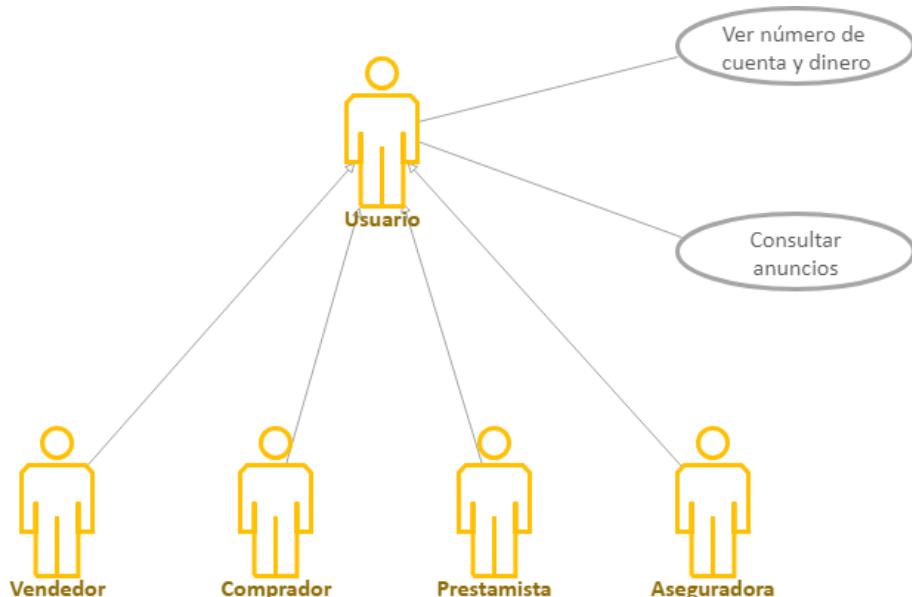


Ilustración 36: Caso de uso común

6.1.2 CASO DE USO VENDEDOR

En el caso del usuario identificado como vendedor, podrá crear anuncios de venta o de alquiler, que serán aquellos que se publicarán en la plataforma. También, se le posibilita consultar sus propios anuncios, navegando por la aplicación, y podrá llegar a cancelarlos. Sus transacciones quedarán firmadas por su cuenta ante las distintas situaciones que se puean dar, para lo que se requiere su identificación.

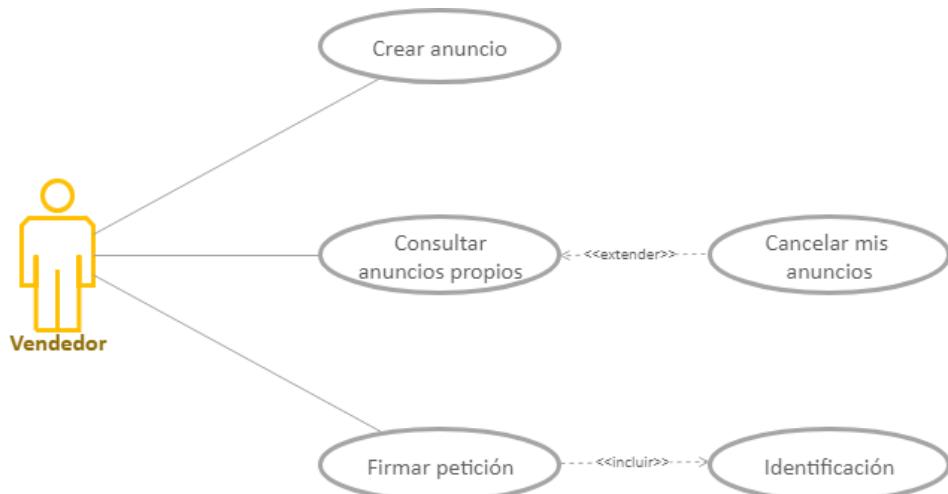


Ilustración 37: Caso de uso vendedor

6.1.3 CASO DE USO COMPRADOR

El usuario identificado como comprador podrá realizar la compra de un anuncio, para ello necesitará verificar su identidad, y tendrá que firmar su transacción. Tendrá la posibilidad de realizar la contratación de la póliza de seguro ofertada por algunas de las aseguradoras que están dadas de alta en el sistema, así como poder publicar la necesidad de un préstamo con sus necesidades, para que los prestamistas decidan aceptarlo o no. Además, al acabar el proceso de compra, deberá incluir el código generado por la plataforma para poder abrir la puerta del inmueble adquirido, para que pueda darse por concluida la compra del inmueble de uso compartido.

En el caso que se desee realizar el alquiler del anuncio, el usuario que se ha registrado como comprador tendrá unas funcionalidades similares a las de la compra de un anuncio, aunque no dispondrá de algunas de ellas. En primer lugar, el usuario deberá identificarse en el sistema, y tendrá que firmar las peticiones que se le presenten en el flujo. Por último, deberá introducir el código generado para poder abrir la puerta del inmueble que se ha alquilado. De esta forma, finalizará el proceso de alquiler del inmueble.

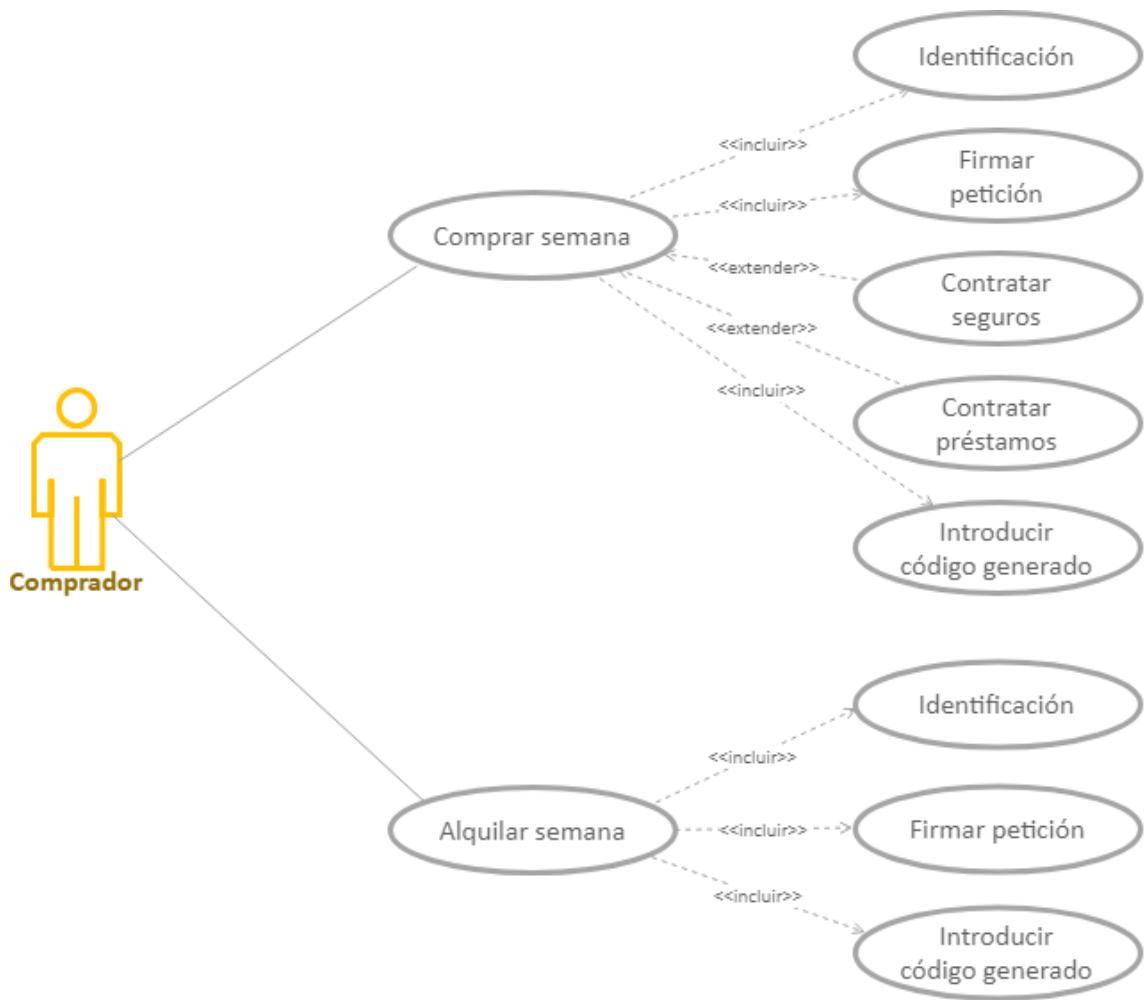


Ilustración 38: Caso de uso comprador

6.1.4 CASO DE USO PRESTAMISTA

El sistema posibilita al actor prestamista la visualización de las peticiones realizadas por los usuarios compradores. Para ello, será necesario que el prestamista se identifique y quede verificada su identidad. Además, el prestamista podrá generar ofertas de préstamos, previamente identificándose en el sistema. La oferta generada tendrá que ser firmada por el usuario prestamista.

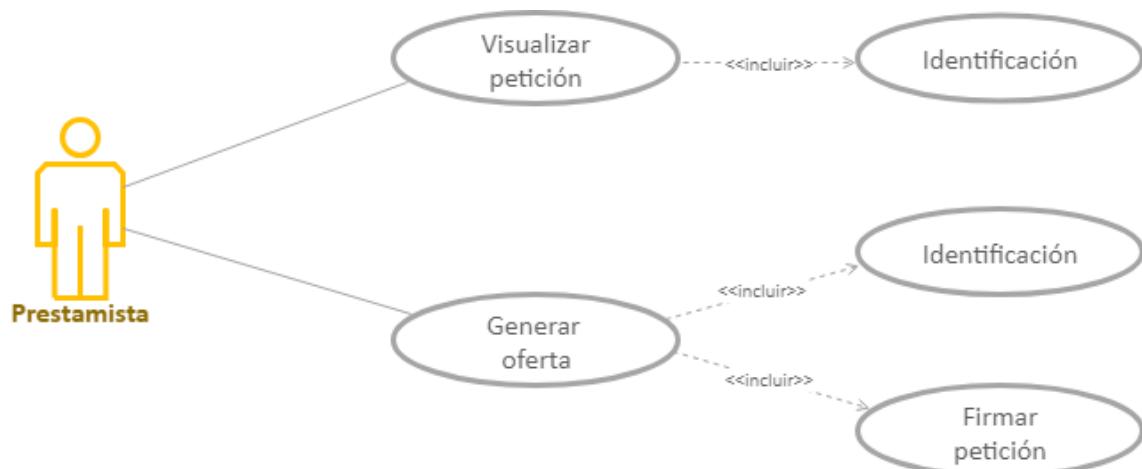


Ilustración 39: Caso de uso prestamista

6.1.5 CASO DE USO ASEGUADORADA

La aseguradora tiene la posibilidad de publicar su oferta de póliza de seguro para los anuncios publicados que se encuentran en venta, siempre que se haya identificado y se haya verificado su identidad. Si algún usuario comprador decide aceptar alguna de sus ofertas de seguro, deberá firmar el acuerdo del seguro para que se dé por finalizado el proceso.

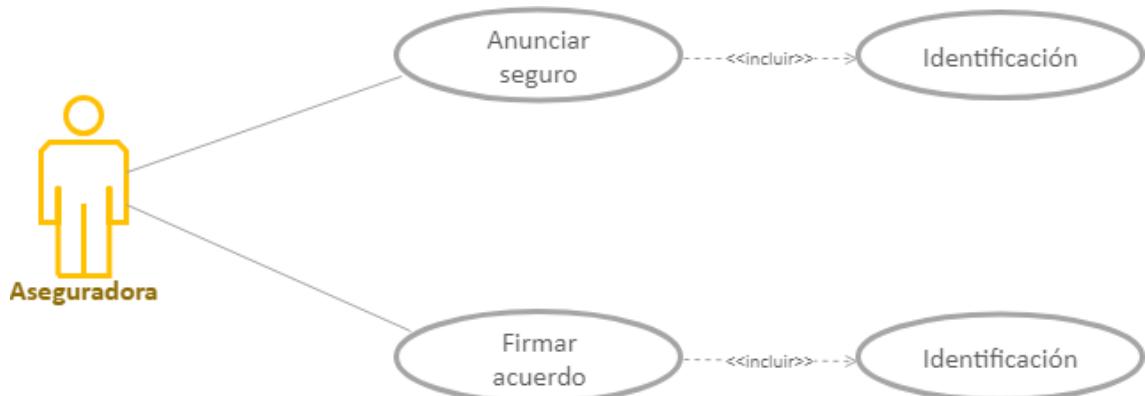


Ilustración 40: Caso de uso aseguradora

6.2 DIAGRAMAS DE SECUENCIA

En la sección presente se escenifica las interacciones del sistema con el comportamiento de las entidades de los casos de uso mencionados previamente.

6.2.1 DIAGRAMA DE SECUENCIA COMPRA

En el caso de realizar un anuncio de compra, en el sistema pueden actuar todos los actores: vendedor, comprador, prestamista y asegurador.

Para empezar, el asegurador deberá autenticarse a través del controlador que se lo retransmite al *backend*. Tras ello, se verifica su identidad a través del DNI. A partir de ahí, el asegurador puede realizar su oferta de condiciones de póliza de seguro que se almacena a través del controlador, que a su vez se comunica con el *backend*.

Por otro lado, el vendedor debe autenticarse a través del controlador que interactuará con el *backend*. Además, el comprador podrá visualizar los anuncios sin necesidad de identificación. En el momento que desee interactuar con la plataforma, deberá identificarse con su DNI. En el proceso de compra, deberá firmar la petición. Si la petición es rechazada, se informará al comprador y se cancelará la petición. Si la firma es correcta, se le mostrará

el estado correcto al vendedor, quien realizará su proceso de firmado. También, en caso de ser inválida, se cancelará la operación y se le mostrará el aviso.

En caso de aceptarse la petición por ambas partes, se muestra la opción del seguro, y el comprador podrá aceptar o rechazar la oferta del asegurador. En caso de aceptación, esta petición deberá firmarse por ambas partes. Si sucede la falta de verificación por alguna parte, se cancela la operación y se muestra cancelada. Dentro de las posibilidades de creación de una oferta de seguro, la aseguradora tiene la opción de realizar un *scoring* sobre el cliente en potencia. A partir de la respuesta que obtenga del *scoring*, podrá firmar o no el contrato.

El comprador tiene la opción de publicar la necesidad de préstamo con sus condiciones, y su situación podrá ser visualizada por el prestamista, quien decidirá aceptarla o no. En caso de aceptarla, podrá realizar un rating adicional por su parte, y el comprador le entregará la documentación necesaria. Finalmente, se producirá la firma por ambas partes, mientras que, en caso de no ser verificada, se mostrará el aviso de rechazo. En caso positivo de firmar el acuerdo por ambas partes, se procede al proceso de compra.

En el proceso de compra, ambos deberán firmar la petición; comprador y vendedor. La petición es validada por el *backend*, que se encarga de detectar errores. Si hubiera errores, se les muestra y se cancela la operación.

Si se produce un proceso satisfactorio, se les muestra a comprador y vendedor. Por consecuencia, se genera un código que se envía al comprador con el fin de que pueda abrir la vivienda. El comprador lo tiene que introducir en la cerradura de la *smartdoor* del inmueble, que se comunicará con el *backend* para comprobar su validez. Además, al producirse la compra del anuncio, se entregan los fondos al vendedor.

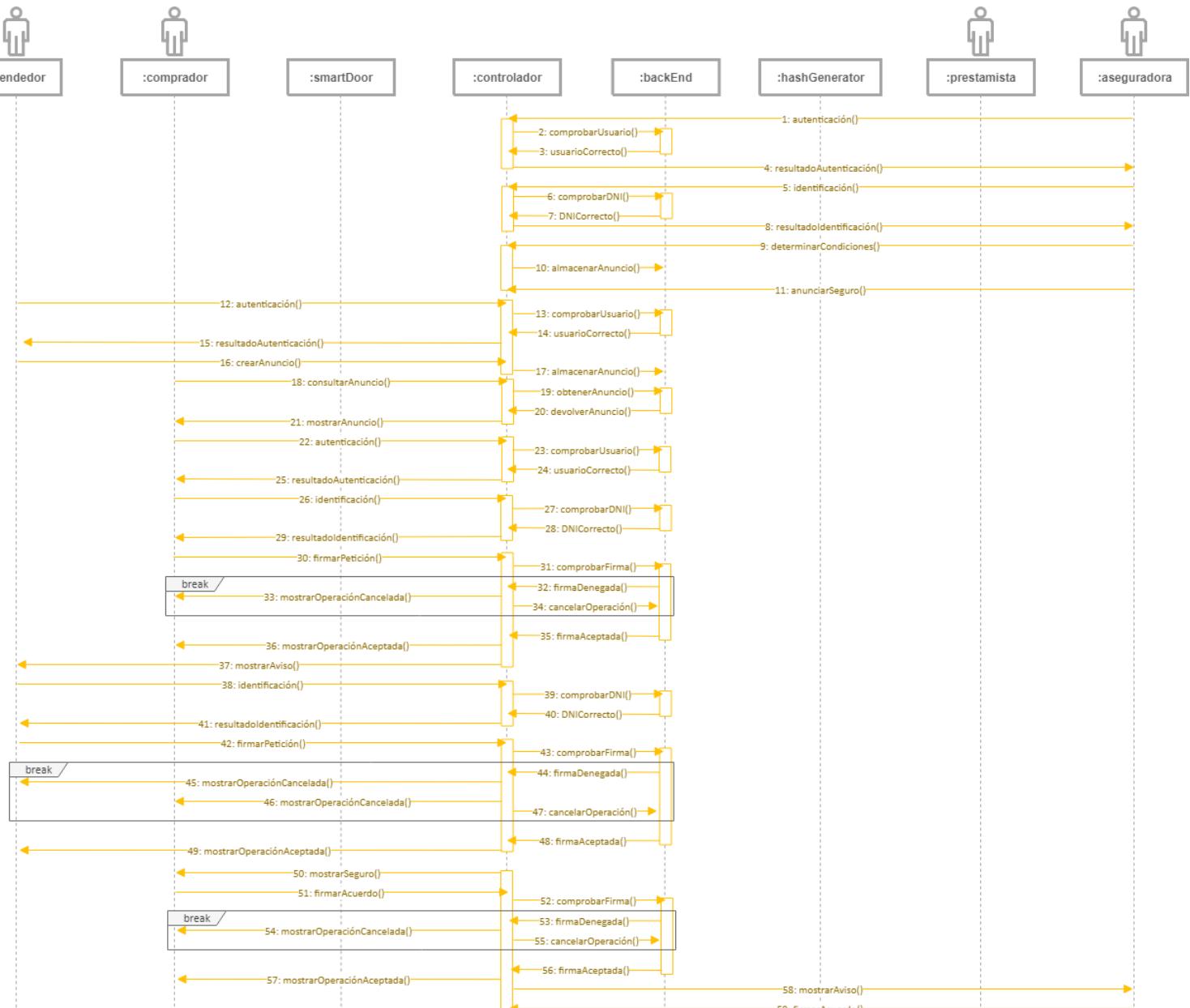


Ilustración 41: Diagrama de secuencia compra (I)

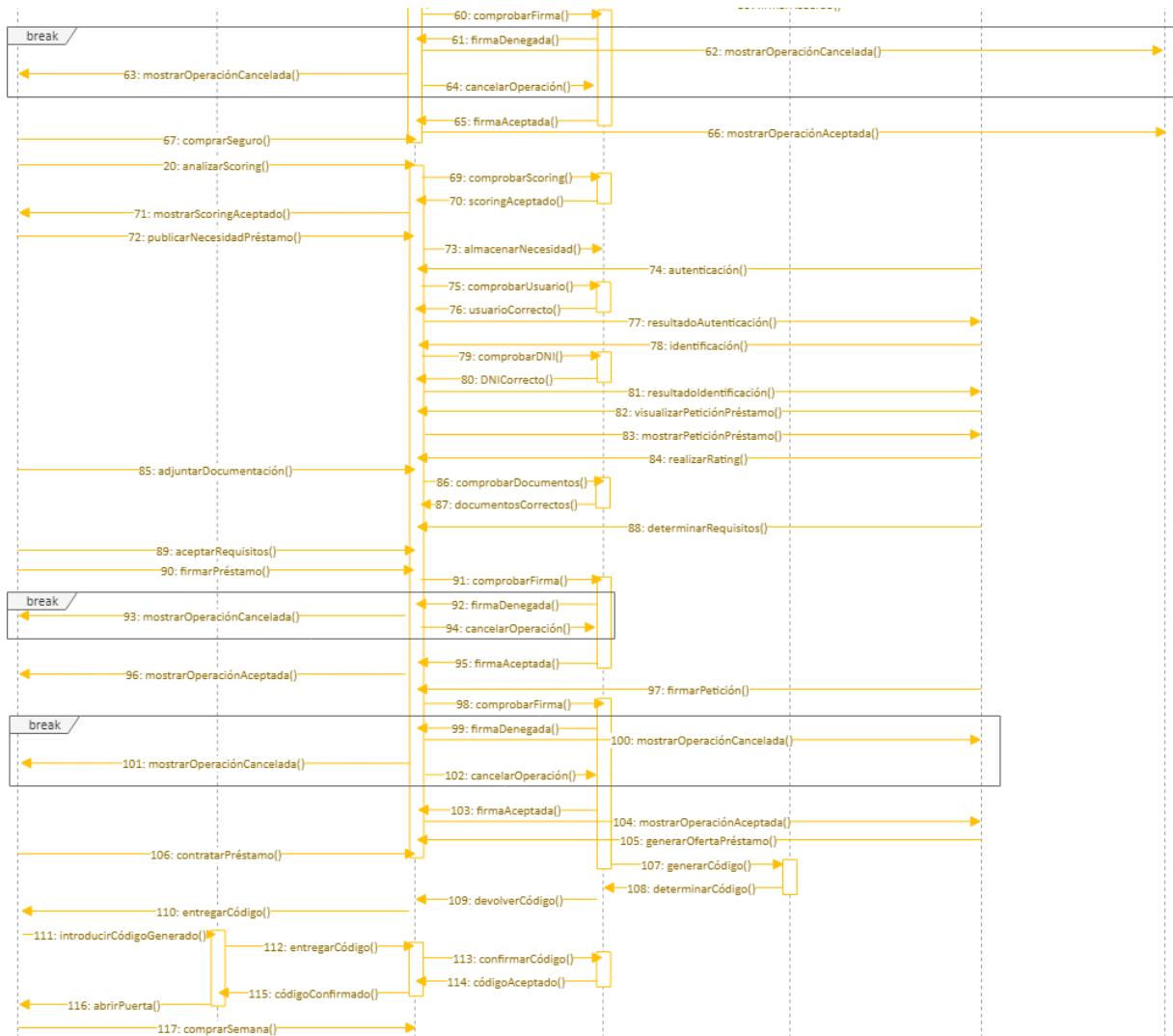


Ilustración 42: Diagrama de secuencia compra (II)

6.2.2 DIAGRAMA DE SECUENCIA ALQUILER

En el caso de realizar un anuncio de alquiler, el sistema se simplifica, eliminando los actores prestamista y asegurador.

En primer lugar el vendedor deberá identificarse, y tras una autenticación correcta podrá realizar la creación del anuncio. El sistema *backend* será el encargado de almacenar el anuncio.

Si el comprador desea consultar el anuncio, será a través del controlador, que se comunicará con el *backend* para mostrárselo. Este proceso será similar a la autenticación del comprador.

En el caso que desee realizar el alquiler, deberá firmar la petición, que será validada por el *backend*. Si el *backend* detecta algo incorrecto en la firma, se cancelará la operación y se muestra al comprador. El vendedor también debe firmar la petición, y si sucediera una verificación incorrecta, se cancelaría el proceso. En el caso que se acepte correctamente la compra, se le muestra el resultado satisfactorio a los dos actores involucrados; vendedor y comprador.

Tras realizar el hecho anterior, supone la generación de un código para que el comprador pueda realizar la apertura del inmueble, que lo deberá introducir en la cerradura de la *smartdoor* del inmueble que se comunicará con el *backend* para verificar que es correcto. De este modo, se producirá el alquiler de ese anuncio, y se entregarán los fondos al vendedor.

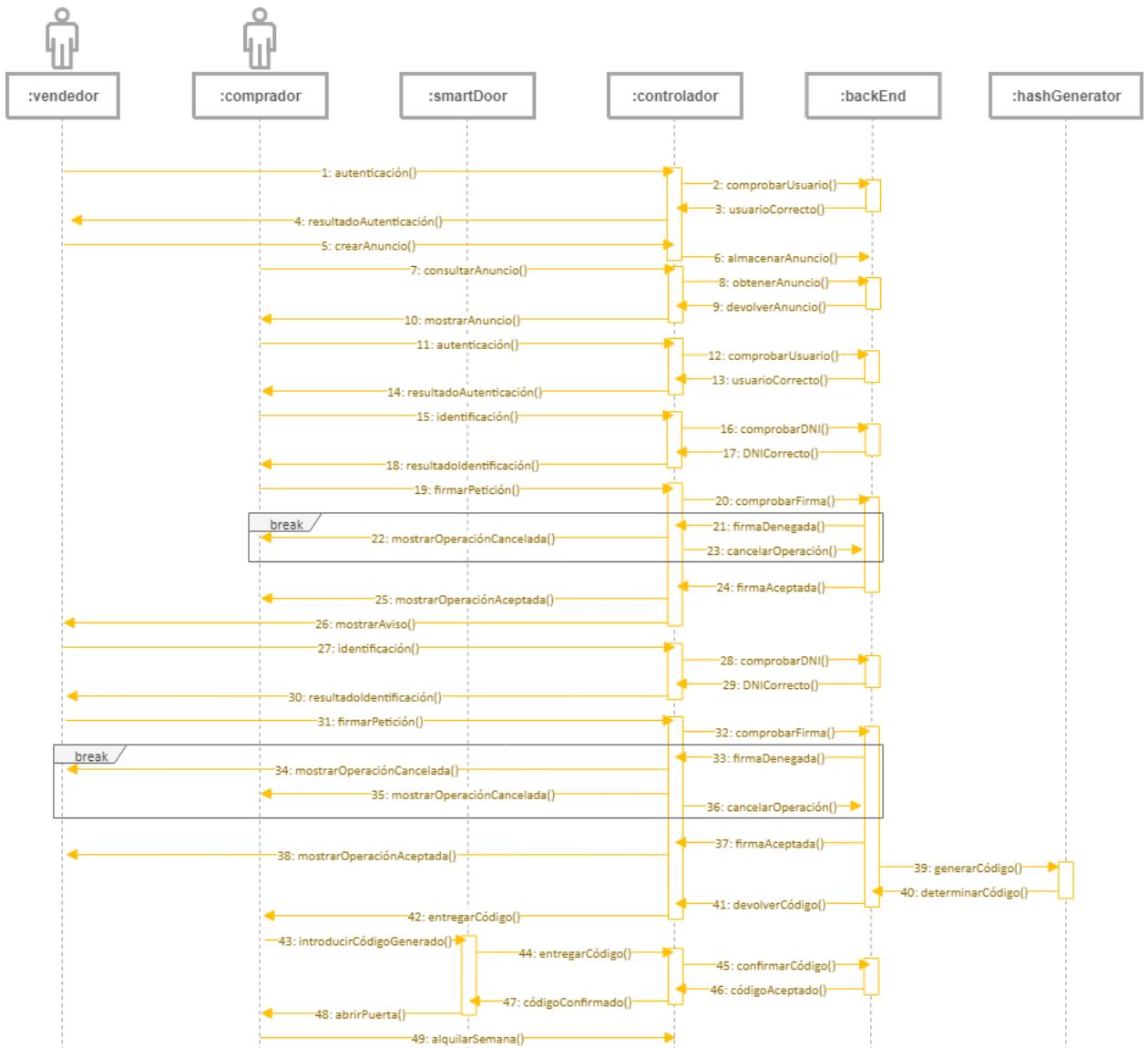


Ilustración 43: Diagrama de secuencia alquiler

6.3 DIAGRAMA DE ACTIVIDAD

En esta apartado se pretende visualizar el flujo de la plataforma, mostrando las actividades realizadas en el sistema para concretar las diferencias entre alquiler y compra de un inmueble.

6.3.1 DIAGRAMA DE ACTIVIDAD COMPRA

En el diagrama que se puede encontrar en la Ilustración 44, se implementa la actividad de compra de un inmueble. En primer lugar, se accede a la página web. Si el usuario no tiene cuenta, debe registrarse. Si el usuario tiene cuenta, debe autenticarse. En consecuencia, deberá identificarse con su DNI. A continuación, el comprador deberá poseer al menos el 20% de dinero para poder continuar con la operación . En caso , de disponer de la cantidad, se genera el contrato, siempre y cuando la firma del vendedor se produzca de forma correcta. Entonces, se realiza un depósito al vendedor del 10% del importe del inmueble y se registra la transacción en un bloque de *Blockchain*.

Por otra parte , el comprador tendrá la opción de firmar una póliza de seguro si la desea. En caso de que así lo señale, se realizará el contrato, que quedará registrado en *Blockchain*.

Si no desea realizarlo, se continúa con el proceso y se ofrece la posibilidad de publicar la petición de un préstamo. Si su petición es aceptada por un prestamista , este podrá realizar un *rating* , para lo cual el comprador podría tener que adjuntar documentación, y si cumple los requerimientos, se hace efectivo el préstamo.

En caso de que en algún punto anterior se incumpla la condición, se verificará si tiene el dinero suficiente . Si no dispone de la cantidad, se rechaza la operación y se devuelve el depósito del 10% al comprador y se registra en *Blockchain*.

Si el comprador ya tiene el dinero suficiente, el comprador obtiene el código de la *smartdoor*, y si en el mes siguiente el comprador no ha validado el código, se realiza una

transacción del 10% al vendedor y se registra. Esta situación será iterativa hasta que el código sea introducido en la cerradura.

Cuando introduce el comprador el código y no le ha funcionado, se rechaza la operación, cancelándose el contrato, y devolviendo al comprador el 10% del valor del inmueble, todo queda registrado en *Blockchain*. Si se produce un funcionamiento correcto, se realiza el pago de la cantidad restante al vendedor, hasta llegar al precio fijado al principio de la operación.

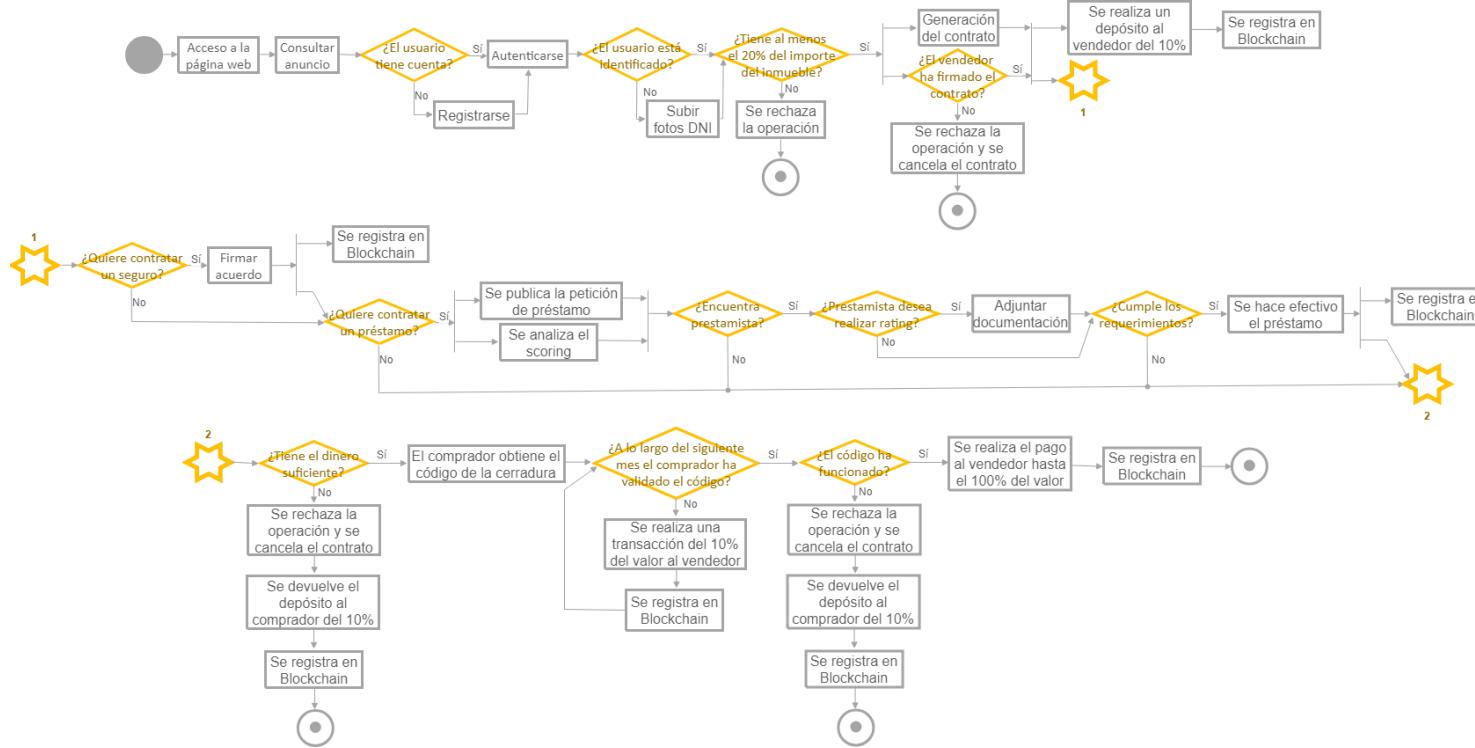


Ilustración 44: Diagrama de actividad compra

6.3.2 DIAGRAMA DE ACTIVIDAD ALQUILER

A continuación, se muestra la Ilustración 45, en la que se presenta el flujo a la hora de realizar la operación de alquiler. Esta operación supone una simplificación respecto al proceso de compra, puesto que desaparece la figura del prestamista y asegurador. Sin embargo, también incluye algunos cambios en los porcentajes de flujos de intercambio de divisas.

Al principio, se accede a la plataforma, y si el usuario no está registrado, debe hacerlo. En cambio si ya está, registrado tiene que autenticarse. En ambos casos tiene que identificarse con su DNI. Después, el arrendatario deberá poseer el 100% de dinero para poder continuar con la operación de alquiler. En caso de disponer de la cantidad, se genera el contrato, siempre y cuando la firma del arrendador se produzca de forma correcta. Entonces, se realiza un depósito al vendedor del 25% del importe del inmueble y se registra la transacción en un bloque de *Blockchain*.

A continuación, el arrendatario obtiene el código de la *smartdoor*, y si en el primer día del mes de alquiler el comprador no ha validado el código, se realiza una rechazo de la transacción. Se registra la cancelación en *Blockchain*.

En caso que haya introducido el código y no funcione correctamente, se rechaza la operación, y se devuelve el depósito al comprador del importe del 25% de la cantidad acordada de alquiler, registrándose en *Blockchain*. Por otro lado, si se realiza de forma correcta, finaliza el pago del alquiler al arrendador de la cantidad restante, es decir del 75%, y se registra en *Blockchain*.

ANÁLISIS DEL SISTEMA

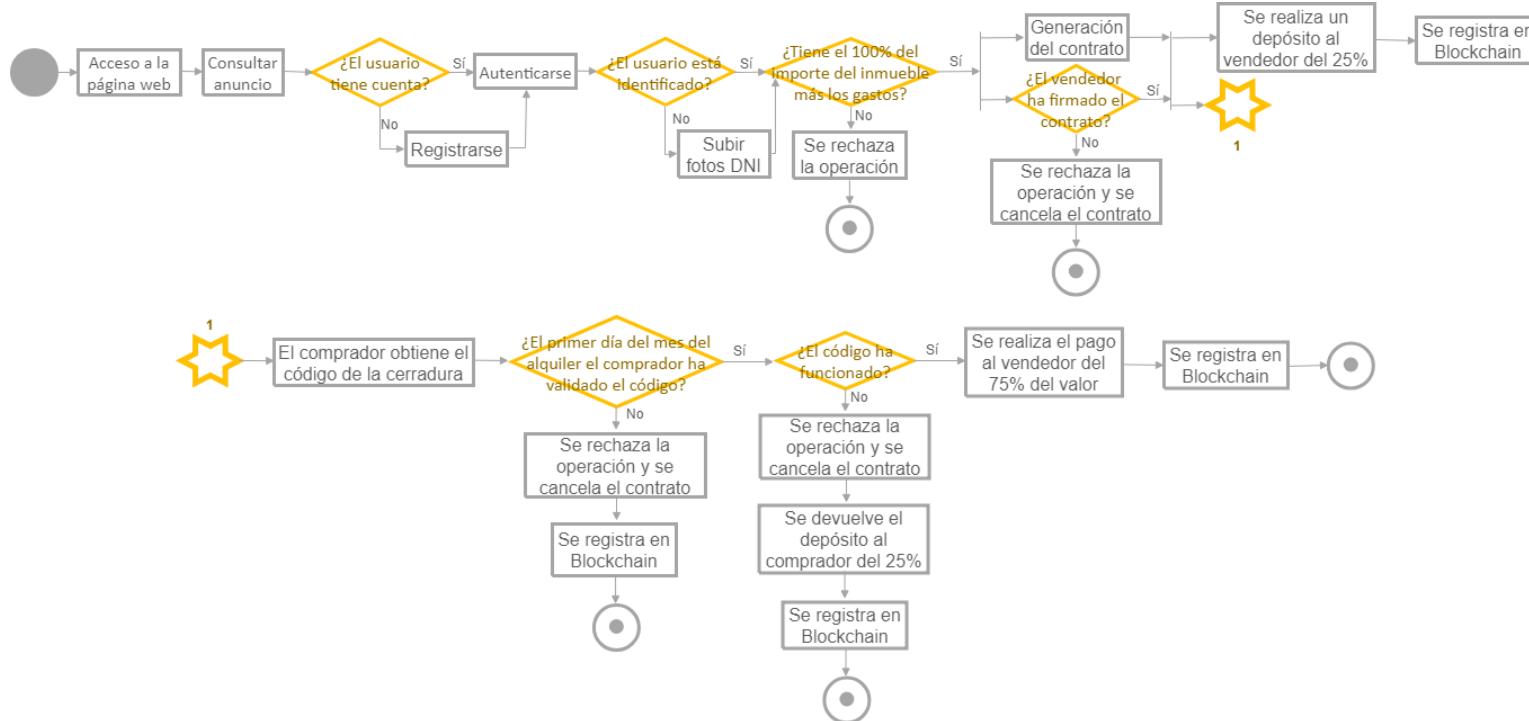


Ilustración 45: Diagrama de actividad alquiler

Capítulo 7. DISEÑO

Tras afrontar la etapa de análisis de caracterización de la plataforma, es el paso de proponer el diseño de cara a la implementación.

7.1 FUNCIONALIDADES COMUNES

A continuación, se detallan funcionalidades que comparten todos los usuarios que acceden al sistema:

- A través del plugin del navegador, *Metamask*, se realiza la identificación. Para ello, el usuario deberá ingresar la contraseña asociada a su cuenta de *Metamask*. Una vez conectado a *Metamask* deberá seleccionar la red sobre la que se encuentra desplegado la *Dapp*: red principal de *Ethereum*, red privada *Ropsten*, red de pruebas *Kovan*, red privada *Rinkeby*, red de pruebas *Goerli*, *localhost* o *RPC* personalizado. Tras seleccionar la red, elegirá la cuenta con la que interactuar en *Blockchain*. En caso de que no disponga de la cuenta con la que quiere trabajar, tiene la opción de crearse la cuenta, o importarla a través de su clave privada o a través de la conexión de un monedero físico. Este sistema aporta gran seguridad, puesto que no ha sufrido ataques hasta la fecha, y además cuenta con su cadena de palabras de recuperación de la cuenta. [81]
- Todos los usuarios, tras identificarse a través de *Metamask*, podrán visualizar en la plataforma su número de cuenta, que es válido para cualquiera de las redes de *Ethereum*, y el balance de divisas que posee, indicado en *Ethers* (ETH). Esta información se encontrará en la parte de arriba de la plataforma, justamente debajo del logo.
- El usuario deberá realizar la autenticación a través del plugin *Metamask*, y a mayores deberá autenticarse con su DNI, subiendo una foto de su documento a través de la red *IPFS*, sobre la que se enlazará el *hash* al usuario y se validará su autenticidad.

- Cualquier usuario, sin estar identificado, podrá visualizar la información publicada en el sistema. El usuario no podrá interactuar con ello sin previa identificación. Existirán las pestañas *Publicar*, *Anuncios*, *Préstamos*, *Seguros* e *Histórico*. Para hacer uso de la ventana de *Publicar*, necesitará encontrarse identificado son su cuenta previamente. Las tres siguientes pestañas podrá visualizarlas pero no interactuar con ellas. El registro histórico estará disponible aunque no se encuentre identificado. Además, el usuario siempre podrá acceder a la página de bienvenida.

7.2 FUNCIONALIDADES DE VENDEDOR

A continuación, se detallan la funcionalidad primordial del vendedor en la *Dapp*:

- El vendedor podrá crear su anuncio, bien sea de alquiler o venta. En el formulario que se encuentra en la pestaña *Publicar*, tendrá que llenar los parámetros de *Operación*, *Localización*, *Resumen del anuncio*, *Descripción del inmueble* y *Precio en Ethers (ETH)*. Antes de ser publicado en la red *Ethereum*, se comprueba que el precio de inmueble sea positivo, y que los campos se encuentren llenos. En caso de error, el usuario será informado para solucionar el problema antes de publicarlo. Al realizar la publicación en la red *Ethereum* de *Blockchain*, de la cuenta del vendedor se descontará el *Gas* de esa transacción. La transacción se realiza mediante el plugin *Metamask*, mostrando una nueva ventana donde se indica el precio de la transacción, que en el caso de una publicación será nulo, y el *Gas* que se descontará en caso de aceptar.

7.3 FUNCIONALIDAD DE COMPRADOR

Se muestra las funcionalidades que puede realizar el comprador:

- El comprador podrá realizar la compra o alquiler de un inmueble anunciado. Para ello, deberá acceder a la pestaña *Anuncios*, y pulsar sobre el botón *Comprar* que aparecerá debajo de cada uno de los anuncios que no se han publicado desde la misma

cuenta que ahora se visita. Se verificará que exista el anuncio para vender, que se disponga de dinero en la cuenta del comprador para realizar la compra o alquiler, que el ID del anuncio se encuentre dentro del rango de los anuncios publicados, que no tenga todavía comprador asignado el anuncio, y que el precio de Ether enviado coincida con la cantidad enviada. Además, el comprador no podrá realizar la compra de un inmueble publicado por él mismo porque no será visible el botón *Comprar* en estos casos. La transacción se realizará mediante el *plugin Metamask*, mostrando una nueva ventana donde se indica el precio de la transacción, que en este caso coincidirá con el importe indicado en el anuncio, y el *Gas* que se descontará en caso de aceptar.

- El comprador podrá solicitar un préstamo haciendo referencia al anuncio que desea adquirir, rellenando un formulario que existe en la pestaña *Publicar*. Los campos a llenar serán *ID Anuncio referencia*, *Condiciones del préstamo* y *Cantidad en Ethers (ETH)*. Para ello se comprobará que existe el ID del anuncio, y que aún está en venta. Se comprobará que el ID del anuncio que solicita no es un anuncio en alquiler. También se comprobará que el anuncio no se encuentra ya en el registro histórico. Además, se comprueba que la cantidad que se solicita es positiva. Por último, también se comprobará que el anuncio asociado para el que se pide el préstamo no ha sido publicado por él mismo. Para poder realizar dicha solicitud y dejarlo registrado en *Blockchain*, deberá pulsar el botón *Publicar*. La transacción se realizará mediante el *plugin Metamask*, mostrando una nueva ventana donde se indica el precio de la transacción, que en el caso de una publicación será nulo, y el *Gas* que se descontará en caso de aceptar.

- Al comprador se le posibilita ver los anuncios de seguros relacionados con el ID del anuncio en venta que desea adquirir. En el caso que desee adquirirlo, deberá dirigirse a la pestaña *Seguros* y, dentro de esta, seleccionar la oferta que se corresponde con el Id del anuncio que desea adquirir. Deberá disponer de dinero en la cuenta para aceptar la transacción y se verificará que el anuncio no tenga comprador. Con la aseguradora se producirá la firma de la aceptación del seguro, almacenándose la

transacción en un bloque en *Blockchain* y cargándose el gasto de *Gas* de la aceptación de la oferta al comprador.

7.4 FUNCIONALIDAD DE PRESTAMISTA

A continuación, se detallan la funcionalidad primordial del usuario prestamista en la *Dapp*:

- El usuario identificado como prestamista accederá a la plataforma y podrá acceder a la sección *Préstamos*. En esta sección, los prestatarios habrán indicado sus condiciones de préstamos. En el caso que el prestamista le encaje alguno de estos préstamos que hacen referencia a los anuncios en venta, y con un ID que existe y no está registrado históricamente, tal como se comprobará antes de su publicación, podrá aceptar la operación. El prestamista pulsará el botón *Prestar* y su cantidad de *Ether* será transferida a través de la red *Ethereum* al prestatario. La cantidad de *Gas* por aceptar el préstamo será cargada a la cuenta del prestamista. La transacción se realizará mediante el *plugin Metamask*, mostrando una nueva ventana donde se indica el precio de la transacción, que en este caso coincidirá con la cantidad que figura en el préstamo, y el *Gas* que se descontará en caso de aceptar.

7.5 FUNCIONALIDAD DE ASEGURADORA

A continuación, se detallan la funcionalidad primordial del usuario aseguradora en la *Dapp*:

- El asegurador podrá crear su póliza de seguro. Para ello, tendrá que cumplimentar el formulario que se encontrará en la pestaña *Publicar*. Los parámetros que debe llenar son *ID Anuncio referencia*, *Tipo de seguro* e *Importe en Ethers (ETH)*. Antes de ser publicado en la red *Ethereum*, se comprueba que el precio de la póliza sea positivo, que los campos se encuentren llenos y que el ID de anuncio al que hace referencia no tiene comprador y es válido. En caso de error, el usuario será informado

para solucionar el problema antes de publicarlo, mediante una ventana emergente. Al realizar la publicación en la red *Blockchain*, se descontará el *Gas* de esa transacción de la cuenta del asegurador. La transacción se realizará mediante el *plugin Metamask*, mostrando una nueva ventana donde se indica el precio de la transacción, que en este caso será nulo, y el *Gas* que se descontará en caso de aceptar.

7.6 FUNCIONALIDAD DEL SISTEMA

El sistema deberá contener las siguientes particularidades:

- El sistema será el encargado de realizar la historificación de cada uno de los anuncios que se muestran en la aplicación. Existirá la pestaña *histórico* donde se registrarán las ventas y alquileres de los inmuebles publicados en la pestaña de anuncios. Al registrarse un comprador o arrendatario en el *Smart Contract*, serán automáticamente mostrados en la pestaña *Histórico*, para poder ser visualizados por cualquier usuario. En caso de no tener comprador, se mostrarán en la pestaña *Anuncios*.
- El sistema al aceptar las distintas transacciones de *Blockchain*, se conectará a través de la inyección de *Web3.js* al *Javascript* principal de la *Dapp*, que permitirá conectarlo con la función necesaria del *Smart Contract* sobre el que se esté trabajando. En la función se encontrará con el código que permita almacenar la transacción, si se cumplen los requerimientos especificados en cada situación.
- El *backend* de la aplicación se encargará de generarle al comprador el código necesario para desbloquear la cerradura digital del inmueble. En caso de producirse un error en el código, se devolverán parte de los fondos al comprador.

7.7 FUNCIONALIDAD USUARIO CON PRIVILEGIOS

Dentro de la aplicación, se contará con usuarios con privilegios que dispondrán de las siguientes funcionalidades:

- Para evitar sobresaturar la aplicación, o evitar ataques malignos en caso de alertarse de anuncios sospechosos, el usuario con privilegios tendrá la posibilidad de desactivar de la plataforma la visualización de estos anuncios, dándolos de baja. Esto quedará señalado por el *backend* de la aplicación en el *Smart Contract* propio del anuncio.
- Este mismo comportamiento se extiende para evitar las cuentas sospechosas de ser timadores o producir estafas. El usuario con privilegios tendrá la capacidad de denegar la entrada a la aplicación a este tipo de usuario. Para ello, se realizará un perfilado de usuarios con la ayuda del *plugin Metamask*, revocando el acceso a aquellas cuentas que presenten comportamientos no deseados.

7.8 NAVEGABILIDAD EN LA DAPP

Para organizar la información de la aplicación descentralizada, se muestra a continuación la estructura de los contenidos de la misma.



Ilustración 46: Navegabilidad en la Dapp

Desde la pantalla principal, podrá accederse a la sección de identificación de usuarios de la plataforma. Además, la pantalla principal contará con un menú desde el cual se podrán

visitar las pestañas de *Publicar*, *Anuncios*, *Solicitudes prestatario*, *Ofertas pólizas de seguro* y *Registro histórico*.

Dentro de la pestaña *Publicar*, se dará la posibilidad de realizar publicaciones de anuncios de inmuebles, préstamos de criptomonedas y pólizas de seguro para los distintos inmuebles.

En la pestaña *Anuncios*, aparecerán todos aquellos anuncios que se han publicado con anterioridad y que todavía no han sido comprados o alquilados. En esta sección, existirá un botón por cada anuncio para hacer posible la compra y el alquiler de los inmuebles. Una vez se presionen estos botones, se moverán los anuncios al registro histórico.

El contenido referente a todos los préstamos que se han publicado y que todavía no han sido aceptados se mostrará en la pestaña *Solicitudes prestatario*. En ella, los prestamistas aceptarán aquellas peticiones realizadas por los potenciales compradores de inmuebles. Se contará con un botón para aceptar los préstamos que redirigirá aquellos préstamos que se ejecuten al registro histórico.

En la pestaña *Ofertas pólizas de seguro*, podrán encontrarse todas las ofertas de seguros que han realizado las aseguradoras y que todavía no han sido aceptadas por los compradores de inmuebles. En esta pestaña existirá un botón para que los compradores contraten aquellas pólizas que consideren oportunas, y además, llevará hasta el registro histórico todos los seguros que se hayan contratado en el sistema.

Por último, dentro de la pestaña *Registro histórico*, aparecerán todos los anuncios que ya han sido contratados, ya sean anuncios de inmuebles, de préstamos o de seguros.

Capítulo 8. ARQUITECTURA

Para poder implementar el diseño señalado, se deben utilizar diferentes soluciones de software integradas y bien estructuradas para permitir la escalabilidad de la plataforma. Una arquitectura que cumple con los requerimientos demandados es la que se muestra a continuación.

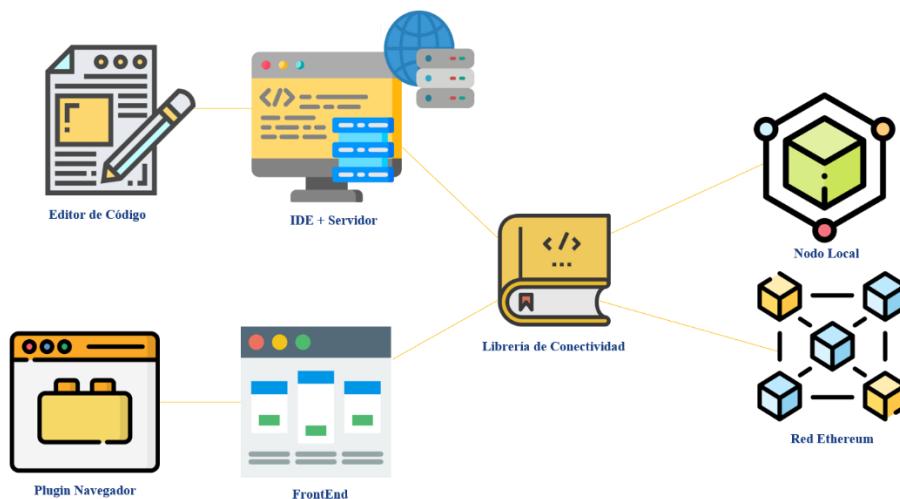


Ilustración 47: Arquitectura Dapp

Por ello, a continuación se presenta el diseño de la arquitectura de la plataforma utilizando las tecnologías descritas en el Capítulo 2. que se han escogido para el modelo.

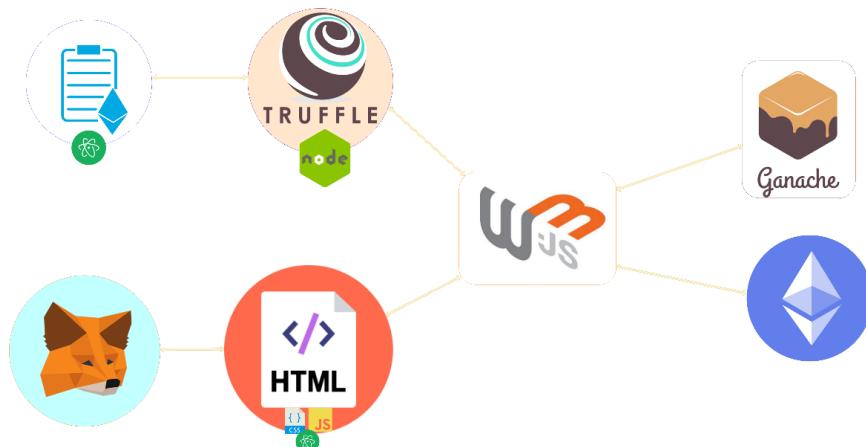
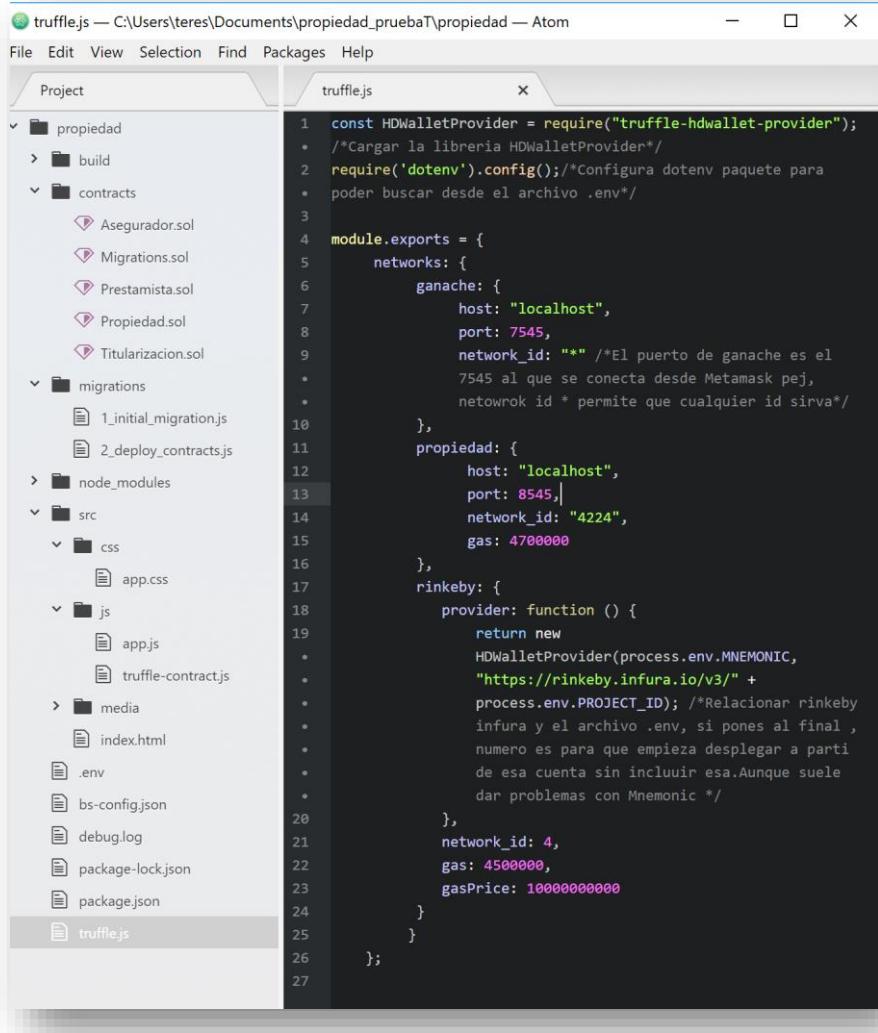


Ilustración 48: Arquitectura Dapp con tecnologías

A la hora de realizar el desarrollo de los *Smart Contracts*, se ha utilizado el entorno de desarrollo de *Truffle*, puesto que permite vincularlos con facilidad a las diferentes redes de *Ethereum*, así como facilitar un entorno para la compilación de los *Smart Contracts* y el despliegue y migración de los mismos a las diferentes redes. También, proporciona la estructura del archivo de proyectos, tal como se visualiza en la Ilustración 49.



The screenshot shows the Atom code editor interface. The left sidebar displays the project structure:

```

Project
  - propiedad
    - build
    - contracts
      - Asegurador.sol
      - Migrations.sol
      - Prestamista.sol
      - Propiedad.sol
      - Titularizacion.sol
    - migrations
      - 1_initial_migration.js
      - 2_deploy_contracts.js
    - node_modules
    - src
      - css
        - app.css
      - js
        - app.js
        - truffle-contract.js
      - media
        - index.html
      - .env
      - bs-config.json
      - debug.log
      - package-lock.json
      - package.json
  - truffle.js

```

The main editor window shows the content of the `truffle.js` file:

```

const HDWalletProvider = require("truffle-hdwallet-provider");
//Cargar la libreria HDWalletProvider/
require('dotenv').config();/*Configura dotenv paquete para
poder buscar desde el archivo .env*/
module.exports = {
  networks: {
    ganache: {
      host: "localhost",
      port: 7545,
      network_id: "*" /*El puerto de ganache es el
      7545 al que se conecta desde Metamask pej,
      netowrok id * permite que cualquier id sirva*/
    },
    propiedad: {
      host: "localhost",
      port: 8545,
      network_id: "4224",
      gas: 4700000
    },
    rinkeby: {
      provider: function () {
        return new
        HDWalletProvider(process.env.MNEMONIC,
        "https://rinkeby.infura.io/v3/" +
        process.env.PROJECT_ID); /*Relacionar rinkeby
        infura y el archivo .env, si pones al final ,
        numero es para que empieza desplegar a parti
        de esa cuenta sin incluir esa.Aunque suele
        dar problemas con Mnemonic */
      },
      network_id: 4,
      gas: 4500000,
      gasPrice: 10000000000
    }
  };
};

```

Ilustración 49: Estructura del proyecto- *Truffle*

- La carpeta de *build* contiene *ABI*, que permite llamar a los Smart Contracts desarrollados.

- La carpeta *contracts* son los *Smart Contracts* que se han creado para el desarrollo de la *Dapp*.
- La carpeta de *migrations* incluye los ficheros *1_initial_migration.js* y *2_deploy_contracts.js*, que son utilizados para poder realizar la implantación en las redes de *Ethereum* basándose en *Truffle*.
- La carpeta *node_modules* es donde se ha realizado la instalación a través de la ventana de comando del servidor *NPM lite-server* de *node.js*.
- La carpeta *src* es la parte de *frontend* de la aplicación junto al *Javascript*, que juega un papel muy importante, pues también se encarga de relacionar la parte del servidor con la del cliente.
- El fichero *.env* es utilizado para la implementación segura a través de *Infura* para la red *Rinkeby*.
- El fichero *bs-config.json* se utiliza para indicar al *lite-server* dónde se encuentran las carpetas a las que tiene que acceder.
- El fichero *debug.log* sirve para almacenar los errores producidos.
- El fichero *package-lock.json* y *package.json* son generados de forma automática por *node.js*, y en concreto sirven para manejar las dependencias del proyecto.
- El fichero de *truffle.js*, que es el que aparece en la Ilustración 49, permite indicar la configuración de la red para conectarnos a nivel local a través de la consola de comando, o a través la aplicación puesto que *Ganache*, tiene los puertos 8545 o 7545. Además, también está realizada la configuración para la red *Rinkeby*. En todos aparece el precio del *Gas* y el *Gas* que se habilita para poder desplegar el *Smart Contract*.

Cabe destacar que el programa utilizado para programar a través del lenguaje *Solidity*, todos los ficheros comentado anteriormente ha sido el editor *Atom*, que tiene gran cantidad de paquetes y customizaciones disponibles junto a una visión estructurada del proyecto.

Tras tener la parte del desarrollo de *Smart Contracts* configurada se necesita un nodo de *Ethereum* para ejecutarlos y testearlos. Los *Smart Contracts* se necesitan ejecutar en *Ethereum Virtual Machine*.

- En el proyecto se ha decidido realizar las pruebas al principio a través de *Ganache*, nodo a nivel local, puesto que la velocidad de transacción es muy rápida. Es un simulador a nivel de red local donde se pueden desplegar los *Smart Contracts*, ejecutarlos e inspeccionar la red. Te permite configurar la generación del número de cuentas junto a su balance. Cada cuenta tiene sus claves: pública y privada. Además, posibilita explorar los bloques y las transacciones desde el gasto de *Gas*, el tipo de transacción, y su dirección en el despliegue del *Smart Contract*, así como mostrar el funcionamiento a través de los logs.
- En segundo lugar, se ha decidido implementar en una red de pruebas de *Ethereum*, en concreto *Rinkeby*, para poder testear con tiempos de ejecución adecuados, ya que se realiza la minación de diversos nodos mientras se resuelve *PoS*. Con *Rinkeby* es posible analizar las transacciones e inspeccionar los parámetros desde una plataforma externa a través de *Etherscan*. La red de pruebas *Rinkeby* tiene un método seguro para evitar el colapso de la misma, por lo que la obtención de criptomonedas gratuitas es lenta. El límite máximo de obtención durante tres días es 18,75 *Ethers*, y se realiza a través de la publicación de un *tweet* con el número de cuenta que deseas obtener dinero. Es recomendable prever esta situación, así como realizar un uso moderado de los *Ethers* durante la fase pruebas.

La pieza angular que permite conectar la parte de *frontend* con la de *backend* y con *Blockchain* es *Web3.js*. *Javascript* necesita disponer de la librería *Web3.js* para poder interactuar con *Smart Contracts*. *Web3.js* proporciona acceso a varias funcionales de forma más sencilla, como para los que no desean hacer *Remote Procedure Calls*. En resumen, el navegador y el servidor de *backend Node.js* interactúan con las funciones de *los Smart Contracts* utilizando *Web3.js*. Las llamadas a las funciones de *los Smart Contracts* producen la creación de las transacciones en *Blockchain*.

Además, *Web3.js* basándose en el *framework Truffle* tiene desarrollado el código para inyectar el proveedor *Web3.js* de la extensión *Metamask*. *Metamask* permite de forma sencilla y fácil, para cualquier tipo de usuario, poder identificarse a cualquiera de las redes

Ethereum, ver su balance, e incluso aceptar o denegar transacciones en la *Dapp*. En la Ilustración 50 se muestra cómo *Metamask* aparece en una venta aparte a la aplicación para realizar la aceptación o no de la transacción.

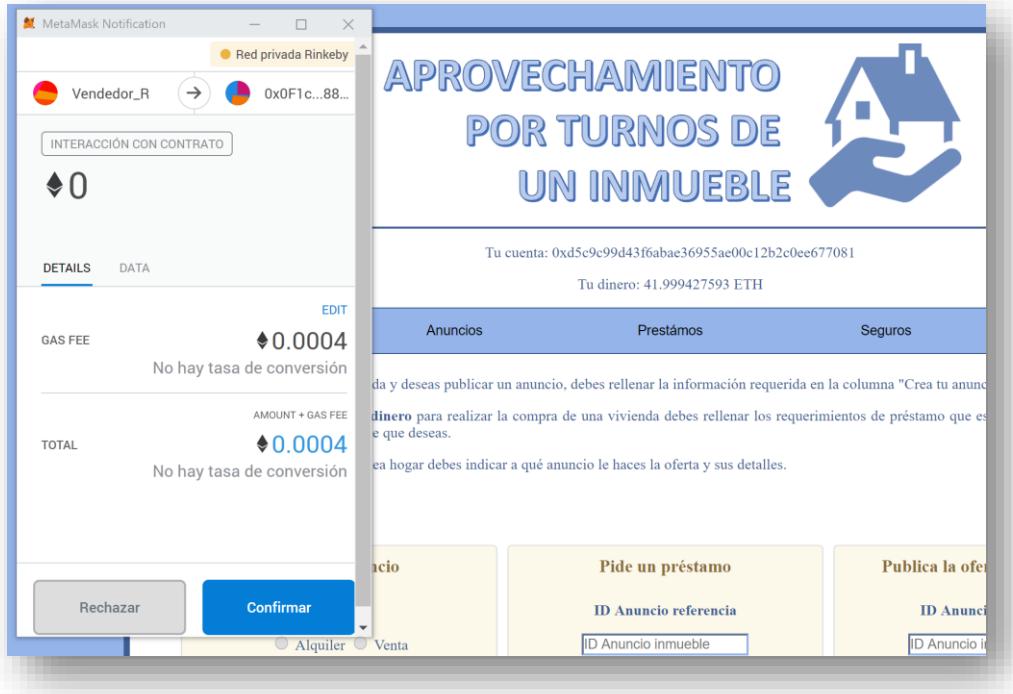


Ilustración 50: Metamask interacción Dapp

En la Ilustración 50, se ha creado la transición al pulsar el botón publicar el anuncio. Esta transacción es transmitida a través de *Web3.js* en formato binario a *Metamask*. En el momento de aceptación o rechazo, se comunica con *Web3.js* a la red *Ethereum*.

Capítulo 9. DESARROLLO DE LA DAPP

9.1 SMART CONTRACTS

Los *Smart Contracts* programados en la *Dapp* están escritos en *Solidity*. En concreto en esta *Dapp* existen cinco *Smart Contracts*:

- Propiedad.sol
- Prestamista.sol
- Asegurador.sol
- Titularizacion.sol
- Migrations.sol

Con el fin de clarificar el comportamiento y funcionamiento de los *Smart Contract*, se realizará diversos aclaraciones respecto a distintas zonas del código de algunos *Smart Contracts*.

Como es una tecnología que se encuentra avanzando, al principio del *script* se indica la versión que se utiliza para que a la hora de compilarlo y usarlo pasado cierto tiempo, no se modifique la intención del *Smart Contract* evitando un mal comportamiento y un derroche de *Ethers*. En el proyecto se ha utilizado esta versión que se muestra

```
pragma solidity ^0.4.18; /*Versión 0.4.18 o superior*/
-----
```

Código 1: Fragmento versión Solidity

El *Smart Contract* de *Titularizacion.sol* es utilizado por *Propiedad.sol*, *Prestamista.sol* y *Asegurador.sol*. Sirve para restringir el acceso a la cuenta en caso de que no seas el titular, y por tanto, es necesario importarlo al principio de cada uno de los *Smart Contracts*.

```
import "./Titularizacion.sol";
```

Código 2: Fragmento import

Tras indicar la versión de *Solidity* y la importación del *Smart Contract*, se define el *Smart Contract* similar a una clase en *Java*. En *Propiedad.sol*, para poder trabajar con mayor comodidad, se ha definido una estructura de datos llamada *Semana* donde se agrupan las características del anuncio. Estos valores permanecen almacenado en el *storage* del *Smart Contract*.

```
contract Propiedad is Titularizacion {
    struct Semana { uint id; address vendedor; address comprador; string operacion; string localizacion; string resumen; string descripcion; uint256 precio; }
```

Código 3: Fragmento *Propiedad.sol* estructura

Se ha utilizado *mapping* en *Propiedad.sol* para almacenar de forma cómoda los anuncios junto a las características almacenadas en su estructura.

```
mapping (uint => Semana) public semanas; /*Variables estaticas*/
uint contSemana;
```

Código 4: Fragmento *Propiedad.sol* mapping

Al llamar a los eventos se guarda los parámetros en el registro que esta asociada a la dirección del *Smart Contract*, en este ejemplo se guarda en la dirección de *Asegurador.sol* facilitando la comprobación de la exitencia de ese bloque. Los atributos indexados posibilitan filtrar por ellos y pueden existir hasta tres parámetros. Se posibilita acceder a la zona de log de estos eventos.

```
event LogPublicarSeguro(uint indexed _idSeguro, address indexed _asegurador, string _referencia, uint256 _cantidad); /*Los eventos se relacionan en la app.js*/
```

Código 5: Fragmento *Asegurador.sol* evento

Las funciones son aquellas que se pueden invocar una vez que se ha implementado en *Blockchain* y nos permiten interactuar. El tipo de la función puede ser:

- *Public*. Cualquier persona puede llamar a esta función con una cuenta de *Ethereum*, accediendo al *Smart Contract*. La mayoría de las funciones son de este tipo
- *Private*. No implementa ningún nivel de seguridad, se utiliza en funciones auxiliares que realizan tareas muy concretas.

- *View o constant.* Se utilizan para cuando se accede a la información pero en ningún momento se modifica.
- *Pure.* Es muy parecido al caso anterior pero va un paso más allá. No modifica ni accederá a los datos del contrato.
- *Payable.* Se utiliza cuando alguna entidad externa intente llamar a una función y enviar dinero.

Además, cada función tiene un nombre y posible información a devolver o no.

La siguiente función permite enviar a la dirección del titular todo el saldo del *Smart Contract*. Se utiliza cuando se quiere eliminar el *Smart Contract* porque al liberar espacio el *Gas* es inferior a la cantidad que costaría enviar los fondos a través de una transacción de envío de *Ethers*.

```
function desactivar() public unicoTitular { /*Desactivar el contrato*/
    selfdestruct(titular);
}
```

Código 6: Fragmento desactivación Smart Contract

La función *venderSemana* de *Propiedad.sol* se fundamenta en la estructura y el *mapping* realizado para almacenar en la estructura la información que ha recibido a través del *frontend* conectado al *Javascript*, así como contabilizar el ID del anuncio a través del contador.

```
/*Función venderSemana está conectada con app.js e index.html*/
function venderSemana(string _operacion, string _localizacion, string
_resumen, string _descripcion, uint256 _precio) public {
    contSemana++; /*Aumentar el contador de semanas nuevas*/
    semanas[contSemana] =
Semana(contSemana,msg.sender,0x0,_operacion,_localizacion,_resumen,_descripcion,
,_precio);/*Guardar la semana segun el formato y secuencia de la estructura*/
    LogVenderSemana(contSemana, msg.sender, _resumen, _precio);
}
```

Código 7: Fragmento Propiedad.sol vender Anuncio

La función *getNumeroPrestamos* de *Prestamos.sol* consiste en contabilizar el número de préstamos que existen.

```
function getNumeroPrestamos() public view returns (uint) { /*Contar número
prestamos*/
    return contPrestamo;
}
```

Código 8: Fragmento Prestamos.sol obtener número de préstamos

La función *aceptarPrestamo* de *Prestamos.sol* es tipo *payable*, porque, cuando se llama a la función, se necesita enviar el dinero. Se basa en realizar diferentes comprobaciones a través de *require* para realizar de forma segura la aceptación del préstamo. Estas comprobaciones añaden seguridad al proceso, y se valida que exista al menos un préstamo en el sistema, se comprueba que el préstamo que se desea aceptar no tiene una dirección de prestamista, que la persona que acepta el préstamo no sea la misma persona que lo está pidiendo, y que la cantidad de *Ethers* que se envía coincida con la que se solicita en el *Smart Contract*.

```
function aceptarPrestamo(uint _idPrestamo) payable public { /*Payable:
permite recibir ether o valor a la función. Si intentas enviar desde una
función a otra que no es payable se rechaza la operación.*/
    require(msg.sender != prestamo.prestatario); /*El prestatario NO puede
aceptar su préstamo*/
    [...]
    prestamo.prestamista = msg.sender; /*Información del prestamista*/
    prestamo.prestatario.transfer(msg.value); /*Prestamista pueda transferir
el dinero al prestatario, es decir, poder prestarle*/
    LogAceptarPrestamo(_idPrestamo,
                        prestamo.prestatario,
                        prestamo.prestamista, prestamo.referencia, prestamo.cantidad); /*Evento*/
}
```

Código 9: Fragmento Prestamos.sol aceptación préstamo

La siguiente función de *Asegurado.sol* nos permite conocer los seguros que se han publicado y aún no tienen asegurado. Se utiliza *memory* por utilizarse una variable temporal y consumir menos que *storage*.

```
function getSegurosPublicados() public view returns (uint[])
{
    uint[] memory seguroIds = new uint[](contSeguro); /*Array de longitud
contSeguro*/
    uint numSegurosPublicados = 0; /*Contador numero de seguros publicados sin
asegurado*/
    [...]
    for(uint j = 0; j < numSegurosPublicados; j++) { /*Copiar los IDs de las
seguros publicados sin asegurado a un array de tamaño menor*/
        publicados[j] = seguroIds[j];
    }
    return publicados; /*Devolver las seguros publicados sin asegurado*/
}
```

Código 10: fragmento Asegurado.sol seguros publicados sin asegurado

La función *getSemanasTotales* de *Propiedad.sol* nos permite obtener todos los anuncios vendidos, así como los no vendidos aún para poder utilizarlo en el registro histórico.

```
function getSemanasTotales() public view returns (uint[]) {
    uint[] memory semanaIds = new uint[](contSemana); /*Array de longitud
contSemana*/
    [...]
    for(uint j = 0; j < numSemanasTotal; j++) { /*Copiar los IDs de las
semanas totales a un array de tamaño menor*/
        Total[j] = semanaIds[j];
    }
    return Total;
}
```

Código 11: fragmento *Propiedad.sol* Obtener todas los anuncios

Por último, señalar que *Migrations.sol*, es un *Smart Contract* ofrecido por *Truffle* para facilitar la migración y despliegue a la red. Este contrato almacena el número del último script que se ha implementado, de tal forma que *Truffle* no vuelva a ejecutarlo. Por otro lado, en una ejecución futura, es posible que la aplicación deba implementar un nuevo *Smart Contract*, y, deberá acceder a *Migrations.sol* para no sobrescribir los *Smart Contracts* anteriores.

9.2 INTERFAZ DE USUARIO

En esta sección se mostrará la vista del cliente, es decir, la interfaz de usuario a través de texto explicativo del funcionamiento así como diferentes capturas de pantallas.

9.2.1 PÁGINA DE BIENVENIDA

Al acceder el usuario a la plataforma, aún sin estar identificado a través de *Metamask* se le mostrará una página de bienvenida para que sepa la competencia de la misma. Para hacer la interfaz más amigable se ha realizado una selección de imágenes que se pueden seleccionar para visualizar tal como se muestra en la Ilustración 51.

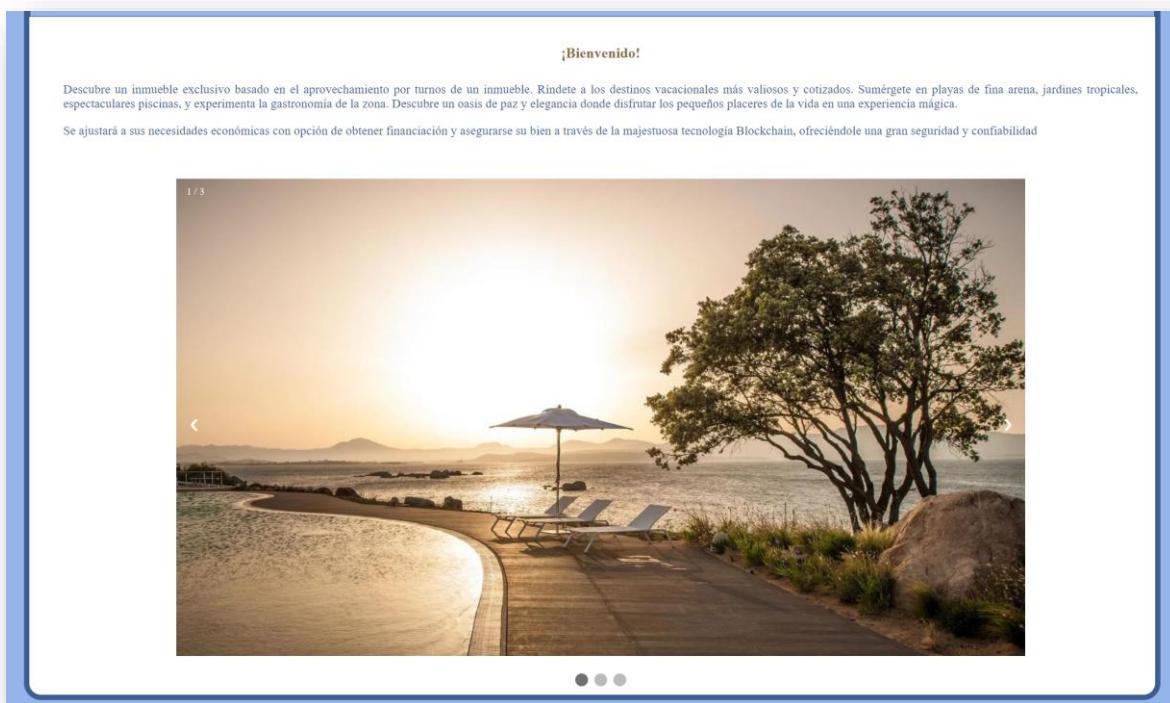


Ilustración 51: Página de bienvenida

9.2.2 CUENTA

En la parte superior de la *Dapp*, se encuentra el rótulo con el nombre de la misma y su logo, junto al número de cuenta y su saldo. Hasta que el usuario no se identifica en *Metamask* aparece sin información tal como se presenta en la Ilustración 52.



Ilustración 52: Cuenta y saldo sin Metamask

Tras identificarse en *Metamask* y tener seleccionado una cuenta por defecto o haberla seleccionado, se muestra su número de cuenta junto a su saldo en *Ethers* como parece en la Ilustración 53.



Ilustración 53: Cuenta y saldo con Metamask

9.2.3 PUBLICAR

Tras identificarse con su cuenta a través del plugin *Metamask*, el usuario tendrá acceso a través de la pestaña *Publicar* a poder publicar anuncios en caso de ser vendedor, solicitar un préstamo en el caso de ser un comprador y si es una aseguradora poder publicar una oferta de póliza de seguro. La pestaña *Publicar* se encuentra en la misma menú horizontal que el resto.

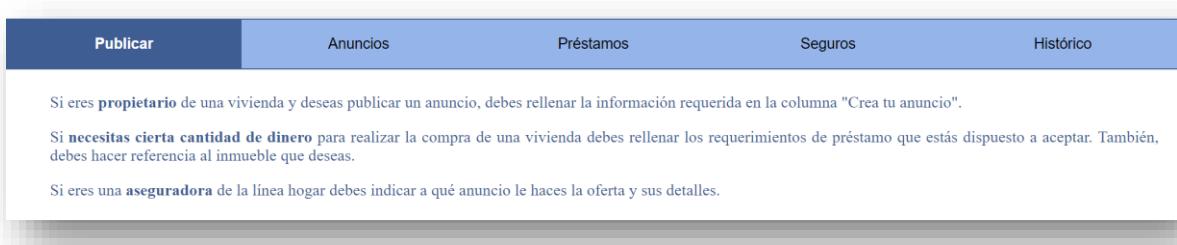


Ilustración 54: Pestaña Publicar

9.2.3.1 Publicar Anuncios de Venta o Alquiler

En caso de ser un vendedor o arrendatario con intención de vender o alquilar un inmueble, se deberá completar el formulario indicado en la Ilustración 55 . Antes de realizarse la publicación, se verificará que la información es válida: los datos se encuentran llenos y el precio es positivo. Habrá que señalar de qué tipo de operación se trata, la localización del inmueble, un breve resumen del anuncio, la descripción del inmueble y el precio en *Ethers*.

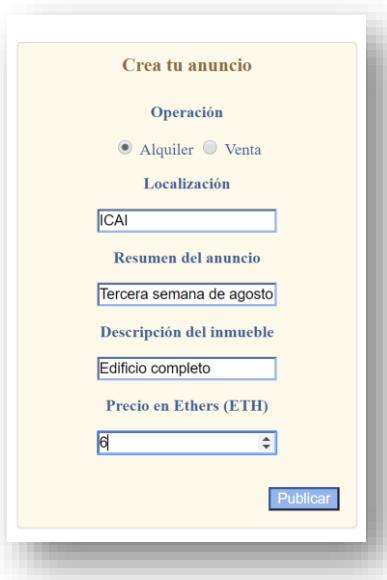
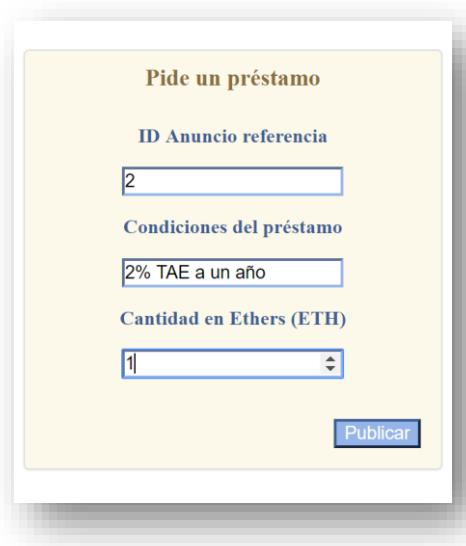


Ilustración 55: Publicación anuncio

9.2.3.2 Publicar Solicitud de Préstamo

Al ojear los inmuebles anuncios con intención de comprarlos, el comprador puede advertir que no dispone de todo el dinero del inmueble que quiere, por lo que necesita solicitar un préstamo. Para ello, se rellenan los campos solicitados: ID del anuncio por el que se desea solicitar un préstamo, condiciones del préstamo que aceptarías y la cantidad que deseas en *Ethers*. Esta información es validada a través del *Javascript*, es decir, que el ID al que se hace referencia realmente sea un anuncio en venta y no de alquiler, que el ID al que se hace referencia no sea de un anuncio del que se es propietario, que el ID exista y no sea

inventado, y que este ID no tenga ya comprador, es decir, que no esté presente en la pestaña de registro *Histórico*. Además, se comprueba que la cantidad deseada de *Ethers* sea positiva.



Pide un préstamo

ID Anuncio referencia
2

Condiciones del préstamo
2% TAE a un año

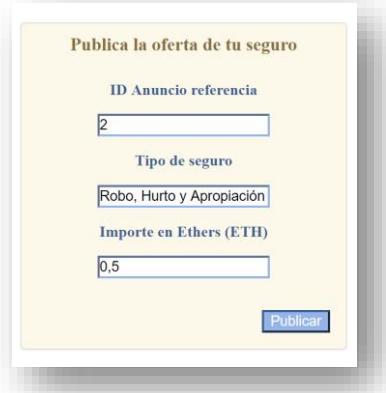
Cantidad en Ethers (ETH)
1

Publicar

Ilustración 56: Solicitud préstamo

9.2.3.3 Publicar Oferta de Póliza de Seguro

A la hora de realizar una oferta de una póliza de seguro, la aseguradora lo deberá realizar para los anuncios en venta, anuncios que aún no tengan comprador, que el ID de anuncio exista. Además, la cantidad debe ser positiva.



Publica la oferta de tu seguro

ID Anuncio referencia
2

Tipo de seguro
Robo, Hurto y Apropiación

Importe en Ethers (ETH)
0.5

Publicar

Ilustración 57: Publicación póliza de seguro

9.2.4 ANUNCIOS

Otra de las funcionalidades de la *Dapp* es, tras publicar los anuncios en la red *Ethereum*, mostrarlos al usuario para poder interactuar con ellos.

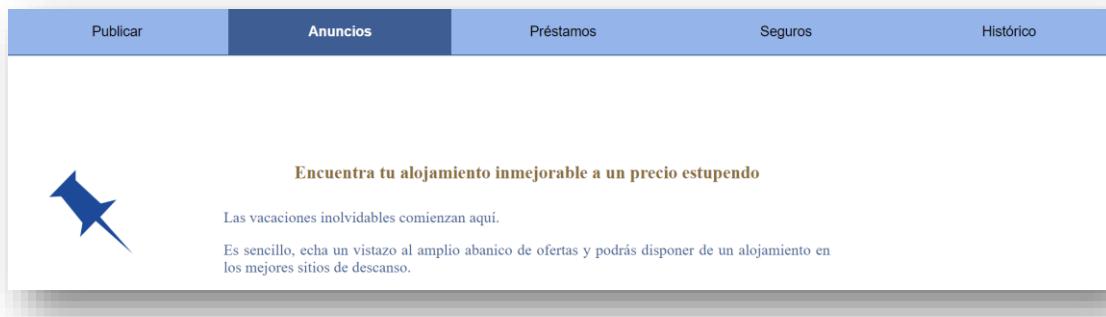


Ilustración 58: Pestaña Anuncios

Para que el futuro comprador pueda ubicar al inmueble con mayor facilidad, a la hora de mostrarlo se utiliza la ubicación y se geoposiciona a través de la *API* de *Google Maps*. Además también se muestran los mismos campos que en el anuncio junto a la cuenta del vendedor. En el momento que el comprador desee realizar la compra del inmueble, deberá pulsar en el botón *Comprar*.

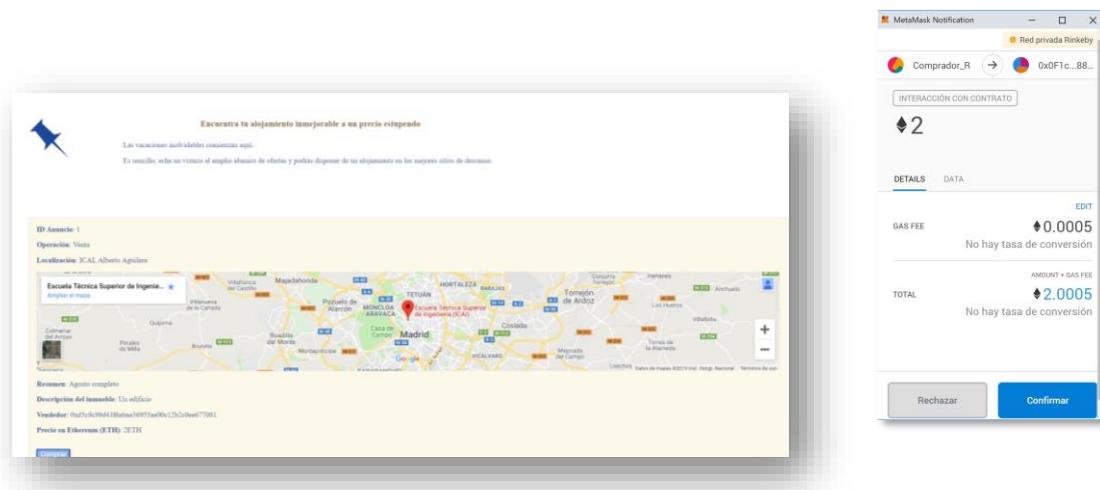


Ilustración 59: Comprar inmueble

Para poder realizar la compra se necesita contar con la ayuda de *Web3.js*, *Metamask* y *Javascript*. Desde *Javascript* a través del Código 12, se detecta la inyección de una instancia de *Web3.js* y junto al plugin *Metamask* permite poder realizar la compra. Además, en el código también se muestra el paso que inicializa la cuenta y la cantidad de *Ethers* que tiene y que lo carga en la cabecera de la página web, así como la funcionalidad para crear una instancia de los tres *Smart Contracts* principales: *Propiedad.sol*, *Prestamista.sol* y *Asegurador.sol*.

```
inicializarWeb3: function() {
    /*Iniciar web3, es decir si ya está definida web3 variable
    inyectada en la página web por Metamask (pej). Si abres la aplicación con Metamask
    habilitado, se inyecta una instancia de web3 en el Windows.object de la pagina*/
    if(typeof web3 !== 'undefined') {
        /*Reutiliza el provider del objeto web3 inyectado por Metamask
        (pej) */
        App.web3Provider = web3.currentProvider;
    } else {
        [...]
    }
    web3 = new Web3(App.web3Provider); /*Se instancia un nuevo objeto de
    Web3. Tras inicializar el Web3 objeto, se carga al nodo local y podemos usarlo
    para dos cosas: mostrar info de la cuenta(número cuenta, balance...) y lo
    segundo para instanciar el contrato.*/
    [...]
    return App.inicializarContrato(); /*Se inicializa la conexión con el
    contrato de Semanas de anuncios*/
},
```

Código 12: Fragmento Javascript *Web3.js* [82]

9.2.5 PRÉSTAMOS

En la sección *Préstamos* se mostrarán las solicitudes de los prestatarios para que los prestamista pueden aceptar en caso que les interese las condiciones del préstamo.

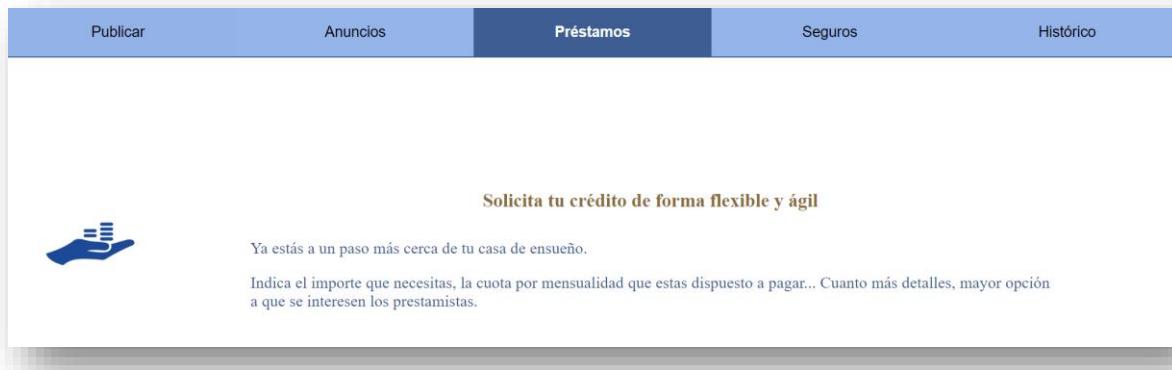


Ilustración 60: Pestaña Préstamos

En la Ilustración 61: Prestar dinero, se muestra la ventana de *Metamask* que aparece después que el prestamista pulsa el botón de prestar del anuncio. En concreto, se observa que el préstamo hace referencia al anuncio en venta con el ID 1 donde se pueden observar las condiciones junto el número de cuenta del prestatario.

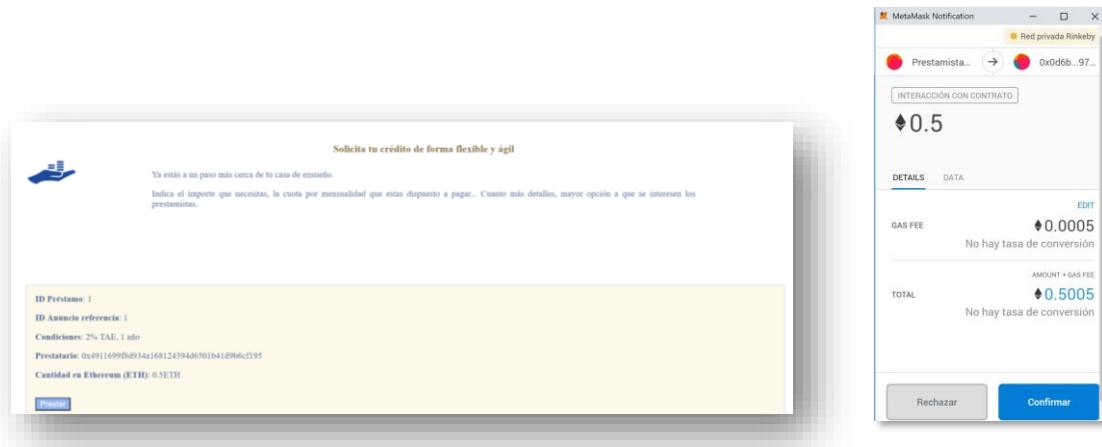


Ilustración 61: Prestar dinero

9.2.6 SEGUROS

En la sección seguros se mostrarán las ofertas de las pólizas realizadas para los anuncios en venta que se encuentran ubicados en la sección *Anuncios*.

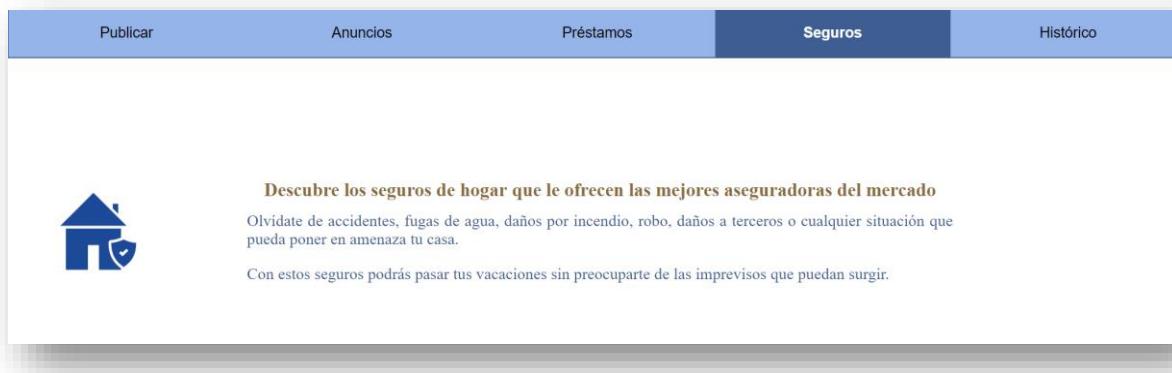


Ilustración 62: Pestaña Seguros

A continuación, se muestra la póliza de un seguro. Se puede contratar pulsando el botón *Contratar* que realizará el registro en *Blockchain* a través de las funciones de *Javascript*. Este se relaciona con las funciones del *Smart Contract: Asegurador.sol* y además se lo comunican a *Metamask* para realizarlo de forma visual.

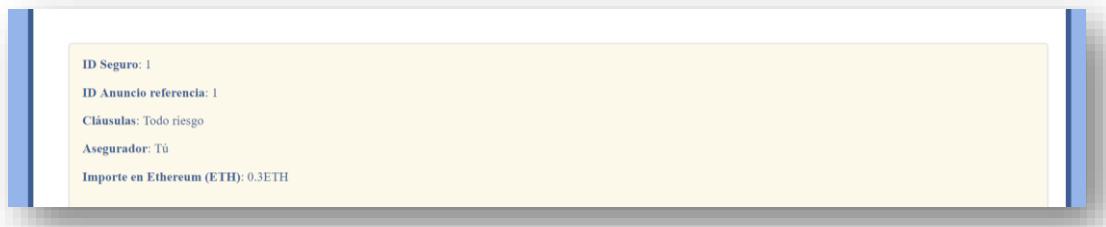


Ilustración 63: Póliza de seguro

9.2.7 REGISTRO HISTÓRICO

La pestaña *Histórico* permite mostrar al usuario los inmuebles que se han alquilado o contratado, así como las contrataciones de seguros y los préstamos realizados. Esta sección es meramente informativa, puesto que, además de mostrarse en la sección, todo se encuentra registrado correctamente en la red *Rinkeby* de *Ethereum*.

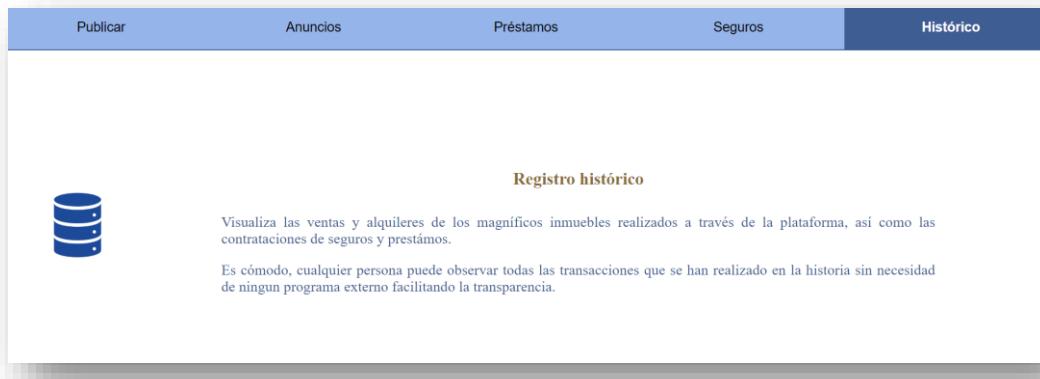
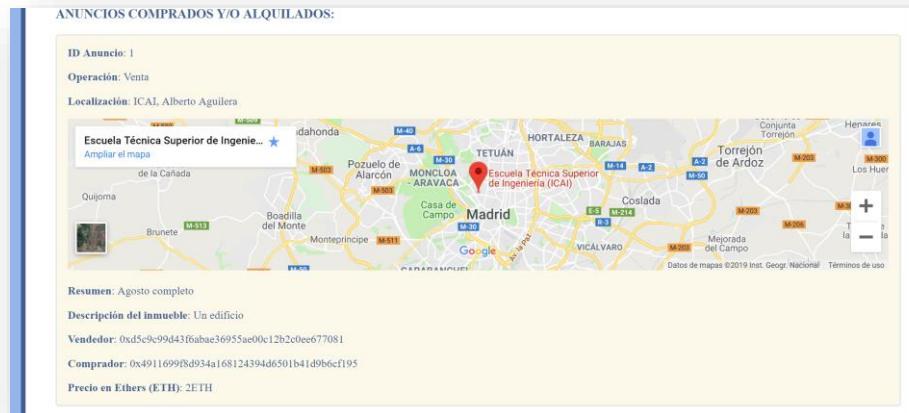


Ilustración 64: Pestaña Registro Histórico

Tras ser comprado o alquilado un inmueble se muestra en *Histórico* y dentro de este en la sección *Anuncios comprados y/o alquilados*. La información aparece la misma que antes de ser comprado o alquilado. La diferencia únicamente reside en que ya no aparece la funcionalidad para poder interactuar con el inmueble, así como aparece el número de cuenta del comprador.



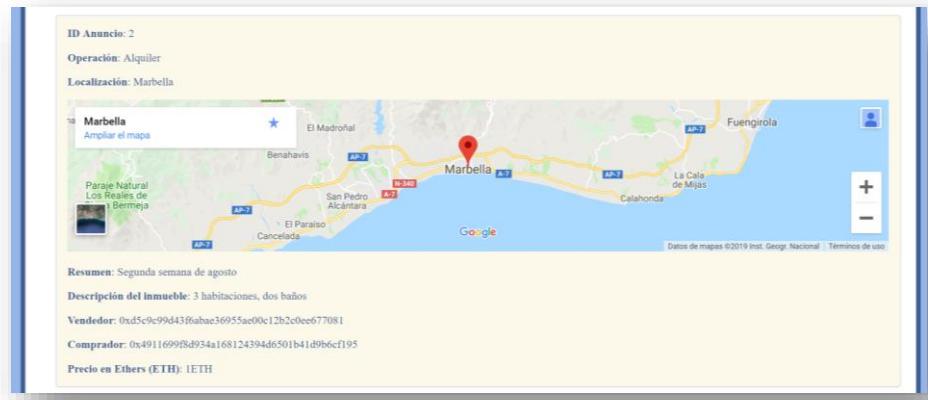


Ilustración 65: Registro histórico anuncios

Tras ser realizado un préstamo respecto a un inmueble de venta se muestra en *Histórico*, y dentro de éste en la sección *Préstamos realizados*. La información que se muestra es: ID de referencia al anuncio del inmueble, ID del préstamo, condiciones del préstamo, número de cuenta del prestatario, número de cuenta del prestamista y el importe prestado.



Ilustración 66: Registro histórico préstamos

Tras ser contratado un seguro se muestra en *Histórico* y dentro de este en la sección *Seguros contratados*. Se enseña los siguientes campos respecto a la póliza: ID de referencia al anuncio en venta del inmueble, ID de la póliza del seguro, cláusulas de la póliza, número de cuenta del asegurador, número de cuenta del asegurado y el importe del seguro.

SEGUROS CONTRATADOS:

ID Seguro: 1
ID Anuncio referencia: 1
Cláusulas: Todo riesgo
Asegurador: Tú
Asegurado: 0x4911699f8d934a168124394d6501b41d9b6cf195
Importe en Ethers (ETH): 0.3ETH

Ilustración 67: Registro histórico seguros

Capítulo 10. DESPLIEGUE EN RINKEBY Y ANÁLISIS DE LAS TRANSACCIONES

En este capítulo del documento se van a analizar las transacciones una vez se ha desplegado la *Dapp* en la red de pruebas *Rinkeby*. Para ello, se va a hacer uso del programa *Etherscan*. A continuación, se muestra el flujo que se va a seguir:

	<i>Vendedor</i>	<i>Comprador</i>	<i>Prestamista</i>	<i>Aseguradora</i>
Paso 1	En venta inmueble 1 (I1)			
Paso 2	En alquiler inmueble 2 (I2)			
Paso 3				Publica seguro 1 (S1), relacionado con I1
Paso 4		Publica préstamo 1 (P1) (relacionado con I1)		
Paso 5			Acepta P1 (relacionado con I1)	
Paso 6		Contrata S1 (relacionado con I1)		
Paso 7		Compra I1		
Paso 8	En alquiler inmueble 3 (I3)			
Paso 9		Alquila I2		
Paso 10		Alquila I3		

Tabla 4. Flujo de transacciones

Para seguir este flujo, se han cargado cuatro cuentas en la red de pruebas *Rinkeby*, de tal forma que cada una de ellas realizará el rol de cada uno de los cuatro perfiles que se muestran. A continuación, se muestra la relación entre cada uno de los perfiles y la cuenta utilizada:

<i>Perfil</i>	<i>Cuenta</i>
Vendedor	0xd5c9c99d43f6abae36955ae00c12b2c0ee677081
Comprador	0x4911699f8D934a168124394d6501B41d9B6cf195
Prestamista	0xa44199021303e147F7ed9F1E6eC597F50baE1706
Aseguradora	0x95f14652f634948aEA5866229a0d89409eA21612

Tabla 5. Relación entre perfil y cuenta

En primer lugar, una vez se ha realizado el despliegue en *Rinkeby*, se realiza una transacción para el despliegue de cada uno de los *Smart Contracts* por los que está formada la *Dapp*. Cabe destacar que el despliegue de un *Smart Contract* en *Rinkeby* tiene un coste en *Gas*. En el fichero *truffle.js*, explicado en el apartado de arquitectura, se indica en el código qué cuenta es la encargada de realizar los despliegues de los *Smart Contracts*, y, por tanto, de realizar el pago en *Gas*.

```
rinkeby: {
    provider: function () {
        return new HDWalletProvider(process.env.MNEMONIC,
        "https://rinkeby.infura.io/v3/" + process.env.PROJECT_ID); /*Relacionar rinkeby
        infura y el archivo .env, si se pone al final ", número" es para que empieza
        desplegar a partir de esa cuenta sin incluir esa*/
    },
    network_id: 4,
    gas: 4500000, //Máxima cantidad de gas dispuesto a pagar por cada
    transacción
    gasPrice: 10000000000 //Cantidad de Ether (en Gwei) dispuesto a
    pagar por cada unidad de Gas
}
```

Código 13. Detalle despliegue en Rinkeby de truffle.js

Se observa cómo, en la línea donde se genera el *HD Wallet Provider*, no se incluye ningún número, y, por tanto, la primera cuenta de la *Wallet* que realiza el despliegue será la encargada de desplegar cada uno de los *Smart Contracts*.

Se ha generado un *HD Wallet*, que es un sistema que permite almacenar claves privadas, públicas y direcciones de las cuentas, a partir de una frase semilla única. Esta *Wallet* conserva de forma segura las cuentas y mantiene facilidad para restaurarlas. Se ha utilizado *BIP32*, que es la estructura de la frase semilla. Estas cuentas permiten trabajar en todas las redes *Ethereum*, puesto que una cuenta sirve tanto para cualquiera de las redes de pruebas, como local o la red principal. En este caso, se ha utilizado en la red *Rinkeby*.

Derived Addresses

Note these addresses are derived from the BIP32 Extended Key

Encrypt private keys using BIP38 and this password: Enabling BIP38 means each key will take several minutes to generate.

Table	CSV				
Path	Toggle	Address	Toggle	Public Key	Toggle
m/44'/60'/0'/0/0		0x4911699f8D934a168124394d6501B41d9B6cf195		0x033c4388ab8334c85dce76660062069088d4f2e40ef5a04edfb77d7fac93adcc62	
m/44'/60'/0'/0/1		0xD5c9c99d43f6aBaE36955AE00C12b2c0eE677081		0x039ef7f27f5f7f6aa7270e48ea461d1024784fde03d013e5ccf71d9f1e5b685678	
m/44'/60'/0'/0/2		0xa44199021303e147F7ed9F1E6eC597F50baE1706		0x034f7a359c766b7bfbd71bc8e2be5868dc4a1aa5caa118b9cbd1adce5b3f0781dc	
m/44'/60'/0'/0/3		0x95f14652f634948aEA5866229a0d89409eA21612		0x03e0728d32fbccc8708ca375a1047ae8af6e15a9a25795ebd1252e9d46a7c1192b	

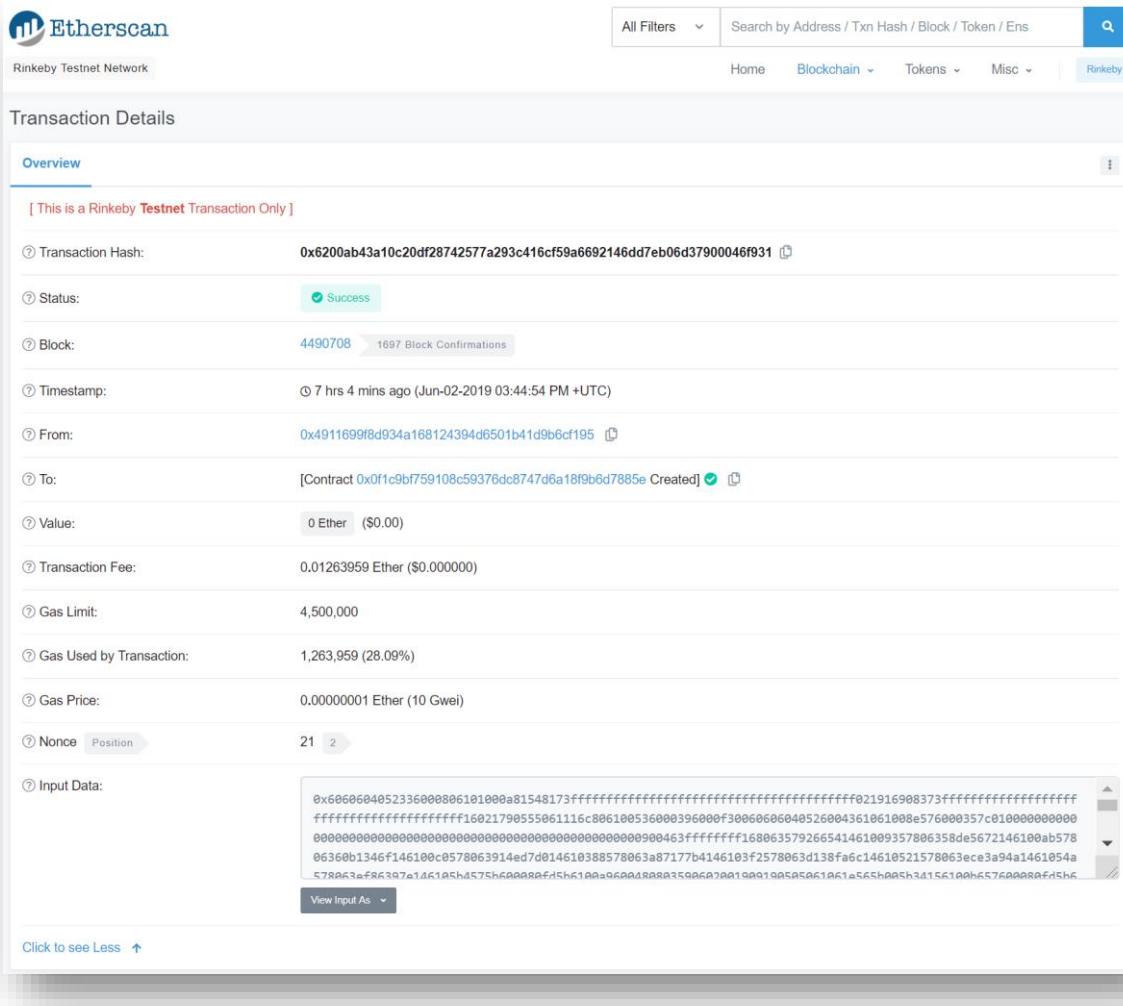
Ilustración 68. Cuentas existentes en Wallet

En la Ilustración 68 se observa, si se compara con el contenido de la Tabla 5, cómo la primera cuenta corresponde con el comprador, la segunda con el vendedor, la tercera con el prestamista y la cuarta con la aseguradora. Así, el encargado de realizar el despliegue de los *Smart Contracts* será el comprador.

Las cuentas mostradas anteriormente han obtenido balance en *Rinkeby* a través de la publicación de un *tweet* con el número de cuenta, y haciendo referencia al enlace a dicho *tweet* en la página principal de *Rinkeby*. Este proceso permite obtener 18,75 ETH cada tres días, para evitar colapsar la red.

10.1 DESPLIEGUE DE LOS SMART CONTRACTS

La primera transacción autogenerada al realizar el despliegue, es por tanto la que genera el *Smart Contract Propiedad.sol* en la red *Rinkeby*.



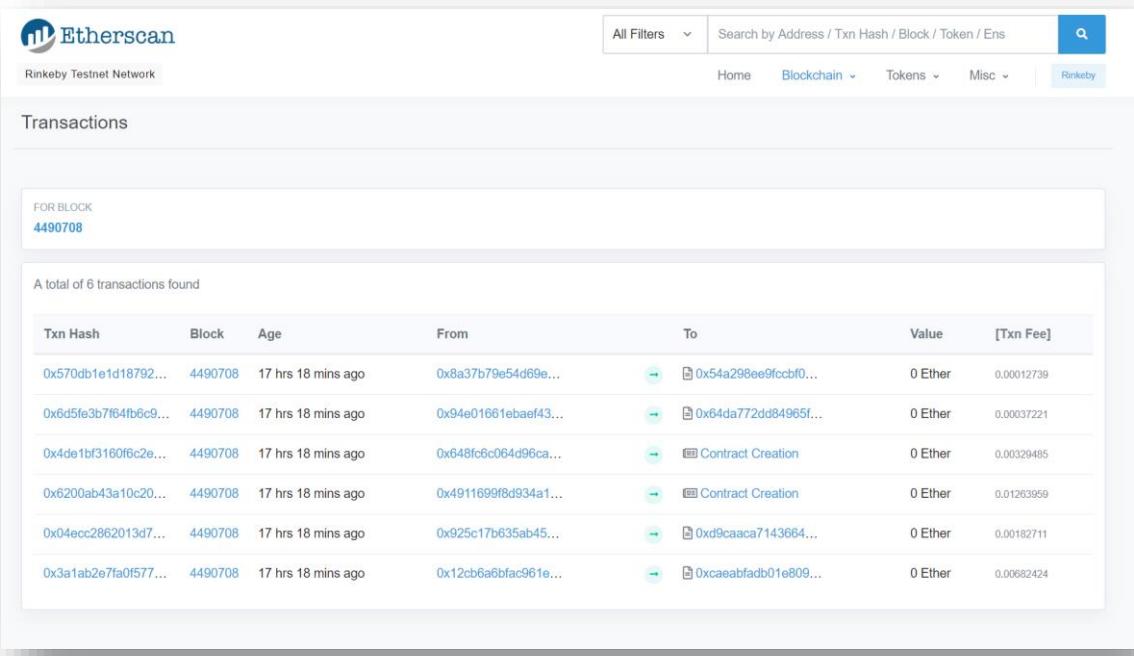
The screenshot shows a transaction details page on Etherscan for a Rinkeby Testnet transaction. The transaction hash is 0x6200ab43a10c20df28742577a293c416cf59a6692146dd7eb06d37900046f931. The status is Success. It was created in Block 4490708 with 1697 confirmations. The timestamp is 7 hrs 4 mins ago (Jun-02-2019 03:44:54 PM +UTC). The transaction originated from 0x4911699f8d934a168124394d6501b41d9b6cf195 and went to [Contract 0x0f1c9bf759108c59376dc8747d6a189b6d7885e Created]. The value was 0 Ether (\$0.000000). The transaction fee was 0.01263959 Ether (\$0.000000). The gas limit was 4,500,000 and the gas price was 0.00000001 Ether (10 Gwei). The nonce was 21. The input data is a large hex string starting with 0x06060604052336000806101000a81548173ffffffffffff... and ending with 578063e486397a146105h4575h600080f15h6100a96004800359060700190050061061a56h005h34156100h657600000000.

Ilustración 69. Transacción Rinkeby: generación Propiedad.sol

Se observa cómo esta transacción tiene en la dirección *From* la cuenta del comprador, por el motivo que se ha explicado anteriormente de que esta cuenta es la primera de la *Wallet*.

Por otro lado, se deduce de la dirección *To* que se ha creado el *Smart Contract* en la dirección 0x0f1c9bf759108c59376dc8747d6a18f9b6d7885e.

Se advierte en la dirección *Block* el bloque dentro de la red *Rinkeby* que forma parte esta transacción. Como se ha comentado en un apartado anterior de este documento, cada bloque de la red está formado por un conjunto de transacciones. En el momento de almacenar una transacción en la red *Rinkeby*, pueden existir varios usuarios que deseen almacenar información, por lo que, en un mismo bloque, coexisten transacciones provenientes de distintos usuarios, como puede observarse en la Ilustración 70.



The screenshot shows the Etherscan interface for the Rinkeby Testnet Network. The top navigation bar includes filters for 'All Filters' (dropdown), 'Search by Address / Txn Hash / Block / Token / Ens' (input field with a magnifying glass icon), and tabs for 'Home', 'Blockchain' (selected), 'Tokens', 'Misc', and 'Rinkeby'. Below the header, a section titled 'Transactions' displays a table of transactions for block 4490708. The table has columns: Txn Hash, Block, Age, From, To, Value, and [Txn Fee]. There are 6 transactions listed:

Txn Hash	Block	Age	From	To	Value	[Txn Fee]
0x570db1e1d18792...	4490708	17 hrs 18 mins ago	0x8a37b79e54d69e...	0x54a298ee9fccbf0...	0 Ether	0.00012739
0x6d5fe3b7f64fb6c9...	4490708	17 hrs 18 mins ago	0x94e01661ebaef43...	0x64da772dd84965f...	0 Ether	0.00037221
0x4de1bf3160f6c2e...	4490708	17 hrs 18 mins ago	0x648fc6c064d96ca...	Contract Creation	0 Ether	0.00329485
0x6200ab43a10c20...	4490708	17 hrs 18 mins ago	0x4911699f8d934a1...	Contract Creation	0 Ether	0.01263959
0x04ecc2862013d7...	4490708	17 hrs 18 mins ago	0x925c17b635ab45...	0xd9caaca7143664...	0 Ether	0.00182711
0x3af1ab2e7fa0f57...	4490708	17 hrs 18 mins ago	0x12cb6a6bfac961e...	0xcaeabfadbd01e809...	0 Ether	0.00682424

Ilustración 70. Bloque generado en la red Rinkeby

La segunda transacción autogenerada al realizar el despliegue, ha sido la que genera el *Smart Contract Prestamista.sol* en la red *Rinkeby*. Al igual que en el caso anterior, se observa que la cuenta que aparece en el *From* es la cuenta del comprador. De igual manera, se concluye que la cuenta 0x0d6b666b6cb9289b1d3203f0dae49981b5519785 que aparece

en el *To* es sobre la que se ha generado el *Smart Contract* en *Rinkeby*. El coste de *Gas* que aparece en *Gwei* es superior a un millón, que, realizando el cambio a *Ethers*, es algo superior a 0,01 ETH. Este es el gasto que ha tenido que realizar el comprador para que se despliegue el *Smart Contract*. De forma ideal, este gasto debería haber sido realizado por el propietario de la *Dapp*, quien, en un caso real, necesitaría una cuenta con cierta cantidad de *Ether* para poder hacerse cargo de los gastos de despliegue.

Ilustración 71. Transacción Rinkeby: generación Prestamista.sol

La última transacción automática que se ha realizado para el despliegue de los *Smart Contracts* es la que genera en la red *Rinkeby Aseguradora.sol*. En el *From* se encuentra la cuenta del comprador, mientras que en el *To*, se encuentra la cuenta sobre la que se ha desplegado el *Smart Contract*, 0xb51c1c9d43cc559a1c649ff1efd786ed37088c2a en este caso. Se observa que esta transacción tiene un valor de cero *Ether*. Esto es porque el despliegue de los *Smart Contracts* en la red no tienen un coste, más allá del *Gas* que es necesario para que el contrato pueda replicarse en cada uno de los nodos que forman la red *Rinkeby*.

Ilustración 72. Transacción Rinkeby: generación Aseguradora.sol

A continuación, en la Tabla 6, se detalla el resumen de las transacciones que han generado los *Smart Contracts* en la red *Rinkeby*. Cada uno de estos *hash* pueden buscarse a través de *Etherscan* u otros buscadores en cualquier momento, y la información que se ha estudiado en este punto aparecerá siempre de forma inmutable, debido a que en *Blockchain* la información siempre queda guardada.

Transacción	Hash
Propiedad.sol	0x6200ab43a10c20df28742577a293c416cf59a6692146dd7eb06d37900046f931
Prestamista.sol	0x696b93138ab9285cd7a84ade60963b081da86262c7cdda7ca7bc170795efc1f6
Aseguradora.sol	0xbab807bbd48738cc689eb338a0196c445c44ac0df6e87f0e72620b823d5e3485

Tabla 6. Hash de las transacciones que han generado Smart Contracts

En la siguiente tabla, se detallan las cuentas sobre las que se han desplegado los *Smart Contracts*. No se debe confundir esta cuenta con el hash mostrado anteriormente. Mientras que en el caso anterior, únicamente es la transacción que despliega, visitando esta dirección de cuenta es posible observar todas las transacciones que se han llevado a cabo en esta cuenta.

Smart Contract	Cuenta
Propiedad.sol	0x0f1c9bf759108c59376dc8747d6a18f9b6d7885e
Prestamista.sol	0xd6b666b6cb9289b1d3203f0dae49981b5519785
Aseguradora.sol	0xb51c1c9d43cc559a1c649ff1efd786ed37088c2a

Tabla 7. Cuentas sobre las que se han desplegado los Smart Contracts

En el momento de desplegar desde la consola en la red *Rinkeby*, se pudo tomar nota de las direcciones en las que se habían desplegado los *Smart Contracts*. Se encuentran en la Ilustración 73 las direcciones que coinciden con los mostrados en la tabla anterior.

```
PS C:\Users\teres\Documents\propiedad_pruebaT\propiedad> truffle migrate --compile-all --reset --network rinkeby
Compiling .\contracts\Asegurador.sol...
Compiling .\contracts\Migrations.sol...
Compiling .\contracts\Prestamista.sol...
Compiling .\contracts\Propiedad.sol...
Compiling .\contracts\Titularizacion.sol...
Writing artifacts to .\build\contracts

Using network 'rinkeby'.

Running migration: 1_initial_migration.js
  Replacing Migrations...
    ...
    ...
  Migrations: 0xe99cb61a556a2e2f87b16a70556ddaa9004a8fb2c
Saving successful migration to network...
  ...
  ...
Saving artifacts...
Running migration: 2_deploy_contracts.js
  Replacing Propiedad...
    ...
    ...
  Propiedad: 0xf1c9bf759108c59376dc8747d6a18f9b6d7885e
  Replacing Prestamista...
    ...
    ...
  Prestamista: 0x696b93138ab9285cd7a84ade60963b081da86262c7cdda7ca7bc170795efc1f6
  Replacing Asegurador...
    ...
    ...
  Asegurador: 0xb51c1c9d43cc559a1c649ff1efd786ed37088c2a
Saving successful migration to network...
  ...
  ...
Saving artifacts...
PS C:\Users\teres\Documents\propiedad_pruebaT\propiedad> npm run dev
```

Ilustración 73. Despliegue desde consola en la red Rinkeby

10.2 FLUJO DE TRANSACCIONES

En este apartado, y una vez se ha realizado el despliegue en la red de cada uno de los *Smart Contracts*, se van a llevar a cabo las transacciones que se han mostrado al comienzo del capítulo.

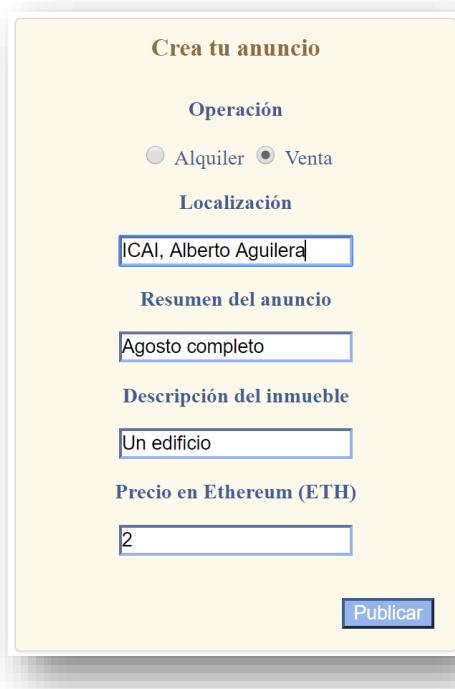
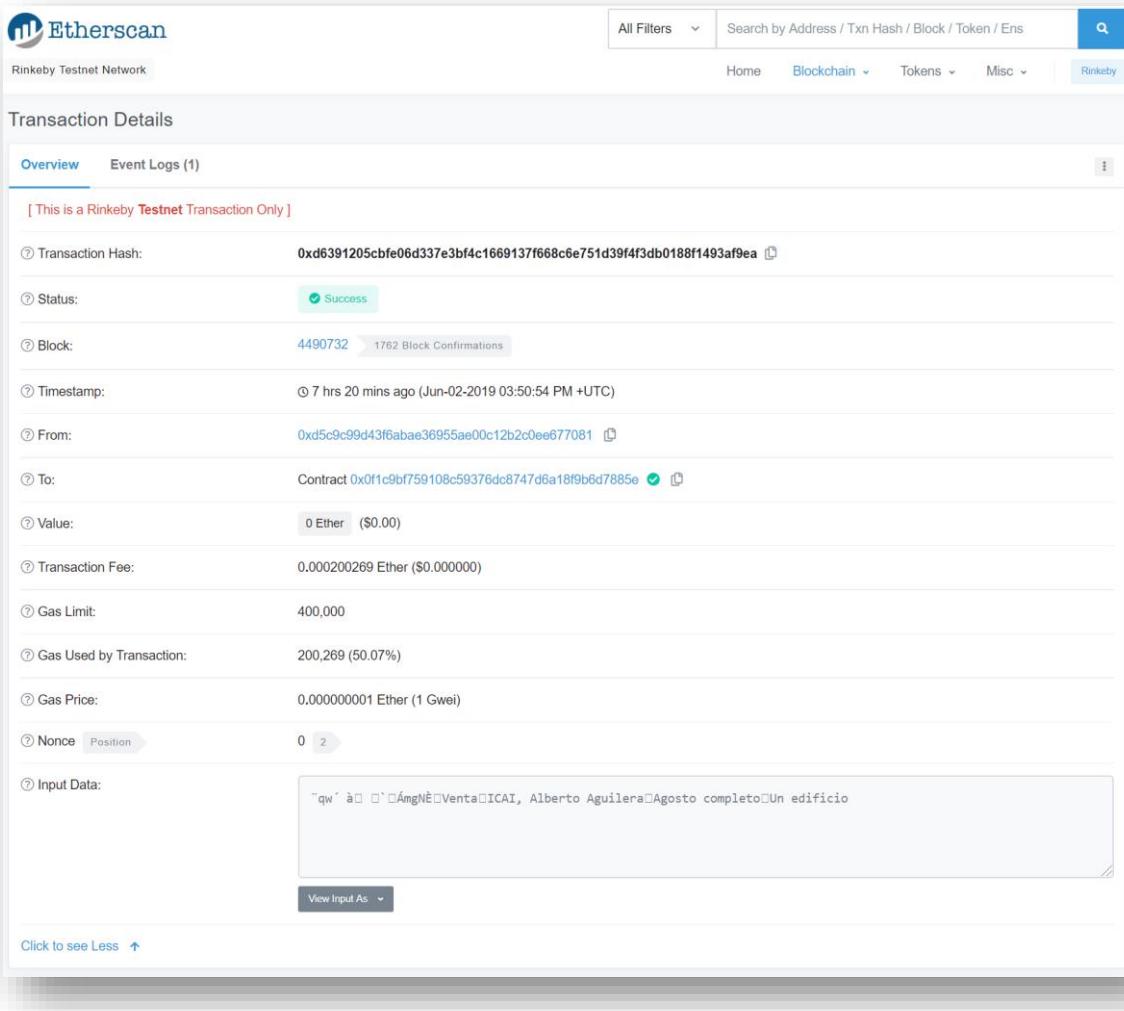


Ilustración 74: Puesta en venta del primer inmueble

En el primer paso del flujo, el vendedor pone a la venta el primer inmueble. En la Ilustración 74 se realiza la puesta en venta desde la *Dapp*, mientras que en la Ilustración 75 puede observarse el detalle de esta transacción desde *Etherscan*.

Se advierte cómo la cuenta del *From* es la del vendedor, mientras que el *To* es la cuenta de *Propiedad.sol*. Esta transacción tiene un valor de cero *Ether*, aunque por ella, el comprador ha realizado un gasto de *Gas* de más de 200.000 *Gwei*. En el *Input Data*, se detallan las características que se incluían en el *front* de la aplicación, ya que en *Blockchain* posibilita la transparencia.



The screenshot shows a transaction details page on Etherscan for a Rinkeby Testnet transaction. The transaction hash is 0xd6391205cbfe06d337e3bf4c1669137f668c6e751d39f4f3db0188f1493af9ea. The status is Success. It occurred in block 4490732, which has 1762 block confirmations. The timestamp is 7 hrs 20 mins ago (Jun-02-2019 03:50:54 PM +UTC). The transaction originated from address 0xd5c9c99d43f6abae36955ae00c12b2c0ee677081 and was sent to a contract at 0x0f1c9bf759108c59376dc8747d6a18f9b6d7885e. The value was 0 Ether (\$0.00). The transaction fee was 0.000200269 Ether (\$0.000000). The gas limit was 400,000, and the gas used was 200,269 (50.07%). The gas price was 0.00000001 Ether (1 Gwei). The nonce was 0, and the position was 2. The input data field contains the text: "qw à Ámp; Venta ICAI, Alberto Aguilera Agosto completo Un edificio". A note indicates "[This is a Rinkeby Testnet Transaction Only]".

Ilustración 75: Transacción Rinkeby: Venta primer inmueble

Además, si se observa el *log* de eventos en la Ilustración 76, se pueden encontrar los parámetros indexados en la zona de *topics*, correspondiendo el primer valor con el ID del anuncio y el segundo valor con la dirección de la cuenta del vendedor, tal y como se implementó en el *Smart Contract*.

Ilustración 76: Transacción Rinkeby: Log de eventos venta primer inmueble

Estableciendo el foco de atención en la parte superior de la *Dapp*, se puede observar cómo ha disminuido el dinero de la cuenta del vendedor, al realizar el gasto de *Gas* que conlleva la publicación del anuncio.

APROVECHAMIENTO POR TURNOS DE UN INMUEBLE



Ilustración 77: Cuenta de vendedor antes de realizar anuncio



Ilustración 78: Cuenta de vendedor después de realizar anuncio

Esto también puede visualizarse desde *Etherscan*. Si en el buscador se localiza la cuenta de vendedor, se observan las transacciones que ha llevado a cabo. Se aprecian las dos ocasiones en las que se han incluido 18,75 ETH en la cuenta, así como la publicación del anuncio. En la parte superior, se localiza el balance actual de la cuenta.

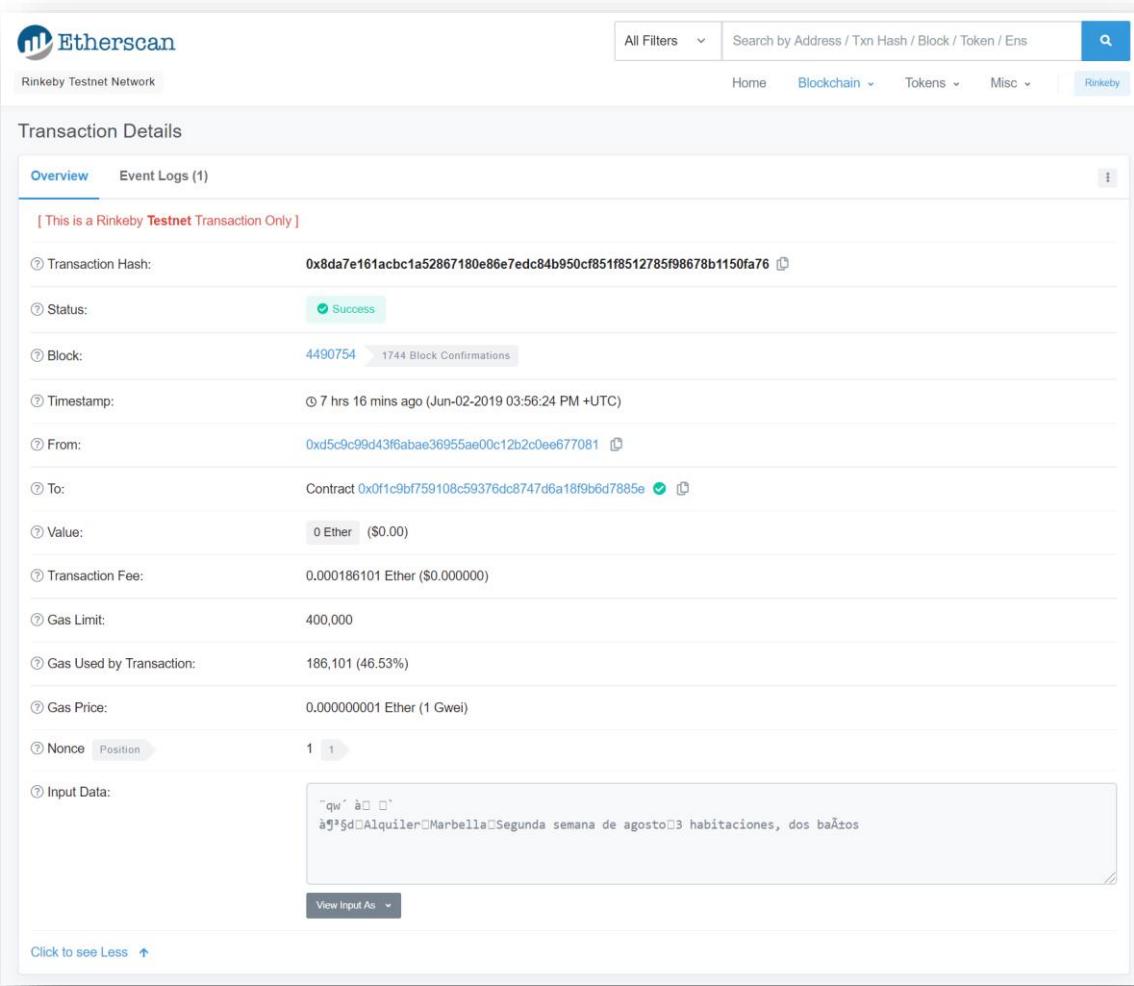
Txn Hash	Block	Age	From	To	Value	[Txn Fee]
0xd6391205cbfe06d...	4490732	2 mins ago	0xd5c9c99d43f6aba...	0x0f1c9bf759108c5...	0 Ether	0.000200269
0xb99b4d96c3b792...	4075024	72 days 4 hrs ago	0x31b98d14007bde...	0xd5c9c99d43f6aba...	18.75 Ether	0.000021
0x9fb32d2d1d1ae2...	3886912	104 days 20 hrs ago	0x31b98d14007bde...	0xd5c9c99d43f6aba...	18.75 Ether	0.000021

Ilustración 79: Dirección de vendedor en Etherscan

A continuación, se realiza el segundo paso del flujo comentado. El vendedor publica el segundo inmueble, siendo en este caso una operación del tipo de alquiler. Al igual que en

la transacción anterior, la cuenta del *From* es la del vendedor, mientras que el *To* es la cuenta de *Propiedad.sol*.

Se muestra en la siguiente ilustración los valores principales de esta transacción, aunque no se muestra a fondo el *log* de eventos, al ser muy similar al paso anterior.



The screenshot shows the Etherscan interface for a Rinkeby Testnet transaction. The transaction details are as follows:

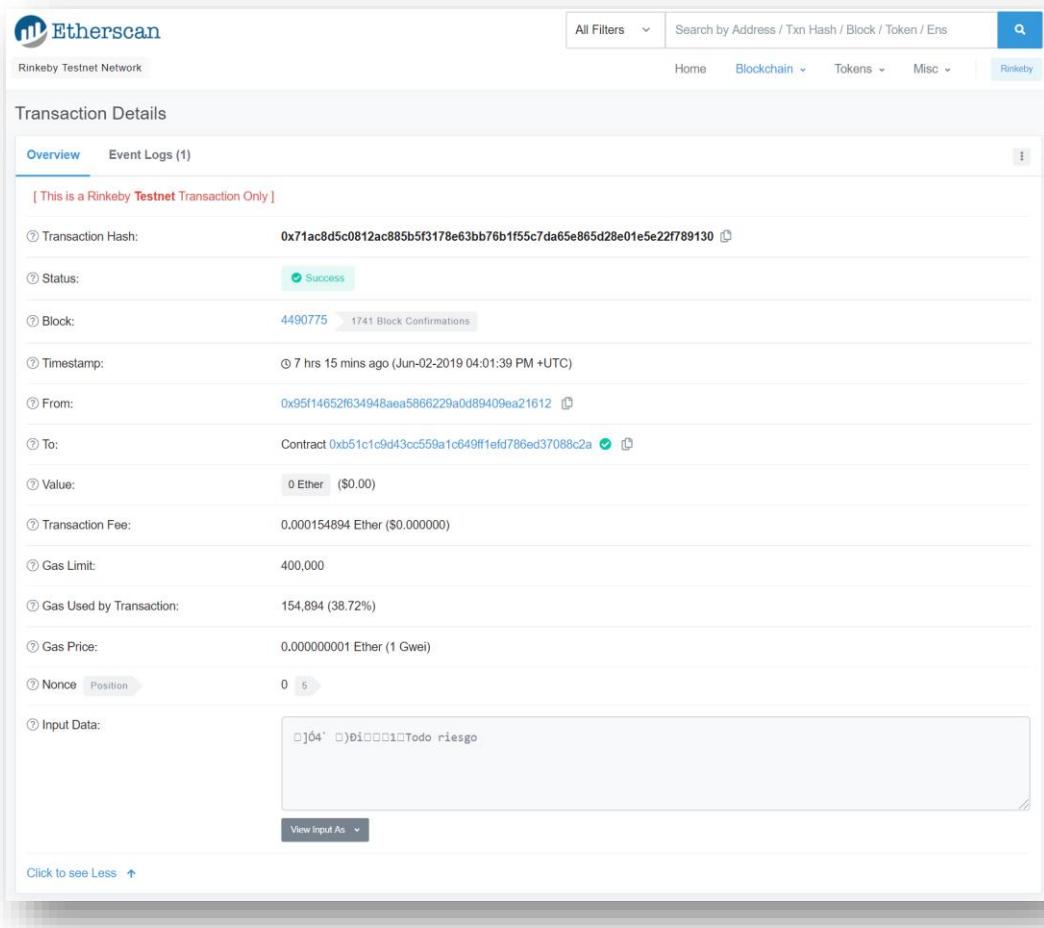
- Transaction Hash: 0x8da7e161acbc1a52867180e86e7edc84b950cf851f8512785f98678b1150fa76
- Status: Success
- Block: 4490754 (1744 Block Confirmations)
- Timestamp: ⑤ 7 hrs 16 mins ago (Jun-02-2019 03:56:24 PM +UTC)
- From: 0xd5c9c99d43f6abae36955aae00c12b2c0ee677081
- To: Contract 0x0f1c9bf759108c59376dc8747d6a18f9b6d7885e
- Value: 0 Ether (\$0.00)
- Transaction Fee: 0.000186101 Ether (\$0.000000)
- Gas Limit: 400,000
- Gas Used by Transaction: 186,101 (46.53%)
- Gas Price: 0.000000001 Ether (1 Gwei)
- Nonce: 1
- Input Data: "qw' □□" à¤§d□Alquiler□Marbella□Segunda semana de agosto□3 habitaciones, dos baños

Ilustración 80: Transacción Rinkeby: Alquiler segundo inmueble

Posteriormente, en el paso 3 del flujo, el asegurador publica una oferta de póliza de seguro para el anuncio de inmueble con ID 1, puesto que es un anuncio con una operación

en venta. En este caso, observamos que la cuenta del *From* es la de la aseguradora, mientras que el *To* es la cuenta de *Aseguradora.sol*.

Este paso también consiste en una publicación, por lo que no existen grandes diferencias con la publicación de anuncio de los inmuebles, más allá de las cuentas que intervienen en la operación y los detalles que se guardan en el *Input Data* y en el *log* de eventos. En el Código 5 que se explicó en el capítulo 9, se observa que en el *log* se guardan de forma indexada el ID del seguro y la dirección del asegurador, mientras que el ID anuncio de referencia y el precio se guardan sin indexar.



The screenshot shows a detailed view of a Ethereum transaction on the Rinkeby Testnet. The transaction hash is 0x71ac8d5c0812ac885b5f3178e63bb76b1f55c7da65e865d28e01e5e22f789130. It was successful and included in block 4490775 with 1741 confirmations. The transaction occurred 7 hours and 15 minutes ago on June 02, 2019, at 04:01:39 PM UTC. The transaction originated from the address 0x95f14652f634948aea5866229a0d89409ea21612 and was sent to the contract address 0xb51c1c9d43cc559a1c649ff1efd706ed37088c2a. The value transferred was 0 Ether (\$0.00). The transaction fee was 0.000154894 Ether (\$0.000000). The gas limit was 400,000, and the gas used was 154,894 (38.72%). The gas price was 0.000000001 Ether (1 Gwei). The nonce was 0. The input data field contains the string: "JÓ4')Đí]]]]] Todo riesgo".

Ilustración 81: Transacción Rinkeby: Seguro primer inmueble

Ilustración 82: Transacción Rinkeby: Log de eventos seguro primer inmueble

En el cuarto paso del flujo, el comprador, realiza una petición de préstamo para poder comprar el anuncio en venta. Se observa la transacción de forma similar a las anteriores.

Ilustración 83: Transacción Rinkeby: Préstamo primer inmueble

Ilustración 84: Transacción Rinkeby: Log de eventos préstamo primer inmueble

Antes de que se publique la transacción, mediante el navegador, aparece una notificación de *MetaMask*. En esta notificación, se muestra la cuenta origen, la cuenta destino, el valor de la transacción y el gasto en *Gas*. Será necesario confirmar esta notificación para que se publique la transacción.

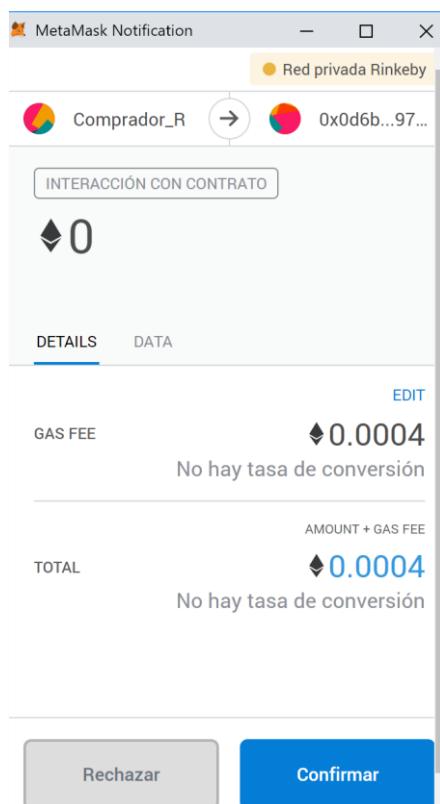


Ilustración 85. Notificación de *MetaMask* previa a transacción

En el paso 5 de la transacción, el prestatario observa las peticiones de préstamos publicadas y se ve interesado por la petición de préstamo realizada en el paso 4, por lo que procede a aceptarla.

Esta transacción liberará la cláusula del *Smart Contract*, con lo cual, el comprador recibirá el pago. La dirección de la cuenta en la que se ha desplegado *Prestamista.sol* será la encargada de hacer de intermediario en este intercambio de criptomonedas que sucede entre comprador y prestamista.

Ilustración 86: Transacción Rinkeby: Aceptación préstamo primer inmueble

Etherscan

Rinkeby Testnet Network

All Filters Search by Address / Txn Hash / Block / Token / Ens

Home Blockchain Tokens Misc Rinkeby

Transaction Details

Overview Internal Transactions Event Logs (1)

Note: There is limited (Beta) support for tracking Internal Transactions on Rinkeby.

The Contract Call From [0xa44199021303e1...](#) To [0x0d6b666b6cb928...](#) Produced 1 Contract Internal Transaction :

Type	Trace	Address	From	To	Value	Gas Limit
↳	call	_0_1	0x0d6b666b6cb928...	0x4911699f8d934a1...	0.5 Ether	0

Ilustración 87: Transacción Rinkeby: Interna aceptación préstamo primer inmueble

En la vista *Overview* de la transacción se observa que esta transacción tiene un valor de 0,5 *Ethers*, además de que la cuenta del *From* es la del prestamista, mientras que el *To* es la cuenta de *Prestamista.sol*. Sin embargo, en esta misma vista se identifica en el *To* referencia a un *Transfer*. Este movimiento puede estudiarse mejor desde la pestaña de *Internal Transactions*, donde aparece la ejecución del *Smart Contract*. Aquí, puede observarse que la cuenta del *From* es la de *Prestamista.sol*, mientras que el *To* es la cuenta de comprador, existiendo un valor de movimiento de 0,5 *Ethers*, que es la cantidad que se indicaba en la petición del paso 4. Al existir una transacción de 0,5 *Ethers* entrantes en el *Smart Contract*, pero liberarse la ejecución de tal forma que también salen 0,5 *Ethers* de éste, el *Smart Contract* queda con un balance de 0 *Ethers*.

En el sexto paso del flujo, el comprador observa los anuncios de pólizas de seguros ofrecidos para la futura casa a comprar, y le entusiasma la póliza ofrecida en el paso 3, por lo que procede a contratarla.



Ilustración 88. Contratación de póliza de seguro desde el Front

Al ejecutar la contratación del seguro, el comprador realizará una transacción de 0,3 Ethers hacia el *Smart Contract*, que liberará la cláusula que ejecutará el pago de la misma cantidad a la aseguradora, tal como sucedía en el paso anterior entre prestamista y prestatario.

Ilustración 89: Transacción Rinkeby: Contratación seguro primer inmueble

Etherscan		All Filters	Search by Address / Txn Hash / Block / Token / Ens	🔍																
Rinkeby Testnet Network		Home	Blockchain	Tokens	Misc	Rinkeby														
Transaction Details																				
Overview	Internal Transactions	Event Logs (1)																		
<p>Note: There is limited (Beta) support for tracking Internal Transactions on Rinkeby.</p> <p>The Contract Call From 0x4911699f8d934a1... To 0xb51c1c9d43cc55... Produced 1 Contract Internal Transaction :</p> <table><thead><tr><th>Type</th><th>Trace</th><th>Address</th><th>From</th><th>To</th><th>Value</th><th>Gas Limit</th></tr></thead><tbody><tr><td>↓</td><td>call</td><td>0_1</td><td>0xb51c1c9d43cc55...</td><td>0x95f14652f634948...</td><td>0.3 Ether</td><td>0</td></tr></tbody></table>	Type	Trace	Address	From	To	Value	Gas Limit	↓	call	0_1	0xb51c1c9d43cc55...	0x95f14652f634948...	0.3 Ether	0						
Type	Trace	Address	From	To	Value	Gas Limit														
↓	call	0_1	0xb51c1c9d43cc55...	0x95f14652f634948...	0.3 Ether	0														

Ilustración 90: Transacción Rinkeby: Interna contratación seguro primer inmueble

Previo a la ejecución de la transacción, *MetaMask* realiza una notificación. En este caso, aparece como valor de la misma el precio del seguro que acaba de contratar el comprador.

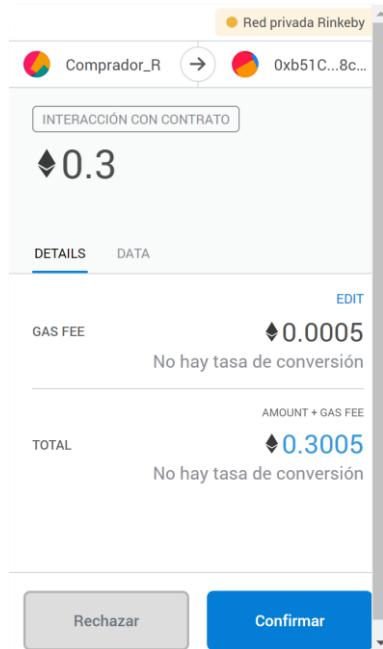


Ilustración 91. Notificación de MetaMask a la hora de contratar el seguro

En el paso 7, tras el comprador haber obtenido el préstamo y el seguro para el hogar, procede a comprar el inmueble. Se muestran a continuación las imágenes de la transacción.

Ilustración 92: Transacción Rinkeby: Compra primer inmueble

Etherscan

Rinkeby Testnet Network

All Filters Search by Address / Txn Hash / Block / Token / Erc

Home Blockchain Tokens Misc Rinkeby

Transaction Details

Overview Internal Transactions Event Logs (1)

Note: There is limited (Beta) support for tracking Internal Transactions on Rinkeby.

The Contract Call From [0x4911699f8d934a1...](#) To [0x0f1c9bf759108c5...](#) Produced 1 Contract Internal Transaction :

Type	Trace	Address	From	To	Value	Gas Limit
↳	call_0_1		0x0f1c9bf759108c5...	0xd5c9c99d43f6aba...	2 Ether	0

Ilustración 93: Transacción Rinkeby: Interna compra primer inmueble

Rinkeby Testnet Network

All Filters Search by Address / Txn Hash / Block / Token / Ens

Home Blockchain Tokens Misc Rinkeby

Transaction Details

Overview Internal Transactions **Event Logs (1)**

Transaction Receipt Event Logs

6

Address 0x0f1c9bf759108c59376dc8747d6a18f9b6d7885e

Topics

- 0 → 0x07119d3da1c6f553a9305cc42f4a4883705beb5ab6a0b6277e3755b67df517d2
- 1 → 0x0001
- 2 → 0x00000000000000000000000000000000d5c9c99d43f6abae36955ae00c12b2c0ee677081
- 3 → 0x000000000000000000000000000000004911699f8d934a168124394d6501b41d9b6cf195

Data

- Num → 64
- Num → 20000000000000000000
- Num → 15
- Text → Agosto completo

Ilustración 94: Transacción Rinkeby: Log de eventos compra primer inmueble

Analizando las ilustraciones superiores, se observa cómo la transacción tiene un valor de 2 *Ethers*, la cuenta del *From* es la del comprador, mientras que el *To* es la cuenta de *Propiedad.sol*, y que además esta transacción ha liberado la ejecución del *Smart Contract*, que tiene en la cuenta del *From* la de *Propiedad.sol*, mientras que el *To* es la cuenta del vendedor, también con un valor de 2 *Ethers*.

Por último, si se observa el *log* de eventos, ahora se detallan tres valores distintos indexados en los *topics*. Esto corresponde a la definición de *log* de compra en *Solidity*, que además de los valores que incluye el *log* de venta, incluye también de forma indexada la dirección del comprador. Es decir, ahora existe un *topic* más.

A continuación, y de tal forma que se compruebe todo lo explicado a lo largo de este punto, se muestra el octavo paso del flujo, en el que el vendedor desea arrendar un inmueble nuevo durante la segunda semana de julio, por lo que desea publicarlo en el portal.

 Etherscan

Rinkeby Testnet Network

All Filters Search by Address / Txn Hash / Block / Token / Ens

Home Blockchain Tokens Misc Rinkeby

Transaction Details

[Overview](#) Event Logs (1)

[This is a Rinkeby **Testnet** Transaction Only]

② Transaction Hash: [0x7d6f7ea1fef280a6f25daf494c5e6df1b76957def8bb8083f9de5218650e8899](#)

② Status: Success

② Block: [4490882](#) 1672 Block Confirmations

② Timestamp: [6 hrs 58 mins ago \(Jun-02-2019 04:28:24 PM +UTC\)](#)

② From: [0xd5c9c99d43f6abae36955ae00c12b2c0ee677081](#)

② To: Contract [0xf1c9bf759108c59376dc8747d6a18fb6d7885e](#) ✓

② Value: 0 Ether (\$0.00)

② Transaction Fee: 0.000186037 Ether (\$0.000000)

② Gas Limit: 400,000

② Gas Used by Transaction: 186,037 (46.51%)

② Gas Price: 0.000000001 Ether (1 Gwei)

② Nonce Position: 2

② Input Data:

qw' à □ □' □Ñ□
{□□Alquiler
Formentera□Segunda semana de julio□2 habitaciones, 2 baños

Click to see Less

Ilustración 95: Transacción Rinkeby: Alquiler tercer inmueble

En el paso número 9 del flujo, el comprador desea alquilar el inmueble publicado en el paso anterior, por lo que procede a alquilar el inmueble, tal y como se muestra en las siguientes imágenes.

Ilustración 96: Transacción Rinkeby: Alquilado tercer inmueble

Etherscan

Rinkeby Testnet Network

All Filters ▾ Search by Address / Txn Hash / Block / Token / Ens 🔍

Home Blockchain ▾ Tokens ▾ Misc ▾ Rinkeby

Transaction Details

Overview Internal Transactions Event Logs (1) ⋮

Note: There is limited (Beta) support for tracking Internal Transactions on Rinkeby.

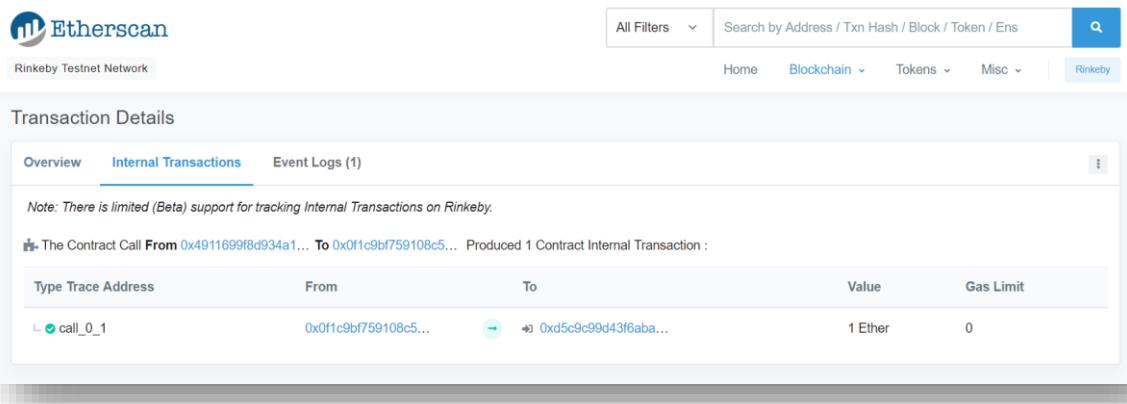
info The Contract Call From [0x4911699f8d934a1...](#) To [0x0f1c9bf759108c5...](#) Produced 1 Contract Internal Transaction :

Type	Trace Address	From	To	Value	Gas Limit
↳	call_0_1	0x0f1c9bf759108c5...	0xd5c9c99d43f0aba...	1.5 Ether	0

Ilustración 97: Transacción Rinkeby: Interna alquilado tercer inmueble

Por último, en el décimo paso del flujo, el arrendatario está interesado en el anuncio de alquiler publicado en el paso 2, por lo que decide alquilarlo. Esta transacción liberará la ejecución del *Smart Contract*.

Ilustración 98: Transacción Rinkeby: Alquilado segundo inmueble



Type	Trace	Address	From	To	Value	Gas Limit
call	0x0f1c9bf759108c5...	0xd5c9c99d43f6aba...			1 Ether	0

Ilustración 99: Transacción Rinkeby: Interna alquilado segundo inmueble

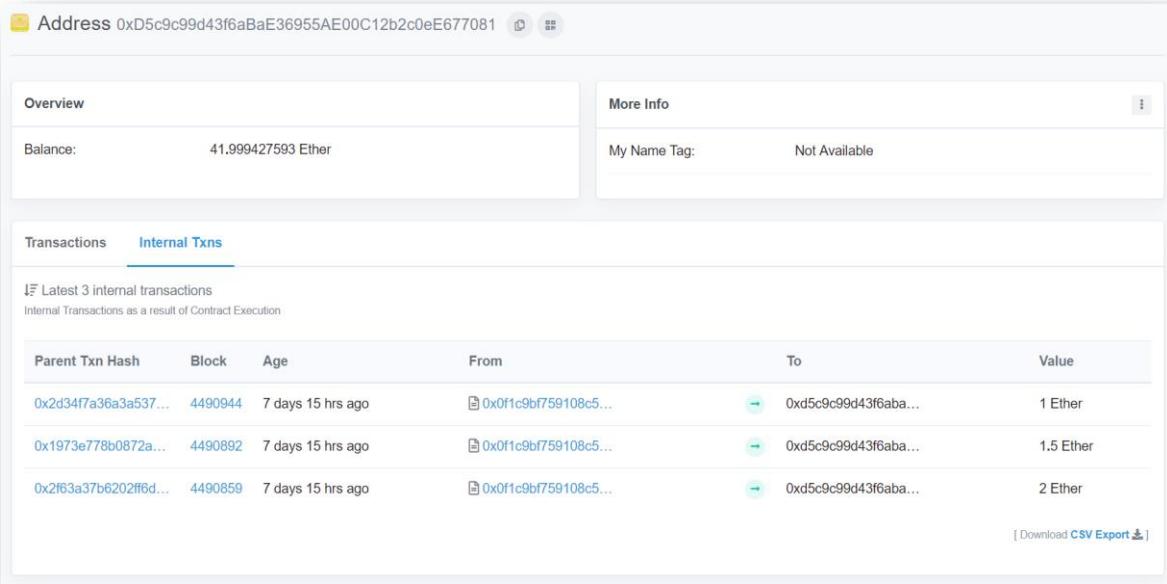
A continuación, se muestra la relación de las transacciones realizadas en el flujo de la actividad diseñado. Estas transacciones siempre estarán disponibles para ser buscadas y analizadas en *Rinkeby* a través de *Etherscan*.

<i>Transacción</i>	<i>Hash</i>
Paso 1	0xd6391205cbfe06d337e3bf4c1669137f668c6e751d39f4f3db0188f1493af9ea
Paso 2	0x8da7e161acbc1a52867180e86e7edc84b950cf851f8512785f98678b1150fa76
Paso 3	0x71ac8d5c0812ac885b5f3178e63bb76b1f55c7da65e865d28e01e5e22f789130
Paso 4	0xffe1339a42b484df6d667a6dc687ba8a55f6e038e49125bd962b78e2a5a4e7b
Paso 5	0x8ed4f6edd5d49798c179a608ea2f5411030d12074850099d4e62bf2f93f4613
Paso 6	0xe649c812993b2ac4fb34dc7a51b52e7baf3801796bfd589c6fa1e7d4ec8f5224
Paso 7	0x2f63a37b6202ff6d2ca88d193a784fce3aab12e3d4935b398c8b3e0419226274
Paso 8	0x7d6f7ea1fef280a6f25daf494c5e6df1b76957def8bb8083f9de5218650e8899
Paso 9	0x1973e778b0872ad486d1038c1453d914818253b6e84ac086799e12e4153119e0
Paso 10	0x2d34f7a36a3a53736651155e936c66ab785175eaf16bbdc55e27618bd4719c87

Tabla 8: Relación transacciones hash en el flujo Rinkeby

10.3 TRANSACCIONES INTERNAS DE LOS SMART CONTRACTS

Se procede a analizar a través del número de cuenta del vendedor las transacciones internas realizadas, se observa cómo se corresponden con el paso 7, que es donde se produce la venta, y en los pasos 9 y 10 que es donde se alquilan los inmuebles. Es decir, las transacciones donde se intercambia el dinero del comprador al vendedor.



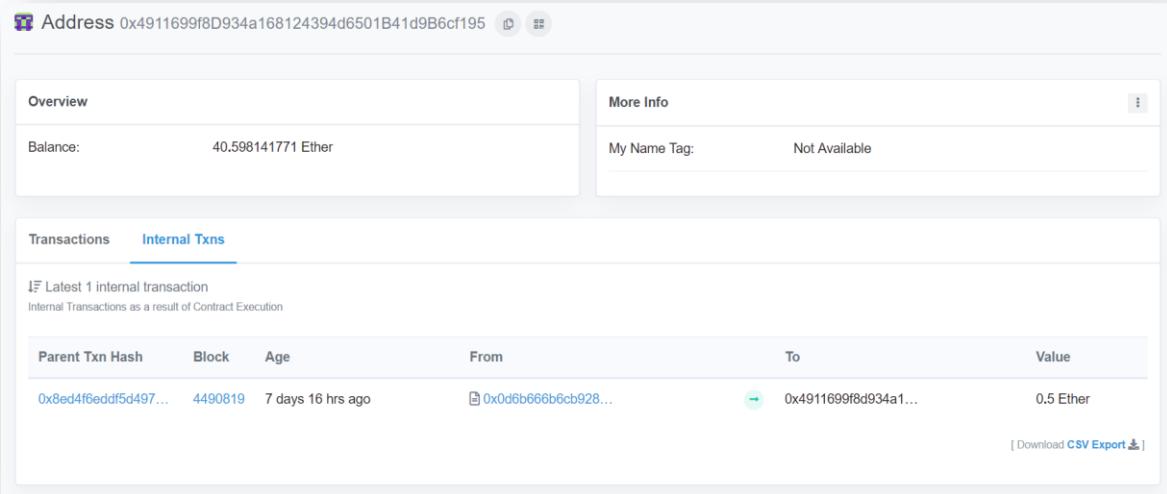
The screenshot shows a blockchain transaction details page for a seller account. The address is 0xD5c9c99d43f6aBaE36955AE00C12b2c0eE677081. The balance is 41.999427593 Ether. There is no name tag available. The page displays the latest 3 internal transactions as a result of contract execution.

Parent Txn Hash	Block	Age	From	To	Value
0xd3417a36a3a537...	4490944	7 days 15 hrs ago	0x0f1c9bf759108c5...	0xd5c9c99d43f6aba...	1 Ether
0x1973e778b0872a...	4490892	7 days 15 hrs ago	0x0f1c9bf759108c5...	0xd5c9c99d43f6aba...	1.5 Ether
0x2f63a37b6202ff6d...	4490859	7 days 15 hrs ago	0x0f1c9bf759108c5...	0xd5c9c99d43f6aba...	2 Ether

[Download CSV Export]

Ilustración 100: Transacción Rinkeby: interna del vendedor

En el caso de la cuenta del comprador, la transacción interna se corresponde con el paso 5, cuando el prestamista acepta la solicitud del prestatario, que en este caso el prestatario es el comprador.



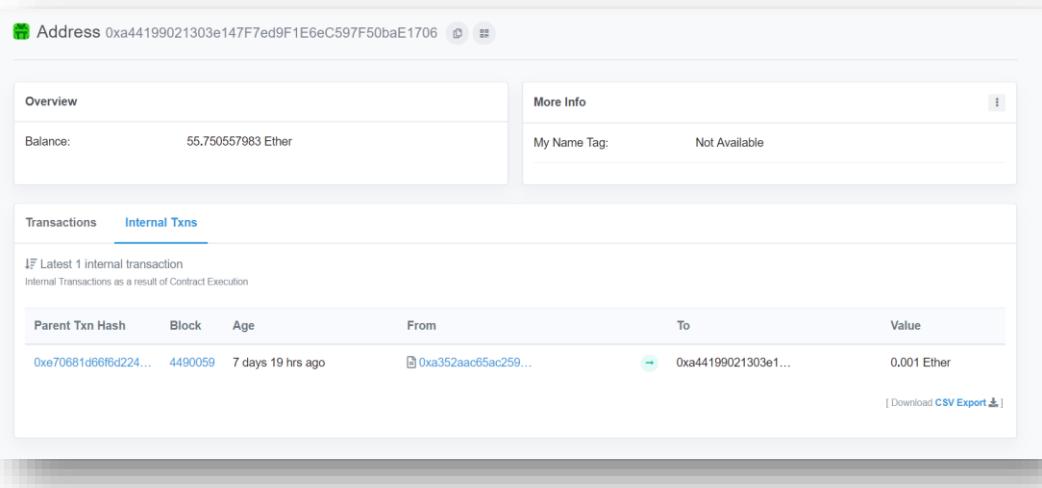
The screenshot shows a blockchain transaction details page for a buyer account. The address is 0x4911699f8D934a168124394d6501B41d9B6cf195. The balance is 40.598141771 Ether. There is no name tag available. The page displays the latest 1 internal transaction as a result of contract execution.

Parent Txn Hash	Block	Age	From	To	Value
0x8ed4f6eddf5d497...	4490819	7 days 16 hrs ago	0x0d6b666b6cb928...	0x4911699f8d934a1...	0.5 Ether

[Download CSV Export]

Ilustración 101: Transacción Rinkeby: interna del comprador

Para la cuenta del prestamista, le aparece una transacción, puesto que se realizaron pruebas antes de realizar el flujo. Además, si se visualizan las horas está se ha producido 3 horas antes que el resto. Por tanto, no pertenece a este flujo.

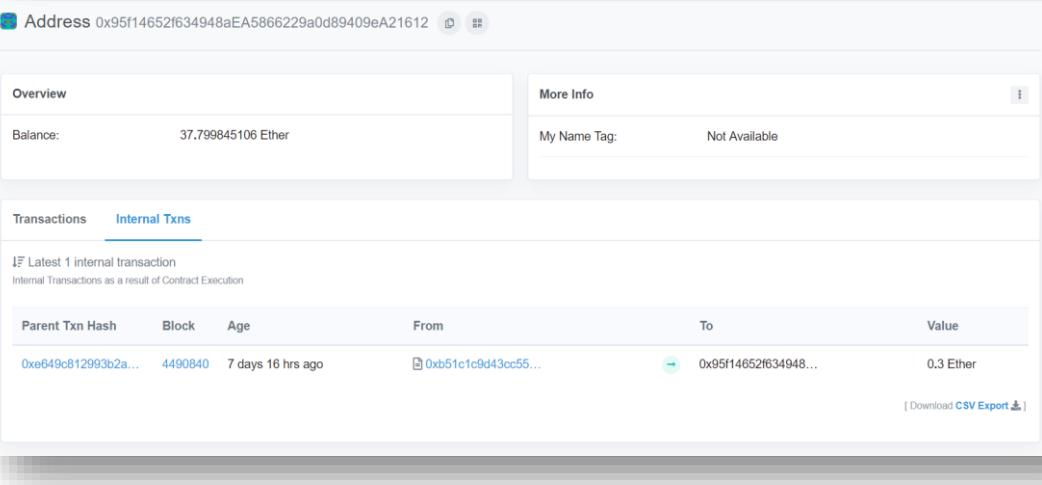


The screenshot shows a blockchain transaction details page for the address 0xa44199021303e147F7ed9F1E6eC597F50baE1706. The 'Internal Txns' tab is selected, showing one internal transaction from 0xe70681d66f6d224... at block 4490059, 7 days 19 hrs ago, to 0xa44199021303e1..., with a value of 0.001 Ether. The 'Transactions' tab shows no external transactions.

Parent Txn Hash	Block	Age	From	To	Value
0xe70681d66f6d224...	4490059	7 days 19 hrs ago	0xa352aac65ac259...	0xa44199021303e1...	0.001 Ether

Ilustración 102: Transacción Rinkeby: interna del prestamista

En la dirección del asegurador, la transacción existente hace referencia al paso 6, donde se produce la aceptación de la póliza por parte del comprador y se transfieren los fondos al asegurador.



The screenshot shows a blockchain transaction details page for the address 0x95f14652f634948aEA5866229a0d89409eA21612. The 'Internal Txns' tab is selected, showing one internal transaction from 0xb51c1c9d43cc55... at block 4490840, 7 days 16 hrs ago, to 0x95f14652f634948..., with a value of 0.3 Ether. The 'Transactions' tab shows no external transactions.

Parent Txn Hash	Block	Age	From	To	Value
0xe649c812993b2a...	4490840	7 days 16 hrs ago	0xb51c1c9d43cc55...	0x95f14652f634948...	0.3 Ether

Ilustración 103: Transacción Rinkeby: interna del asegurador

Para concluir, se muestra a continuación las transacciones realizadas en el *Smart Contract Propiedad.sol*. Esta información se ha obtenido a través de la búsqueda del número del cuenta del *Smart Contract*. La siguiente ilustración nos permitirá relacionar los hash de las transacciones realizadas con las que aparecen en ella.

Contract Overview								More Info																																																									
Transactions				Internal Txns		Contract		Events																																																									
View Latest 7 txns								More																																																									
<table border="1"> <thead> <tr> <th>Txn Hash</th> <th>Block</th> <th>Age</th> <th>From</th> <th>To</th> <th>Value</th> <th>[Txn Fee]</th> </tr> </thead> <tbody> <tr> <td>0xd34f7a36a3a537...</td> <td>4490944</td> <td>7 days 15 hrs ago</td> <td>0x4911699f8d934a1...</td> <td>0x0f1c9bf759108c5...</td> <td>1 Ether</td> <td>0.000055517</td> </tr> <tr> <td>0x1973e778b0872a...</td> <td>4490892</td> <td>7 days 16 hrs ago</td> <td>0x4911699f8d934a1...</td> <td>0x0f1c9bf759108c5...</td> <td>1.5 Ether</td> <td>0.000055517</td> </tr> <tr> <td>0x7d6f7ea1fef280a6...</td> <td>4490882</td> <td>7 days 16 hrs ago</td> <td>0xd5c9c99d43f6aba...</td> <td>0x0f1c9bf759108c5...</td> <td>0 Ether</td> <td>0.000186037</td> </tr> <tr> <td>0x2f63a37b6202ff6d...</td> <td>4490859</td> <td>7 days 16 hrs ago</td> <td>0x4911699f8d934a1...</td> <td>0x0f1c9bf759108c5...</td> <td>2 Ether</td> <td>0.000055517</td> </tr> <tr> <td>0x8da7e161acbc1a...</td> <td>4490754</td> <td>7 days 16 hrs ago</td> <td>0xd5c9c99d43f6aba...</td> <td>0x0f1c9bf759108c5...</td> <td>0 Ether</td> <td>0.000186101</td> </tr> <tr> <td>0xd391205cbfe06d...</td> <td>4490732</td> <td>7 days 16 hrs ago</td> <td>0xd5c9c99d43f6aba...</td> <td>0x0f1c9bf759108c5...</td> <td>0 Ether</td> <td>0.000200269</td> </tr> <tr> <td>0x6200ab43a10c20...</td> <td>4490708</td> <td>7 days 16 hrs ago</td> <td>0x4911699f8d934a1...</td> <td>Contract Creation</td> <td>0 Ether</td> <td>0.01263059</td> </tr> </tbody> </table>								Txn Hash	Block	Age	From	To	Value	[Txn Fee]	0xd34f7a36a3a537...	4490944	7 days 15 hrs ago	0x4911699f8d934a1...	0x0f1c9bf759108c5...	1 Ether	0.000055517	0x1973e778b0872a...	4490892	7 days 16 hrs ago	0x4911699f8d934a1...	0x0f1c9bf759108c5...	1.5 Ether	0.000055517	0x7d6f7ea1fef280a6...	4490882	7 days 16 hrs ago	0xd5c9c99d43f6aba...	0x0f1c9bf759108c5...	0 Ether	0.000186037	0x2f63a37b6202ff6d...	4490859	7 days 16 hrs ago	0x4911699f8d934a1...	0x0f1c9bf759108c5...	2 Ether	0.000055517	0x8da7e161acbc1a...	4490754	7 days 16 hrs ago	0xd5c9c99d43f6aba...	0x0f1c9bf759108c5...	0 Ether	0.000186101	0xd391205cbfe06d...	4490732	7 days 16 hrs ago	0xd5c9c99d43f6aba...	0x0f1c9bf759108c5...	0 Ether	0.000200269	0x6200ab43a10c20...	4490708	7 days 16 hrs ago	0x4911699f8d934a1...	Contract Creation	0 Ether	0.01263059	Download CSV Export	
Txn Hash	Block	Age	From	To	Value	[Txn Fee]																																																											
0xd34f7a36a3a537...	4490944	7 days 15 hrs ago	0x4911699f8d934a1...	0x0f1c9bf759108c5...	1 Ether	0.000055517																																																											
0x1973e778b0872a...	4490892	7 days 16 hrs ago	0x4911699f8d934a1...	0x0f1c9bf759108c5...	1.5 Ether	0.000055517																																																											
0x7d6f7ea1fef280a6...	4490882	7 days 16 hrs ago	0xd5c9c99d43f6aba...	0x0f1c9bf759108c5...	0 Ether	0.000186037																																																											
0x2f63a37b6202ff6d...	4490859	7 days 16 hrs ago	0x4911699f8d934a1...	0x0f1c9bf759108c5...	2 Ether	0.000055517																																																											
0x8da7e161acbc1a...	4490754	7 days 16 hrs ago	0xd5c9c99d43f6aba...	0x0f1c9bf759108c5...	0 Ether	0.000186101																																																											
0xd391205cbfe06d...	4490732	7 days 16 hrs ago	0xd5c9c99d43f6aba...	0x0f1c9bf759108c5...	0 Ether	0.000200269																																																											
0x6200ab43a10c20...	4490708	7 days 16 hrs ago	0x4911699f8d934a1...	Contract Creation	0 Ether	0.01263059																																																											

Ilustración 104: Transacción Rinkeby: Cuenta Smart Contract Propiedad.sol

Desde arriba hacia abajo, la transacción donde se alquiló el segundo inmueble, la siguiente es la transacción correspondiente al inmueble tercero alquilado, la transacción de publicación del alquiler del tercer inmueble, a transacción donde se transfieren los fondos de la venta del inmueble primero, puesta en alquiler del segundo inmueble, la publicación del anuncio en venta del primer inmueble y la última corresponde a la creación y despliegue del *Smart Contract*.

Capítulo 11. **ALCANCE COMERCIAL**

Actualmente las criptomonedas tienen una volatilidad alta. Este hecho puede verse afectado por algunas de las siguientes situaciones:

– **Especulación**

Las opiniones de los inversores con prestigio de los *criptoactivos* han supuesto un comportamiento alcista o a la baja del valor de las criptomonedas. Además, unido a la facilidad de obtener dinero a corto plazo, comentado en los grandes medios de comunicación, han convocado a las masas para verlo como un sistema de inversión similar a la bolsa, pero con un riesgo en algunas ocasiones más elevado. [83]

– **Oferta y demanda**

La mayoría de las criptomonedas tienen una masa monetaria limitada. Por ejemplo, en el caso de la criptomoneda de *Bitcoin* su número máximo de *Bitcoin* es de 21 millones. El hecho de no poder seguir generando un mayor número de criptomoneda, unido al equilibrio que debe existir entre la oferta y la demanda, hace que las criptomonedas tiendan a la deflación y a tener un valor constante. Por ejemplo, en el año 2018, al aumentar la fama de las criptomonedas, supuso que comenzase a disminuir la cantidad restante por minarse de una forma más radical, lo que produjo un aumento de su valor. [83]

– **Regulaciones**

Las regulaciones suponen una fuente importante de seguridad y por lo tanto, cuando se lanza una nueva ley que afecta a las criptomonedas, produce cambios en el valor de las mismas. Por ejemplo, en Septiembre de 2017, China anunció la prohibición de rondas de inversión basadas en *ICOs* (*Initial Coin Offerings*), y esto llevó a que el precio del *Ethereum* bajase cerca de un 12% de su valor de forma inmediata.

Tal como comentó José García Caballero, el creador de *Real Fund*, existe gran deficiencia en España acerca de los marcos jurídicos para el desarrollo de procesos de *tokenización* y *tokenomics*, y esto supone un freno al desarrollo de una *Dapp* en curso legal en España basada en el sector inmobiliario. [84] Al no disponer de unas

regulaciones precisas, claras y al existir una normativa indecisa, supone inseguridad para los inversores, y los usuarios de negocio prefieren hacer uso de otras tecnologías más asentadas en el mercado. [83]

– **Cambios internos**

Las criptomonedas no dependen de una entidad central, por lo que necesitan la opinión y consenso de la comunidad para realizar diferentes acciones. Cuando la división del consenso se divide, se producen las bifurcaciones, también llamadas *fork*, que implican cambio en el funcionamiento de la red o una modificación en las reglas. Este hecho puede tener una influencia fuerte en el flujo de la criptomoneda como sucedió con *Bitcoin* y *Bitcoin Cash*, o como ha sucedido con *Ethereum* y *Bizancio*, o el futuro de *Constantinople*. [83] [85]

Uno de los mayores inconvenientes al tratarse de inmuebles es la volatilidad comentada de las criptomonedas, en concreto, en este caso, de *Ethereum*. *Ethereum* ha ofrecido gran rentabilidad a corto plazo para un gran número de inversores, aunque en el sector inmobiliario podría producir grandes pérdidas a largo plazo en el momento que se devalúe su precio en *Ether*. Poniendo un ejemplo, en el caso que se realice la venta de un inmueble con un valor de 400.000€, en Ethereum, su precio equivaldría a fecha 8 jun. 17:54 UTC a 1863,16 ETH. El problema sucedería si al día siguiente, cuando el vendedor va a cambiar su valor a euros, se encuentra con un descenso del *Ether* respecto al euro, teniendo un valor del *Ether* respecto al euro un 10% más bajo que el cambio realizado el día anterior, produciéndole grandes pérdidas. Una posible solución sería *tokenizar* el valor del inmueble para transferir el derecho al uso.

Otra traba existente acerca de este mercado en España sucede cuando se intercambian criptomonedas. Es decir, si se ha comprado, por ejemplo, cierta cantidad de euros en *Ethereum*, y después se decide cambiarlo a *Bitcoin*. A pesar de no haberlos vuelto a cambiar a euros, al tener un precio de valor de beneficio, se deberá tributar en España. Este hecho supone un alejamiento hacia el mundo de las criptomonedas, suponiendo un obstáculo para llegar a un gran alcance comercial. Este efecto también se ve implicado en la venta de la

vivienda al obtener beneficios, que también dependerá de la volatilidad de la moneda que deberás tributarlos [86]

Además, en el plan antifraude implantado en España, se obliga a quienes realicen operaciones con criptomonedas a informar a Hacienda de ello, ya sean en el extranjero o aquí en España. Esta situación, es contraria al anonimato que supone *Blockchain*. Esta situación, beneficia el uso de la aplicación al disponer de identificación fiscal a la hora de realizar la compra o alquiler que se solicita en la aplicación. [87]

Tras el análisis de los puntos comentados anteriormente, se puede observar cómo a día de hoy la implantación en España de la *Dapp* desarrollada, de tal forma que cumpla con el marco legislativo, resulta poco atractivo tanto para implantarla como para que los usuarios la utilicen. Para el implantador, no es fácil cuadrar el caso de uso con la legislación vigente, mientras que los usuarios mostrarían una gran incertidumbre en la aplicación y el uso que se pueda realizar de sus datos, al no existir una normativa sobre la que tributar y una escritura digital legal

Capítulo 12. CONCLUSIONES Y TRABAJOS FUTUROS

Tras realizar el proyecto, se ha permitido obtener distintas conclusiones, resultados y evolutivos para el proyecto, que se presentan a continuación.

12.1 CONCLUSIONES

En este proyecto se observa cómo la gestión de inmuebles, préstamos y seguros puede reducir sus costes de tramitación basándose en la tecnología *Blockchain*, que reafirma la confianza y fiabilidad en ambas partes. Se exponen las ventajas primordiales ofrecidas en este proyecto desde la perspectiva de estudio realizada:

- **Elaboración, documentación y síntesis de la tecnología *Blockchain*.** El motor de desarrollo del proyecto ha sido la tecnología innovadora de *Blockchain*, de manera divulgativa, sin opiniones al respecto, junto a lenguaje más sencillo y en español, puesto que la mayoría de la información actualmente se encuentra en inglés, permitiendo un acercamiento de la tecnología a los centros educativos.

Esta tecnología permite reducir los cambios y modificaciones por algunas de las partes, puesto que para realizarlos se necesitaría modificar una cantidad inmensa de información, y para ello se necesitaría tener unas capacidades de procesamientos superior a todos los nodos de donde se encuentra replicado. Es decir, una misión prácticamente imposible. Además de que este sistema está disponible de lunes a domingo a diferencia de la operativa de los bancos, de lunes a viernes en horario comercial, el coste de la transacción por *Gas* es muy baja, siendo una comisión prácticamente inexistente, a diferencia de operación habitual de banca.

- **Documentación y comparativa de las plataformas para la aplicación del caso de uso.** No se trata de un simple caso de realizar una aplicación *Java* basándose en un lenguaje conocido por la comunidad de desarrolladores. La fuerza del proyecto

recaen en el acercamiento a unas tecnologías no estudiadas previamente y en pleno esplendor. Para ello, basándose en el aprendizaje y conocimientos de otras tecnologías similares, trasladarlos para realizar una valoración y utilidad correcta de las mismas. Es decir, mostrar las ventajas y desventajas en su uso, así como mostrar en el proyecto la aplicación de estas.

- **Aplicación de la gestión de inmuebles a través de *Smart Contracts* en una Dapp a través de las fases de análisis, diseño, desarrollo y testeo del caso de uso.** La *Dapp* desarrollada ha permitido mostrar una mayor transparencia en un mercado oscuro como es la gestión del derecho de aprovechamiento de inmuebles. Esto ha sido debido a que las transacciones son visibles para cualquier usuario. Además, el código de la *Dapp* es abierto, a diferencia de las aplicaciones tradicionales que sólo se puede inspeccionar el *frontend*. Por otro lado, este proyecto supone la implicación de la comunidad, y una mayor decisión al no depender de un tercero que decide por sus acciones, ya que para tomar una decisión se debe llegar a un consenso desde la minación para validar al nodo, tal y como se dio la situación que bifurcó las criptomonedas *Bitcoin* y *Bitcoin Cash* [79]. También ha beneficiado al cliente utilizar una *Dapp*, que con una única cuenta en una *Wallet* puede acceder a cualquier *Dapp* de forma muy segura a través de los métodos criptográficos.

Cabe destacar que el proyecto no se ha basado simplemente en documentar y realizar una síntesis de la tecnología *Blockchain* disponible en la red, sino que incluye un caso de uso desarrollado y analizado de la tecnología *Blockchain* en la red *Ethereum*, suponiendo un proyecto base para realizar sobre éste diferentes evolutivos. Desde el punto de vista de la documentación de la tecnología, es trascendental ir completando y modificando según aparezcan mejoras o florezcan nuevas ideas sobre la tecnología, puesto que a día de hoy está actualizado, pero, al estar creciendo continuamente, siempre se podrá actualizar a una mejor versión. Por otra parte, desde la perspectiva del desarrollo e implantación del caso práctico de la tecnología *Blockchain*, al estar siendo desplegado a través de un nodo remoto en la red de pruebas *testnet Rinkeby*, permite tener una mayor escalabilidad, puesto que no supone

precisar de gran capacidad de procesamiento ni estar pendiente de sincronizar y actualizar el nodo que se instalaría en el ordenador.

Además, en la siguiente sección se mostrarán gran cantidad de puntos para poder realizar progresivas versiones de la misma. También, para facilitar el progreso del proyecto, se ha facilitado varios anexos para que cualquier persona sin experiencia y conocimientos de *Blockchain* puede disponer de la instalación y funcionamiento del proyecto paso a paso para poder mejorarlo y crear un evolutivo de Trabajo de Fin de Grado, o simplemente visualizarlo y comprobar el funcionamiento del proyecto en *Blockchain*.

El desarrollo del caso de uso, a través de una *Dapp*, conlleva un desafío por la integración de las tecnologías que, al estar en pleno auge, aún tienen diferentes defectos y falta de información relevante para el desarrollo.

En la siguiente tabla se muestran las tareas principales realizadas, tal como se detallaron en el apartado de planificación. Cabe recordar que estas tareas han estado subdivididas en actividades más concretas. Estas tareas se han vinculado con los objetivos marcados al comienzo del proyecto. Ambos se han asociado con el grado de realización y estado de finalización del mismo para observar de manera particularizada cada uno de estos hechos.

<i>Tarea principal</i>	<i>Objetivo</i>	<i>Cumplimiento</i>
Planteamiento del proyecto	-	Se ha realizado satisfactorio el planteamiento que ha servido para poder abordar el alcance del proyecto, tal como se muestra en el Capítulo 4.

Estudio <i>Blockchain y su aplicación</i>	Investigar la tecnología <i>Blockchain y su aplicación</i> de <i>Smart Contracts</i> con vistas a desplegar el caso de uso en la red de pruebas <i>Rinkeby</i>	Se ha estudiado la tecnología, tal como se ha mostrado en el Blockchain Capítulo 5., que ha servido de motor para las siguientes tareas. Además, la integración de las tecnologías se ha realizado en el Capítulo 8.
<i>Smart Contracts:</i> plataformas	Estudiar y comparar las plataformas para la aplicación de <i>Smart Contracts</i> con objeto de diseñar, desarrollar y testear el caso de uso.	En el Capítulo 2. se han descrito las tecnologías estudiadas y usadas, así como la comparativa con otras herramientas disponibles.
Gestión de inmuebles	Solucionar el problema de gestión de inmuebles con tecnología <i>Blockchain</i> .	Tras entender el contexto de la situación actual, se ha planteado y resuelto el problema. Para ello, se ha realizado a través de un estudio detallado, tal como se ha especificado en los Capítulo 1. Capítulo 3. Capítulo 6. y Capítulo 7. Además, se ha introducido un estudio del alcance comercial del mismo en el Capítulo 11.
<i>Smart Contract:</i> desarrollo	Estudiar y comparar las plataformas para la aplicación de <i>Smart Contracts</i> con objeto de diseñar, desarrollar y testear el caso de uso.	Una parte del proyecto se basaba en la realización de <i>Smart Contracts</i> , que se ha cumplido a través de tres principales y dos secundarios. A mayores del objetivo marcado al comienzo, se ha realizado un gran avance al haberlo integrado en un aplicación descentralizada: <i>Dapp</i> , permitiendo la interacción de cualquier usuario sin nociones de programador.

		Se muestra en el Capítulo 9.
<i>Smart Contract:</i> testear	Estudiar y comparar las plataformas para la aplicación de <i>Smart Contracts</i> con objeto de diseñar, desarrollar y testear el caso de uso.	Se han realizado las pruebas del funcionamiento en local, a través de <i>Ganache</i> antes de ser desplegado en <i>Rinkeby</i> . Por lo que se ha logrado adecuadamente esta tarea.
Despliegue en <i>Rinkeby</i> y análisis	Investigar la tecnología <i>Blockchain</i> y su aplicación de <i>Smart Contracts</i> con vistas a desplegar el caso de uso en la red de pruebas <i>Rinkeby</i>	Se ha realizado el despliegue de la Dapp en <i>Rinkeby</i> tras haber probado el correcto funcionamiento en <i>Ganache</i> . El despliegue se ha realizado de forma novedosa utilizando un nodo remoto: <i>Infura</i> . Tras haber sido desplegado, se han analizado las transacciones y el comportamiento del mismo como se observó en el Capítulo 10.

Tabla 9: Relación tarea principal, objetivo y grado de cumplimiento

12.2 TRABAJOS FUTUROS

El trabajo presente se ha basado en una fase de investigación y exploración de la tecnología *Blockchain*, que está en pleno descubrimiento, desarrollo y auge, seguido de una implantación de la tecnología y análisis del funcionamiento de las transacciones a través de una *Dapp*. Aprovechándose y basándose en lo obtenido en este proyecto, se detallan futuros posibles avances tal como se muestran a continuación:

- Para asegurar un mejor funcionamiento de la plataforma, evitando posibles estafas y fraudes, se deben poder realizar depósitos de dinero, similar a la fianza a la que estamos acostumbrados. Se deberá realizar a través de *Smart Contract escrow*.

TO BE IN ESCROW → EXTR EN DEPOSITO

Esto posibilita que hasta que el *Smart Contract* no se firma por comprador y vendedor, no se libera los fondos, asegurando la fiabilidad de la venta, teniendo retenido en la cuenta del *Smart Contract* el dinero hasta que ambos no marquen las directrices de recibido. En la Ilustración 105, se muestra claramente como la cuenta del *Smart Contract* retiene un total de cuatro *Ether*, cuando realmente el bien a comprar únicamente vale un *Ether*, depositando y asegurando la confiabilidad para las partes involucradas.

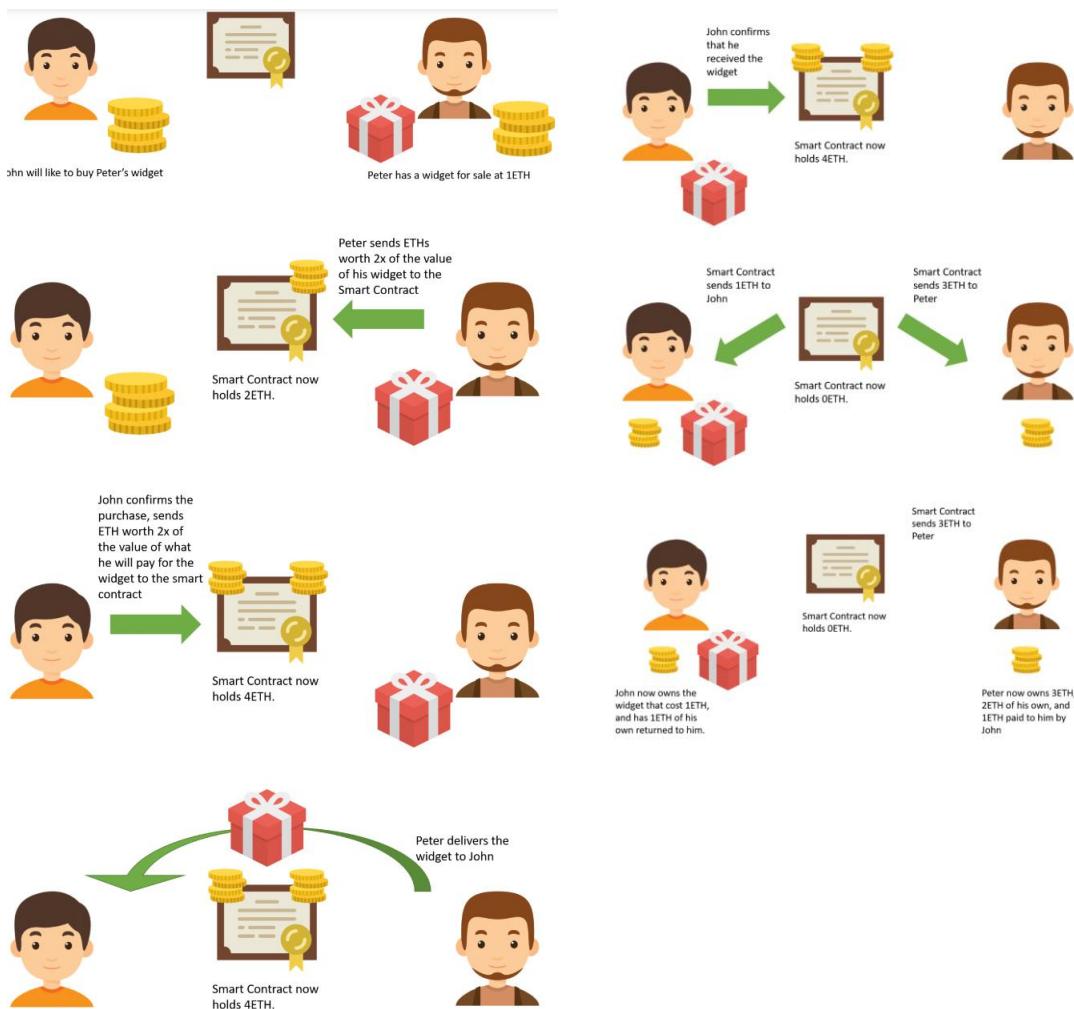


Ilustración 105: Smart Contract Escrow [88]

- Realización de una evaluación de *scoring* de los prestatarios. Valorar a través de los movimientos transaccionales que ha tenido la cuenta del prestatario en *Blockchain*,

la capacidad para devolver el dinero. Esto permitiría disminuir el riesgo de impago. Una gran opción sería seguir la línea de la normativa *IFRS9*. La normativa se implantó tras la gran crisis financiera que sucedió debido a la venta de conductas inadecuado de los empleados de varias entidades financieras, unido a la venta de productos calificados con un bajo riesgo cuando realmente eran de alto riesgo. Este modelo se basa en la pérdida esperada, clasificando en tres tipos de *stages* tal como se muestra en la Ilustración 106.

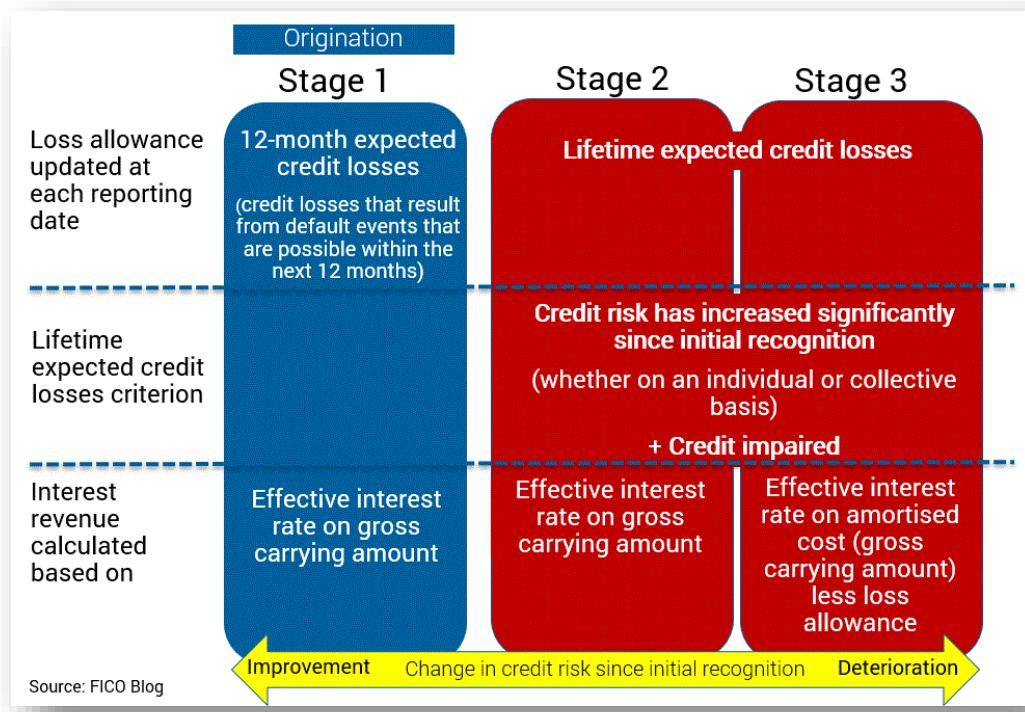


Ilustración 106: Stages de la normativa IFRS9 [89]

- El almacenamiento en *Blockchain* es muy costoso en términos monetarios. Por eso se propone subir documentos adjuntos a través de *IPFS: InterPlanetary File System*. Este sistema *peer to peer* permite subir el fichero e indexar el *hash* del fichero a la transacción de *Blockchain*. En caso que se desee acceder a la información, se necesita

el hash del fichero *IPFS* junto a la clave privada de encriptación del documento, manteniendo con seguridad la información.

Se podrán subir documentos para verificar la posesión del inmueble como las escrituras de la casa. Además como la red de *Blockchain* mantiene el anonimato a la cuenta de *Ethereum*, se propone realizar la subida de un documento de DNI para verificar la autenticidad de la persona en asuntos inmobiliarios.

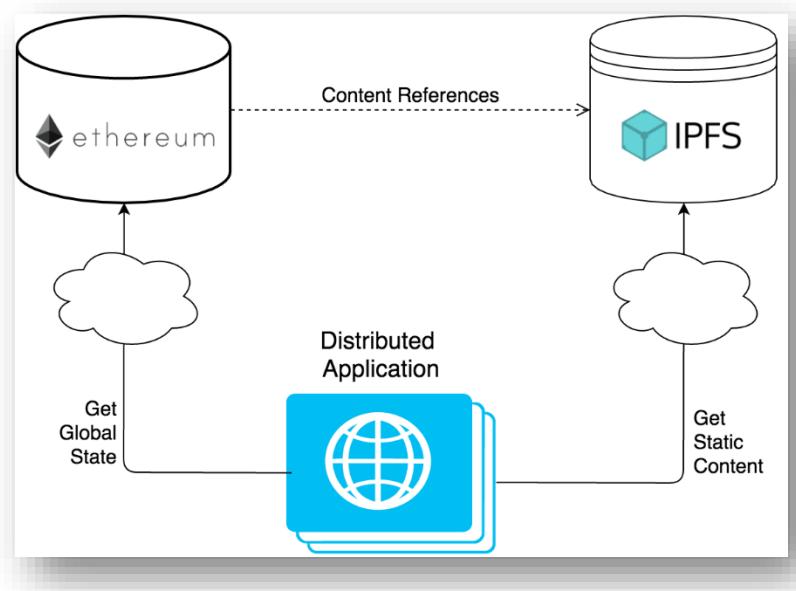


Ilustración 107: IPFS [90]

- Tras realizar la compra o el arrendamiento del inmueble, realizar la implantación de la conexión automática desde la *Dapp* con la *Smart Door*. El comprador o arrendatario deberá disponer del acceso en el período indicado en el *Smart Contract*: si es compra, indefinido, y si es alquilado, únicamente en el plazo establecido. En caso de producirse un error en la generación del código, el sistema deberá detectar a que se debe y devolver los fondos.



Ilustración 108: Smart Door [91]

- Otra posibilidad a realizar es que la *Dapp* se podría enfocar hacia la *tokenización*. Cuando el comprador compra el inmueble recibe un *token* junto a la escritura subida a *IPFS* vinculada con el *hash* a la transacción. De este modo el *token* permite reconocer la propiedad del activo agilizando el proceso tradicional. Además, también se podrían realizar otros tipos de *token* a la hora de realizar un despliegue real de la *Dapp* a través de una *ICO*.
- Para acercarnos a los gastos de comunidad de un aprovechamiento por turnos de un inmueble, se deben realizar pagos de mantenimiento. En el caso de una cuota de mantenimiento de la comunidad, se debería añadir a los *Smart Contract* una condición inversa. Para ello se propone abastecerse de los fondos de una *wallet*, y que, si el receptor del servicio no firma la nulidad de los pagos periódicos, se realicen.
- Se propone un llamamiento a la innovación de las aseguradoras, así como mostrar los beneficios que obtienen a través de la *Dapp*. Intentar obtener una oferta de pólizas innovada y mejorada. Una de las preocupaciones para las viviendas que se encuentran

vacías durante ciertos períodos de tiempo, es la ocupación ilegal de estas. Se propone la creación de un seguro *antiokupas*.

- Con el fin de obtener un mayor alcance en la *Dapp*, se propone una gestión del inmueble en la que se incluya la opción de alquilar únicamente una de las habitaciones del inmuebles, así como mantener la opción de por ciertos períodos de tiempo. También, considerar si estos períodos de tiempo deberían realizar semanalmente, o caber la posibilidad de únicamente unos días, realizando el tratamiento de las fechas correctamente, evitando el solapamiento de las mismas.
- Al tratarse de una tecnología reciente, la legalidad de realizar en España compra y venta de viviendas a través de Blockchain no se reconoce. Se deberá realizar los avances únicamente en una testnet. A no ser que la implantación se realice en otro país con otro marco regulatorio.

Capítulo 13. BIBLIOGRAFÍA

- [1] [En línea]. Available: <http://blog.ruralzoom.com/breve-historia-del-turismo-en-espana/>. [Último acceso: 31 Mayo 2019].
- [2] W. E. Forum, «Spain - Travel & Tourism Competitiveness Index,» [En línea]. Available: <http://reports.weforum.org/travel-and-tourism-competitiveness-report-2017/country-profiles/#economy=ESP>. [Último acceso: 8 Noviembre 2018].
- [3] [En línea]. Available: <https://es.statista.com/estadisticas/489156/gasto-total-de-los-turistas-internacionales-en-espana-por-motivo-del-viaje/>. [Último acceso: 31 Mayo 2019].
- [4] 24 05 2019. [En línea]. Available: <http://www.rtve.es/noticias/20190116/espana-cierra-2018-record-826-millones-turistas-extranjeros/1869862.shtml>.
- [5] [En línea]. Available: <https://es.statista.com/estadisticas/802182/ingresos-anuales-de-airbnb-en-espana/>. [Último acceso: 31 Mayo 2019].
- [6] [En línea]. Available: <https://es.statista.com/estadisticas/475605/numero-de-turistas-internacionales-en-viviendas-de-alquiler-en-espana/>. [Último acceso: 31 Mayo 2019].
- [7] A. Riaño, «Reclamador: Problema vivienda en multipropiedad,» 23 Mayo 2018. [En línea]. Available: <https://www.reclamador.es/blog/contrato-a-perpetuidad-vivienda-en-multipropiedad/>. [Último acceso: 1 Noviembre 2018].
- [8] [En línea]. Available: <https://www.criptonoticias.com/mercados/precios-trading/precio-bitcoin-sobrepasa-9000-marca-precio-maximo-meses/>. [Último acceso: 31 Mayo 2019].
- [9] [En línea]. Available: <https://www.criptonoticias.com/criptopedia/quien-es-vitalik-buterin-creador-ethereum/>. [Último acceso: 25 Mayo 2019].

- [10] Y. Zhu. [En línea]. Available: <https://www.rankia.com/blog/blockchain-criptomonedas-bitcoin-ethereum/3683082-ethereum-que-como-nacio-cuales-son-sus-ventajas>. [Último acceso: 25 Mayo 2019].
- [11] M. Mulders. [En línea]. Available: <https://hackernoon.com/comparison-of-smart-contract-platforms-2796e34673b7>. [Último acceso: 24 Mayo 2019].
- [12] «Logos vg png,» [En línea]. Available: <https://www.logosvgpng.com/solidity-logo-vector/>. [Último acceso: 35 Mayo 2019].
- [13] «Truffle Framework,» [En línea]. Available: <https://truffleframework.com/docs/truffle/overview>. [Último acceso: 25 Mayo 2019].
- [14] «Truffle Framework,» [En línea]. Available: <https://truffleframework.com/ganache>. [Último acceso: 25 Mayo 2019].
- [15] «Truffle Framework,» [En línea]. Available: <https://truffleframework.com/docs/ganache/workspaces/creating-workspaces>. [Último acceso: 24 Mayo 2019].
- [16] «Remix IDE Doc,» [En línea]. Available: <https://media.readthedocs.org/pdf/remix/latest/remix.pdf>.
- [17] «Remix,» [En línea]. Available: <https://remix.ethereum.org>. [Último acceso: 25 Mayo 2019].
- [18] «Visual Studio Solidity,» [En línea]. Available: <https://marketplace.visualstudio.com/items?itemName=JuanBlanco.solidity>. [Último acceso: 1 Noviembre 2018].
- [19] [En línea]. Available: <https://blogs.msdn.microsoft.com/caleteet/2016/04/01/solidity-integration-with-visual-studio/>. [Último acceso: 24 Mayo 2019].

- [20] «Integraciones Solidity,» [En línea]. Available: <https://media.readthedocs.org/pdf/solidity-es/latest/solidity-es.pdf>. [Último acceso: 10 Noviembre 2018].
- [21] «Metamask,» [En línea]. Available: <https://chrome.google.com/webstore/detail/metamask/nkbihfbeogaeaoehlefnkodbefgpgknn?hl=es-419>. [Último acceso: 26 Mayo 2019].
- [22] «Etherscan Transacciones,» [En línea]. Available: <https://etherscan.io/txs>. [Último acceso: 1 Noviembre 2018].
- [23] «Etherscan,» [En línea]. Available: <https://etherscan.io/>. [Último acceso: 26 Mayo 2019].
- [24] [En línea]. Available: <https://etherscan.io/>. [Último acceso: 2 Junio 2019].
- [25] [En línea]. Available: https://www.w3schools.com/nodejs/nodejs_intro.asp. [Último acceso: 27 Mayo 2019].
- [26] «Nodejs,» [En línea]. Available: <https://nodejs.org/es/about/resources/>. [Último acceso: 26 Mayo 2019].
- [27] [En línea]. Available: https://www.youtube.com/watch?v=DzAmQcR_79g. [Último acceso: 26 Mayo 2019].
- [28] «Ethereum Stack Exchange,» [En línea]. Available: <https://ethereum.stackexchange.com/questions/27048/comparison-of-the-different-testnets>. [Último acceso: 27 Mayo 2019].
- [29] «Rinkeby,» [En línea]. Available: <https://www.rinkeby.io/#stats>. [Último acceso: 26 Mayo 2019].
- [30] M. Wuehler. [En línea]. Available: <https://blog.infura.io/infura-debuts-new-visual-identity-as-daily-requests-top-6-billion-644c6e8f389a>. [Último acceso: 23 Mayo 2019].

- [31] «Expansión- Turno de bienes inmuebles,» [En línea]. Available: <http://www.expansion.com/2014/09/05/ahorro/1409920458.html>. [Último acceso: 8 Noviembre 2018].
- [32] J. Correa, «Timeshare Lawyer - Aprovechamiento por Turno de Bienes,» [En línea]. Available: <http://www.correaguimera.com/el-blog/20/2/2014/un-poquito-de-historia-sobre-el-tiempo-compartido-multipropiedad-hoy-aprovechamiento-por-turno-de-bienes>. [Último acceso: 2018 Noviembre 7].
- [33] «Cancelar Tiempos Compartidos,» [En línea]. Available: <http://www.cancelartiemposcompartidos.com/blog/115-companias-tiempo-compartido/>. [Último acceso: 8 Noviembre 2018].
- [34] «BBC - 5 paginas web competencia Airbnb,» [En línea]. Available: <https://www.bbc.com/mundo/noticias-39693721>. [Último acceso: 12 Noviembre 2018].
- [35] J. F. Bolaños, «Contratos ricardianos vs contratos inteligentes,» 24 Octubre 2018. [En línea]. Available: <https://www.academiablockchain.com/2018/10/24/contratos-ricardianos-vs-contratos-inteligentes/>. [Último acceso: 12 Noviembre 2018].
- [36] G. González, «NICK SZABO DEFINIÓ LOS CONCEPTOS BÁSICOS DE BLOCKCHAIN 14 AÑOS ANTES DE SU LANZAMIENTO,» 4 Abril 2018. [En línea]. Available: <https://www.criptonoticias.com/adopcion/nick-szabo-definio-conceptos-basicos-blockchain-14-anos-antes-lanzamiento/>. [Último acceso: 13 Noviembre 2018].
- [37] N. Szabo, «@NickSzabo4 Twitter,» 14 Octubre 2018. [En línea]. Available: <https://twitter.com/NickSzabo4/status/1051603179526270976>. [Último acceso: 2 Noviembre 2018].

- [38] V. Buterin, «Twitter Vitalik Non-giver of Ether,» @VitalikButerin, 13 Octubre 2018. [En línea]. Available: <https://twitter.com/VitalikButerin/status/1051160932699770882>. [Último acceso: 2018 Noviembre 2].
- [39] E. Mundo, 12 Diciembre 2017. [En línea]. Available: <https://www.elmundo.es/economia/vivienda/2017/12/12/5a2ecb6e46163ffd528b45c9.html>. [Último acceso: 10 Enero 2019].
- [40] «El Mundo "ProntoPiso acelera el proceso de las tasaciones de viviendas gracias al 'blockchain' ",» 26 Octubre 2018. [En línea]. Available: <https://www.elmundo.es/economia/vivienda/2018/10/26/5bd049e546163f06ac8b459d.html>. [Último acceso: 4 Noviembre 2018].
- [41] «Alastria,» [En línea]. Available: <https://alastria.io/#1>. [Último acceso: 10 Enero 2019].
- [42] «Comillas,» [En línea]. Available: <https://www.comillas.edu/es/prensa-comunicacion/16882-comillas-presenta-el-primer-nodo-universitario-blockchain>. [Último acceso: 9 Enero 2019].
- [43] S. Blázquez, «Metrovacesa lanza dos blockchain para clientes e inversores inmobiliarios,» 7 Noviembre 2018. [En línea]. Available: <https://www.blockchaineconomia.es/tag/metrovacesa/>. [Último acceso: 2018 Noviembre 8].
- [44] [En línea]. Available: <https://observatorioblockchain.com/real-fund-el-primer-proyecto-de-tokenizacion-inmobiliaria-espanola-estara-en-demo-day-madrid/>. [Último acceso: 31 Mayo 2010].
- [45] [En línea]. Available: <http://www.expansion.com/economia-digital/innovacion/2018/03/27/5ab522c0e2704ecb098b4580.html>. [Último acceso: 10 Junio 2019].
- [46] [En línea]. Available: <https://www.bbva.com/es/bbva-indra-realizan-primer-prestamo-corporativo-tecnologia-blockchain-mundo/>. [Último acceso: 10 Junio 2019].

- [47] [En línea]. Available: <https://www.businessinsider.es/caixabank-comienza-utilizar-blockchain-comercio-exterior-201750>. [Último acceso: 10 Junio 2019].
- [48] [En línea]. Available: <https://es.cointelegraph.com/news/banco-santander-plans-to-explore-blockchain-s-potential-in-securities-trade>. [Último acceso: 10 Junio 2019].
- [49] [En línea]. Available: <https://www.bbva.com/es/azlo-ultima-el-lanzamiento-de-su-comunidad-empresarial-de-pymes/>. [Último acceso: 10 Junio 2019].
- [50] [En línea]. Available: <https://es.cointelegraph.com/news/santander-enters-700-million-deal-to-use-ibms-tech-including-blockchain>. [Último acceso: 10 Junio 2019].
- [51] [En línea]. Available: https://www.axa.es/documents/1119421/143282252/NP_alastralia.pdf/47658e8d-36e8-a944-566e-d60cace4e146. [Último acceso: 10 Junio 2019].
- [52] [En línea]. Available: <http://www.finanzas.com/noticias/mercados/bolsas/20180621/mapfre-crea-grupo-expertos-3863920.html>. [Último acceso: 10 Junio 2019].
- [53] [En línea]. Available: https://www.axa.es/documents/1119421/143282252/NP_alastralia.pdf/47658e8d-36e8-a944-566e-d60cace4e146. [Último acceso: 10 Junio 2019].
- [54] [En línea]. Available: <https://www.santalucia.es/sobre-santalucia/gabinete-de-prensa/santalucia-elege-a-las-5-startups-que-participaran-en-su-programa-de-aceleracion.html>. [Último acceso: 6 Junio 2019].
- [55] [En línea]. Available: <https://www.ictworks.org/blockchain-use-cases-self-sovereign-digital-identities/#.XP056ogzbIU>. [Último acceso: 9 Junio 2019].

- [56] [En línea]. Available: <https://www.criptonoticias.com/analisis-investigacion/blockchain-impulsa-identidad-digital-auto-soberana/>. [Último acceso: 9 Junio 2019].
- [57] [En línea]. Available: <https://www.criptonoticias.com/analisis-investigacion/blockchain-impulsa-identidad-digital-auto-soberana>. [Último acceso: 9 Junio 2019].
- [58] [En línea]. Available: <https://cointelegraph.com/news/new-blockchain-app-for-quick-identity-verification-to-help-protect-users-data>. [Último acceso: 9 Junio 2019].
- [59] [En línea]. Available: <https://www.comillas.edu/es/noticias-prensa-comunicacion/16882-comillas-presenta-el-primer-nodo-universitario-blockchain>. [Último acceso: 2019 Junio 10].
- [60] [En línea]. Available: <https://www.economista.es/ecoaula/noticias/8680380/10/17/Comillas-ICAIICADE-albergara-el-primer-laboratorio-de-investigacion-blockchain-en-Espana.html>. [Último acceso: 10 Junio 2019].
- [61] [En línea]. Available: https://medium.com/@alastria_es/c%C3%B3mo-est%C3%A1-conformado-alastria-y-en-qu%C3%A9-puedo-participar-f4f89c9e2a10. [Último acceso: 10 Junio 2019].
- [62] [En línea]. Available: <https://www.blockchaineconomia.es/i-jornada-blockchain-economia/>. [Último acceso: 10 Junio 2019].
- [63] [En línea]. Available: <https://www.blockchaineconomia.es/proteum-inyectara-hasta-20-millones-de-euros-por-start-up/>. [Último acceso: 10 Junio 2019].
- [64] R. Canaan, «Tipos métodos de investigación,» [En línea]. Available: <https://www.lifeder.com/tipos-metodos-de-investigacion/>. [Último acceso: 12 Noviembre 2018].
- [65] «LeanKit Kanban,» [En línea]. Available: <https://leankit.com/learn/kanban/what-is-kanban/>. [Último acceso: 13 Noviembre 2018].

- [66] K. board. [En línea]. Available: <https://docs.microsoft.com/en-us/azure/devops/boards/boards/split-columns?view=vsts&tabs=new-nav>. [Último acceso: 13 Noviembre 2018].
- [67] «ICAI,» [En línea]. Available: <https://www.comillas.edu/es/noticias-comillas/16882-comillas-presenta-el-primer-nodo-universitario-blockchain>. [Último acceso: 29 Mayo 2019].
- [68] A. Brownworth. [En línea]. Available: <https://anders.com/blockchain/blockchain.html>. [Último acceso: 30 Mayo 2019].
- [69] [En línea]. Available: <https://101blockchains.com/es/algoritmos-de-consenso-blockchain/#1>. [Último acceso: 29 Mayo 2019].
- [70] [En línea]. Available: <https://www.etoro.com/es/markets/btc/stats>. [Último acceso: 30 Mayo 2019].
- [71] [En línea]. Available: <https://economipedia.com/definiciones/altcoin-criptomoneda.html>. [Último acceso: 30 Mayo 2019].
- [72] [En línea]. Available: <https://www.bbva.com/en/what-is-a-token-and-what-is-it-for/>. [Último acceso: 30 Mayo 2019].
- [73] [En línea]. Available: <https://www.infobae.com/cripto247/educacion-cripto247/2018/08/10/cuales-son-los-tres-tipos-de-tokens-y-como-se-diferencian-de-las-criptomonedas/>. [Último acceso: 30 Mayo 2019].
- [74] [En línea]. Available: <https://searchcompliance.techtarget.com/definition/smart-contract>. [Último acceso: 30 Mayo 2019].
- [75] [En línea]. Available: <https://www.miethereum.com/smart-contracts/dapps/>. [Último acceso: 30 Mayo 2019].
- [76] [En línea]. Available: <https://www.stateofthedapps.com/dapps>. [Último acceso: 30 Mayo 2019].

- [77] [En línea]. Available: <https://kubide.es/bienvenidos-al-mundo-de-las-ico/>. [Último acceso: 30 Mayo 2019].
- [78] [En línea]. Available: <https://bitcoin.es/actualidad/brillante-lado-de-las-ico-no-malas-noticias/>. [Último acceso: 30 Mayo 2019].
- [79] F. Gutierrez. [En línea]. Available: <https://es.cointelegraph.com/explained/bitcoin-cash>. [Último acceso: 29 Mayo 2019].
- [80] [En línea]. Available: <https://www.bbva.com/es/bbva-colabora-r3-desarrollo-proyecto-trade-finance-tecnologias-dlt/>. [Último acceso: 29 Mayo 2019].
- [81] «Mi ethereum,» [En línea]. Available: <https://www.miethereum.com/guias/metamask/>. [Último acceso: 4 Junio 2019].
- [82] [En línea]. Available: <https://www.trufflesuite.com/docs/truffle/getting-started/truffle-with-metamask>. [Último acceso: 15 Abril 2019].
- [83] [En línea]. Available: <https://criptotario.com/4-razones-de-la-volatilidad-del-precio-de-bitcoin>. [Último acceso: 8 Junio 2019].
- [84] [En línea]. Available: <https://observatorioblockchain.com/real-fund-el-primer-proyecto-de-tokenizacion-inmobiliaria-espanola-estara-en-demo-day-madrid/>. [Último acceso: 8 Junio 2019].
- [85] [En línea]. Available: <https://criptoinforme.com/lo-que-necesitas-saber-sobre-el-proximo-hard-fork-de-ethereum-eth-constantinople/>. [Último acceso: 8 Junio 2019].
- [86] [En línea]. Available: https://www.elconfidencial.com/tecnologia/2018-05-31/bitcoin-ethereum-criptomonedas-tributacion-renta_1572381/. [Último acceso: 6 Junio 2019].
- [87] [En línea]. Available: https://www.elplural.com/economia/impuesto-digital-y-el-nuevo-plan-antifraude-gobierno_204951102. [Último acceso: 8 Junio 2019].

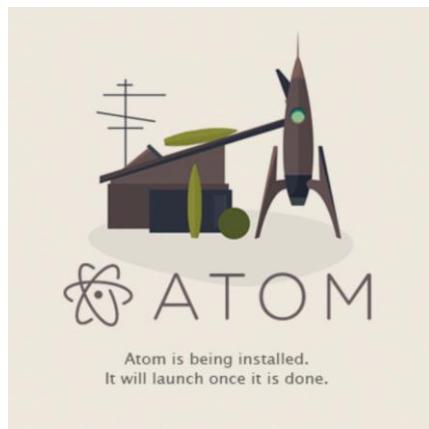
- [88] [En línea]. Available: <https://medium.com/coinmonks/escrow-service-as-a-smart-contract-the-business-logic-5b678ebe1955>. [Último acceso: 6 Junio 2019].
- [89] [En línea]. Available: <https://www.fico.com/blogs/collections-recovery/ifrs-9-and-collections-the-31-day-time-bomb/>. [Último acceso: 6 Junio 2019].
- [90] [En línea]. Available: <https://karl.tech/simple-dapp-architecture/>. [Último acceso: 30 Mayo 2019].
- [91] [En línea]. Available: <http://time.com/4639242/kwikset-premis-smart-lock-deadbolt-home-homekit/>. [Último acceso: 6 Junio 2019].
- [92] «Ganache,» [En línea]. Available: <https://truffleframework.com/docs/ganache/overview>. [Último acceso: 10 Noviembre 2018].
- [93] «Truffle,» [En línea]. Available: <https://truffleframework.com/docs/truffle/overview>. [Último acceso: 10 Noviembre 2018].
- [94] [En línea]. Available: <https://github.com/ethereum/web3.js/>. [Último acceso: 2019 Mayo 18].
- [95] [En línea]. Available: <https://blog.citowise.com/the-basics-coin-vs-token-what-is-the-difference-5cd270591538>. [Último acceso: 29 Mayo 2019].

A. ANEXO A: GUÍA DE INSTALACIÓN PARA EL USUARIO

Se recuerda al lector, que el sistema operativo utilizado ha sido *Windows 10* y desde la consola de comando se ha trabajado principalmente desde *Microsoft Powershell*. Por otro lado, se muestra algunas ilustraciones con cierta información oculta debido a que se trata de claves privadas, contraseñas, e incluso cuentas de usuario; es decir, información sensible.

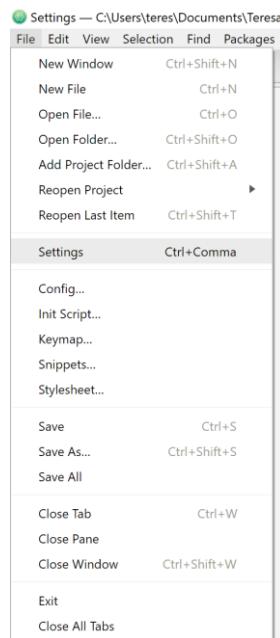
A.1 ATOM

Es un editor de texto especial puesto que permite tener la visión del proyecto y tener diferentes paquetes. Se puede descargar el ejecutable del siguiente enlace: <https://atom.io/>. Al ejecutar la instalación aparece la siguiente ventana:

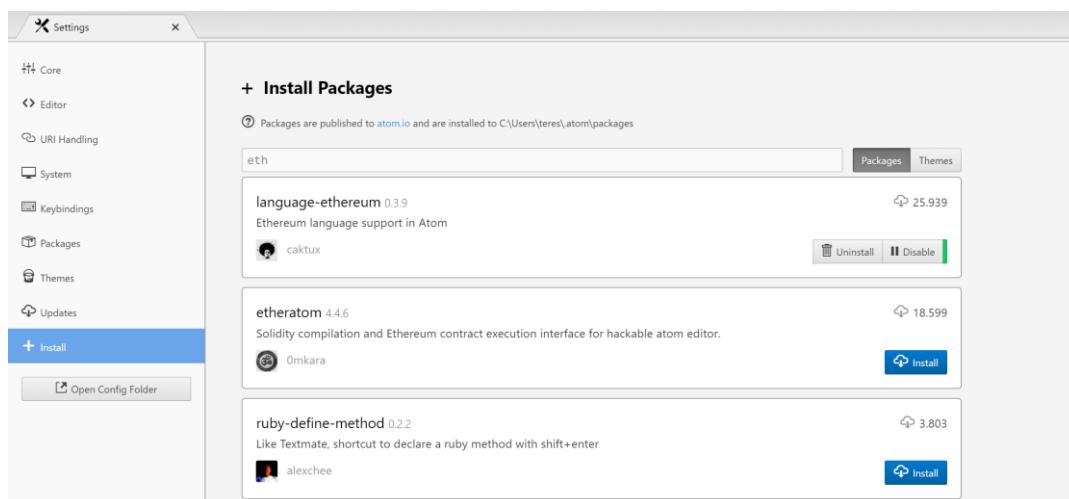


Tras instalar el editor, como se quiere resaltar con colores los scripts realizados en el lenguaje *Solidity*. Se debe instalar el **paquete language-ethereum**.

En primer lugar, se accede a file y dentro de este a la sección *Settings*.



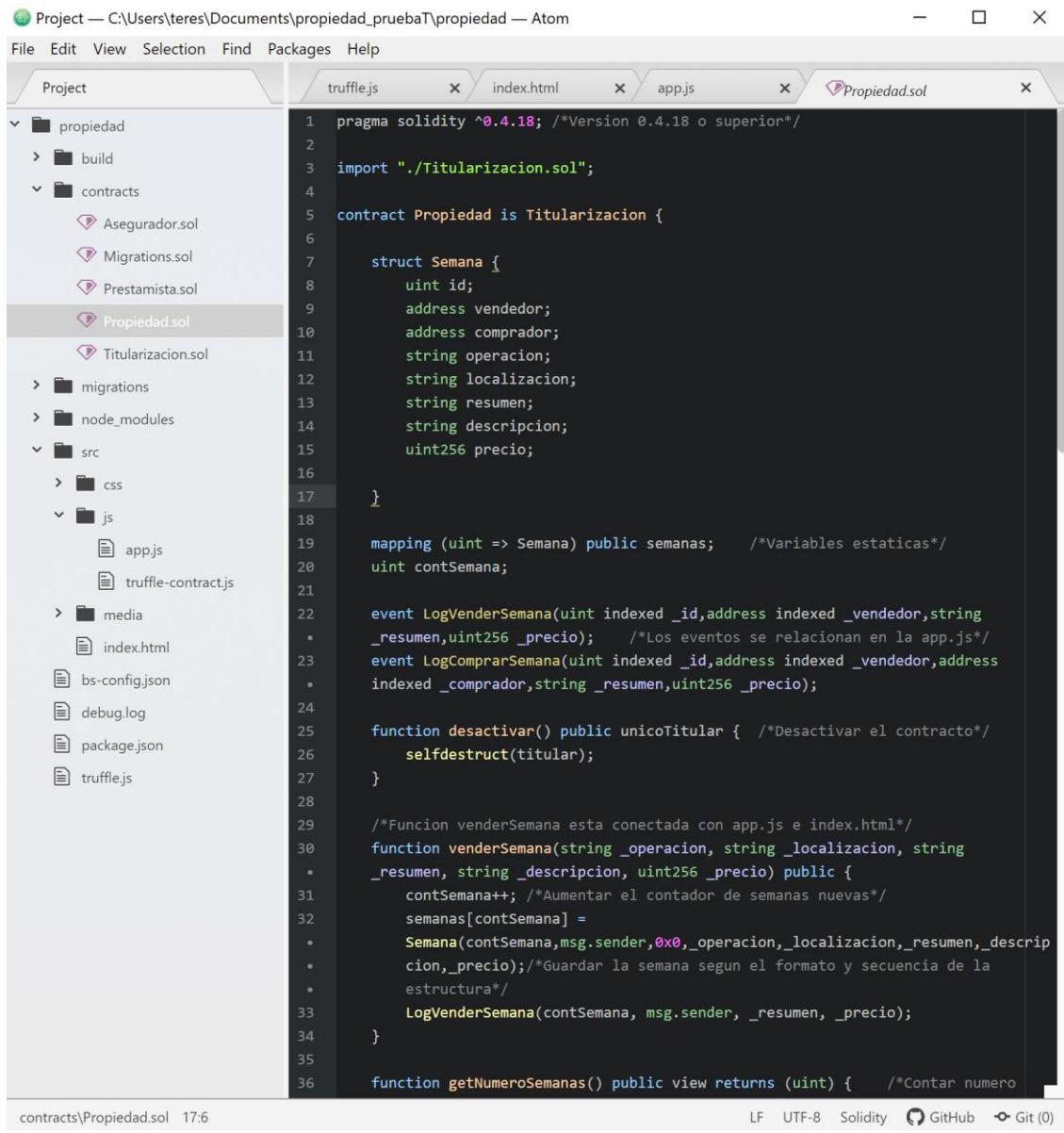
En la pestaña de *Settings*, en la zona *Install*:



Se instala el paquete **language-ethereum**:



Su correcto funcionamiento sucede cuando al abrir un script en *.sol* aparece un diamante en la pestaña y diferentes colores como en este ejemplo:



```

Project — C:\Users\teres\Documents\propiedad_pruebaT\propiedad — Atom
File Edit View Selection Find Packages Help
Project
  propiedad
    build
    contracts
      Asegurador.sol
      Migrations.sol
      Prestamista.sol
      Propiedad.sol
      Titularizacion.sol
    migrations
    node_modules
  src
    css
    js
      app.js
      truffle-contract.js
    media
    index.html
  bs-config.json
  debug.log
  package.json
  truffle.js

1 pragma solidity ^0.4.18; /*Version 0.4.18 o superior*/
2
3 import "./Titularizacion.sol";
4
5 contract Propiedad is Titularizacion {
6
7   struct Semana {
8     uint id;
9     address vendedor;
10    address comprador;
11    string operacion;
12    string localizacion;
13    string resumen;
14    string descripcion;
15    uint256 precio;
16
17  }
18
19  mapping (uint => Semana) public semanas; /*Variables estaticas*/
20  uint contSemana;
21
22  event LogVenderSemana(uint indexed _id,address indexed _vendedor,string
23  _resumen,uint256 _precio); /*Los eventos se relacionan en la app.js*/
24  event LogComprarsemana(uint indexed _id,address indexed _vendedor,address
25  indexed _comprador,string _resumen,uint256 _precio);
26
27  function desactivar() public unicoTitular { /*Desactivar el contrato*/
28    selfdestruct(titular);
29
30    /*Funcion venderSemana esta conectada con app.js e index.html*/
31    function venderSemana(string _operacion, string _localizacion, string
32    _resumen, string _descripcion, uint256 _precio) public {
33      contSemana++; /*Aumentar el contador de semanas nuevas*/
34      semanas[contSemana] =
35        Semana(contSemana,msg.sender,0x0,_operacion,_localizacion,_resumen,_descrip
36        cion,_precio);/*Guardar la semana segun el formato y secuencia de la
estructura*/
37      LogVenderSemana(contSemana, msg.sender, _resumen, _precio);
38
39    function getNumeroSemanas() public view returns (uint) { /*Contar numero
de semanas*/
40      return contSemana;
41    }
42  }
43
44  function comprarSemana(string _operacion, string _localizacion, string
45  _resumen, uint256 _precio) public {
46    contSemana++; /*Aumentar el contador de semanas nuevas*/
47    semanas[contSemana] =
48      Semana(contSemana,msg.sender,0x0,_operacion,_localizacion,_resumen,_descrip
49        cion,_precio);/*Guardar la semana segun el formato y secuencia de la
estructura*/
50    LogComprarsemana(contSemana, msg.sender, _resumen, _precio);
51
52  }
53
54  function consultarSemana(uint _id) public view returns (string, string, string,
55  string, uint256) {
56    require(_id <= contSemana);
57    return (semanas[_id].operacion, semanas[_id].localizacion, semanas[_id].resumen,
58    semanas[_id].descripcion, semanas[_id].precio);
59  }
60
61  function consultarTitular() public view returns (address) {
62    return titular;
63  }
64
65  function cambiarTitular(address _nuevoTitular) public {
66    require(msg.sender == titular);
67    titular = _nuevoTitular;
68  }
69
70  function consultarVendedor(uint _id) public view returns (address) {
71    return semanas[_id].vendedor;
72  }
73
74  function consultarComprador(uint _id) public view returns (address) {
75    return semanas[_id].comprador;
76  }
77
78  function consultarOperacion(uint _id) public view returns (string) {
79    return semanas[_id].operacion;
80  }
81
82  function consultarLocalizacion(uint _id) public view returns (string) {
83    return semanas[_id].localizacion;
84  }
85
86  function consultarResumen(uint _id) public view returns (string) {
87    return semanas[_id].resumen;
88  }
89
90  function consultarDescripcion(uint _id) public view returns (string) {
91    return semanas[_id].descripcion;
92  }
93
94  function consultarPrecio(uint _id) public view returns (uint256) {
95    return semanas[_id].precio;
96  }
97
98  function consultarContador() public view returns (uint) {
99    return contSemana;
100}
101
102}
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136

```

contracts\Propiedad.sol 17:6

LF UTF-8 Solidity GitHub Git (0)

A.2 GANACHE

Ganache es una herramienta de los creadores del *Framework Truffle* que permite ejecutar una *Blockchain* local para poder probar los *Smart Contract* e inspeccionar el estado de la *Blockchain*

Ganache se puede instalar desde el *Powershell* y/o con ejecutable dependiendo del uso. Primero se mostrará desde la consola de comandos para más tarde realizarlo a través del ejecutable.

Se instala *Ganache* desde el *powershell*:

```
PS C:\Windows\system32> npm install -g ganache-cli
C:\Users\teres\AppData\Roaming\npm\ganache-cli -> c:\users\teres\AppData\Roaming\npm\node_modul
+ ganache-cli@6.2.5
added 54 packages from 46 contributors in 6.977s
PS C:\Windows\system32>
```

Se comprueba su instalación:

```
PS C:\Windows\system32> ganache-cli
Ganache CLI v6.2.5 (ganache-core: 2.3.3)

Available Accounts
=====
(0) 0xd7ed32ccf4807ee34435e6499d8fc0e9bee6caa5 (~100 ETH)
(1) 0x8538e05a82b045472d2113ee4dcfd5e49781b5df (~100 ETH)
(2) 0x9e13ee6bc81fb219845114cf5d6e808f8176db2f (~100 ETH)
(3) 0xadbd2841e3f401eadc961e69ee00bc0017a69d2c2 (~100 ETH)
(4) 0x201a3e94e7449000957b59f458543ab3cb0ac4fd (~100 ETH)
(5) 0x3c329ac1ba988c0ac71774f4f1a8dc7faa16cf70 (~100 ETH)
(6) 0x52a09e292fe6799e80147c7d2d6b9b642396f4ee (~100 ETH)
(7) 0x100d7129c73588aff20491a7384ffe382d494fb (~100 ETH)
(8) 0x77a8c91b0c85d26271a7015750c312e33443b4bf (~100 ETH)
(9) 0xfd153fe19eff81646ab59fa50d5928a79cb2324d (~100 ETH)

Private Keys
=====
(0) 0x?
(1) 0?
(2) 0?
(3) 0?
(4) 0?
(5) 0?
(6) 0?
(7) 0?
(8) 0?
(9) 0?

HD Wallet
=====
Mnemonic: [REDACTED]
Base HD Path: m/44'/60'/0'/0/{account_index}

Gas Price
=====
200000000000

Gas Limit
=====
6721975

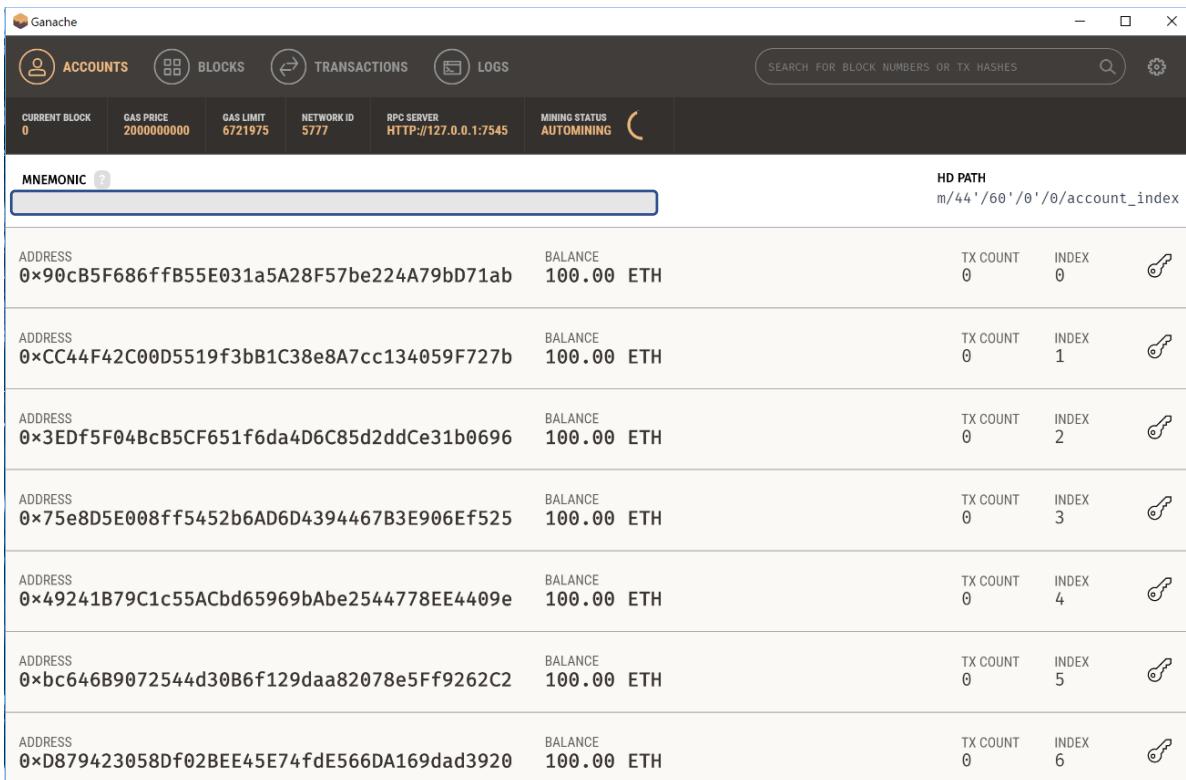
Listening on 127.0.0.1:8545
```

Teniendo en *Ganache* diferentes cuentas disponibles, diferentes claves privadas, el precio y límite del *Gas*, así como el RPC 127.0.0.1:8545. Es decir, se ha creado un entorno simulado para empezar a ejecutar con los *Smart Contracts* en el ordenador. Por lo que se crean 10 cuentas asociadas con 10 cuentas privadas cada cuenta con 100 *Ethers*. Para poder

trabajas con *Ganache* en venta de comandos, se debe tener dos ventanas de comandos abiertas, la de ejecución de *Ganache* y la segunda con lo que se esté trabajando.

Tras mostrar el proceso de instalación de *Ganache* por ventana de comando, se enseña también a instalar la interfaz de *Ganache* para *Windows* desde el siguiente enlace: <https://truffleframework.com/ganache>. Simplemente, consiste en aceptar los pasos de la instalación que es muy rápida.

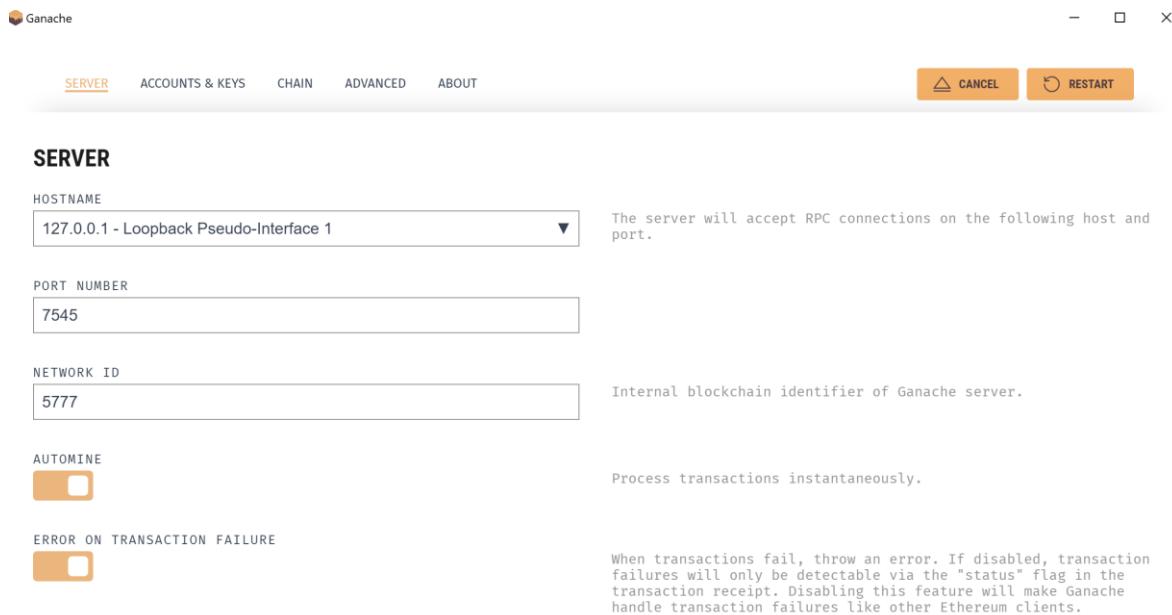
En el proyecto se ha trabajado a través de la interfaz por disponer de una interfaz gráfica más amigable. A continuación, se muestra la interfaz tras la instalación de *Ganache*.



The screenshot shows the Ganache graphical user interface. At the top, there is a navigation bar with tabs: ACCOUNTS, BLOCKS, TRANSACTIONS, and LOGS. Below the navigation bar, there are several status indicators: CURRENT BLOCK (0), GAS PRICE (2000000000), GAS LIMIT (6721975), NETWORK ID (5777), RPC SERVER (HTTP://127.0.0.1:7545), and MINING STATUS (AUTOMINING). A search bar at the top right allows searching for block numbers or transaction hashes. The main area displays a table of accounts:

MNEMONIC	ADDRESS	BALANCE	TX COUNT	INDEX	KEY
	0x90cB5F686ffB55E031a5A28F57be224A79bD71ab	100.00 ETH	0	0	🔑
	0xCC44F42C00D5519f3bB1C38e8A7cc134059F727b	100.00 ETH	0	1	🔑
	0x3EDf5F04BcB5CF651f6da4D6C85d2ddCe31b0696	100.00 ETH	0	2	🔑
	0x75e8D5E008ff5452b6AD6D4394467B3E906Ef525	100.00 ETH	0	3	🔑
	0x49241B79C1c55ACbd65969bAbe2544778EE4409e	100.00 ETH	0	4	🔑
	0xbc646B9072544d30B6f129daa82078e5Ff9262C2	100.00 ETH	0	5	🔑
	0xD879423058Df02BEE45E74fdE566DA169dad3920	100.00 ETH	0	6	🔑

Se exhibe la información posible a configurar en *Ganache* como es el puerto de la dirección de *loopback*. Para *Ganache* de interfaz se ha utilizado el puerto 7545 mientras que en el *Ganache* de la consola de comando se ha utilizado 8545, es decir, los puertos por defecto que le son asignados.



A.3 NODE.JS

A continuación, a través de la ventana de *Powershell* se realiza la instalación de *Node.js*, el entorno de ejecución de *JavaScript* con los comandos que aparecen:

```
PS C:\Windows\system32> npm install -g npm
C:\Users\teres\AppData\Roaming\npm\npx -> C:\Users\teres\AppData\Roaming\npm\node_modules\npm\bin\npx-cli.js
C:\Users\teres\AppData\Roaming\npm -> C:\Users\teres\AppData\Roaming\npm\node_modules\npm\bin\npm-cli.js
+ npm@6.5.0
added 389 packages from 781 contributors in 23.349s
PS C:\Windows\system32>
```

```
PS C:\Windows\system32> npm install -g --production windows-build-tools
> windows-build-tools@5.1.0 postinstall C:\Users\teres\AppData\Roaming\npm\node_modules\windows-build-tools\index.js
  eted in 118ms
> node ./dist/index.js

downloading python-2.7.15.amd64.msi
[> ] 0.0% (0 B/s)
Downloaded python-2.7.15.amd64.msi. Saved to C:\Users\teres\.windows-build-tools\python-2.7.15.amd64.msi.
downloading vs_BuildTools.exe
[> ] 0.0% (0 B/s)
Downloaded vs_BuildTools.exe. Saved to C:\Users\teres\.windows-build-tools\vs_BuildTools.exe.

starting installation...
Launched installers, now waiting for them to finish.
This will likely take some time - please be patient!

status from the installers:
----- Visual Studio Build Tools -----
2019-01-14T10:41:41 : Verbose : Calling SetupEngine.Installer.InstallProduct. [channelId: visualstudio.15.Release, productId: Microsoft.VisualStudio.Setup.Engine]
2019-01-14T11:25:17 : Verbose : Relaunching the application with new commands. Args: ["vs_installershell.exe","/finalizeInstall","images\\bootstrapper\\vs_setup_bootstrapper_201901141125120083.json","--norestart","--quiet","--includeRecommended","-add","Microsoft.VC.140.CRT","80609b65-d110-4f5d-ba60-d7f2b09c0864"]
2019-01-14T11:25:17 : Error : Parece que está instalando o modificando otro producto en este momento.
Espere hasta que el resto de operaciones se hayan completado y vuelva a intentarlo.
----- Python -----
Successfully installed Python 2.7
```

Tras realizar la instalación se muestra, la versión de *Node.js* instalada.

```
PS C:\Users\teres> node -v  
v9.3.0
```

Se ha instalado *Node.js*, pero a mayores se necesita instalar *NPM lite-server* que es un servidor de desarrollo local. Sirve para ejecutar *javascript* con *css* con *html*... Para ello lo primero es poder crear el archivo *package.json*. Se sitúa dentro del proyecto y dentro de este ejecuta el comando que aparece a continuación, presionando el botón *enter*, cuando se le solicita información.

```
C:\Users\teres\Documents\Teresa_ICAI_4\TFG\propiedad>npm init
This utility will walk you through creating a package.json file.
It only covers the most common items, and tries to guess sensible defaults.

See `npm help json` for definitive documentation on these fields
and exactly what they do.

Use `npm install <pkg>` afterwards to install a package and
save it as a dependency in the package.json file.

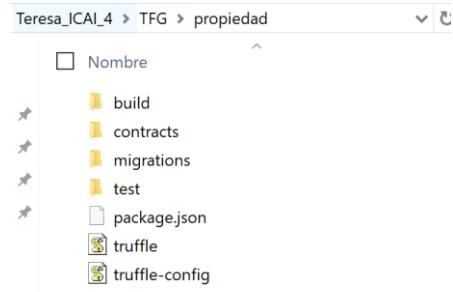
Press ^C at any time to quit.
package name: (propiedad)
version: (1.0.0)
description:
git repository:
keywords:
author:
license: (ISC)
About to write to C:\Users\teres\Documents\Teresa_ICAI_4\TFG\propiedad\package.json:

{
  "name": "propiedad",
  "version": "1.0.0",
  "main": "truffle.js",
  "directories": {
    "test": "test"
  },
  "scripts": {
    "t": "truffle",
    "test": "truffle test",
    "testnet": "ganache-cli"
  },
  "author": "",
  "license": "ISC",
  "dependencies": {
    "truffle": "^4.1.14"
  },
  "devDependencies": {
    "ganache-cli": "^6.1.8"
  },
  "description": ""
}

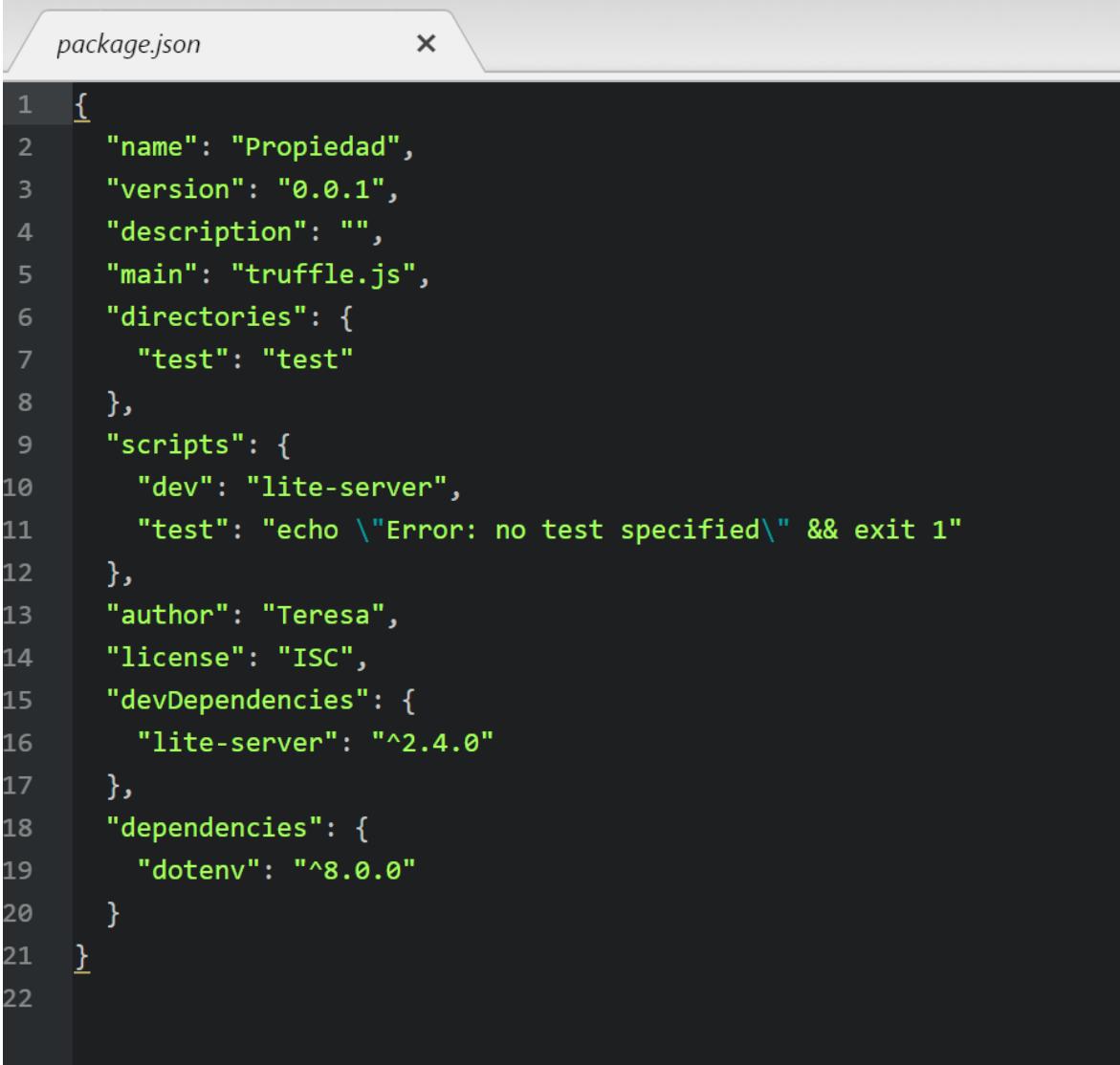
Is this OK? (yes)

C:\Users\teres\Documents\Teresa_ICAI_4\TFG\propiedad>
```

Con estos pasos se crea el archivo *package.json*. Este archivo le indica que debe ejecutar *lite-server* al realizar los comandos *npm run dev* .



El contenido es el que se muestra en la figura, es muy importante que dentro del fichero aparezca en el *script*, él que corresponde con *lite-server*, *dev*.



```

1  {
2    "name": "Propiedad",
3    "version": "0.0.1",
4    "description": "",
5    "main": "truffle.js",
6    "directories": {
7      "test": "test"
8    },
9    "scripts": {
10      "dev": "lite-server",
11      "test": "echo \\"Error: no test specified\\" && exit 1"
12    },
13    "author": "Teresa",
14    "license": "ISC",
15    "devDependencies": {
16      "lite-server": "^2.4.0"
17    },
18    "dependencies": {
19      "dotenv": "^8.0.0"
20    }
21  }
22

```

Muy importante para instalar *lite-server* estar dentro de la carpeta donde está *package.json*. Para posibilitar instalar el nodo se debe escribir los siguientes comandos.

```
C:\Users\teres\Documents\Teresa_ICAI_4\TFG\propiedad>npm install lite-server --save-dev
[.....] \ extract:localtunnel: sill extract localtunnel@1.9.1 extracted to
```

Al instalarlo le aparecerá en el proyecto *node_modules* es una carpeta usada por *NPM* para guardar los paquetes que requiere la aplicación como *lite-server*. Es el servidor *http* que servirá la aplicación al ejecutar *npm run dev*.

Se necesita tener el archivo *bs-config.json* pues esto le indica a *lite-server* donde está el directorio a utilizar.



The screenshot shows a code editor with two tabs: "package.json" and "bs-config.json". The "package.json" tab contains the following code:

```
1  {
2      "server": {
3          "baseDir": ["./src", "./build/contracts"]
4      }
5 }
```

Node.js es un servidor de *javascript* para crear la aplicación e interactuar con el nodo de *Ethereum*. Cuando se instala el *Node.js* también hay que instalar *NPM*, *Node Package Manage* que es el que tiene las herramientas y las librerías. Por tanto, las versiones utilizadas en el proyecto son las siguientes del *node.js* y de *npm*.

```
Windows PowerShell
Copyright (C) 2016 Microsoft Corporation. Todos los derechos reservados.

PS C:\Users\teres> node -v
v9.3.0
PS C:\Users\teres> npm -v
5.5.1
PS C:\Users\teres>
```

A.4 TRUFFLE

Se realiza la instalación del *framework truffle*, permite mantener un entorno de desarrollo más asequible a la hora de desplegar, compilar y migrar los *Smart Contracts*. Con la ayuda de *NPM* con el siguiente comando, la versión que se instala es la 4.0.4, esta versión condiciona la versión de *Solidity* a compilar.

```
PS C:\Users\teres> npm install -g truffle@4.0.4
C:\Users\teres\AppData\Roaming\npm\truffle -> C:\Users\teres\AppData\Roaming\npm\node_modules\truffle\build\cli_bundled.js
+ truffle@4.0.4
added 93 packages in 10.667s
PS C:\Users\teres> truffle version
Truffle v4.0.4 (core: 4.0.4)
Solidity v0.4.18 (solc-js)
```

En la siguiente imagen se prueba que la palabra *truffle* la reconozca el sistema.

```
PS C:\Windows\system32> truffle
Truffle v5.0.1 - a development framework for Ethereum

Usage: truffle <command> [options]

Commands:
  build      Execute build pipeline (if configuration present)
  compile    Compile contract source files
  console    Run a console with contract abstractions and commands available
  create     Helper to create new contracts, migrations and tests
  debug      Interactively debug any transaction on the blockchain (experimental)
  deploy     (alias for migrate)
  develop    Open a console with a local development blockchain
  exec       Execute a JS module within this Truffle environment
  help       List all commands or provide information about a specific command
  init       Initialize new and empty Ethereum project
  install   Install a package from the Ethereum Package Registry
  migrate   Run migrations to deploy contracts
  networks  Show addresses for deployed contracts on each network
  opcode    Print the compiled opcodes for a given contract
  publish   Publish a package to the Ethereum Package Registry
  run       Run a third-party command
  test      Run Javascript and Solidity tests
  unbox     Download a Truffle Box, a pre-built Truffle project
  version   Show version number and exit
  config    Set user-level configuration options
  watch     Watch filesystem for changes and rebuild the project automatically

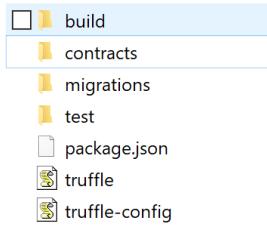
See more at http://truffleframework.com/docs
```

A.5 ESTRUCTURA Y FICHEROS EN EL PROYECTO

Con la ayuda del *framework truffle*, se crea la estructura de trabajo del proyecto a través del comando que aparece en la siguiente figura:

```
C:\Users\teres\Documents\Teresa_ICAI_4\TFG\propiedad>truffle init
```

Tras ejecutarlo en la carpeta del proyecto deben aparecer estos ficheros:

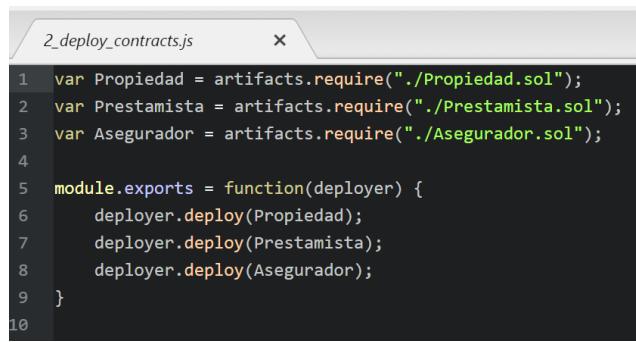


- **build**: es la carpeta donde se aloja el código compilado de cada *Smart Contracts*.
- **contracts**: donde se aloja el código fuente de los *Smart Contracts*.
- **migrations**: esta carpeta se utiliza para poder migrar y desplegar los *Smart Contracts* en una red de *Ethereum*. Para ello deberán contener los siguientes ficheros mostrados, rellenos con el nombre del código de cada *Smart Contract*. El archivo *2_deploy_contracts.js*, se debe tener cuidado porque sólo recompila los ficheros si ve cambios en el código fuente, por ello siempre se debe usar al ejecutar el comando *–reset*.

```

1_initial_migration.js
var Migrations = artifacts.require("./Migrations.sol");

module.exports = function(deployer) {
  deployer.deploy(Migrations);
};
  
```



```

1 var Propiedad = artifacts.require("./Propiedad.sol");
2 var Prestamista = artifacts.require("./Prestamista.sol");
3 var Asegurador = artifacts.require("./Asegurador.sol");
4
5 module.exports = function(deployer) {
6     deployer.deploy(Propiedad);
7     deployer.deploy(Prestamista);
8     deployer.deploy(Asegurador);
9 }
10

```

- **truffle.js:** documento de configuración para poder migrarlo a la red de pruebas en este caso *Rinkeby* o en local para el *Ganache* de ventana de comandos o de interfaz ejecutable.



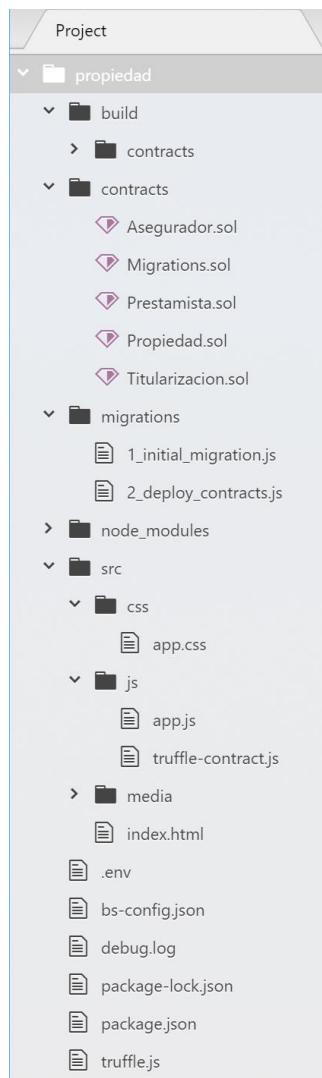
```

1 const HDWalletProvider = require("truffle-hdwallet-provider");
2 /*Cargar la libreria HDWalletProvider*/
3 require('dotenv').config();/*Configura dotenv paquete para
4 poder buscar desde el archivo .env*/
5
6 module.exports = {
7     networks: {
8         ganache: {
9             host: "localhost",
10            port: 7545,
11            network_id: "*" /*El puerto de ganache es el
12            7545 al que se conecta desde Metamask pej,
13            netowrok id * permite que cualquier id sirva*/
14        },
15        propiedad: {
16            host: "localhost",
17            port: 8545,
18            network_id: "4224",
19            gas: 4700000
20        },
21        rinkeby: {
22            provider: function () {
23                return new
24                HDWalletProvider(process.env.MNEMONIC,
25                "https://rinkeby.infura.io/v3/" +
26                process.env.PROJECT_ID); /*Relacionar rinkeby
27                infura y el archivo .env, si pones al final ,
28                numero es para que empieza desplegar a parti
29                de esa cuenta sin incluir esa. */
30            },
31            network_id: 4,
32            gas: 4500000,
33            gasPrice: 10000000000
34        }
35    }
36 };
37

```

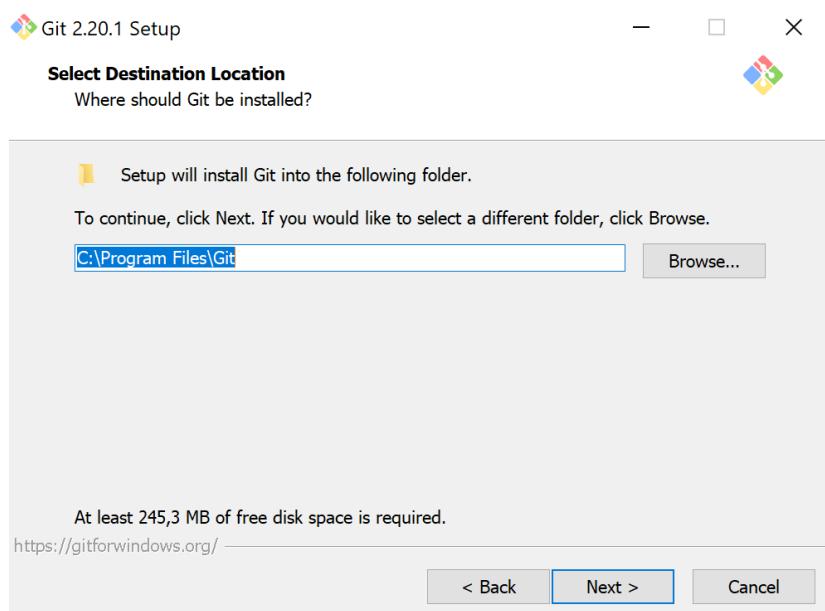
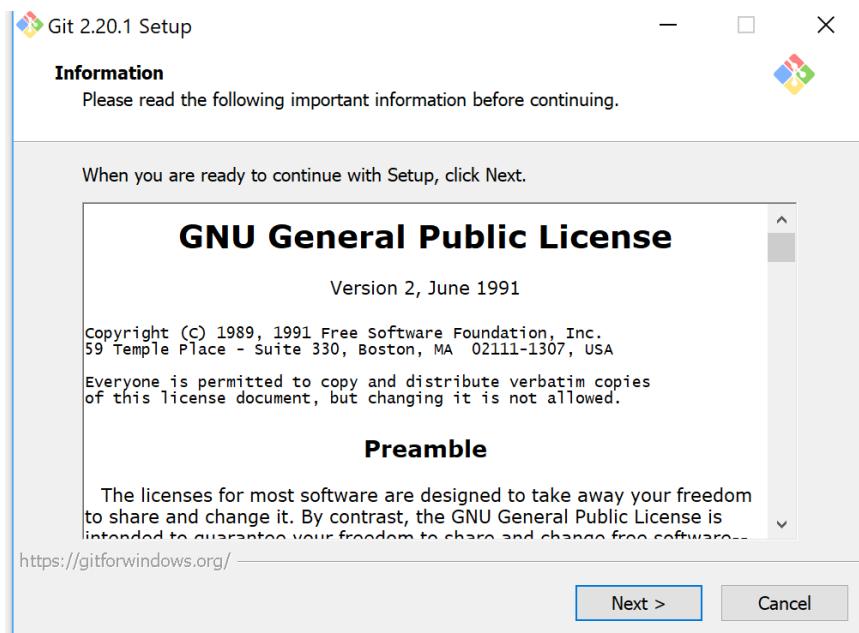
- El archivo **truffle-config.js** se utiliza si no empleas el *Powershell* para ejecutarlo y utilizas el *cmd*.

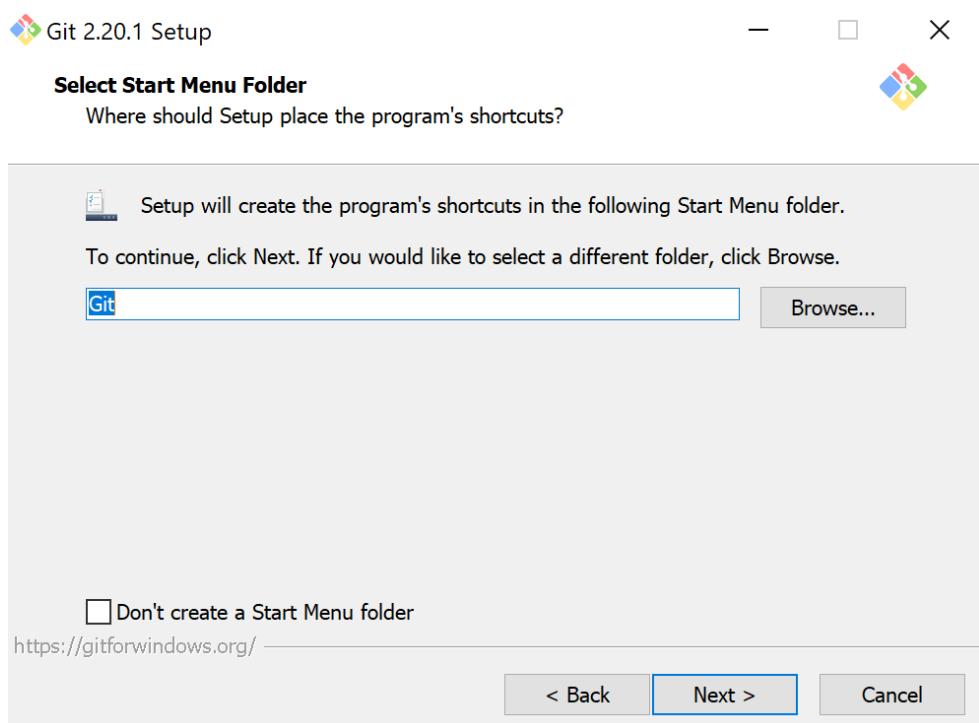
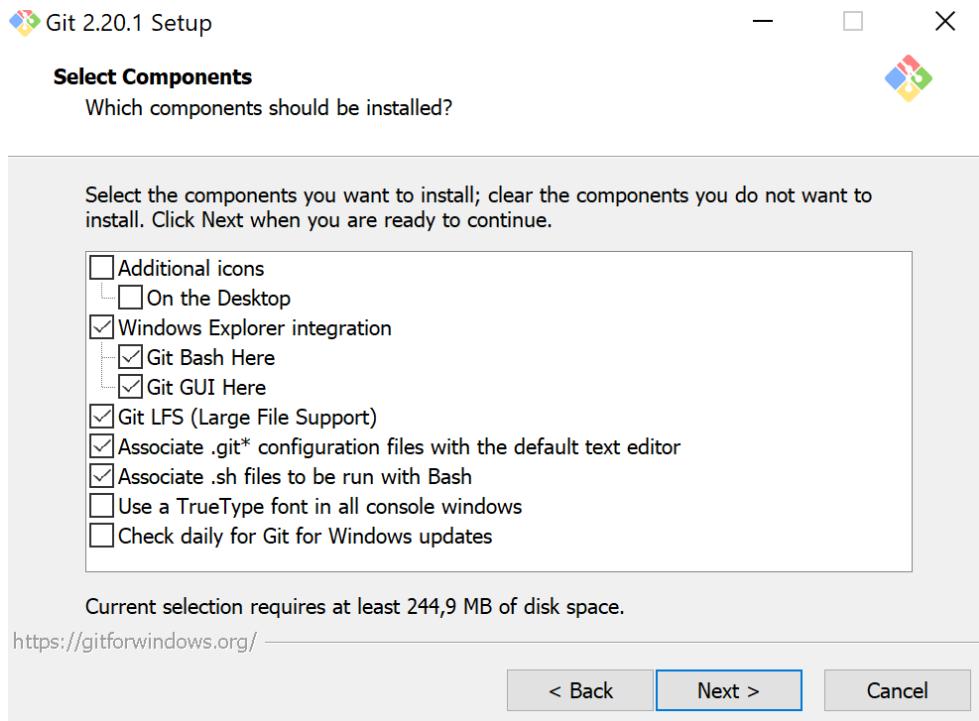
Además, en el proyecto debe existir el fichero *package.json* junto a *bs-config.json*, explicados en el apartado de *Node.js*. También debe aparecer el fichero *.env* utilizado al desplegar a través de un nodo remoto, *Infura*, a la red de pruebas *Rinkeby*. Para clarificar, a continuación se muestra la estructura del proyecto.

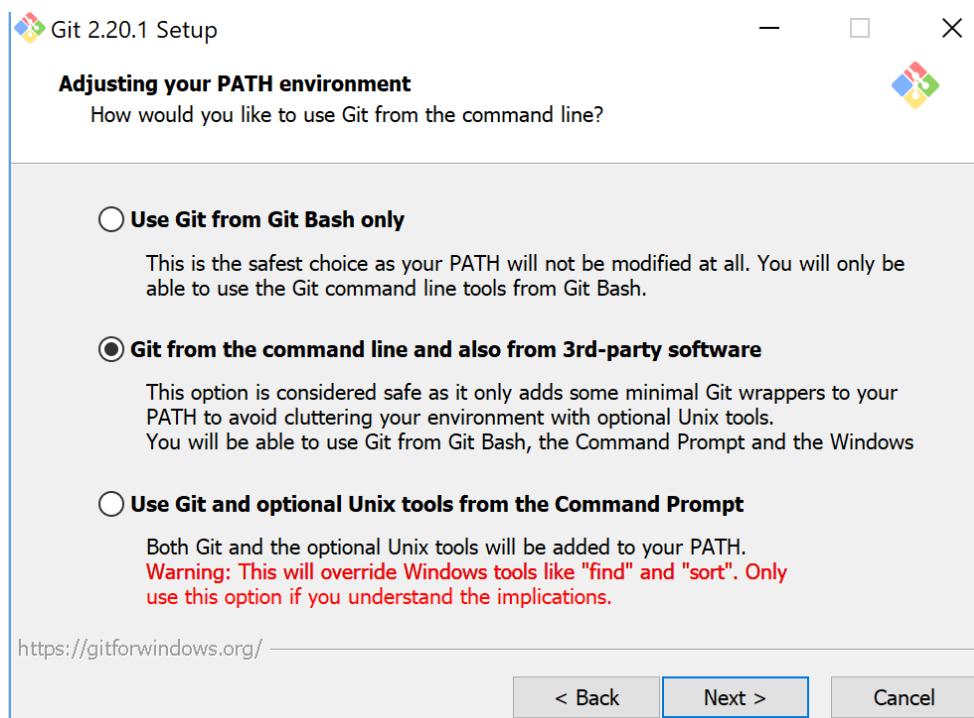
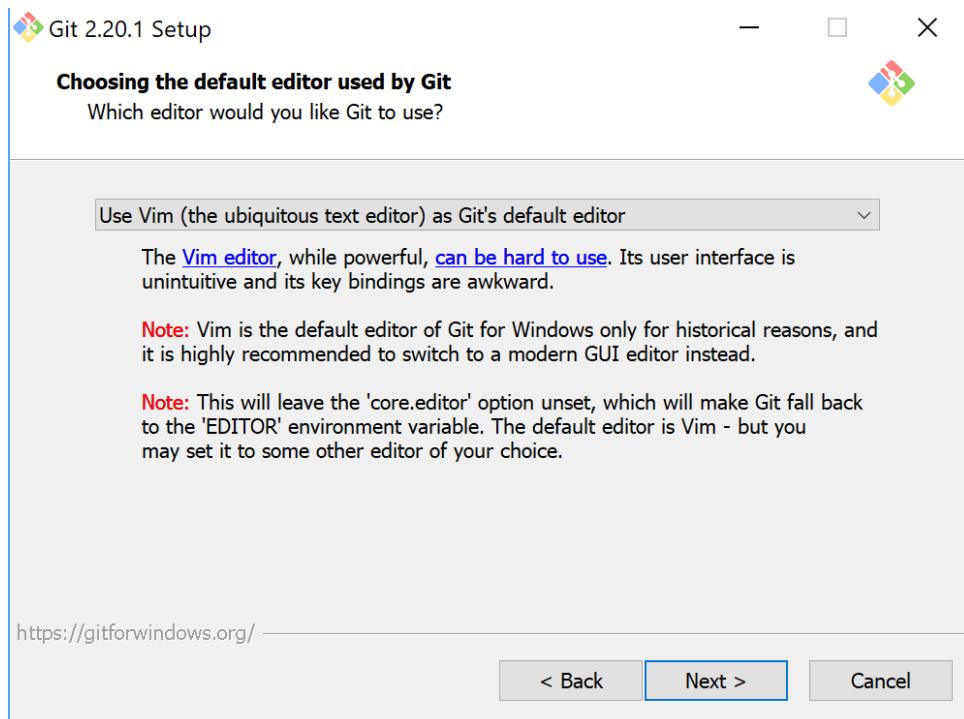


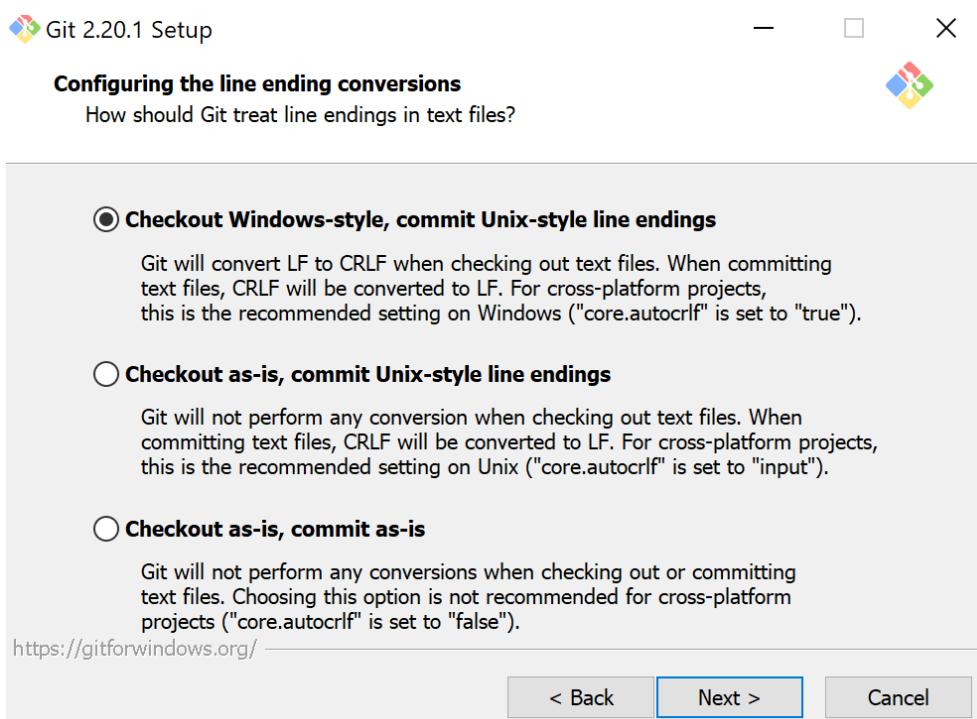
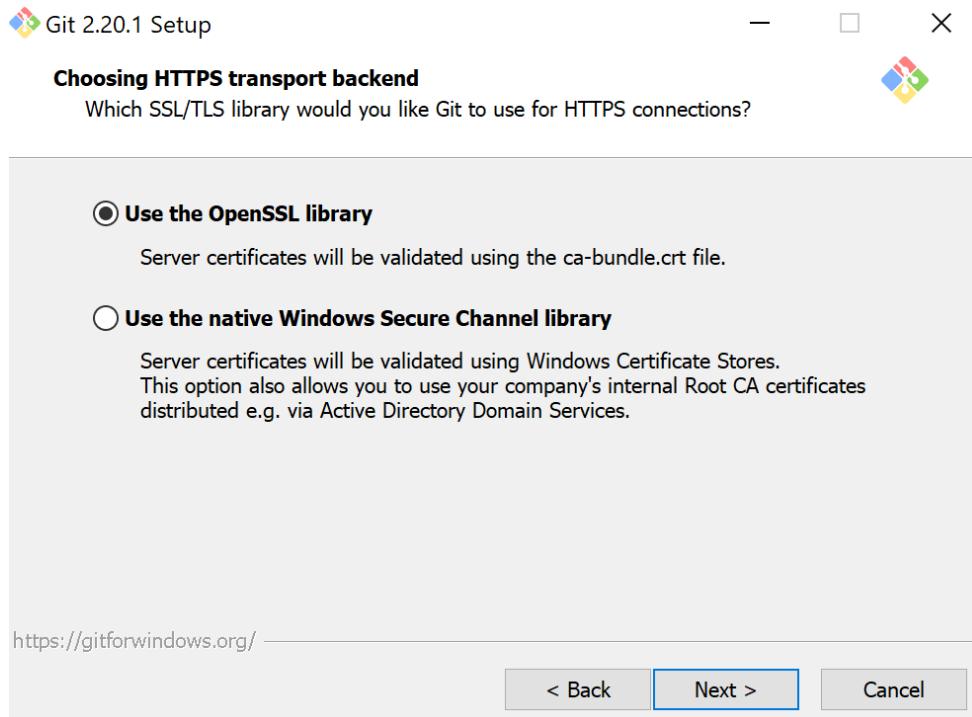
A.6 GIT

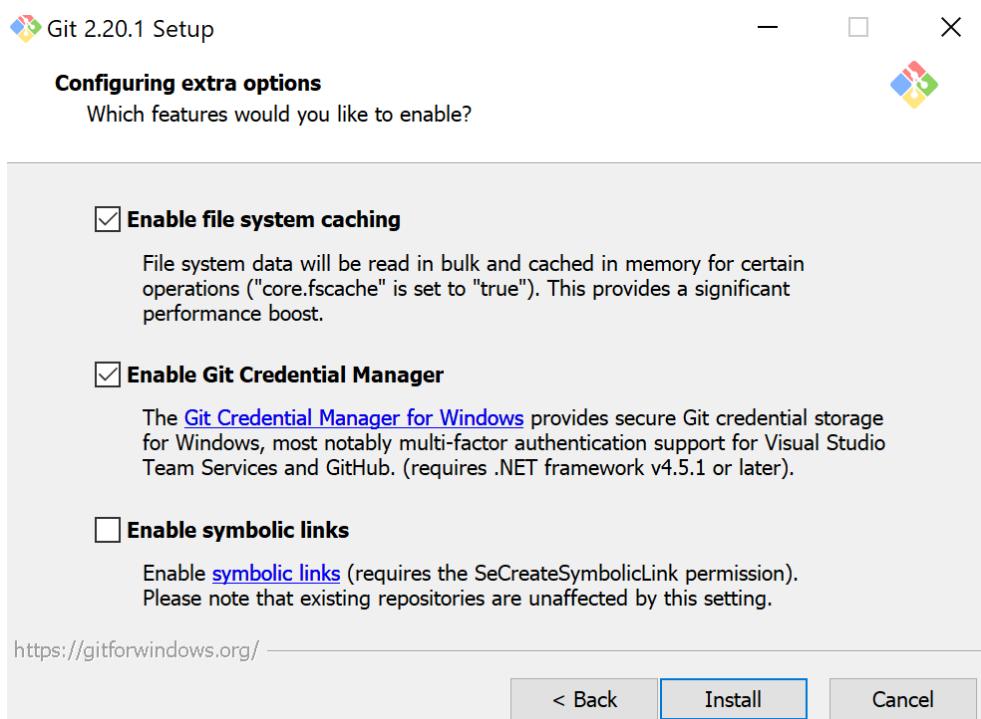
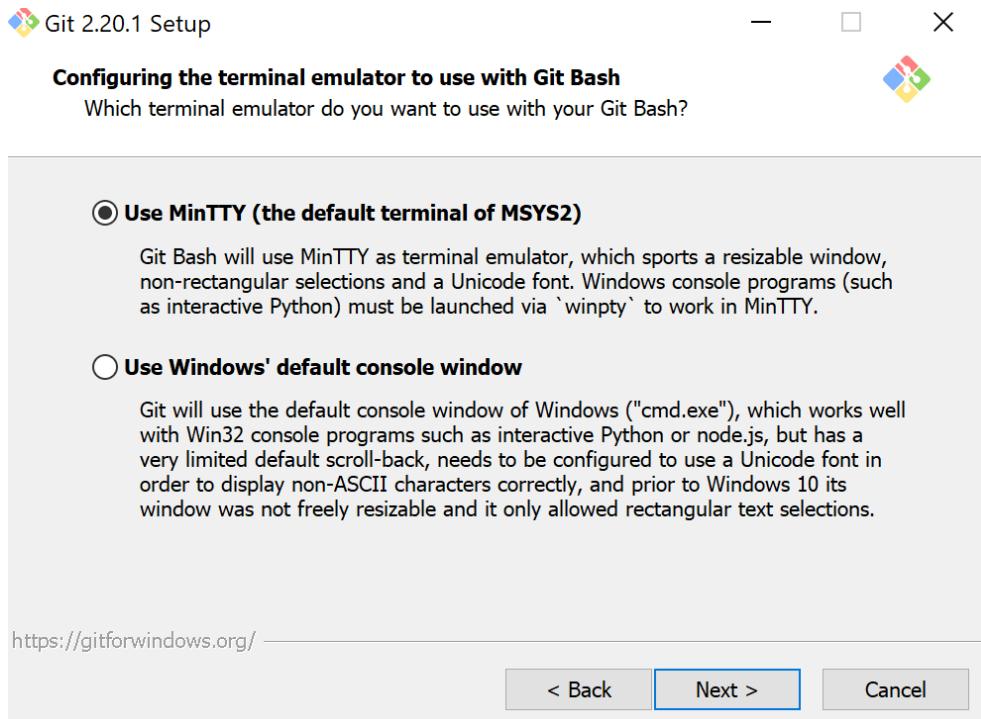
Git es un software de control de versiones que es muy usado para *NPM*. Así que cuando instalamos paquetes *NPM* necesitan *Git*. Se puede descargar a través de la página web <https://git-scm.com/downloads>. Se instala siguiendo estos pasos del ejecutable:



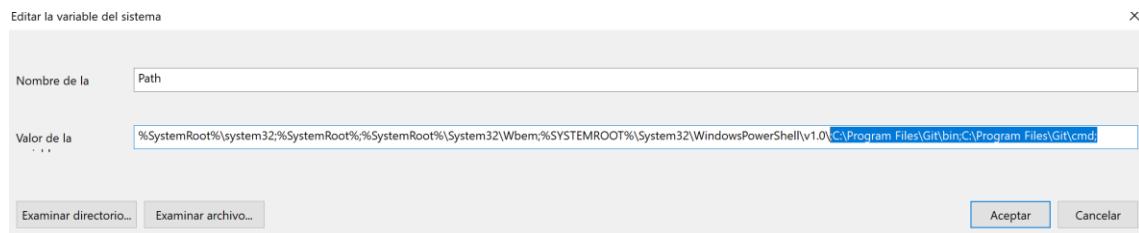
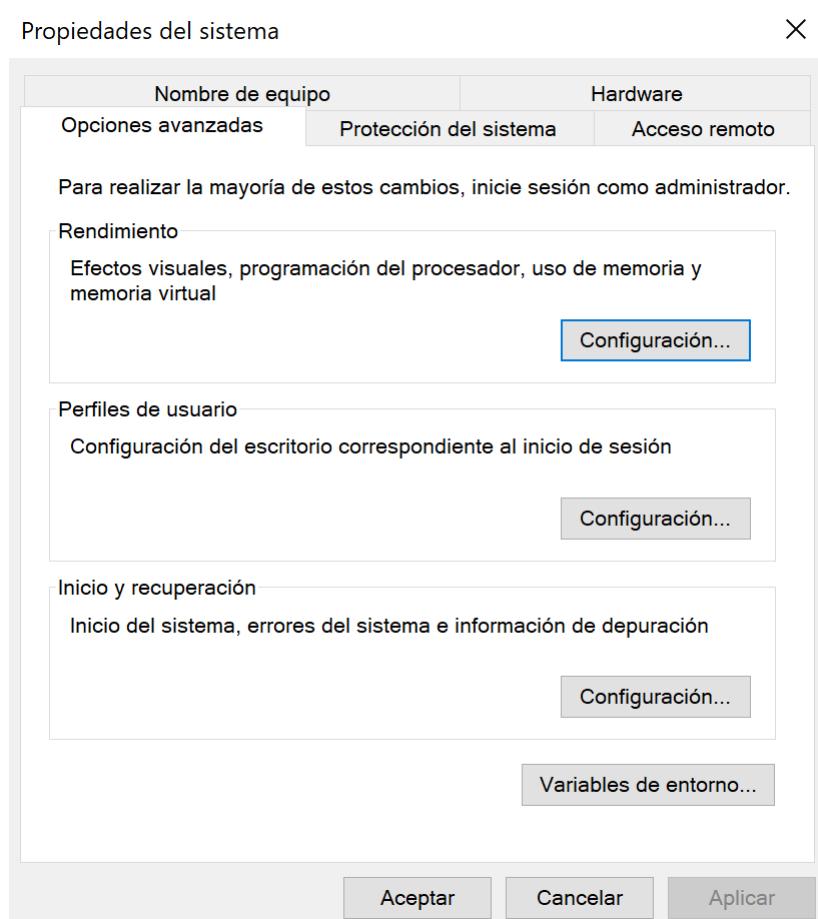








Después de instalar *Git*, se configura la variable de entorno para ello hay que mirar la dirección donde se encuentra el bin y el cmd, que en este caso es C:\Program Files\Git\cmd y ponérselo al PATH :



Se comprueba que funciona *Git* a través de escribir el comando en la consola.

 Símbolo del sistema

```
Microsoft Windows [Versión 10.0.15063]
(c) 2017 Microsoft Corporation. Todos los derechos reservados.

C:\Users\teres>git
usage: git [--version] [--help] [-C <path>] [-c <name>=<value>]
           [--exec-path[=<path>]] [--html-path] [--man-path] [--info-path]
           [-p | --paginate | -P | --no-pager] [--no-replace-objects] [--bare]
           [--git-dir=<path>] [--work-tree=<path>] [--namespace=<name>]
           <command> [<args>]

These are common Git commands used in various situations:

start a working area (see also: git help tutorial)
  clone      Clone a repository into a new directory
  init       Create an empty Git repository or reinitialize an existing one

work on the current change (see also: git help everyday)
  add        Add file contents to the index
  mv         Move or rename a file, a directory, or a symlink
  reset     Reset current HEAD to the specified state
  rm         Remove files from the working tree and from the index

examine the history and state (see also: git help revisions)
  bisect    Use binary search to find the commit that introduced a bug
  grep      Print lines matching a pattern
  log       Show commit logs
  show      Show various types of objects
  status    Show the working tree status

grow, mark and tweak your common history
  branch   List, create, or delete branches
  checkout Switch branches or restore working tree files
  commit   Record changes to the repository
  diff     Show changes between commits, commit and working tree, etc
  merge   Join two or more development histories together
  rebase   Reapply commits on top of another base tip
  tag      Create, list, delete or verify a tag object signed with GPG

collaborate (see also: git help workflows)
  fetch   Download objects and refs from another repository
  pull    Fetch from and integrate with another repository or a local branch
  push    Update remote refs along with associated objects

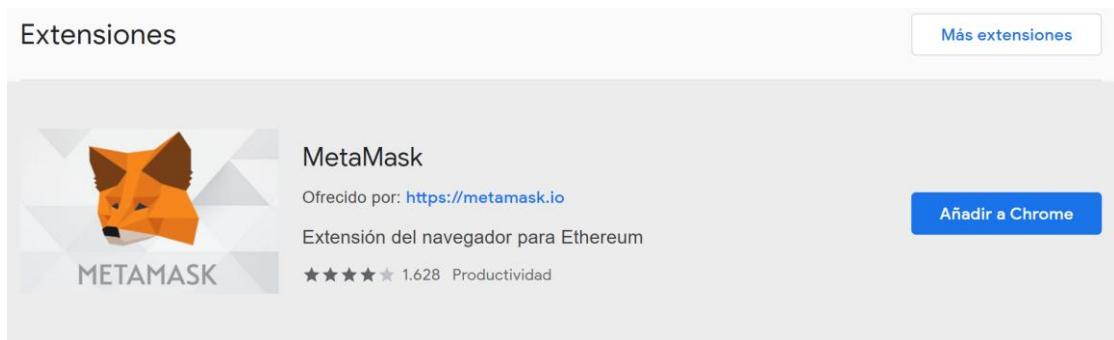
'git help -a' and 'git help -g' list available subcommands and some
concept guides. See 'git help <command>' or 'git help <concept>'
to read about a specific subcommand or concept.
```

La versión que ha sido instalada de *Git* es la mostrada.

```
PS C:\Users\teres> git version
git version 2.20.1.windows.1
PS C:\Users\teres>
```

A.7 METAMASK

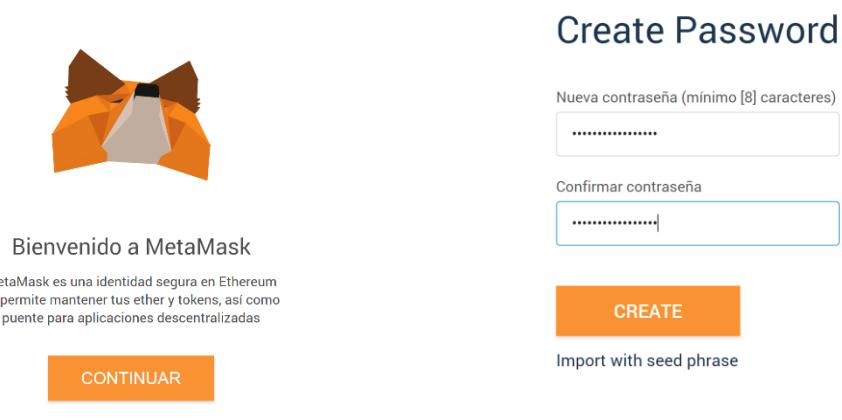
Es una extensión para los navegadores, es la manera más simple de interactuar con la red de *Ethereum*. Para acceder a la instalación deberá ir al siguiente enlace <https://chrome.google.com/webstore/detail/metamask/nkbihfbeogaeaoehlefknkodbefgpgknn?hl=es-419>



Tras instalar el plugin de *Metamask*, en el navegador de *Chrome* aparece en la URL el símbolo:



Se debe registrarse creando una contraseña, no se necesita nombre de usuario.



Tras crearlo, aparecerán las siguientes pantallas en la que puedes modificar la imagen de la cuenta, así como te indican la frase para recuperar la cuenta en caso de que te olvides de la contraseña.



Your unique account image

This image was programmatically generated for you by your new account number.

You'll see this image everytime you need to confirm a transaction.

NEXT

○ ○ ○



Secret Backup Phrase

Your secret backup phrase makes it easy to back up and restore your account.

WARNING: Never disclose your backup phrase. Anyone with this phrase can take your Ether forever.



Tips:

Store this phrase in a password manager like 1Password.

Write this phrase on a piece of paper and store in a secure location. If you want even more security, write it down on multiple pieces of paper and store each in 2 - 3 different locations.

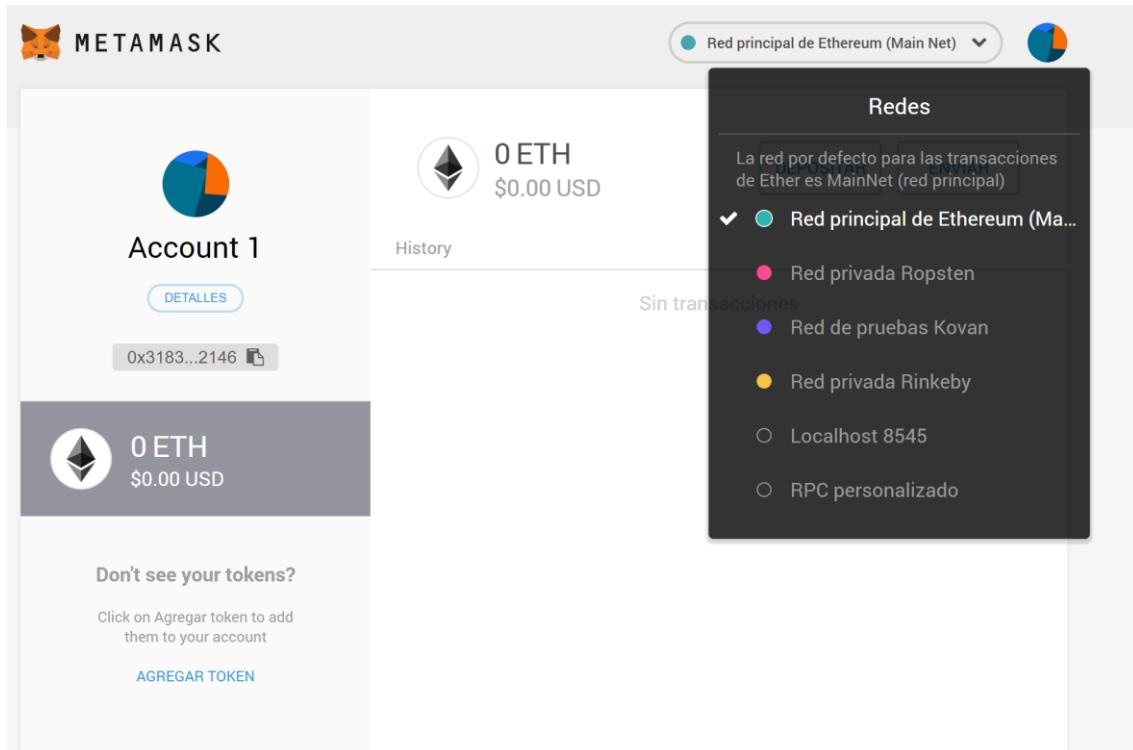
Memorize this phrase.

Download this Secret Backup Phrase and keep it stored safely on an external encrypted hard drive or storage medium.

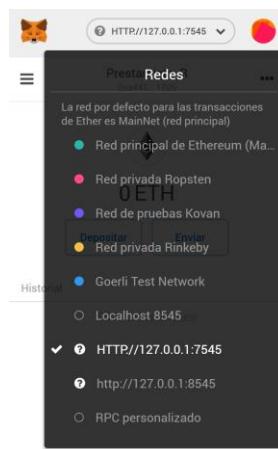
NEXT

○ ○ ○

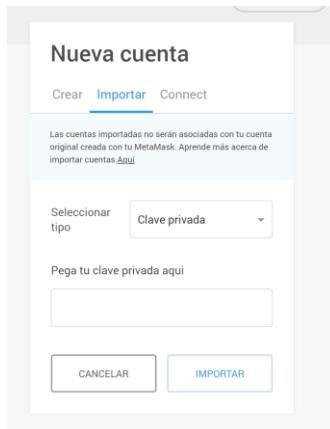
Después, de realizar los pasos indicados, aparece el inicio que se muestra a continuación.



En el caso de querer probarlo en local con *Ganache* se debe conectar al puerto 7545 que se ha configurado añadiendo un *RPC* personalizado tal como se muestra a continuación.



Con *Ganache* se crean cuentas, entonces puedes importar las cuentas que por defecto tienen 100 *Ethers*, con la clave privada dándole a importar cuenta cuando se está en la red privada de *localhost*.



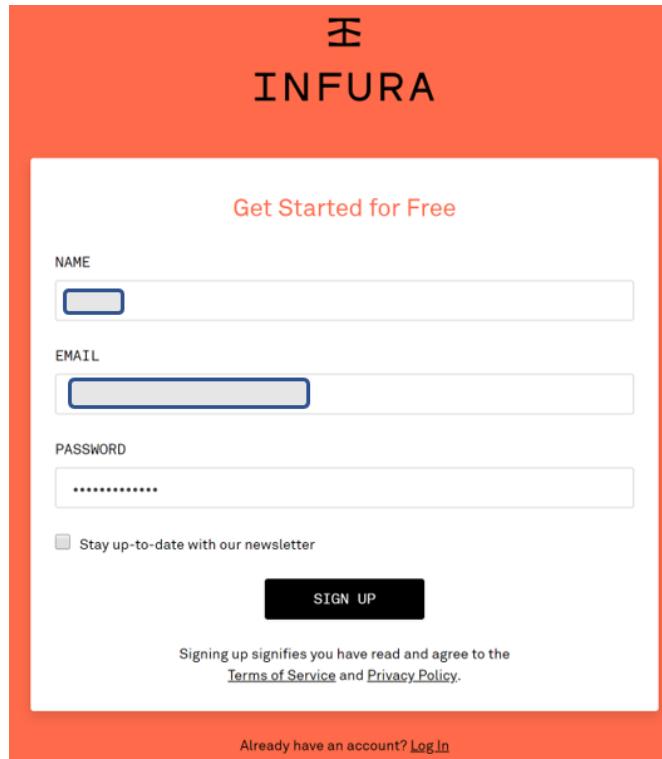
Puede que de vez en cuando *Metamask* deje de funcionar correctamente, para ello debes vigilar que *Metamask* no esté en modo privado. En caso de que continúe debes abrir y cerrar la red privada y ponerse en la red principal de *Ethereum* para iniciar sesión en *Metamask*. Otra posible solución es reiniciar el equipo. El último recurso consiste en deshabilitar y habilitar la extensión de nuevo.

En el caso de que deseas conectarte a la red de pruebas, *Rinkeby*, deberás seleccionar la red e importar las cuentas al igual que sucedió con *Ganache*. En la sección siguiente se enseña cómo realizar la conexión en *Rinkeby*.

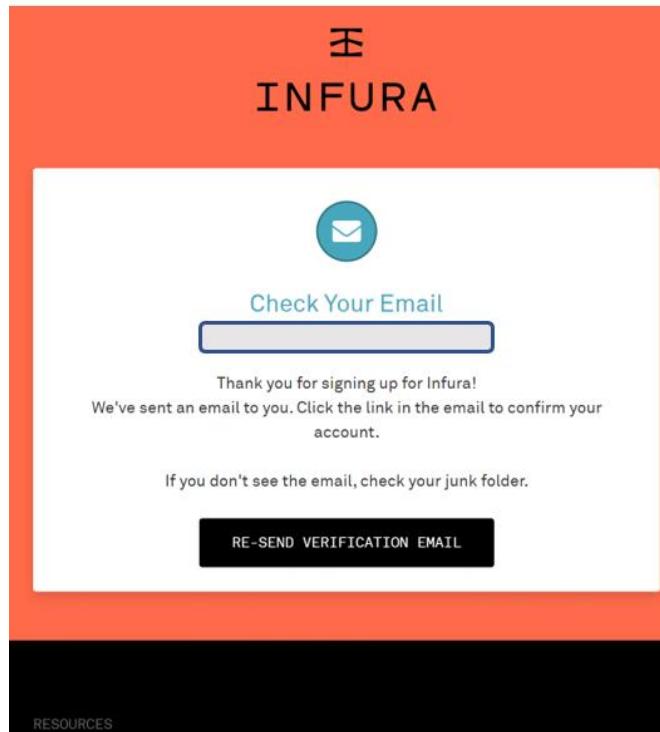
A.8 INFURA RINKEBY

Para poder realizar el despliegue en *Rinkeby*, se ha elegido realizarlo a través de un nodo remoto para eliminar problemas de sincronización y de actualización.

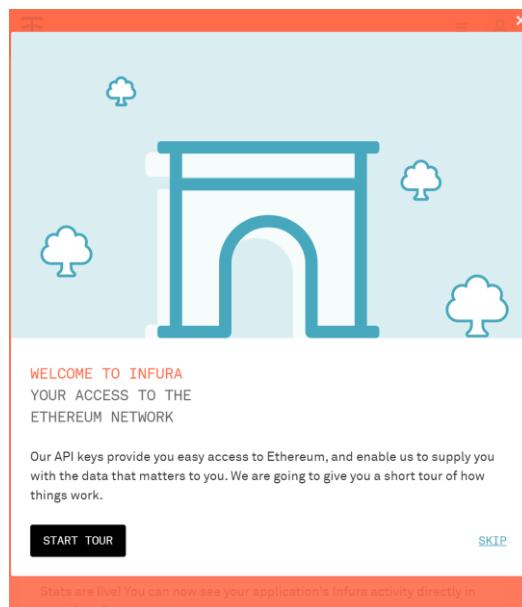
Solamente la primera vez se deberá crear una cuenta en *Infura* en la página web, <https://infura.io/> Para ello se debe llenar únicamente un nombre, email y contraseña.



The image shows a sign-up form for Infura, set against a red background. The form is contained within a white rectangular box. At the top center, it says "INFURA" in large, bold, black capital letters. Below this, in smaller red text, is "Get Started for Free". The form fields are labeled "NAME", "EMAIL", and "PASSWORD", each with a corresponding input field. There is also a checkbox labeled "Stay up-to-date with our newsletter". A "SIGN UP" button is located at the bottom left of the form area. Below the form, a small note states: "Signing up signifies you have read and agree to the [Terms of Service](#) and [Privacy Policy](#)". At the very bottom, there is a link "Already have an account? [Log In](#)".

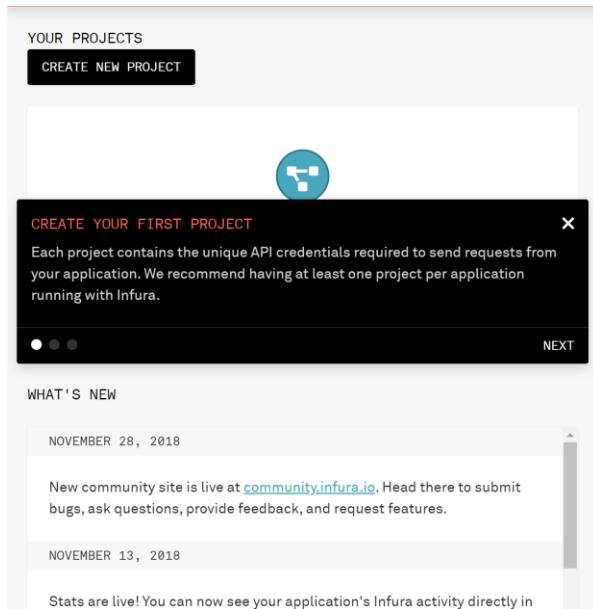


Al iniciar la cuenta aparecerán las siguientes pantallas.

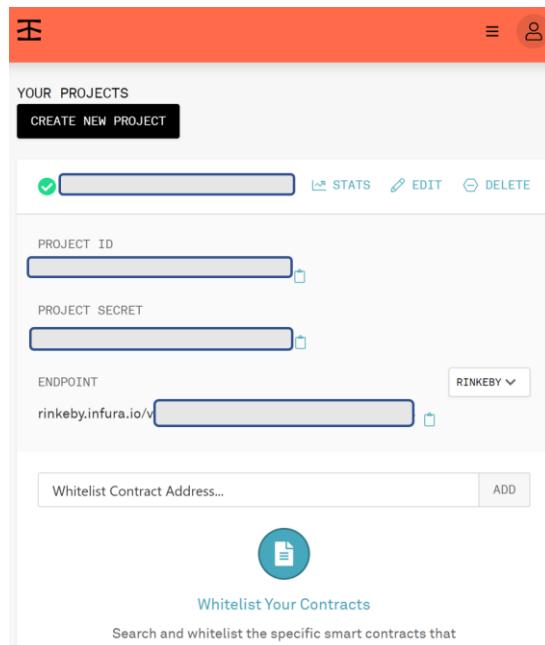
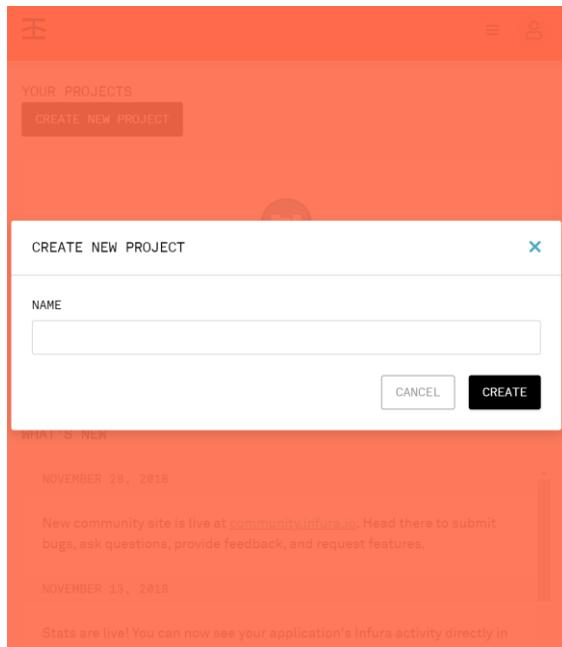


Se muestra la pantalla donde se puede crear un nuevo proyecto para desplegarse en *Rinkeby*. Se puede crear tantos proyectos como se quiera y vincularlos a los *Smart Contracts*

que se implementarán en *Rinkeby*. La creación del proyecto se realiza porque para implementar Smart Contracts con *Infura*, se debe obtener una clave *API* para interactuar con el nodo alojado en *Infura*.



Los siguientes pasos exhiben la creación del proyecto indicando el nombre del proyecto, se obtiene el ID del proyecto será la *API*, la clave secreta del proyecto y la dirección del *endpoint*. Hay que tener cuidado de que el *endpoint* tenga seleccionada la red *Rinkeby*, puesto que si cambias la red , el *endpoint* también cambia.



El siguiente paso es crear un *Wallet HD* para mantener la seguridad y no guardar la clave privada en un sitio inseguro ya que la clave privada es la que posibilita gastar sus criptomonedas o firmar transacciones. La *Wallet HD* es un sistema que le permite generar una secuencia de claves privadas a partir de una frase semilla única. Esta secuencia de claves es la misma si utilizas esa misma frase puesto que se trata de un recurso determinista.

Hay diferentes recursos de *Wallet*, la utilizada ha sido <https://iancoleman.io/bip39/>. Puesto que utilizar la de *Metamask* podría poner en peligro las cuenta en caso de se trabaje con la red principal de *Etherum*, ya que hay información conectada al proyecto a través del archivo *.env*.

A continuación, se muestra la creación de la *Wallet* a través de la herramienta indicada.

BIP39 - Mnemonic Code

Mnemonic Code Converter

Mnemonic

You can enter an existing BIP39 mnemonic, or generate a new random one. Typing your own twelve words will probably not work how you expect, since the words require a particular structure (the last word is a checksum).
For more info see the BIP39 spec (<https://github.com/bitcoin/bips/blob/master/bip-0039.mediawiki>).

Generate a random mnemonic: **GENERATE** words, or enter your own below.

Show entropy details
 Hide all private info

Mnemonic Language English 日本語 Español 中文(简体) 中文(繁體) Français Italiano 한국어

BIP39 Mnemonic

BIP39 Passphrase (optional)

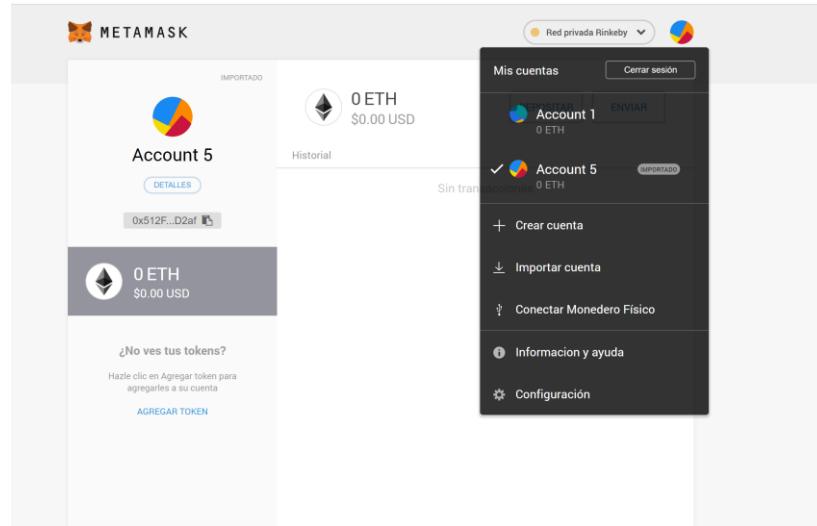
BIP39 Seed

Coin ETH - Ethereum

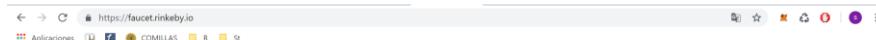
BIP32 Root Key

Path	Address	Public Key
m/44'/60'/0'/0/0	0x4911699f80934a168124394d6501b41d986cf195	0x033c438ab8334c85dce76660062069088d4f2e48ef5a04edfb77d7fac93
m/44'/60'/0'/0/1	0x05c9c99d43f6baE36955AE00C12b2c8eE677081	0x039ef7f2f7f5f7f6aa7270e48ea461d1024784fd03d013e5ccf71d9f1e5b
m/44'/60'/0'/0/2	0xa4199021303e147F7ed9f1e6eC597F50baE1706	0x034f7a359c766b7bfd71bc8e2be5868dc4a1aa5caa118b9cbd1adce5b3f
m/44'/60'/0'/0/3	0x95f14652f634948aE5866229a0d8949eA21612	0x03e0728d32fbccc8708ca375a1047ae8af6e15a9a25795ebd1252e9d46a7

Se tiene que abrir *Chrome*, la extensión de *Metamask* y seleccionar la red *Rinkeby*. Para realizar la importación de estas cuentas creadas al igual que se hizo con las cuentas de *Ganache* en local.



La siguiente fase es obtener saldo en las cuentas para poder implementar los *Smart Contracts* e interactuar con ellos. *Rinkeby* por seguridad para no colapsar la red, no te ofrece todo el *Ether* disponible. Se debe obtener a través de la publicación de un tweet con la cuenta que deseas obtener saldo. Después, indicando la dirección en esta página, <https://www.rinkeby.io/#faucet> se consigue el dinero



Rinkeby Authenticated Faucet

https://twitter.com/smartproperty11/status/1095361927373557760 Give me Ether ▾

0x72b72338887273a69f62c9fc8946be61e1a4e132 funded
8 peers 3857374 blocks 0.046256971665328e+56 Ethers 254516 funded

How does this work?

This Ether faucet is running on the Rinkeby network. To prevent malicious actors from exhausting all available funds or accumulating enough Ether to mount long running spam attacks, requests are fed to common 3rd party social network accounts.

Anyone having a Twitter, Google+ or Facebook account may request funds within the permitted limits.

To request funds via Twitter, make a [tweet](#) with your Ethereum address pasted into the contents (surrounding text doesn't matter).

Copy-paste the [tweets URL](#) into the above input box and fire away!

To request funds via Google Plus, publish a new [public post](#) with your Ethereum address embedded into the content (surrounding text doesn't matter).

Copy-paste the [posts URL](#) into the above input box and fire away!

To request funds via Facebook, publish a new [public post](#) with your Ethereum address embedded into the content (surrounding text doesn't matter).

Copy-paste the [posts URL](#) into the above input box and fire away!

You can track the current pending requests below the input field to see how much you have to wait until your turn comes.

The faucet is running invisible reCaptcha protection against bots.



El siguiente paso se debe instalar un paquete llamado *HDWalletProvider* para configurar el archivo del proyecto *truffle.js*. Para ello a través del *Powershell* y utilizando el comando indicado se instala.

```
PS C:\Users\teres\Documents\propiedad_pruebaT\propiedades> npm install truffle-hdwallet-provider
> websocket@1.0.28 install C:\Users\teres\Documents\propiedad_pruebaT\propiedad\node_modules\websocket
> (node-gyp rebuild 2>> builderror.log) || (exit 0)

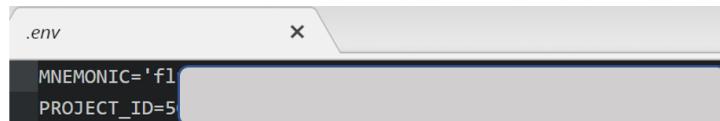
C:\Users\teres\Documents\propiedad_pruebaT\propiedad\node_modules\websocket>if not defined npm_config_node_gyp (node "C:\Program Files\nodejs\node-gyp\bin\node-gyp.js" rebuild --verbose --libsass_ext= --libsass_cflags= --libsass_ldflags= --libsass_library=)

```

Como la clave *API* de *Infura* pertenece a la cuenta proyecto. Se instala el paquete *dotenv* para mantener de forma confidencial los datos a través del siguiente comando.

```
PS C:\Users\teres\Documents\propiedad_pruebaT\propiedades> npm install dotenv
npm WARN Propiedad@0.0.1 No description
npm WARN Propiedad@0.0.1 No repository field.
npm WARN          SKIPPING OPTIONAL DEPENDENCY: fsevents@1.2.7 (node_modules\fsevents):
npm WARN          SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents@1.2.7: wanted {"os":"darwin","arch":"any"} (current: {"os":"win32","arch":"x64"})
+ dotenv@6.2.0
added 68 packages in 6.695s
PS C:\Users\teres\Documents\propiedad_pruebaT\propiedades>
```

En la carpeta del proyecto, se crea un archivo llamado *.env* para el paquete *dotenv*. La primera línea es la frase semilla que va entre comillas simples y la segunda línea es el ID del proyecto de *Infura*.



La siguiente fase es el *fichero truffle.js* agregar la información para poder conectarlo a la red *Rinkeby* tal como se muestra en la siguiente imagen donde se basará del fichero *.env* para conectarse al ID del proyecto y la *Wallet*. Se recuerda que la ID de la red de *Rinkeby* es la 4 mientras que la principal es la 0. Por otro lado, si no se indica en la configuración se utiliza la primera cuenta que aparece en la *Wallet* para realizar el despliegue.



```

truffle.js          .env
1 const HDWalletProvider = require("truffle-hdwallet-provider"); /*Cargar la
2 libreria HDWalletProvider*/
3 require('dotenv').config();/*Configura dotenv paquete para poder buscar desde el
4 archivo .env*/
5
6 module.exports = {
7   networks: {
8     ganache: {
9       host: "localhost",
10      port: 7545,
11      network_id: "*" /*El puerto de ganache es el 7545 al que se conecta
12      desde Metamask pej, netowrok id * permite que cualquier id sirva*/
13    },
14    propiedad: {
15      host: "localhost",
16      port: 8545,
17      network_id: "4224",
18      gas: 4700000
19    },
20    rinkeby: {
21      provider: function () {
22        return new HDWalletProvider(process.env.MNEMONIC, "https://
23        rinkeby.infura.io/v3/" + process.env.PROJECT_SECRET,1);
24        /*Relacionar rinkeby infura y el archivo .env, si pones al final
25        , numero es para que empieza desplegar a parti de esa cuenta
26        sin incluir esa*/
27      },
28      network_id: 4,
29      gas: 4500000,
30      gasPrice: 10000000000
31    }
32  };
33};

```


B. ANEXO B: GUÍA DE FUNCIONAMIENTO PARA EL USUARIO

Tras haber instalado los programas y herramientas necesarias para el desarrollo y despliegue de la *Dapp*, se indica a continuación cómo ejecutar la plataforma.

B.1 EJECUCIÓN EN LOCAL A TRAVÉS DE GANACHE

Antes de migrar se debe tener abierto la aplicación de *Ganache* u otra venta de comandos con *Ganache-cli* funcionando. Es decir o esta imagen o la siguiente figura.

```
Windows PowerShell
Copyright (C) 2016 Microsoft Corporation. Todos los derechos reservados.

PS C:\Windows\system32> ganache-cli
Ganache CLI v6.2.5 (ganache-core: 2.3.3)

Available Accounts
=====
(0) 0xa8566d421fb98c8f2ae1fcfd9be00954ec8cd58b (~100 ETH)
(1) 0xcf87895cf410fa0d206cdc037708e9749d3496ea (~100 ETH)
(2) 0x7f8b88dcraf181f97b5f499fc87f9e1299336220c (~100 ETH)
(3) 0xf86905416d572bb8bc1d27a43f17e362f2bc51e2 (~100 ETH)
(4) 0x07762b47582e86f406dsb6210a0e424337db1237 (~100 ETH)
(5) 0xfdad1699c8a109b43dbe0effece4b006d1663b0f (~100 ETH)
(6) 0x53fc784b575ffec3ae66321cc30e33a4f00ad951 (~100 ETH)
(7) 0x6e8865e712c4e35aeaefac42a510c52c102fdb41f7d (~100 ETH)
(8) 0xcea3094bd5d80ffa9d9b65b6c2f6f627a06ad81 (~100 ETH)
(9) 0xa9b344e86d8a7121d2ecb30cd565a08cd978ba59 (~100 ETH)

Private Keys
=====
(0) 0x0e5b4813db1ea0c437f3d12dc31dffcc388d92054b1692a4a7d755f509e97d7d6
(1) 0x66c03a70fbdf5717c5b8551016ae6bde9c3e7c8c085e628ea5fa8139d6aaef827
(2) 0xb0d0d08000014be38e84ab6cf89ea3aeec7c9f181f9fc710d2413c9f418d1d1
(3) 0xd5d5a9f567be9328656a1119232d4430d1076fed337ale46b36143b028685228
(4) 0x9ddde60e4412ae361c51343b562b69004ada2689a4977bf604ad1795022b9ff
(5) 0x62b63dd2b103e1da3292094caa4e29447968fc7424e4ec568561e90b1acee9
(6) 0x85f3637fc85f3ae87bee4f1d42d4fa34abe853ead5e0916fb7492d78e397a1a
(7) 0xfd20a885215ea1c12916881960c8339f0559d0dfac32d19e0fbda998641fd
(8) 0x7bd50b43bcc5ceb799259454ce00a77e79f9ea/6a87b62a116f9b2f33910d2d
(9) 0xd4708ef1a6e42b7afbdfe0028ea8f9285d369de4af13c68836fa689c8eb5f03c

HD Wallet
=====
Mnemonic: often ankle luxury caution mobile snow happy fabric cattle legend able brown
Base HD Path: m/44'/60'/0'/0/{account_index}

Gas Price
=====
20000000000

Gas Limit
=====
6721975

Listening on 127.0.0.1:8545
net_version
eth_accounts
eth_getBlockByNumber
eth_accounts
eth_getBlockByNumber
net_version
eth_getBlockByNumber
eth_getBlockByNumber
net_version
eth_estimateGas
eth_getBlockByNumber
eth_blockNumber
net_version
eth_sendTransaction

Transaction: 0x9bc6d308f3cod76b939d1b78f9a303810a079c27324843a2ecf513b9c6588db3
Contract created: 0x5de27652fe7b359943afed327a88403ed858493
Gas usage: 284908
Block Number: 1
Block Time: Mon Jan 14 2019 19:34:46 GMT+0100 (GMT+01:00)

eth_getTransactionReceipt
eth_getCode
```



MNEMONIC	ADDRESS	BALANCE	TX COUNT	INDEX	KEY
	0x90cB5F686ffB55E031a5A28F57be224A79bD71ab	200.00 ETH	0	0	🔑
	0xCC44F42C00D5519f3bB1C38e8A7cc134059F727b	200.00 ETH	0	1	🔑
	0x3EdF5F04BcB5CF651f6da4D6C85d2ddCe31b0696	200.00 ETH	0	2	🔑
	0x75e8D5E008ff5452b6AD6D4394467B3E906Ef525	200.00 ETH	0	3	🔑
	0x49241B79C1c55ACbd65969bAbe2544778EE4409e	200.00 ETH	0	4	🔑
	0xbc646B9072544d30B6f129daa82078e5Ff9262C2	200.00 ETH	0	5	🔑
	0xD879423058Df02BEE45E74fdE566DA169dad3920	200.00 ETH	0	6	🔑

A continuación, tras tener ejecutando *Ganache*, se prodcede a compilar y migrar el proyecto con el comando que aparece.

```
PS C:\Users\teres\Documents\propiedad_pruebaT\propiedad> truffle migrate --compile-all --reset --network ganache
```

B.2 EJECUCIÓN EN RINKEBY

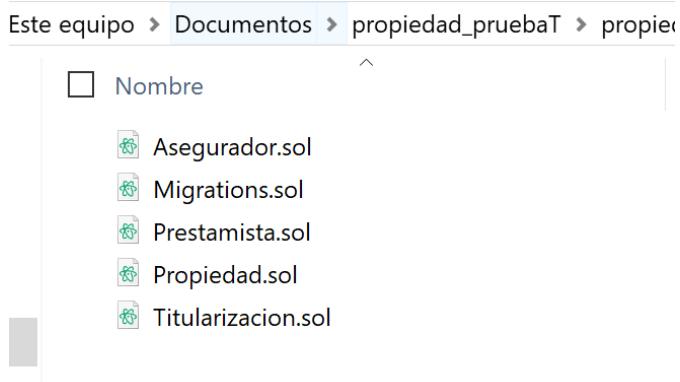
Para realizar la ejecución en la red de pruebas *Rinkeby*, se debe realizar primero la conexión en el proyecto a través del nodo remoto *Infura* explicado en las pasos previos de instalación. Tras tener listo el proyecto, se migra y compila el código fuente del proyecto indicando la red a la que lo queremos desplegar la *Dapp*. Se debe tener en cuenta que la ejecución en *Rinkeby* es mucho más lenta que en el nodo local.

```
PS C:\Users\teres\Documents\propiedad_pruebaT\propiedad> truffle migrate --compile-all --reset --network rinkeby
Compiling ./contracts\Asegurador.sol...
Compiling ./contracts\Migrations.sol...
Compiling ./contracts\Prestamista.sol...
Compiling ./contracts\Propiedad.sol...
Compiling ./contracts\Titularizacion.sol...
Writing artifacts to ./build\contracts

Using network 'rinkeby'.

Running migration: 1_initial_migration.js
Replacing Migrations...
... 0x4c72607ac05f5209d2338c94026218a7efc1017d9cda7c5b6a7e15eec137f6d3
Migrations: 0xe99cb61a556a2e2f87b16a70556dda9004a8fb2c
Saving successful migration to network...
... 0xe034f1e8780e80e80bdaaac7594b5433ab16c3b65cf59aeb3cedb021394f0af
Saving artifacts...
Running migration: 2_deploy_contracts.js
Replacing Propiedad...
... 0x6200ab43a10c20df28742577a293c416cf59a6692146dd7eb06d37900046f931
Propiedad: 0xf1c9bf759108c59376dc8747d6a18f9b6d7885e
Replacing Prestamista...
... 0x696b93138ab9285cd7a84ade60963b081da86262c7cdda7ca7bc170795efc1f6
Prestamista: 0xd6b666b6cb9289b1d3203f0dae49981b5519785
Replacing Asegurador...
... 0xbab807bbd48738cc689eb338a0196c445c44ac0df6e87f0e72620b823d5e3485
Asegurador: 0xb51c1c9d43cc559a1c649ff1efd786ed37088c2a
Saving successful migration to network...
... 0x81da7cf81a0b012b4dad24cd5ed4db1e5054c95a8f6ff91289de119c6d29a18
Saving artifacts...
PS C:\Users\teres\Documents\propiedad_pruebaT\propiedad> npm run dev
```

Como la compilación ha ido correctamente crea los archivos *.json* dentro de la carpeta propiedad/build/contracts



A continuación, se procede a ejecutar la *Dapp* con el siguiente comando que hace uso de *lite-server*.

```
PS C:\Users\teres\Documents\propiedad_pruebaT\propiedad> npm run dev
> Propiedad@0.0.1 dev c:\Users\teres\Documents\propiedad_pruebaT\propiedad
> lite-server

** browser-sync config **
{
  injectChanges: false,
  files: [ './**/*.{html,htm,css,js}' ],
  watchOptions: { ignored: 'node_modules' },
  server: {
    baseDir: [ './src', './build/contracts' ],
    middleware: [ [Function], [Function] ] }
}

[Browsersync] Access URLs:
-----
  Local:
  External:
-----
      UI:
  UI External:
-----
[Browsersync] Serving files from:
[Browsersync] Serving files from:
[Browsersync] Watching files...
19.06.10 12:08:59 200      /index.html
19.06.10 12:08:59 200      /css/app.css
19.06.10 12:08:59 200      /media/logo.png
19.06.10 12:08:59 200      /media/1.jpg
19.06.10 12:09:00 200      /js/app.js
19.06.10 12:09:00 200      /media/anuncio_b.png
19.06.10 12:09:00 200      /media/prestamo_b.png
19.06.10 12:09:00 200      /js/truffle-contract.js
19.06.10 12:09:00 200      /media/3.jpg
19.06.10 12:09:00 200      /media/asegurador_b.png
19.06.10 12:09:00 200      /media/historico_b.png
19.06.10 12:09:00 200      /media/8.jpg
19.06.10 12:09:01 200      /Prestamista.json
19.06.10 12:09:01 200      /Asegurador.json
19.06.10 12:09:01 200      /Propiedad.json
19.06.10 12:09:01 404      /favicon.ico
```