



COMILLAS

UNIVERSIDAD PONTIFICIA

ICAI

MASTER UNIVERSITARIO EN INGENIERÍA DE TELECOMUNICACIONES

TRABAJO FIN DE MASTER

DESARROLLO DE TECNOLOGÍA BLOCKCHAIN PARA EL INTERCAMBIO DE EXPEDIENTES ACADÉMICOS EN EL ÁMBITO INTERUNIVERSITARIO

Autor: Alfonso Villarino Arias

Director: Atilano Fernández-Pacheco Sánchez-Migallón

Director: Jose Luis Gahete Díaz

Madrid

Declaro, bajo mi responsabilidad, que el Proyecto presentado con el título
**DESARROLLO DE TECNOLOGÍA BLOCKCHAIN PARA EL INTERCAMBIO DE
EXPEDIENTES ACADÉMICOS EN EL ÁMBITO INTERUNIVERSITARIO**

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.: Alfonso Villarino Arias

Fecha: 14/06/2019

Autorizada la entrega del proyecto

EL DIRECTOR DEL PROYECTO

Fdo.: Atilano Fernández-Pacheco Sánchez-Migallón

Fecha://

Fdo.: Jose Luis Gahete Díaz

Fecha://

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

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

El autor D. Alfonso Villarino Arias DECLARA ser el titular de los derechos de propiedad intelectual de la obra: DESARROLLO DE TECNOLOGÍA BLOCKCHAIN PARA EL INTERCAMBIO DE EXPEDIENTES ACADÉMICOS EN EL ÁMBITO INTERUNIVERSITARIO, 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) Reproducir la 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, a14.. dejunio..... de ...2019.

ACEPTA



Fdo.....

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



COMILLAS

UNIVERSIDAD PONTIFICIA

ICAI

MASTER UNIVERSITARIO EN INGENIERÍA DE TELECOMUNICACIONES

TRABAJO FIN DE MASTER

DESARROLLO DE TECNOLOGÍA BLOCKCHAIN PARA EL INTERCAMBIO DE EXPEDIENTES ACADÉMICOS EN EL ÁMBITO INTERUNIVERSITARIO

Autor: Alfonso Villarino Arias

Director: Atilano Fernández-Pacheco Sánchez-Migallón

Director: Jose Luis Gahete Díaz

Madrid

Agradecimientos

Quiero agradecerle este trabajo especialmente a mis directores, Atilano y Jose Luis, los cuales han hecho un esfuerzo extraordinario para guiarme a lo largo de este proceso.

Así mismo, quiero mostrar un sincero agradecimiento a mi familia, sin los cuales no podría estar aquí.

Finalmente agradecer a toda la gente que ha estado ahí durante estos dos últimos años. A todos, Gracias.

DESARROLLO DE TECNOLOGÍA BLOCKCHAIN PARA EL INTERCAMBIO DE EXPEDIENTES ACADÉMICOS EN EL ÁMBITO INTERUNIVERSITARIO

Autor: Villarino Arias, Alfonso.

Director: Fernández-Pacheco Sánchez-Migallón, Atilano.

Director: Gahete Díaz, Jose Luis.

Entidad Colaboradora: ICAI – Universidad Pontificia Comillas)

RESUMEN DEL PROYECTO

En este proyecto se ha llevado a cabo, con éxito, la realización de una plataforma para la interconexión entre alumnos, universidades y empresas para la aplicación a ofertas empresariales, por parte de los alumnos, certificando títulos académicos, a través de las universidades, impulsado todo a través de la tecnología BockChain.

Palabras clave: BlockChain, Identidad digital, Ethereum

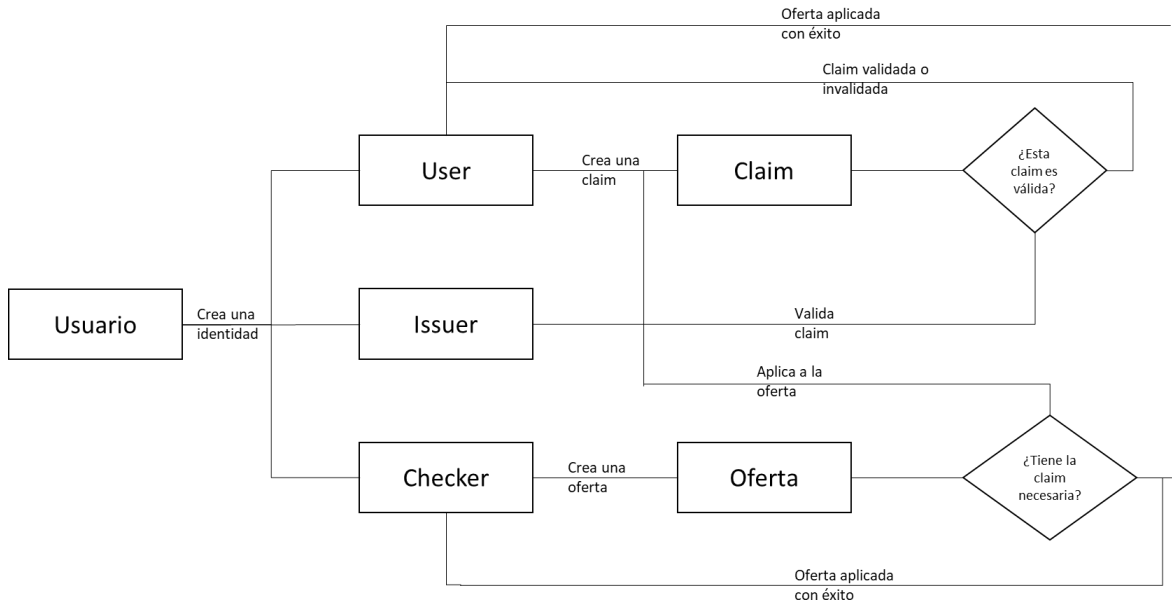
1. Introducción

Con el tiempo ha ido evolucionando la manera de demostrar quién se es antes los demás. Al principio se desarrollaron sistemas primitivos como el sello, o bandera, hasta llegar al actual sistema de pasaporte y carnet de identificación (dependiendo del país). Sin embargo, estos sistemas han quedado obsoletos ante la aparición de internet, y la complejidad de su uso.

2. Definición del proyecto

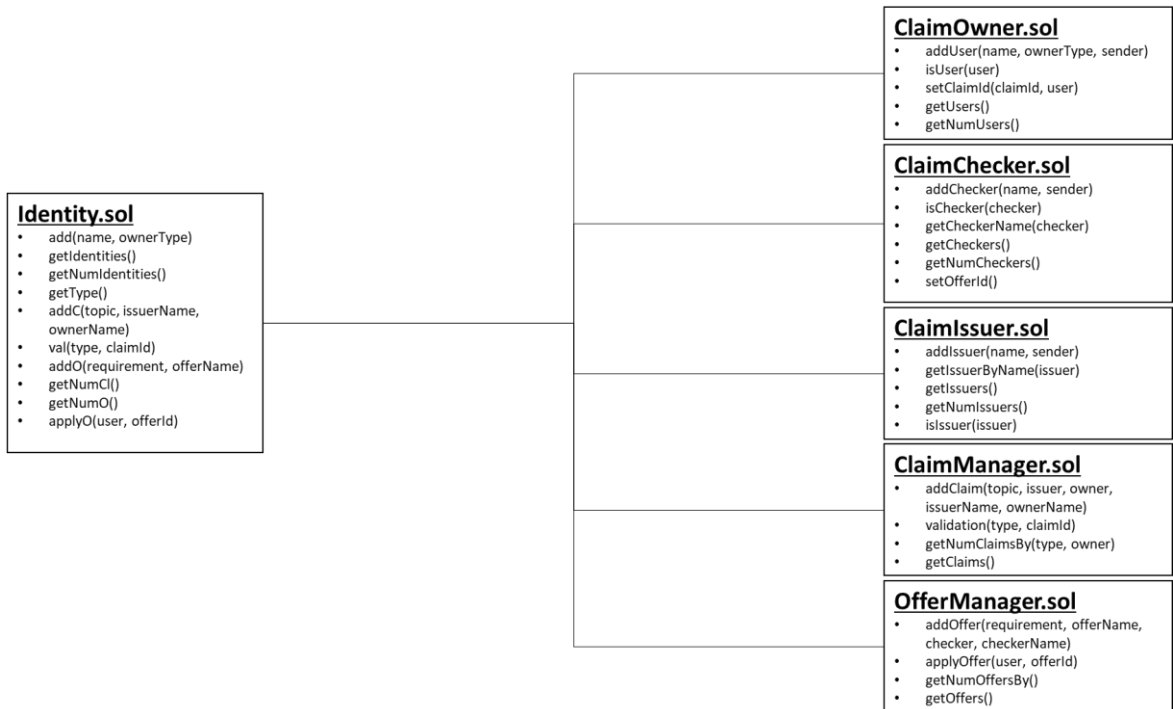
Para el desarrollo del proyecto de conexión entre los tres tipos de identidades, y la simulación de sus interacciones, se han desarrollado una plataforma para permitir el registro de las identidades, creación y gestión de certificaciones académicas y creación y aplicación a ofertas de trabajo.

Dicha plataforma se basa en el siguiente diagrama de flujo:



3. Descripción del sistema

El Desarrollo de dicha plataforma ha sido posible gracias al desarrollo de seis Smart contracts, los cuales se muestran a continuación:



Esto, sumado a un servidor node, permitió lanzar una plataforma web que, conectada a una red local de Ethereum (Ganache) a través de MetaMask, permitiese realizar la interconexión entre las diferentes identidades.

Así mismo, también se desarrollaron una serie de pruebas, a través del framework de test JavaScript Mocha, que permitieron comprobar el correcto funcionamiento del sistema.

4. Resultados

El sistema funcionó de manera correcta, y sin ningún fallo, cumpliendo con los objetivos más importantes, y destacados del proyecto, los cuales eran:

1. Crear claims, respaldadas por un Issuer, y emplearlas para aplicar a ofertas de trabajo, creadas por los Checker, para tratar de optar a un puesto vacante, siempre y cuando la claim creada se encuentre validada y el User disponga de los requisitos, claims, necesarios para aplicar a la oferta.
2. Validar o invalidar las claims creados por los usuarios contra tu institución, si eres un Issuer.
3. Crear ofertas de trabajo que necesiten de unos requisitos, para poder aplicar a ellas, si eres un Checker.

5. Conclusiones

La plataforma ha permitido construir un entorno en el cual tanto alumnos como universidades y empresas, se sintieran cómodos, y seguros, a la hora de poder aplicar a ofertas de trabajo para incorporarse al mundo laboral.

6. Referencias

- [1] Ian Sommerville, Pearson, 2010. “Software Engineering 9”. <https://www.pearson.com/us/higher-education/product/Sommerville-Software-Engineering-9th-Edition/9780137035151.html>.
- [2] Preukschat, Alexander. “Blockchain: la revolución industrial de internet”. https://books.google.es/books/about/Blockchain_la_revoluci%C3%B3n_industrial_de.html?id=Lb7DDgAAQBAJ&source=kp_book_description&redir_esc=y
- [3] Pastor, Javier. “Qué es blockchain: la explicación definitiva para la tecnología más de moda”. <https://www.xataka.com/especiales/que-es-blockchain-la-explicacion-definitiva-para-la-tecnologia-mas-de-moda>

DEVELOPMENT OF BLOCKCHAIN TECHNOLOGY FOR THE EXCHANGE OF ACADEMIC RECORDS AT INTER-UNIVERSITY LEVEL

Author: Villarino Arias, Alfonso.

Supervisor: Fernández-Pacheco Sánchez-Migallón, Atilano.

Supervisor: Gahete Díaz, Jose Luis.

Collaborating Entity: ICAI – Universidad Pontificia Comillas)

ABSTRACT

This project has successfully carried out the creation of a platform for interconnection between students, universities and companies for application to business offers, by students, certifying academic degrees, through universities, driven all through BockChain technology.

Keywords: BlockChain, Digital identity, Ethereum

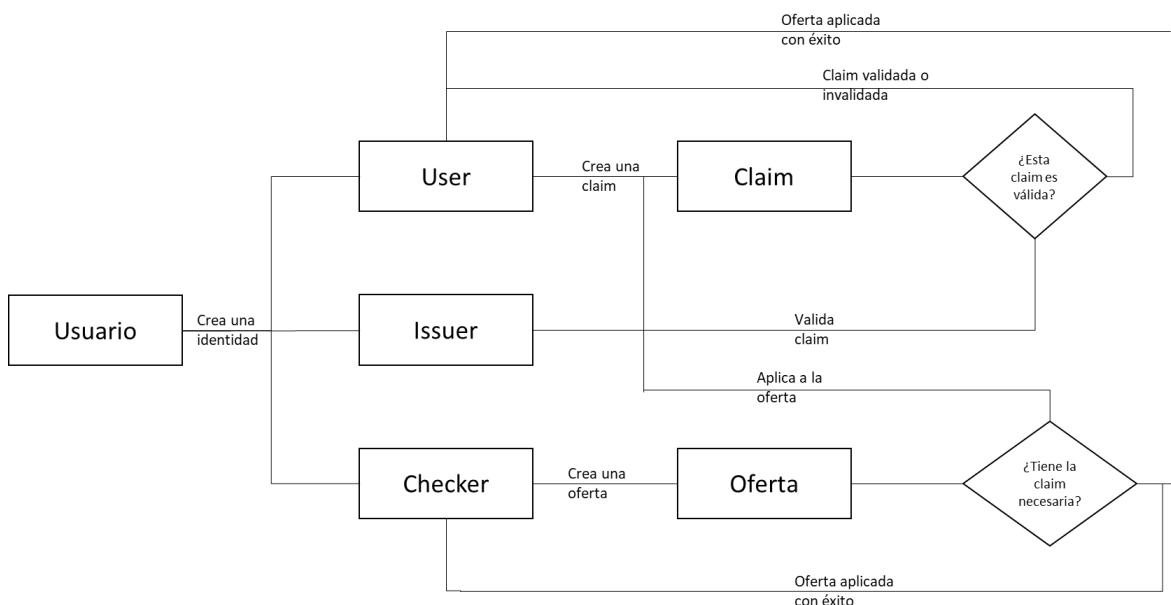
1. Introduction

Over time, the way we show who we are before others has evolved. In the beginning primitive systems were developed as the seal, or flag, until arriving at the current system of passport and identification card (depending on the country). However, these systems have become obsolete due to the appearance of the Internet and the complexity of its use.

2. Project definition

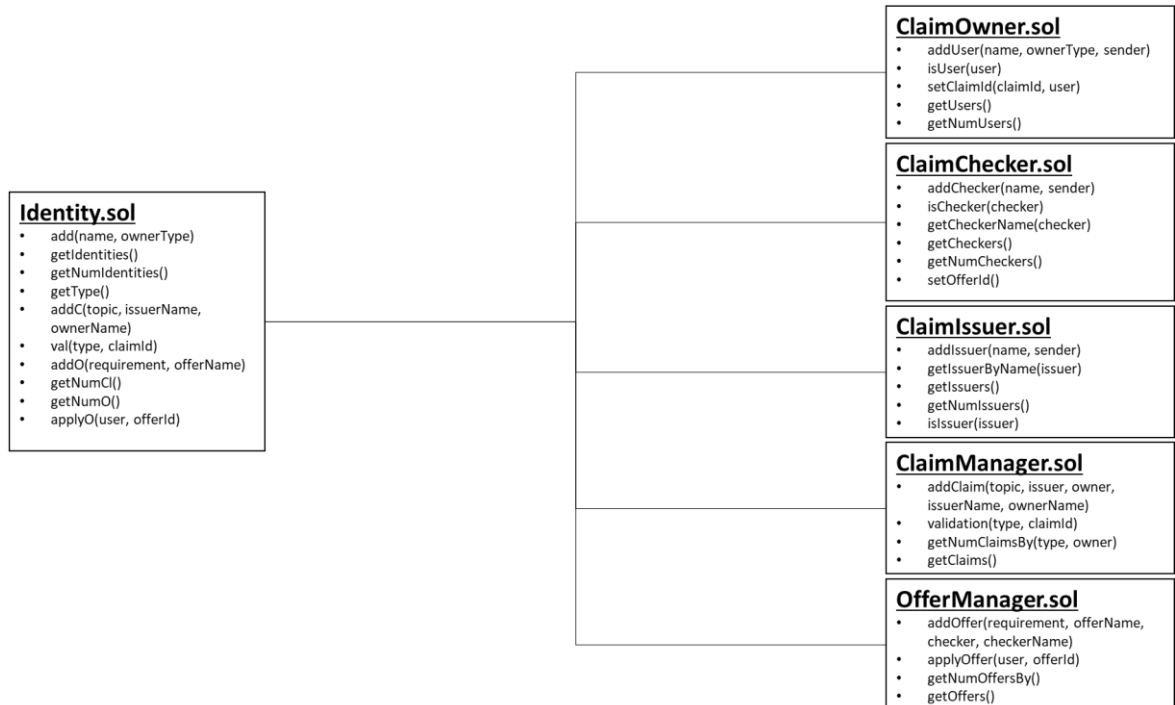
For the development of the project of connection between the three types of identities, and the simulation of their interactions, a platform has been developed to allow the registration of identities, creation and management of academic certifications and creation and application to job offers.

This platform is based on the following flow chart:



3. System description

The development of this platform has been possible thanks to the development of six Smart contracts, which are shown below:



This, added to a node server, allowed to launch a web platform that, connected to a local network of Ethereum (Ganache) through MetaMask, allowed the interconnection between the different identities.

Also, a series of tests were developed, through the Mocha JavaScript test framework, which allowed to check the correct functioning of the system.

4. Results

The system worked correctly, without any failure, fulfilling the most important and outstanding objectives of the project, which they were:

1. Create claims, supported by an Issuer, and use them to apply for job offers, created by the Checkers, to try to apply for a vacant position, provided that the claim created is validated and the User has the requirements, claims, necessary to apply to the offer.
2. Validate or invalidate claims created by users against your institution, if you are an Issuer.
3. Create job offers that need some requirements, to be able to apply to them, if you are a Checker.

5. Conclusions

The platform has made it possible to build an environment in which students, universities and companies feel comfortable and safe when applying for job offers to join the world of work.

6. References

- [1] Ian Sommerville, Pearson, 2010. “Software Engineering 9”. <https://www.pearson.com/us/higher-education/product/Sommerville-Software-Engineering-9th-Edition/9780137035151.html>.
- [2] Preukschat, Alexander. “Blockchain: la revolución industrial de internet”. https://books.google.es/books/about/Blockchain_la_revoluci%C3%B3n_industrial_de.html?id=Lb7DDgAAQBAJ&source=kp_book_description&redir_esc=y
- [3] Pastor, Javier. “Qué es blockchain: la explicación definitiva para la tecnología más de moda”. <https://www.xataka.com/especiales/que-es-blockchain-la-explicacion-definitiva-para-la-tecnologia-mas-de-moda>

Índice de la memoria

1. Introducción	7
2. Descripción de las Tecnologías.....	11
2.1 Visual Studio Code.....	11
2.2 Ganache	12
2.3 Truffle Framework	13
2.4 Mocha.....	14
2.5 VirtualBox.....	14
2.6 Ubuntu.....	15
2.7 Node 8	16
2.8 NPM	16
2.9 MetaMask.....	17
2.10 Solidity	18
2.11 Web3JS.....	18
2.12 Trello	19
2.13 Ethereum	20
3. Estado de la Cuestión	21
3.1 Historia de la identidad.....	21
3.1.1 La identidad digital	22
3.2 Blockchain (BC).....	23
3.2.1 ¿Qué es un bloque y cómo se forma?.....	23
3.2.2 Funcionamiento del Blockchain	25
3.3 La identidad en la empresa.....	29
3.3.1 El Blockchain y la identidad digital	30
3.3.2 La identidad digital y la universidad.....	30
4. Definición del Trabajo	32
4.1 Motivación	32
4.2 Objetivos	34
4.3 Metodología.....	34
4.4 Planificación y Estimación Económica.....	37

5. Sistema/Modelo Desarrollado	39
5.1 Portal de empleo	39
5.2 Diagrama de flujo interfaz de usuario	40
5.3 Diagrama de flujo de un “User”	41
5.4 Diagrama de flujo de un “Issuer”	42
5.5 Diagrama de flujo de un “Checker”	43
5.6 Smart contracts de la aplicación	45
5.6.1 Identity.sol	46
5.6.2 Código Identity.sol	47
5.6.3 Código ClaimOwner.sol	48
5.6.4 Código ClaimIssuer.sol	48
5.6.5 Código ClaimChecker.sol	49
5.7 Servidor de la aplicación	50
5.7.1 Inicio del servidor	50
5.7.2 Conexión con la red ethereum	51
5.7.3 Function “createIdentity”	51
5.8 Tests con Mocha	53
5.8.1 Prueba de creación de un “User”	54
5.8.2 Prueba de error al crear dos identidades con la misma cuenta	54
5.8.3 Prueba de creación de un “Issuer”	55
5.8.4 Prueba de creación de un “Checker”	56
5.8.5 Prueba de creación de una claim	56
5.8.6 Prueba de validación la claim	57
5.8.7 Prueba de crear una oferta de trabajo	58
5.8.8 Prueba de aplicación a la oferta de trabajo	59
5.8.9 Prueba de devolución de todas la información guardada	60
5.8.10 Prueba cantidad de identidades creadas	63
6. Análisis de Resultados	64
7. Conclusiones y trabajos futuros	66
8. Bibliografía	68
ANEXO A: Guía de instalación	70

8.1	VirtualBox	70
8.2	Descarga de Ubuntu e instalación de la máquina virtual.....	76
8.3	Instalación de Ubuntu en la máquina virtual	84
8.4	Instalación de Node 8	89
8.5	Instalación NPM.....	90
8.6	Instalación Truffle	91
8.7	Instalación proyecto en Truffle	93
8.8	Instalación Visual Studio Code	93
8.9	Instalación Ganache	98
8.10	Instalación de Mocha.....	100
<i>ANEXO B: Manual de usuario.....</i>		<i>104</i>
8.1	Puesta en marcha y Vista inicial.....	104
8.2	Crear un Issuer.....	105
8.3	Crear un checker.....	107
8.4	Creación de una oferta.....	108
8.5	Crear un User.....	110
8.6	Creación de una claim	111
8.7	Validación de la claim.....	113
8.8	Aplicar a una oferta	116

Índice de figuras

Figura 1 - Representación esquemática de la organización de Bloques en BC.....	24
Figura 2 - Representación esquemática de la Cadena de bloques de BC. El primer Bloque Génesis se representa en negro, los Bloques Principales en azul y los Bloques Huérfanos en gris.	25
Figura 3 - Diagrama de GANTT del proyecto	35
Figura 4 - Trello de planificación del Proyecto.....	36
Figura 5 - Diagrama de flujo de interfaz de usuario.....	40
Figura 6 - Diagrama de flujo de un "User".....	41
Figura 7 - Diagrama de flujo de un "Issuer".....	42
Figura 8 - Diagrama de flujo de un "Checker"	43
Figura 9 - Diagrama de flujo de la Plataforma	44
Figura 10 - Diagrama de contratos de la Plataforma.....	45
Figura 11 - Código de la función "add" del contrato "Identity.sol"	48
Figura 12 - Código de la función "addUser" del contrato "ClaimOwner.sol".....	48
Figura 13 - Código de la función "addIssuer" del contrato "ClaimIssuer.sol"	49
Figura 14 - Código de la función "addChecker" del contrato "ClaimChecker.sol"	49
Figura 15 - Código de la función de inicio del servidor.....	50
Figura 16 - Código de la función de conexión del servidor al contrato "Identity.sol"	51
Figura 17 - Código de la función "CreateIdentity" del servidor.....	52
Figura 18 - Resultado de todos los test de Mocha.....	53
Figura 19 - Código del test de creación de un "User"	54
Figura 20 - Resultado del test de creación de un "User"	54
Figura 21 - Código del test de error al crear dos identidades con la misma cuenta	54
Figura 22 - Resultado del test de error al crear dos identidades con la misma cuenta.....	55
Figura 23 - Código del test de creación de un "Issuer"	55
Figura 24 - Resultado del test de creación de un "Issuer"	55

Figura 25 - Código del test de creación de un "Checker"	56
Figura 26 - Resultado del test de creación de un "Checker"	56
Figura 27 - Código del test de creación de una claim	56
Figura 28 - Resultado del test de creación de una claim	57
Figura 29 - Código del test de validación de una claim	57
Figura 30 - Resultado del test de validación de una claim	58
Figura 31 - Código del test de creación de una oferta de trabajo	58
Figura 32 - Resultado del test de creación de una oferta de trabajo	58
Figura 33 - Código del test de aplicación a una oferta de trabajo	59
Figura 34 - Resultado del test de aplicación a una oferta de trabajo	59
Figura 35 - Código del test de devolución de toda la información guardada (1)	60
Figura 36 - Código del test de devolución de toda la información guardada (2)	61
Figura 37 - Código del test de devolución de toda la información guardada (3)	62
Figura 38 - Resultado del test de devolución de toda la información guardada.....	63
Figura 39 - Código de test de cantidad de identidades creadas	63
Figura 40 - Resultado del test de cantidad de identidades creadas.....	63

Índice de tablas

Tabla 1 - Estimación económica del Proyecto 38

1. INTRODUCCIÓN

El Diccionario de la Real Academia Española (RAE) define identidad como “Conjunto de rasgos propios de un individuo o de una colectividad que los caracterizan frente a los demás”. Es decir, es la suma de aquellos atributos o cualidades que le otorgan a una persona, o colectivo, ser quien es. Por lo tanto, es importante la posesión de algo que le permita a la persona acreditar que es quien dice ser y puede hacer todo lo que se espera de sí.

Históricamente en las sociedades tradicionales y hasta mediados del XV, el tipo de reconocimiento de identidad que predominaba era la identificación cara a cara. En esa época comienzan a aparecer y a desarrollarse signos de identidad sustitutivos de la persona, como el sello, los escudos de armas, las insignias o incluso marcas permanentes, como las que recibían los delincuentes, como castigo por sus delitos.

A partir del siglo XV el reconocimiento mediante lo escrito desempeña un papel más destacado, así las iglesias realizan cuidadosos registros de fieles mediante libros parroquiales, que también serán utilizados por el poder civil, con la justicia, la recaudación de impuestos y la milicia como ámbitos prioritarios de utilización.

Durante el siglo XIX, las revoluciones liberales y sobre todo los procesos de construcción de los estados provocan la transferencia de los registros de la iglesia al estado y la proliferación de los documentos de identidad emitidos por diversas instituciones. Conforme transcurre el siglo XX y como consecuencia de las crisis políticas, las guerras, el terrorismo y el control de la inmigración se produce un avance en el registro de información. En la Primera Guerra Mundial, los países requerían un férreo control de acceso y de vigilancia de quién salía del mismo, y fue donde se introdujeron los primeros requisitos de pasaporte, que se normalizó definitivamente en 1980. En España, se obligó a la creación del DNI a finales de la Segunda Guerra Mundial, con el objetivo de controlar el censo y los movimientos migratorios de los españoles.

En la sociedad moderna existen una serie de documentos que permiten la acreditación de la identidad tanto de manera internacional (Pasaporte) como diversos documentos de carácter nacional, Documento Nacional de Identidad (DNI), Número de Identificación Fiscal (NIF), licencia de conducir, tarjeta de la Seguridad Social, etc. Con la creación de todos estos documentos se produce el control de la población, posibilitando que se pueda identificar cada individuo como portador de una serie de características y atributos que lo definen y que conforman nuestra identidad analógica.

La aparición de Internet en 1969 y sobre todo de la World Wide Web en 1990 ha complicado de nuevo la forma de identificarse, pues el anonimato de la red puede burlar la realización de transacciones veraces a través de ella. Nace así la necesidad de ser capaz de verificar que una persona que está haciendo una serie de acciones en Internet, es quien dice ser y tiene las habilidades, atributos o cualidades que la caracterizan en la Red.

La identidad digital o identidad 2.0, es todo lo que nos identifica en el entorno online. Cada vez utilizamos más todas las posibilidades y servicios que nos ofrece Internet para hacer todo tipo de operaciones. Normalmente la identidad digital se acredita mediante un certificado digital, que es un fichero informático firmado electrónicamente por un proveedor de servicios de certificación que en España puede ser una persona física o jurídica que se rige por el Reglamento del Parlamento Europeo eIDAS (910/2014) y que otras entidades lo consideran con autoridad para emitirlo.

En la actualidad, muchos países tratan de producir sus identificaciones nacionales, DNI en el caso de España, de manera que puedan ser utilizados como certificados digitales para firmar en la red. Estos certificados tienen la ventaja de ahorrar tiempo y dinero al usarlos para realizar múltiples trámites en cualquier momento y lugar. Sin embargo, estos podrían ser copiados o robados por alguien con el conocimiento suficiente.

El papel que tiene la identidad digital es fundamental en el futuro ya que cualquier actividad de tipo económico o de tipo social se traslada al entorno digital. Los recientes y enormes fallos de seguridad y robos de datos a entidades tan destacadas como en Facebook (que

afectó a aproximadamente 30 millones de usuarios) ponen de manifiesto el grave problema que existe en el proceso de gestión de las identidades.

Paralelamente también aumenta nuestra preocupación porque nuestros datos sean seguros, y aunque las técnicas de verificación de identidad así como la prevención del fraude “on line” son cada vez más sofisticadas, lo cierto es que se pueden producir fraudes de identidad que afectan tanto a las personas como a las empresas. Otra de las preocupaciones inherentes a la identidad digital es la privacidad de los datos que los usuarios suministran con el uso de la red y que determinadas empresas ceden a terceros. Cuando se comparte la información de los usuarios la empresa ya no controla que se hace con esos datos y además no puede garantizar su almacenamiento de forma segura. La responsabilidad de proteger los datos debe entonces regularse para garantizar un uso legítimo y controlado de ellos.

Las transacciones digitales son cada vez más frecuentes y a su vez también están sujetas a regulación para garantizar que no se produzca fraude, blanqueo de capitales etc. Surge por tanto la necesidad de una serie de tecnología que permita a la persona identificarse y autenticarse obteniendo entonces permiso para el acceso a los recursos y transacciones en Internet. Las tecnologías que usan las empresas para la verificación de la identidad implican claras ventajas para ellas pues por un lado permiten al usuario una experiencia mejorada en las transacciones que realice convirtiéndose en más rápidas y fáciles. Por otra parte, les ayuda a reducir las pérdidas millonarias provocadas por el fraude online. En cuanto al usuario, les aporta facilidad de uso, seguridad en sus datos personales y reducción del riesgo de suplantación de identidad. Sin embargo, estos procesos también desencadenan la aparición de nuevos problemas de confidencialidad y uso de datos y por ello es fundamental instaurar un nuevo modelo de relación entre los sectores implicados, usuarios y empresas, para facilitar el alta y la contratación de los posibles productos y servicios que se ofrecen la Web.

Hasta el momento la mayoría de los sistemas de identidad asocian datos con individuos, pero esta información no es propiedad del usuario además se multiplican en distintas bases de datos en cada sitio Web. Por ello, han surgido una serie de iniciativas que promueven la identidad digital auto soberana. Este proceso sería ventajoso ya que la autoridad acreditadora

de identidad no es única y al descentralizar el proceso se conseguiría un sistema de identidad distribuida donde el usuario tendría el control, y ninguna institución podría poner en compromiso los datos de su identidad.

BlockChain o Cadena de Bloques (BC) es una base de datos pública, compartida y descentralizada donde se registran las operaciones que se realizan en cualquier transacción. Es una tecnología que almacena información de manera distribuida y permanente, garantizando su confidencialidad e inmutabilidad y sin la necesidad de que un tercero verifique su validez. A esta gran ventaja, además podemos sumar la de que se puede añadir a la cadena el historial de evolución que tenga ese acuerdo consiguiendo así un registro fehaciente de cada tarea que se realice.

BC se utilizó en un principio como un sistema de contabilidad, pero esta tecnología puede tener múltiples usos. Muchos especialistas creen que se trata de la solución óptima para procurar un sistema con el que conseguir validar identidades de forma fehaciente, segura e inalterable. Por ello, la aparición de BC supuso una nueva implementación en los sistemas de gestión de la identidad consiguiendo el intercambio de datos de los usuarios de manera segura y permitiendo dar solución a esta problemática de identidad en la red, por ello muchas empresas están desarrollando servicios para aplicar la tecnología BC en este ámbito.

En este sentido es destacable la importancia que ha tenido la identidad personal en el contexto histórico. Por ello es de necesario desarrollar las posibilidades que ofrecen las nuevas tecnologías, como BlockChain, en esta área para poder dar soluciones a los problemas de autenticación actuales.

2. DESCRIPCIÓN DE LAS TECNOLOGÍAS

2.1 *VISUAL STUDIO CODE*



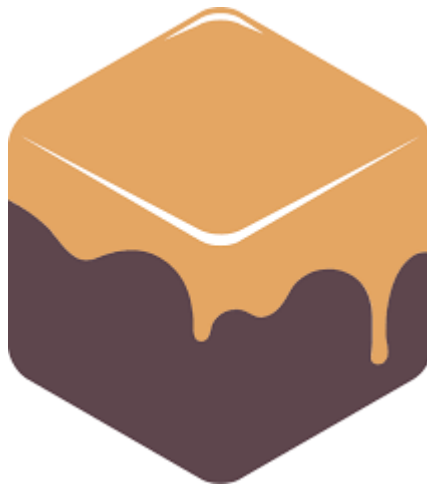
Visual Studio Code es un editor de código fuente, esto es, una plataforma a través de la cual se pueden escribir programas en cualquier lenguaje de programación, permitiendo este la descarga de diferentes tipos de componentes y extensiones, que ayudan a la edición, y corrección de errores del código. Así mismo, Visual Studio Code, permite recorrer las diferentes carpetas y documentos del proyecto, de una manera ordenada, a través del explorador de archivos.

Entre las ventajas que destacan de Visual Studio Code y que han hecho que este proyecto sea desarrollado empleándolo destacan su interfaz simple e intuitivo, que permite que pueda ser usado de una manera fácil por el usuario para el desarrollo del código.

Otro punto a favor de este editor de código es la posibilidad de emplear Git, el cual permite realizar un control de versiones, del código escrito, permitiendo que sea más fácil saber que cambios se han realizado y guardarlos. Además, Visual Studio Code dispone de IntelliSense, el cual permite resaltar el texto que puede estar mal escrito, además de autocompletar lo que escribas.

Por último, cabe destacar que Visual Studio Code es un software gratis, de libre distribución, de código abierto y que es soportado tanto en Windows, Linux y MacOS.

2.2 *GANACHE*



Ganache es un simulador de la red BlockChain de Ethereum que permite realizar pruebas de desarrollo sobre el comportamiento de una aplicación BlockChain sin saturar la red Ethereum principal y permitiendo realizar estas pruebas sin ningún tipo de coste.

Entre las ventajas que Ganache proporciona se pueden destacar las siguientes: muestra el historial de ejecución de BlockChain para poder conocer lo que está sucediendo en la red y poder detectar errores; permite configurar actividades de minado para fijar los bloques de manera más sencilla y adaptada a las necesidades del desarrollo; permite explorar los diferentes bloques que hay en la red de BlockChain; brinda las últimas características de Ethereum para el desarrollo de aplicaciones enfocadas en esta tecnología.

En último lugar, cabe destacar que es una tecnología completamente gratuita que, además, está disponible para los principales Sistemas Operativo del mercado.

2.3 TRUFFLE FRAMEWORK



Truffle es un entorno de desarrollo y un framework de pruebas que permite que la programación para Ethereum sea más fácil. Para ello permite la creación del entorno de programación de Ethereum mediante un comando.

Entre las características que brinda Truffle destacan: permite la prueba del correcto funcionamiento de los Smart Contracts; permite la realización de las migraciones de los Smart Contracts a la red de BlockChain; brinda una consola que permite realizar, con facilidad, gestiones con los Smart Contracts desarrollados.

En último lugar, cabe destacar que es una tecnología completamente gratuita que, además, está disponible para los principales Sistemas Operativo del mercado.

2.4 MOCHA



Mocha es un framework de pruebas de JavaScript que se ejecuta tanto en Nodejs como en el explorador de internet.

Entre sus características principales es que permite la realización de tests, tanto síncronos como asíncronos, que es un framework gratuito y con distribución para cualquiera de los Sistemas Operativos más importantes del mercado.

2.5 VIRTUALBOX



Virtual es un software que permite la virtualización de otro Sistema Operativo desde el que trabajar, a través del SO que se tenga instalado en una máquina. Ha sido desarrollado por Oracle y en este proyecto se ha empleado para virtualizar una distribución de Linux para el desarrollo del proyecto.

Entre las características más destacables de VirtualBox cabe resaltar: interfaz sencillo de usar que permite ejecutar diferentes Máquinas Virtuales; facilidad para desplegar cualquiera de los Sistemas Operativos; facilidad para la configuración de las Máquinas Virtuales y flexibilidad para ello; cuenta con cifrado de las máquinas virtuales, lo que permite añadir mayor seguridad a las mismas.

Por último, cabe destacar que es un software gratuito y de código abierto, además de ser compatible tanto para Windows, Linux, MacOS y Solaris.

2.6 UBUNTU



Ubuntu es un Sistema Operativo, distribución de Linux, empleado, en este proyecto para el desarrollo del entorno de programación, mediante su virtualización, a través de VirtualBox.

Entre las mayores ventajas que brinda Ubuntu es que es un Sistema Operativo abierto y sin limitaciones, que permite al desarrollador configurarlo a su gusto. Además, es un software de Código Abierto y Gratuito.

2.7 *NODE 8*



Node es un entorno en tiempo de ejecución multiplataforma que permite el desarrollo de la capa del servidor. En este proyecto ha sido empleado para el despliegue del servidor para el desarrollo del interfaz web que conecta la red Ethereum con la plataforma.

Entre las ventajas por las que ha sido escogido destacan su facilidad de uso, su alto rendimiento y por la facilidad de su aprendizaje. Así mismo es una tecnología gratuita y compatible con cualquiera de los principales Sistemas Operativos del Mercado actual.

2.8 *NPM*



Es el gestor de paquetes por defecto de Node, por lo que en este proyecto ha sido empleado junto a Node para el despliegue de los archivos necesarios para el desarrollo del interfaz web y del servidor.

Destaca por ser un software gratuito y, al igual que Node, ser compatible con los principales Sistemas Operativos del Mercado actual.

2.9 *METAMASK*



Metamask es una extensión para el explorador de internet que funciona de conexión entre la red de BlockChain Ethereum y el servidor de una página web así como gestionar las cuentas que hay dentro de la red.

Por lo tanto, en este proyecto se ha empleado para realizar la conexión entre la plataforma y la red de pruebas desplegada con Ganache.

Entre sus principales características destaca que es gratuito y compatible con Google Chrome y Firefox.

Finalmente es importante destacar que es muy intuitivo de usar, así como compatible con diferentes redes de BlockChain.

2.10 SOLIDITY



Solidity es el lenguaje de programación orientado a objetos empleado para el desarrollo de Smart Contracts dentro de la tecnología BlockChain en Ethereum. Por lo tanto, en este proyecto ha sido empleado para el desarrollo de los contratos que permiten el funcionamiento de la plataforma.

Las ventajas que destacan solidity es la facilidad de su uso y la posibilidad de crear llamadas en bucles a los contratos. Así mismo, la aparición del GAS como mecanismo de evitar bucles infinitos también es una ventaja a tener en cuenta.

2.11 WEB3JS



Web3JS es un conjunto de librerías de JavaScript que permite la conexión entre el interfaz web y la red BlockChain. En este proyecto, por lo tanto, se ha empleado para conectar el interfaz de la plataforma con los contratos desarrollados en Solidity.

Entre las ventajas que brinda Web3JS destacan su facilidad a la hora de usarlo y de implementarlo.

2.12 TRELLO



Trello es una plataforma que permite la planificación de proyectos, de manera colaborativa, de metodología ágil mediante la creación de los diferentes sprints que deben de realizarse durante el transcurso del proyecto.

Trello fue la plataforma elegida para realizar el seguimiento del proyecto, junto a los directores del mismo, y para realizar la planificación para el transcurso del mismo.

Entre sus ventajas destacan que es una plataforma gratuita y con un interfaz muy sencillo de usar.

2.13 ETHEREUM



Ethereum es una plataforma que permite la creación de contratos inteligentes, de manera descentralizada, mediante el uso de la metodología BlockChain. Ethereum ha sido la plataforma seleccionada para albergar los contratos inteligentes en el presente proyecto.

Entre las ventajas que brinda Ethereum destaca que es una plataforma gratuita, de fácil acceso y muy utilizada en el mundo del BlockChain, lo que permite que tenga una comunidad activa y un equipo amplio de desarrollo detrás.

3. ESTADO DE LA CUESTIÓN

3.1 HISTORIA DE LA IDENTIDAD

El Diccionario de la Real Academia Española (RAE) define identidad como “Conjunto de rasgos propios de un individuo o de una colectividad que los caracterizan frente a los demás”. Es decir, es la suma de aquellos atributos o cualidades que le otorgan a una persona, o colectivo, ser quien es. Por lo tanto, es importante la posesión de algo que le permita a la persona acreditar que es quien dice ser y puede hacer todo lo que se espera de sí.

Históricamente en las sociedades tradicionales y hasta mediados del XV, el tipo de reconocimiento de identidad que predominaba era la identificación cara a cara. En esta época comienzan a aparecer y a desarrollarse signos de identidad sustitutivos de la persona, como el sello, los escudos de armas, las insignias o incluso marcas permanentes, como las que recibían los delincuentes, como castigo por sus delitos.

A partir del siglo XV el reconocimiento mediante lo escrito desempeña un papel más destacado, así las iglesias realizan cuidadosos registros de fieles mediante libros parroquiales que también serán utilizados por el poder civil, con la justicia, la recaudación de impuestos y la milicia como ámbitos prioritarios de utilización.

Durante el siglo XIX, las revoluciones liberales y sobre todo los procesos de construcción de los estados provocan la transferencia de los registros de la iglesia al estado y la proliferación de los documentos de identidad emitidos por diversas instituciones. Conforme transcurre el siglo XX y como consecuencia de las crisis políticas, las guerras, el terrorismo y el control de la inmigración se produce un avance en el registro de información. En la Primera Guerra Mundial, donde los países requerían un control duro de acceso y de vigilancia de quién salía del mismo, y fue en esta época donde se introdujeron los primeros requisitos de pasaporte, que se normalizó definitivamente en 1980. En España, se obligó a

la creación del DNI a finales de la Segunda Guerra Mundial, con el objetivo de controlar el censo y los movimientos migratorios de los españoles.

En la sociedad moderna existen una serie de documentos que permiten la acreditación de la identidad tanto de manera internacional (Pasaporte) como diversos documentos de carácter nacional, Documento Nacional de Identidad (DNI), Número de Identificación Fiscal (NIF), licencia de conducir, tarjeta de la Seguridad Social, etc. Con la creación de todos estos documentos se produce el control de la población, permitiendo a los países de vigilar los accesos de sus fronteras y posibilitando que la población pueda identificarse como portador de una serie de características y atributos que le definen. Además, en la actualidad el acceso a todos los servicios está vinculado a la previa identificación del usuario, lo que convierte este proceso en algo permanente y a la vez imperceptible.

3.1.1 LA IDENTIDAD DIGITAL

La aparición de Internet en 1969 y sobre todo de la *World Wide Web* en 1990 ha complicado de nuevo la forma de identificarse, pues el anonimato de la red complica la realización de transacciones veraces a través de ella. Nace así la necesidad de ser capaz de verificar que una persona que está haciendo una serie de acciones, es quien dice ser y tiene las habilidades, atributos o cualidades que la caracterizan.

En la actualidad, muchos países tratan de producir sus identificaciones nacionales, DNI en el caso de España, de manera que puedan ser utilizados como certificados digitales para firmar en la red, sin embargo, estos podrían ser copiados o robados por alguien con el conocimiento suficiente.

La aparición de *BlockChain (BC)* ha surgido como respuesta a la necesidad de una tecnología que fuera capaz de aportar la confidencialidad y seguridad necesarias para la identificación personal. Es una tecnología que almacena información de manera distribuida y permanente, garantizando su confidencialidad e inmutabilidad y permite dar solución a esta problemática de identidad en la red o identidad 2.0.

3.2 *BLOCKCHAIN (BC)*

BlockChain (BC), nació en 2009 siendo la primera implementación conocida la moneda digital *Bitcoin*. BC según su propia traducción literal al castellano, significa “Cadena de bloques”. Es un conjunto de bloques relacionados entre sí, en cada uno de los cuales se encuentran un conjunto de transacciones, realizadas, que han sido previamente verificadas.

BC es una tecnología que conforma una base de datos distribuida y descentralizada para contener todas las transacciones económicas que se producen alrededor del mundo. La distribución permite una mayor robustez de la tecnología frente a cualquier tipo de adversidad, ya que, mientras se mantenga uno de los nodos de la red, el BC se podrá recomponer de cualquier adversidad.

La tecnología BC se suele relacionar con el ámbito económico, al ser la tecnología que emplean monedas como el *Bitcoin* o el *Ethereum*, pero permite un amplio abanico de posibilidades extendiéndose sus aplicaciones a campos como finanzas, registros médicos, derechos de autor, censos de inmigrantes y refugiados, educación etc.

3.2.1 ¿QUÉ ES UN BLOQUE Y CÓMO SE FORMA?

Un bloque es una estructura de datos que contiene un conjunto de transacciones realizadas entre diferentes cuentas. Estos bloques suelen tener un tamaño de 1MB de media, pudiendo llegar a los 8MB, si fuera necesario.

Los bloques están formados por una cabecera, con información referente al bloque, y un campo de datos, que contiene toda la información que almacena.

En la cabecera, se encontrará el identificador, o *hash*, del propio bloque y el identificador del bloque previo, al que está unido. Mientras tanto, en el campo de datos se encuentran

todas las transacciones almacenadas. A continuación, en la imagen 1 se muestra un esquema que representa la organización de los bloques en BC.

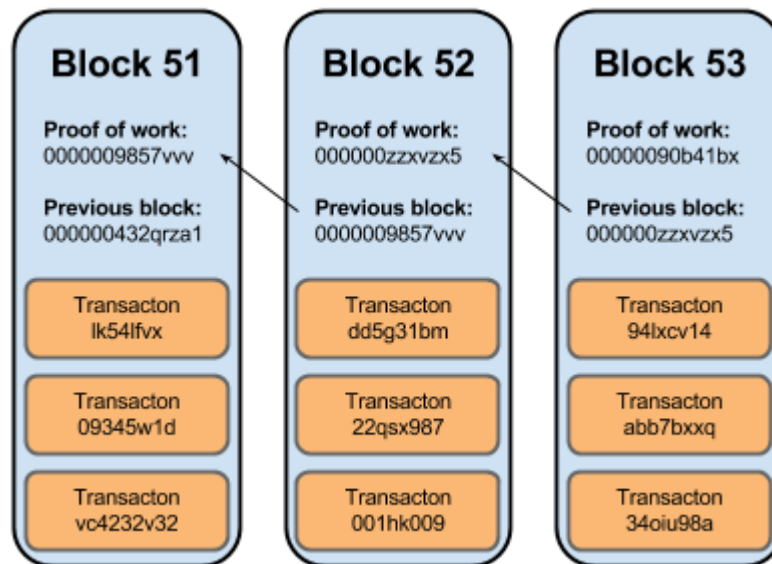


Figura 1 - Representación esquemática de la organización de Bloques en BC

Como se puede ver en la imagen, existe la parte de la cabecera de bloque, con el *Proof of work*, o identificador, un puntero al *Proof of work* del bloque previo, y las transacciones guardadas en el bloque. Por lo tanto, BC la conforman un conjunto de bloques que crecen cada vez que se realiza una transacción, y que permiten recuperar cualquier información al estar todos los bloques relacionados.

El bloque inicial que dio comienzo a la cadena y, del cual, provienen y dependen el resto de bloques, se denomina Génesis (como el libro de la creación de la Biblia) ya que dio lugar a la aparición del BC. Este primer bloque fue creado en 2009, con la aparición del *Bitcoin* por Satoshi Nakamoto, creador del mismo. Este primer bloque contenía la siguiente información: “*The Times 03/ene/2009 Canciller al borde del segundo rescate para los bancos.*” El cual era un titular del periódico estadounidense *The Times*, del 3 de enero del año 2009.

Una vez creado el primer bloque o Génesis, el resto de los bloques que se han ido creando, o Bloques Principales se van uniendo al Génesis, dando forma al cuerpo del BC. Así mismo, también pueden existir bloques que son descartados, llamados Bloques Huérfanos, debido a

la disparidad en la orden de llegada de los bloques a los nodos. En la figura 2 se hace la representación de una posible estructura en bloques de BC.

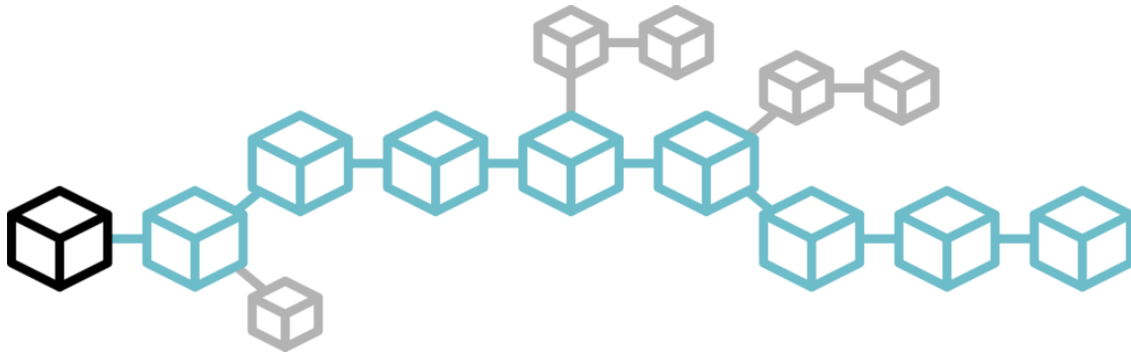


Figura 2 - Representación esquemática de la Cadena de bloques de BC. El primer Bloque Génesis se representa en negro, los Bloques Principales en azul y los Bloques Huérfanos en gris.

3.2.2 FUNCIONAMIENTO DEL BLOCKCHAIN

Una vez entendida la estructura del BlockChain, será necesario comprender como se forman los bloques que dan nombre y vida a esta tecnología. Antes debemos indicar que las carteras de dinero de los usuarios son muy parecidas a las físicas que cualquier persona puede tener en su bolsillo, con la simple diferencia de que el dinero empleado es virtual, no se puede tocar, y el nombre que recibe es *Wallet*, que significa “cartera” en inglés.

Como esta cartera es un ente virtual, es necesario concederle un identificador, que nos permita hacer referencia a la hora de realizar cualquier tipo de transacción con ella, tanto enviarle como pedirle dinero.

Las transacciones en Bitcoin se realizan entre usuarios anónimos (con identidad desconocida) mediante la criptografía asimétrica, a través de dos claves: una de las claves es pública y la otra es privada.

3.2.2.1 Criptografía asimétrica

Antes de entrar en detalle sobre el funcionamiento de los identificadores, de las carteras, es necesario explicar que es la criptografía asimétrica y como funciona.

La criptografía asimétrica es el método de encriptación de datos compuesto por dos claves: una pública la cual puede tener cualquier persona y una privada que es intransferible para que la comunicación no se vea comprometida, es decir, solo puede tenerla uno de los extremos de la comunicación.

Para entender de manera más fácil su funcionamiento, se va a explicar mediante el ejemplo de la comunicación entre una página Web y un visitante de la misma. La página Web tiene una clave privada, que almacena, y una clave pública ligada a esta. La clave pública se difunde a través de Internet, de manera que cualquier cliente pueda enviarle mensajes de manera cifrada y que solamente la página Web pueda descifrarlos.

Así, por ejemplo, cuando se quiere negociar la clave simétrica para la comunicación en HTTPS, el cliente emplea la clave privada para encriptar la información necesaria para dicho trámite. Una vez cifrado el mensaje, la única clave capaz de descifrarlo es la clave privada que posee la página Web, haciendo el mensaje confidencial e imposibilitando su lectura para cualquier tercero.

3.2.2.2 Criptografía asimétrica en BlockChain

De la misma manera que con la página Web, las carteras de los usuarios disponen de una clave pública, que difunden, y otra privada que se guardan para descifrar los mensajes recibidos. Así, cuando una entidad o persona quiere recibir dinero virtual de otra, tiene que hacer una petición del dinero empleando la clave pública de la *wallet* del emisor del dinero.

Si la petición recibida es conforme al dueño del *wallet*, simplemente lo firma con su clave privada, dando validez y aprobando la transacción. Así mismo, una persona o entidad puede realizar un envío de dinero sin necesidad de que haya una petición, simplemente dando la orden de realizar una transacción a un *wallet* con su propia firma privada.

En este punto, es necesario explicar qué son estas transacciones y cómo se realizan y funcionan.

3.2.2.3 Transacciones

Una transacción es cualquier envío de dinero que se realice entre dos cuentas de BC. Estas transacciones van siendo almacenadas, una tras otra, en un *pool* temporal de transacciones, donde serán almacenadas hasta que un “minero” las retire, verifique, y pasen a formar un nuevo bloque de la cadena.

3.2.2.3.1 Mineros

Un “minero” es una persona, o grupo de personas, que se dedican a la confirmación de la veracidad de una transacción a cambio de una recompensa, por su trabajo, en forma de monedas virtuales.

En este método, se denomina “*Proof of Work*” para evitar cualquier tipo de comportamiento sospechoso en el sistema. Esto significa que el “minero” debe de realizar un gran esfuerzo, un alto gasto de consumo de CPU y energía para poder llevar a cabo dicha tarea, y, a cambio, recibe una recompensa económica.

La comprobación de la transacción se realiza de la siguiente manera:

1. En primer lugar, es necesario comprobar que el emisor del dinero recibió en el pasado una cantidad igual, o superior, a la cantidad de dinero que trata de enviar, esto es, constatar que el balance de entrada, de dinero, en la cuenta del emisor ha sido igual, o superior, a la cantidad de dinero a enviar.
2. En segundo lugar, es necesario comprobar la cantidad de dinero que tiene, actualmente, el emisor. Para ello se le resta, al balance de entrada de la cuenta, el balance de salida, esto es, quitarle a los ingresos, los gastos.

Como resultado se obtiene la cantidad de moneda virtual de la que dispone el emisor de moneda y, si esta es igual o superior a la que se pretende emitir, la transacción es aprobada.

Pero aquí no acaba todo, ya que la tarea no es nada sencilla de realizar, al encontrarse los “mineros” en competición con otros “mineros” que quieren realizar la misma tarea, para la obtención de la recompensa.

3.2.2.4 Creación de los bloques de datos

Una vez que se ha realizado la comprobación de la transacción, el “minero”, o grupo de “mineros”, pasará a introducirla en su propio bloque. Una vez creado el bloque, este se sella, con la información más importante del bloque, su firma digital.

La firma digital está conformada por tres *hash*, el primero está formado por la firma digital del bloque anterior; el segundo está formado por una lista de transacciones, que ya hayan sido confirmadas recientemente; y la tercera es un valor denominado *Nonce*.

3.2.2.4.1 Nonce

El *Nonce* se conoce por sus siglas en inglés “*Number that can only used once*”, es decir, un número que tiene un único uso. Este número es insertado en la huella digital del bloque de manera aleatoria, de esta forma, los “mineros” solo pueden conocer 2 de los *hashes* descritos con anterioridad.

Finalmente, este bloque es insertado en la cadena de bloques, aunque este bloque puede ser rechazado, la manera de que este bloque se mantenga en la cadena de bloques es que el resto de los “mineros” comprueben la veracidad del bloque mediante una nueva prueba de trabajo, tratando de hallar el *Nonce*, mediante prueba y error.

Una vez que este bloque ha sido comprobado por una amplia mayoría de “mineros”, este se implantará para siempre en la cadena de bloques.

Todo este largo camino para la comprobación de las transacciones permite que todas ellas sean verificadas siempre y evitando así la posibilidad de cualquier tipo de ataque de *hackers* o de cualquier intento de fraude financiero, convirtiendo así a las monedas virtuales en la alternativa más segura del dinero físico.

El mundo financiero fue el primero en aprovecharse de esta tecnología BC mediante el *Bitcoin*. Sin embargo, en los últimos años son múltiples las aplicaciones de uso que despiertan en otros campos, surgidas de las posibilidades para certificar la autenticidad e identidad en todo tipo de objetos y actos.

En general, las posibilidades del uso de esta tecnología crecen exponencialmente y puede verse reflejada en todos los ámbitos sociales, desde el Internet de las cosas a la educación, pasando por servicios legales, entidades gubernamentales y en la industria.

3.3 LA IDENTIDAD EN LA EMPRESA

Actualmente las empresas que quieran ser punteras tienen que adaptarse a la digitalización sustentada en las nuevas tecnologías para acoplar sus productos o servicios a las necesidades reales del cliente.

BlockChain tiene el potencial, para que una empresa digitalizada, recopile la información, la ordene, la interprete y la comparta con la confianza y la privacidad necesarias que posibilite una mayor fiabilidad, seguridad y calidad de datos. Esta tecnología puede aplicarse dentro de una empresa en varios campos que puedan agilizar ámbitos tan distintos como en la producción o en los departamentos de recursos humanos (RRHH).

El Servicio de RRHH es básico en toda organización, es el encargado de seleccionar, contratar y formar el personal de la organización. Esta tarea es fundamental ya que un proceso de selección inadecuado afectará a la empresa en capacitación, esfuerzo y tiempo. Cuando una empresa comunica que hay una vacante en un departamento, el servicio de RRHH, lo primero que hace es publicitar dicha oferta en las plataformas destinadas a la contratación de personal. Estas ofertas son vistas por las personas que estén interesadas en el puesto de trabajo y aplican para tratar de conseguirlo.

La empresa necesita un tipo de trabajadores que tengan una preparación previa, que les posibilite comenzar a realizar el trabajo que se oferta, como, por ejemplo: un título de haber terminado una carrera, un título de un master, títulos de idiomas o títulos de habilidades personales. Por lo tanto, la persona que aplica al puesto debe de adjuntar la información que acredite que cumple todos los requisitos que le permiten aplicar a ese puesto de trabajo.

Cada vez que alguien quiera aplicar a un puesto de trabajo, deberá enviar cada uno de los títulos necesarios a través de una entidad, y la empresa por su parte deberá comprobar que ese título es veraz y ha sido expedido por la institución correspondiente.

Con el objetivo de facilitar la gestión por parte de empresas y personas en los procesos de aplicación a los puestos de trabajo y de verificación de los correspondientes curriculums de los aspirantes, en el presente trabajo, se va a crear una plataforma que permita a las personas gestionar su identidad digital con todas las acreditaciones propias que sean testimonio de veracidad e idoneidad en este tipo de gestiones.

Esto facilitará todo el proceso de aplicación a una oferta de trabajo en una empresa. Para la realización de la plataforma, se empleará la tecnología BC, explicada en puntos anteriores aplicada a la identidad digital.

3.3.1 EL BLOCKCHAIN Y LA IDENTIDAD DIGITAL

Como se ha explicado el BC es una tecnología muy potente para almacenar información de manera permanente, segura e inmutable en un conjunto de bloques que conforman la red. Todas estas características producen que el BC, que inicialmente nació como una tecnología centrada únicamente en el mundo financiero, evolucione hacia otro campo de posibilidades.

3.3.2 LA IDENTIDAD DIGITAL Y LA UNIVERSIDAD

Uno de los testimonios más importantes que puede tener una persona, corresponden a los títulos académicos que acreditan que dicha persona ha superado la universidad y puede acceder al mundo laboral a cubrir los puestos que requieran dichos estudios.

Como se ha comentado anteriormente, en el presente trabajo se va a desarrollar una plataforma que permita la creación de una identidad digital, de una persona, la creación de la identidad de una institución que dispensa títulos académicos, una universidad, y la identidad digital de la empresa que publica una oferta de trabajo.

Esto permitirá la interacción entre la empresa, el exalumno y la institución académica de la siguiente manera:

- 1.- La universidad emite un testimonio de que un alumno ha superado unos estudios y tiene un título, Graduado GITT.
- 2.- El alumno acepta este testimonio y lo incorpora a su identidad
- 3.- La empresa en cuestión, confía en la Universidad
- 4.- El alumno presenta el testimonio a la empresa para acceder a un puesto de trabajo

Mediante esto se comprobará si el estudiante tiene el susodicho certificado académico y si está verificado por la institución académica, de así serlo se le permitirá aplicar a la oferta y, de lo contrario, no tendrá la posibilidad de aplicar.

Esto automatizará los procesos de envío de información académica, por parte del exalumno y la verificación de la autenticidad de los títulos, por parte de la empresa.

4. DEFINICIÓN DEL TRABAJO

En el actual capítulo se describirán características del proyecto tales como la motivación, los objetivos, la metodología y la planificación y estimación económica. El entendimiento de los siguientes puntos mostrará el porqué de la realización del proyecto.

4.1 MOTIVACIÓN

Las nuevas tecnologías son los motores de la economía, en la actualidad, llevando a todas las empresas a realizar profundas transformaciones digitales, si no quieren quedarse estancadas en el pasado.

Dentro de las nuevas tecnologías se pueden destacar varias que están sonando muy fuerte, en la actualidad: BigData, IOT, Blockchain, etc. Aunque todas ellas están en su apogeo y máxima difusión, se puede remarcar Blockchain como la tecnología moderna que más seguidores ha tenido, en los últimos años, con la aparición de las criptomonedas y, los ya famosos, Bitcoin y Ethers.

Aunque el Blockchain sea una tecnología que haya creado un gran interés en el mercado, sigue siendo muy reciente y aún se desconoce mucho sobre el potencial que puede ofrecer y su manejo.

Aun cuando el mayor cuando el mayor potencial que se ha mostrado en los medios de comunicación, es la posibilidad de asegurar las transacciones financieras, haciendo posible el control del dinero, controlando así, en mayor medida, el lavado de dinero, no se han explorado otros potenciales usos que permiten, justamente, el control financiero, el cual es: la identificación única e inmutabilidad.

Uno de los mayores problemas que han surgido, a la hora de navegar a través de la red, ha sido la identificación de la persona que está navegando en internet, para lo cual se han tratado

de emplear diferentes métodos: DNIE, el uso de las sesiones (en el servidor), el uso de cookies, etc.

Sin embargo, estas soluciones propuestas solo han servido de parche al problema, ya que cualquier persona podía “robar” la identidad de otro y suplantar su identidad, mediante el robo del SESSION ID o del DNIE, por lo que, a raíz del parche, surgieron problemas como la veracidad y seguridad de las comunicaciones.

Tratando de solucionar, este último problema, tuvo que darse lugar a una guerra de cifrado y rotura del cifrado, por parte de crackers y los encargados de la seguridad informática. No obstante, esto solo puede utilizarse como solución temporal, ya que solo es cuestión de tiempo que los crackers encuentren una vulnerabilidad a un cifrado, ya que el software no es perfecto, al estar creado por humanos.

El primer problema, es fácilmente solucionable gracias a blockchain, ya que cada persona es única, debido a su identificación personal, o address, que permite su identificación, y de carácter inmutable, esa identificación pertenece a esta persona para siempre.

El segundo problema, también puede solventarse gracias a las características intrínsecas a Blockchain. Esto se refiere a su seguridad e inmutabilidad, ya que, si una persona tratara de robar la información de otra, este intento quedaría grabado en la red, para siempre y sería rápidamente detectable el problema con la consecuente restauración de la identidad de la persona original.

Por lo tanto, el Blockchain se presenta como una solución más que viable para la revolución de la red internet, tal como la conocemos, y cambiar las nuevas tecnologías hacia una mejora en la seguridad y confidencialidad de las comunicaciones.

Todo ello, provoca que la investigación de este campo en expansión, y de reciente creación, se presente como un área con infinitas posibilidades y altas posibilidades de incorporación laboral, en el ámbito internacional.

4.2 OBJETIVOS

En el actual proyecto va a tratar de darse solución a los problemas de identificación de las personas en la red siendo como objetivos principales los siguientes:

1. Creación y gestión de identidades, tanto de usuarios como de instituciones académicas y de empresas. Creación, gestión y validación de credenciales de los usuarios (“Claims”).
2. Suministrar la confianza y seguridad, necesarias para el funcionamiento de la plataforma, gracias a los beneficios que confiere la tecnología Blockchain.
3. Creación de la plataforma que permita cumplir los dos primeros objetivos y su normal desempeño.

4.3 METODOLOGÍA

Un problema muy común, a la hora de realizar un proyecto, es la planificación del mismo, lo cual suele provocar situaciones de “delay” en los plazos de entrega del mismo. Por lo tanto, es de vital importancia realizar una buena planificación de las tareas que se quieren realizar, y cuando deben estar terminadas.

Para la consecución de la planificación a realizar, se emplean metodologías de desarrollo de proyectos, las cuales son plantillas que permiten organizar las tareas a realizar, para organizar las fechas de comienzo y finalización, de cada una de ellas. Esto ayuda a conocer si el proyecto se está realizando dentro del tiempo previsto y tener una mayor percepción de las tareas que se consideran “críticas” para la finalización del proyecto.

Para el desarrollo del presente trabajo se ha elegido la metodología la metodología Ágil. Esto se debe, por un lado, a que es la metodología que se recomienda, y más utilizada, en el mundo del desarrollo de proyectos software.

Por otro lado, la metodología Ágil a que permite, como su propio nombre indica, agilidad en la hora del desarrollo de aplicaciones, al presentarse como una metodología dividida en sprints, u objetivos poco ambiciosos, los cuales se tratan de ir alcanzando. Una vez que han sido cumplidos, estos son mejorados mediante la realización de iteraciones, alrededor de los objetivos alcanzados.

Esto permite una mayor velocidad en el desarrollo de la programación, así como la realización de programas más robustos y seguros, al ser revisados continuamente y estar en constante evolución.

Una vez elegida la metodología que se va a emplear, es necesario plasmar la planificación del proyecto a través de un diagrama de GANTT. Este, permite la visualización, en forma de diagrama, de las diferentes tareas a realizar y sus fechas de inicio y finalización, así como de otras funcionalidades importantes como puede ser la visualización del camino crítico del proyecto.

A continuación, se muestra una imagen del diagrama de GANTT del proyecto.

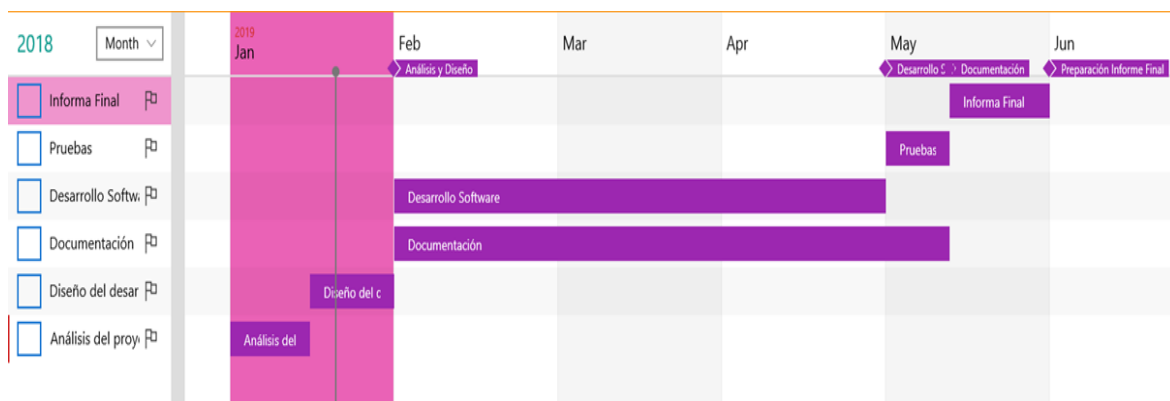


Figura 3 - Diagrama de GANTT del proyecto

Como se puede observar, el proyecto se ha desglosado en una serie de actividades, a lo largo de 6 meses de duración, para el desarrollo de cada uno de las tareas a realizar. Así mismo, se ha marcado, en paralelo, la realización del presente documento, para finalizar todo al mismo tiempo.

Además del diagrama de GANTT, se ha realizado una planificación, más detallada, a través de la plataforma TRELLO, que permite desglosar, mediante tableros, las tareas que se pretenden realizar y realizar comentarios sobre las mismas.

La siguiente ilustración muestra un ejemplo del tablón de TRELLO creado para el desarrollo del proyecto.



Figura 4 - Trello de planificación del Proyecto

Como se puede observar, el tablón está dividido en las diferentes semanas que hay, poniendo cada una de las actividades que se pretenden llevar a cabo cada una de ellas. Estas actividades se dividen en dos:

1. **Objetivos en programación:** esto es las tareas en el ámbito de desarrollo del software que se pretende terminar en dicha semana.
2. **Objetivos en documentación:** los cuales engloban las partes del desarrollo del presente documento, que se pretende tener listo dicha semana.

Así mismo, en cada una de las tareas a realizar, se encuentra una pequeña descripción de en qué consiste la misma, para realizar un seguimiento más exhaustivo de si se ha conseguido implantar toda la funcionalidad descrita.

4.4 PLANIFICACIÓN Y ESTIMACIÓN ECONÓMICA

Otro de los grandes retos a los que enfrentarse cuando se trata de realizar la planificación de un proyecto, es la estimación de los gastos que van a ser necesarios para la finalización del proyecto, con éxito.

Esto es importante ya que deben de ser lo más realistas posibles tratando de que no sean muy bajos, para no tener que pedir ampliaciones de presupuesto, las cuales podrían echar a perder el proyecto, ni muy altas para que el proyecto no sea demasiado caro y se pueda llegar a realizar.

Para el presente proyecto, se han empleado una serie de herramientas de software de libre acceso y gratuitas, lo cual han permitido abaratar el coste final del proyecto. Aun así, ha sido necesaria la utilización de diferentes elementos hardware que han producido que para la consecución del proyecto sea necesario realizar un desembolso económico.

Los elementos empleados para la consecución del proyecto se muestran desglosados en la siguiente tabla.

Concepto	Justificación	Coste
Ordenador	Se ha usado un ordenador portátil con un procesador i7, 8 GB de RAM y 1 TB de disco duro	1.138,25 €
Ratón	Se ha empleado un ratón para la comodidad del desarrollador	6 €
Trabajadores	Se ha utilizado a un trabajador para la	34.525,22 €

Concepto	Justificación	Coste
	realización del proyecto a 56,89 € la hora	
Herramientas	Las herramientas empleadas para el desarrollo del proyecto se encuentran descritas en el apartado de descripción de las herramientas	0 €
Total		35.669,47 €

Tabla 1 - Estimación económica del Proyecto

5. SISTEMA/MODELO DESARROLLADO

En el presente capítulo se desarrollará la explicación sobre el trabajo desarrollado a lo largo del proyecto. Esta tratará de hacerse lo más detallada y mejor explicada posible.

5.1 PORTAL DE EMPLEO

En el presente proyecto se ha desarrollado una plataforma que permita a usuarios, que puedan ser cualquier ciudadano, demostrar una serie de estudios académicos a través de la aprobación de una institución académica, con el objetivo de poder aplicar a ofertas de trabajo, que requieran de dichas cualificaciones.

Por lo tanto, para el desarrollo de la plataforma ha sido necesaria la identificación de los 3 usuarios que se han diferenciado, los cuales son: “user”, “Issuer” y “Checker”.

Cada uno de ellos desempeñará un rol diferente dentro de la plataforma, el cual será explicado a continuación.

1. User: el usuario será cualquier persona, que haya realizado una serie de estudios académicos, y que quiera demostrar que los ha estudiado, mediante un Issuer y, finalmente, pueda aplicar a una oferta laboral, ofertada por un Checker.
2. Issuer: este usuario corresponde a cualquier institución académica sobre la cual se respaldará el título académico que dice tener un user. Su responsabilidad radica a la hora de validar, o invalidar, cualquier título que diga tener un user, si esta institución no tiene constancia de que el usuario haya estudiado dichos estudios. Por lo tanto, será el garante de confianza sobre los estudios de los user.

3. Checker: este usuario será la empresa que quiera ofertar una vacante en un trabajo a la cual, solo podrán optar los user que cumplan los requisitos académicos pedidos por el checker.

Así, en el proyecto se entrelazan 3 diferentes usuarios los cuales dependen, unos de los otros, para alcanzar sus objetivos. Por un lado, está el user, el cual necesita del Issuer para probar su formación y optar así a la plaza vacante. Por otro lado, se encuentra el Issuer que tiene el poder de revocar y aprobar los estudios que un user dice tener, además de presentarse como institución sobre la cual confía el checker para garantizar los conocimientos de los user. En último lugar, está el checker, el cual presentará las ofertas a las cuales puedan optar los user y que necesitarán del respaldo de los Issuers para poder optar.

Para realizar una explicación más detallada de las acciones que puede realizar cada uno de los usuarios se mostrará un diagrama de flujo, para cada uno de ellos, con las diferentes posibilidades que tienen.

5.2 DIAGRAMA DE FLUJO INTERFAZ DE USUARIO

Para comenzar, se mostrará las diferentes opciones que tiene un usuario para identificarse como uno de los tres tipos diferentes.

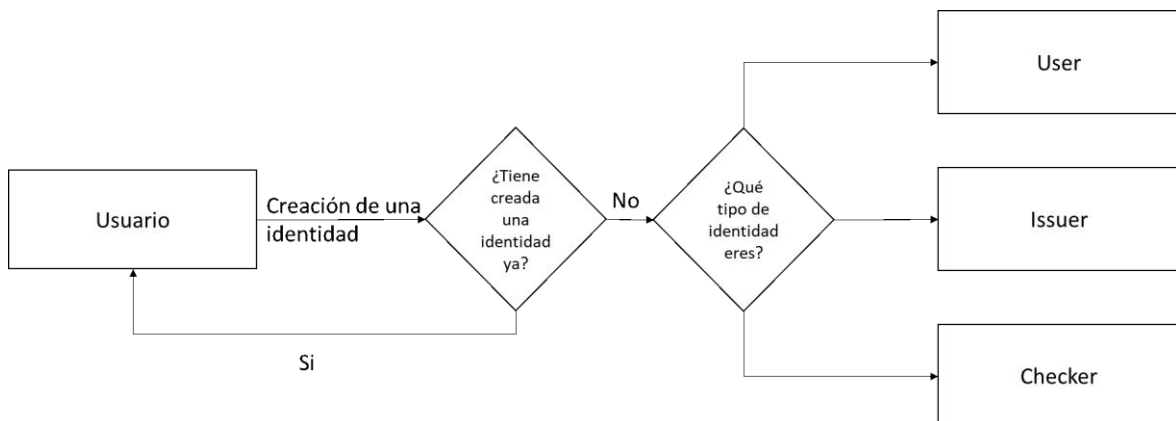


Figura 5 - Diagrama de flujo de interfaz de usuario

Como se muestra, en la ilustración anterior, cuando un usuario entra en la aplicación, lo primero que podrá hacer será crear una identidad. Si este usuario ya tiene creada una identidad, no podrá continuar con la creación. Si el usuario no tiene ninguna identidad creada, se podrá registrar como una de las tres identidades posibles esto es: un user, un Issuer o un Checker.

Una vez que la identidad haya sido creada, este será redireccionado a la página principal, a través de la cual podrá comenzar a realizar acciones propias de su identidad.

5.3 DIAGRAMA DE FLUJO DE UN “USER”

Si el usuario se ha identificado como un User lo que podrá realizar a través de la plataforma serán las siguientes acciones.

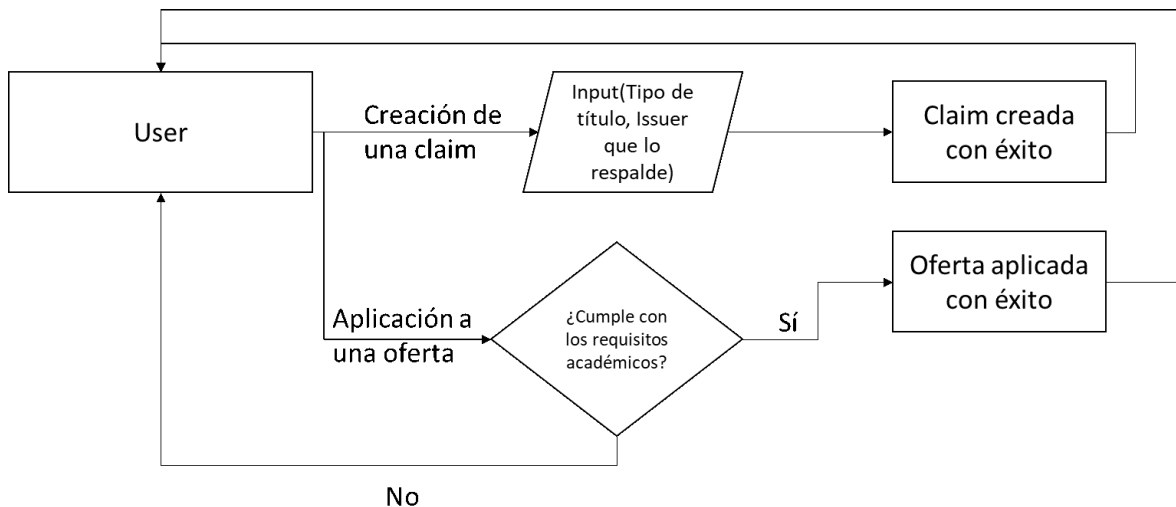


Figura 6 - Diagrama de flujo de un "User"

Como se puede observar, en la ilustración anterior, un user puede realizar dos acciones distintas.

1. Por un lado, se encuentra la creación de claims, las cuales son como “títulos universitarios” acreditados por uno de los issuers que se encuentran registrados en la

plataforma. Si esta acción se realiza de manera satisfactoria permitirá al user aplicar a una oferta de un Checker.

2. Por otro lado, se encuentra la aplicación a una oferta de trabajo, que haya sido creada por un checker. Si el user dispone de los requisitos académicos, si tiene validades las claims necesarias, la aplicación a la oferta se realizará con éxito y se optará a la vacante creada por el Checker. Si el user no dispone de las claims necesarias, no podrá aplicar a la vacante.

5.4 DIAGRAMA DE FLUJO DE UN "ISSUER"

Una vez explicado y entendido el papel que realiza el User, se explicará el rol y las acciones que puede realizar un Issuer, con el siguiente diagrama de flujo.

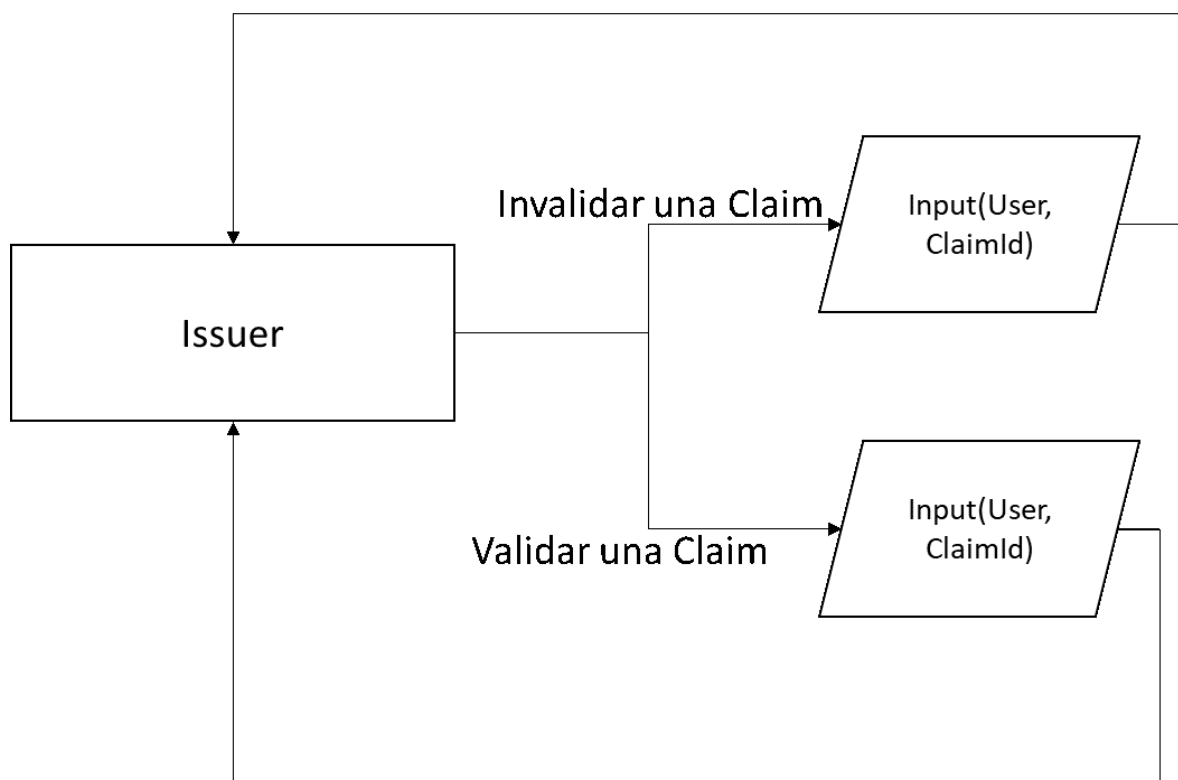


Figura 7 - Diagrama de flujo de un "Issuer"

Como se observa, en la ilustración anterior, la tarea de un Issuer no es más que la de ser responsable de las acreditaciones académicas que expide de la manera que, pueda validar o invalidar un título académico que un User diga poseer. Así, las acciones que puede realizar son las siguientes.

1. Invalidar un claim: a través de esta acción, una claim que haya sido creada por un User quedará invalidada e inservible, con lo que no podrá ser empleada para optar a una vacante creada por un Checker.
2. Validar una claim: a través de esta acción se validará una claim que hubiera sido invalidada, previamente, por el Issuer, permitiendo así que pueda volver a ser utilizada por un User para aplicar a ofertas de trabajo creadas por un Checker.

Queda así patente la intención mediadora del Issuer, siendo a la vez habilitante y garante del cumplimiento de ambas partes para la entrada al mundo laboral de un User.

5.5 DIAGRAMA DE FLUJO DE UN “CHECKER”

A continuación, se explicará mediante un diagrama de flujo las acciones que puede realizar un Checker, para tratar de entender su funcionamiento.

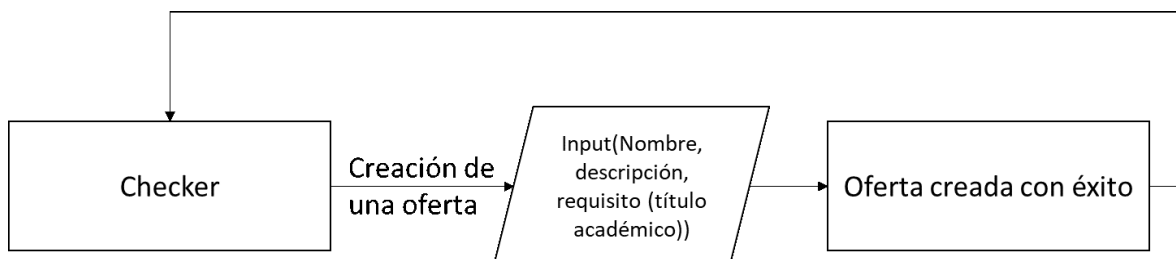


Figura 8 - Diagrama de flujo de un "Checker"

Como se observa, en la ilustración anterior, la única función de un Checker es la de crear una oferta de trabajo, insertando un título y una pequeña descripción, para la misma, y el título académico necesario para la aplicación al puesto de trabajo.

Para que se pueda tener una visión total, del funcionamiento del sistema, a continuación, se muestra una imagen del flujo completo de la página:

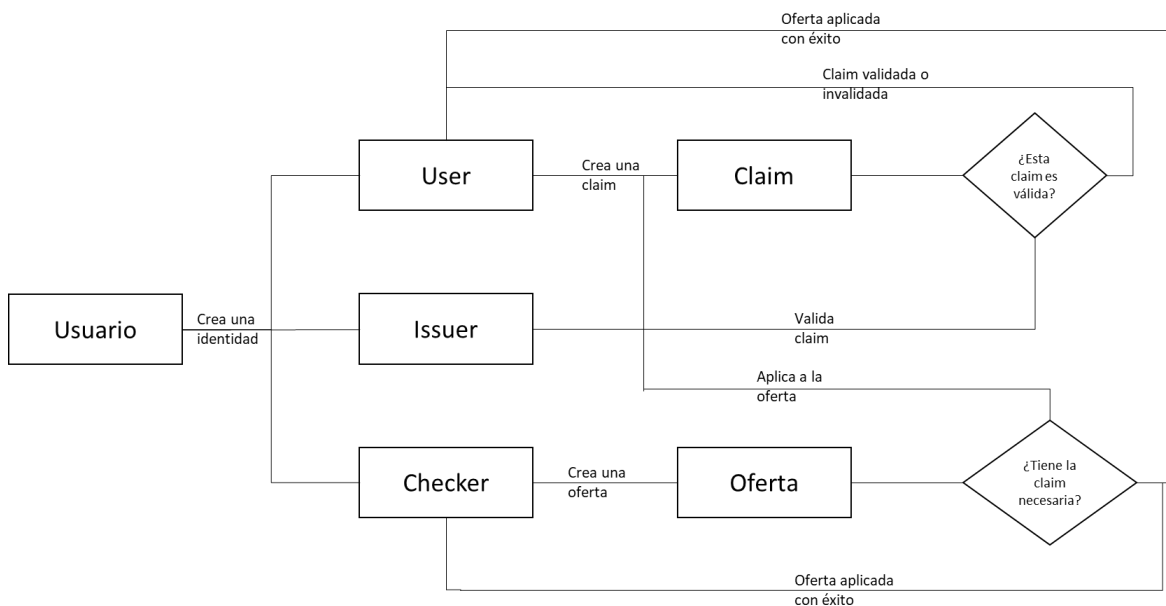


Figura 9 - Diagrama de flujo de la Plataforma

Pudiendo entenderse así, la unión de los tres tipos de identidades, en el portal, y su interacción entre ellos.

Con estos diagramas de flujo, además de las explicaciones posteriores, queda claro el funcionamiento de la plataforma para cada una de las identidades que existen dentro del proyecto. Sin embargo, es necesario explicar el funcionamiento de la plataforma desde el punto de vista técnico de funcionamiento de Blockchain.

Para el desarrollo de la programación se empleó solidity para el desarrollo de los Smart contracts que componen el núcleo de funcionamiento de la plataforma.

5.6 SMART CONTRACTS DE LA APLICACIÓN

Los Smart Contrats que se han desarrollado son 6: Identity.sol, ClaimOwner, ClaimIssuer, ClaimChecker, ClaimManager, OfferManager. Cada uno de ellos tiene un cometido específico dentro de la plataforma.

Este cometido, tratará de explicarse en la siguiente ilustración, la cual muestra un diagrama de clases que posibilitará la identificación de las relaciones existentes entre los diferentes Smart contracts de la plataforma, así como de las diferentes funciones que los conforman.

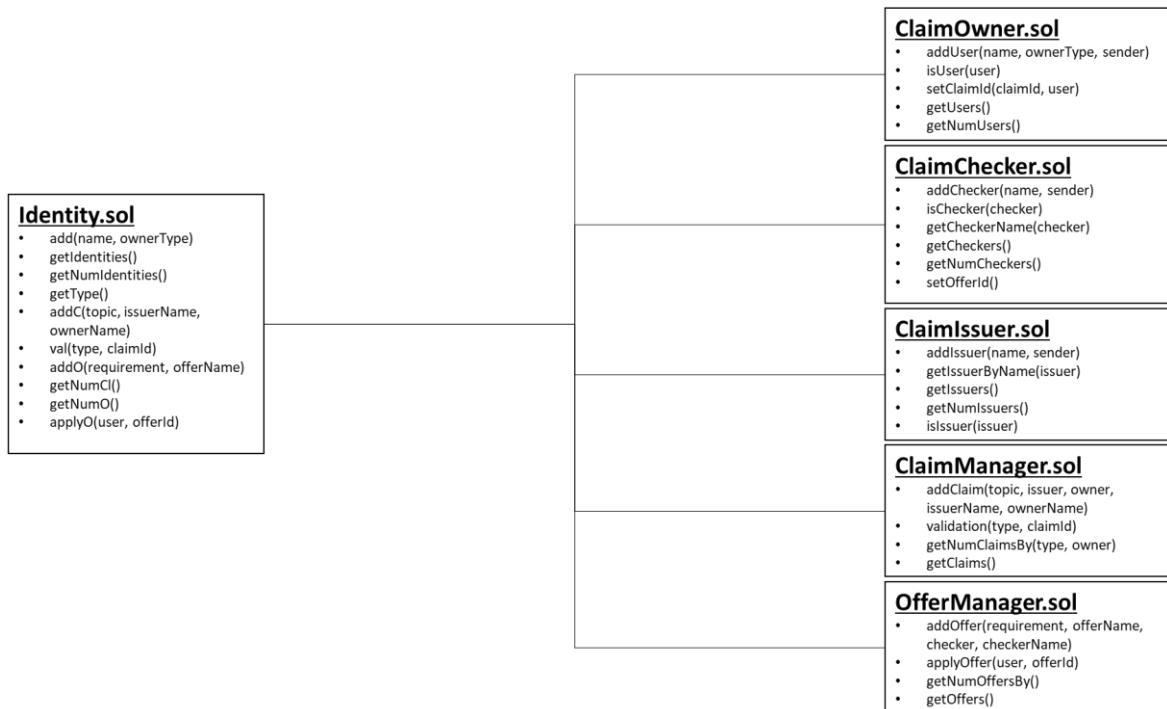


Figura 10 - Diagrama de contratos de la Plataforma

Como se puede observar, la aplicación está formada por cinco contratos, los cuales están organizados dependiendo de un sexto denominado “Identity.sol”.

Este contrato, funciona como manager del resto de contratos y como “entrada” para conectar el interfaz de usuario con los otros contratos. Así, se puede acceder a todas las

funcionalidades de la aplicación desde un mismo punto de entrada, facilitando la seguridad y cualquier comprobación que se quiera hacer a las peticiones.

5.6.1 IDENTITY.SOL

A continuación, se explicará cada uno de los métodos que componen al contrato “Identity.sol”:

- **add**: este método es llamado cuando se quiere crear una de las tres diferentes identidades disponibles. Para ello se le debe suministrar los siguientes valores:
 - **name**: nombre de la identidad que se crea, por ejemplo, “Comillas” si se crea una identidad del tipo Issuer.
 - **ownerType**: representa el tipo de identidad que se va a registrar. Si el entero recibido es un 1 será una identidad del tipo “User”; si el entero recibido es un 2 será una identidad del tipo “Issuer”; si el entero recibido es un 3 será una identidad del tipo “Checker”;
- **getIdentities**: este método es llamado para obtener todas las identidades, claims y ofertas que han sido “registradas”. Así mismo, devuelve el número de identidades, claims y ofertas creadas.
- **getNumIdentities**: este método devuelve el número de identidades que han sido “registradas”.
- **getType**: este método devuelve el tipo de identidad que es la key pública que se le suministra. Esta puede ser de tipo 1, “User”, de tipo 2, “Issuer”, o de tipo 3, “Checker”.
- **addC**: este método es llamado cuando un User quiere crear una claim. Para ello se le suministran los siguientes valores:
 - **topic**: representa el tipo de claim que se va a crear y puede ser de tipo 1, si el usuario tiene el título “GITT”, o de tipo 2 si tiene el título “GITT”.
 - **issuerName**: representa el nombre del Issuer que respalda esta claim y donde ha sido cursado dicho título.
 - **ownerName**: representa el nombre del User que ha creado la claim.

- val: este método lo emplea el Issuer para validar o invalidar cualquier claim que haya sido creada, por un User, y que esté acreditado por él. Para ello se le suministran los siguientes valores:
 - type: representa 1 si la claim quiere ser invalidada, o 2 si quiere ser validada.
 - claimId: este entero representa la id del claim que se quiere validar, o invalidar.
- addO: este método se emplea para la creación de una oferta de trabajo, por parte de un Checker. Para ello se le suministran los siguientes valores:
 - requirement: este entero representa el requerimiento, académico, necesario para aplicar a la oferta. Si este valor es un 1, será necesario un título “GITI”, si el valor es un 2, será necesario un título “GITT”.
 - offerName: nombre de la oferta que se está creando, y que se quiere que aparezca para que puedan aplicar los Users.
- getNumCl: este método devuelve el número de Claims que han sido creadas, por un User.
- getNumO: este método devuelve el número de Ofertas que han sido creadas, por un Checker.
- applyO: este método es empleado, por un User, cuando cumple con los requisitos necesarios para aplicar a una oferta. Para ello se le suministrarán los siguientes valores:
 - user: representa la key pública del User que aplica a la oferta.
 - offerId: representa la id de la oferta a la que el User trata de aplicar.

5.6.2 CÓDIGO IDENTITY.SOL

Una vez que uno de estos métodos es llamado invocará, al método pertinente que se encuentre en el contrato oportuno. Así, por ejemplo, si se invoca el método “add” se ejecutará el código siguiente:

```
function add(string memory _name, uint256 _ownerType) public {
    if(_ownerType == 1){
        addUser(_name, _ownerType, msg.sender);
    }else if(_ownerType == 2){
        addIssuer(_name, msg.sender);
    }else if(_ownerType == 3){
        addChecker(_name, msg.sender);
    }
}
```

Figura 11 - Código de la función "add" del contrato "Identity.sol"

5.6.3 CÓDIGO CLAIMOWNER.SOL

Con esto, dependiendo del tipo de identidad que se quiera crear se llamará a otro contrato. Si el tipo es 1 se invocará el método "addUser" del contrato "ClaimOwner.sol" ejecutando el siguiente trozo de código:

```
function addUser(string memory _name, uint256 _ownerType, address sender) public{
    if(addressToUsers[sender].init != true){
        uint256[] memory claimId;
        addressToUsers[sender] = Identidad(sender, _name, 0, _ownerType, true, claimId);
        usersToAddress[_name] = sender;
        identidades[numUsers] = sender;
        namesArray.push(_name);
        numUsers = numUsers + 1;
    }
}
```

Figura 12 - Código de la función "addUser" del contrato "ClaimOwner.sol"

En este método se procede al registro del User, si esta public key no ha creado un User previamente.

5.6.4 CÓDIGO CLAIMISSUER.SOL

Por otro lado, si el tipo de identidad que se quiere crear es del tipo 2, se invocará el método "addIssuer" del contrato "ClaimIssuer.sol" ejecutando el siguiente trozo de código:


```
function addIssuer(string memory _name, address sender) public{
    if(addressToIssuers[sender].init != true){
        addressToIssuers[sender] = Issuer(sender, _name, true);
        issuersToAddress[_name] = sender;
        issuers[numIssuers] = sender;
        namesArray.push(_name);
        numIssuers = numIssuers + 1;
    }
}
```

Figura 13 - Código de la función "addIssuer" del contrato "ClaimIssuer.sol"

En este método se procede al registro del Issuer, si esta public key no ha creado un Issuer previamente.

5.6.5 CÓDIGO CLAIMCHECKER.SOL

Finalmente, si el tipo de identidad que se quiere crear es del tipo 3, se invocará al método "addChecker" del contrato "ClaimChecker.sol" ejecutando el siguiente trozo de código:

```
function addChecker(string memory _name, address sender) public{
    if(addressToCheckers[sender].init != true){
        uint256[] memory checkerId;
        addressToCheckers[sender] = Checker(sender, _name, true, checkerId);
        checkersToAddress[_name] = sender;
        checkers[numCheckers] = sender;
        namesArray.push(_name);
        numCheckers = numCheckers + 1;
    }
}
```

Figura 14 - Código de la función "addChecker" del contrato "ClaimChecker.sol"

En este método se procede al registro del Checker, si esta public key no ha creado un Checker previamente.

5.7 SERVIDOR DE LA APLICACIÓN

Una vez se ha comprendido el funcionamiento de los contratos de la aplicación y las interacciones entre ellos, se procederá a la explicación del servidor que realiza la conexión con la red Ethereum y los Smart Contracts.

El servidor de la aplicación está escrito mediante Web3JS y se encuentra desarrollado en el archivo “app.js”.

5.7.1 INICIO DEL SERVIDOR

Esta esta formada por una serie de funciones, que se encargan de iniciar el servidor, conectar a la red Ethereum y realizar las llamadas al contrato “Identity.sol”, el cual realizará las interacciones pertinentes con el resto de los contratos.

```
initWeb3: async function() {
  if (window.ethereum) {
    App.web3Provider = window.ethereum;
    try {
      await window.ethereum.enable();
    } catch (error) {
      console.error("User denied account access")
    }
  }
  else if (window.web3) {
    App.web3Provider = window.web3.currentProvider;
  }
  else {
    App.web3Provider = new Web3.providers.HttpProvider('http://localhost:7545');
  }
  web3 = new Web3(App.web3Provider);

  return App.initContract();
},
```

Figura 15 - Código de la función de inicio del servidor

En primer lugar, se realiza el despliegue del servidor en la url “http://localhost:7545”.

5.7.2 CONEXIÓN CON LA RED ETHEREUM

Si toda la ejecución se produce de manera satisfactoria, se procede a la conexión con la red Ethereum.

```
initContract: function() {
  $.getJSON('Identity.json', function(data) {
    var IdentityArtifact = data;
    App.contracts.Identity = TruffleContract(IdentityArtifact);

    App.contracts.Identity.setProvider(App.web3Provider);

    return App.Identity();
  });

  return App.Identity();
},
```

Figura 16 - Código de la función de conexión del servidor al contrato "Identity.sol"

Con la ejecución de este código, se procede a la descarga del JSON que contiene el esqueleto del contrato "Identity.sol". Este es producido al realizar la migración de los contratos.

Por último, se mostrará un ejemplo es una de las funciones que interactúan con el contrato "Identity.sol". Para ello se comentará la función "createIdentity" dentro del archivo "app.js".

5.7.3 FUNCTION "CREATEIDENTITY"

A continuación, se muestra el código de la función que permite la conexión para la creación de una identidad, a través de "Identity.sol".

```
createIdentity: function(event) {
  event.preventDefault();

  var identityInstance;

  web3.eth.getAccounts(function(error, accounts) {
    if (error) {
      console.log(error);
    }

    var account = accounts[0];

    App.contracts.Identity.deployed().then(function(instance) {
      identityInstance = instance;
      var name = $('#identityName').val();
      var type = $('#identityType').val();
      $('#identityName').val('');
      $('#myModal').modal('toggle');

      return identityInstance.add(name,type, {from: account});
    }).then(function(result) {
      return App.Identity();
    }).catch(function(err) {
      console.log(err.message);
    });
  });
},
```

Figura 17 - Código de la función "CreateIdentity" del servidor

En este trozo de código, se obtiene la public key del usuario que está realizando la conexión con el contrato, en la variable "account".

Así mismo, se almacena en la variable "name" el nombre de la identidad que se está creando y en la variable "type", el tipo de identidad que se quiere crear.

A continuación, se realiza la conexión al contrato "Identity.sol", llamando al método "add" del mismo. En la llamada del método, se suministra las variables que contienen el nombre y el tipo de identidad a crear. Además, se añade la public key del usuario que está realizando la transacción para el cobro de Ether, si fuera necesaria para la transacción.

Posteriormente, se realizaron una serie de pruebas del funcionamiento del portal a través del framework de pruebas sobre JavaScript Mocha.

5.8 TESTS CON MOCHA

Para comprobar el correcto funcionamiento de la aplicación se ha empleado Mocha.js. Este es un framework para la realización de pruebas en javascript.

Para realizar un sistema de pruebas completo se ha comprobado que todos los métodos, que se encuentran dentro del contrato “Identity.sol” funcionan correctamente. Este set de pruebas se realiza, solamente, en este contrato, debido a que es el contrato de entrada y llamada, al resto de contratos, a través del cual se controla la totalidad de la plataforma.

Con el fin de realizar la comprobación, se han realizado diez pruebas diferentes, las cuales han sido pasadas con éxito, como se puede ver a continuación:

```
Contract: Identity
  ✓ Deberia de crear una identidad del tipo User con nombre Alfonso (164ms)
  ✓ Deberia de dar error al crear dos identidades con la misma cuenta (71ms)
  ✓ Deberia de crear una identidad del tipo Issuer con nombre Comillas (104ms)
  ✓ Deberia de crear una identidad del tipo Checker con nombre BBVA (136ms)
  ✓ Deberia de crear una claim del tipo GITT (1) al usuario Alfonso teniendo Comillas como Issuer (143ms)
  ✓ Deberia de validar la claim creada por Alfonso (72ms)
  ✓ Deberia de crear una oferta de trabajo con requisito GITT de software developer (191ms)
  ✓ Deberia de permitir aplicar a la oferta de trabajo de Software Developer (79ms)
  ✓ Deberia de Devolver todo lo creado y probado antes (309ms)
  ✓ Deberia devolver la suma de todas las identidades

10 passing (1s)
```

Figura 18 - Resultado de todos los test de Mocha

Con el fin de comprender mejor cada uno de los tests realizados se mostrará cada una de las pruebas y se acompañará de una breve explicación:

5.8.1 PRUEBA DE CREACIÓN DE UN “USER”

```
it("Debería de crear una identidad del tipo User con nombre Alfonso", () =>
  Identity.deployed()
    .then(instance => instance.add("Alfonso",1,{from:accounts[0]}))
    .then(result => {
      assert.equal(
        result.logs[0].args.created,
        true,
        "El usuario ha sido creado con éxito"
      );
    })
);
```

Figura 19 - Código del test de creación de un "User"

Como se ve, se crea un User con la cuenta [0] de Ganache con el nombre “Alfonso”. Una vez realizada la creación del User, se compara el evento emitido por la función con el booleano “true” para comprobar si se ha realizado la operación con éxito.

El resultado de esta prueba es satisfactorio:

```
✓ Debería de crear una identidad del tipo User con nombre Alfonso (164ms)
```

Figura 20 - Resultado del test de creación de un "User"

5.8.2 PRUEBA DE ERROR AL CREAR DOS IDENTIDADES CON LA MISMA CUENTA

```
it("Debería de dar error al crear dos identidades con la misma cuenta", () => {
  return Identity.deployed()
    .then(instance => {
      return instance.add("Alfonso",1,{from:accounts[0]});
    })
    .then(result => {
      assert.equal(
        result.logs[0].args.created,
        false,
        "La cuenta ya ha creado más de 1 usuario"
      );
    })
});
```

Figura 21 - Código del test de error al crear dos identidades con la misma cuenta

Como se ve, se trata de crear un User con la cuenta [0] de Ganache con el nombre “Alfonso”. Una vez realizada la creación del User, se compara el evento emitido por la función con el booleano “false” para comprobar si se ha habido error al tratar de crear el User.

El resultado de la prueba es satisfactorio:

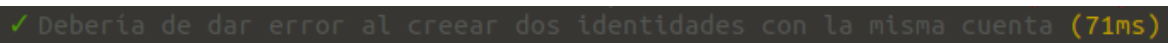


Figura 22 - Resultado del test de error al crear dos identidades con la misma cuenta

5.8.3 PRUEBA DE CREACIÓN DE UN “ISSUER”

```
it("Debería de crear una identidad del tipo Issuer con nombre Comillas", () =>
  Identity.deployed()
    .then(instance => instance.add("Comillas",2,{from:accounts[1]}))
    .then(result => {
      assert.equal(
        result.logs[0].args.created,
        true,
        "El usuario ha sido creado con éxito"
      );
    }));
```

Figura 23 - Código del test de creación de un "Issuer"

Como se ve, se crea un Issuer con la cuenta [1] de Ganache con el nombre “Comillas”. Una vez realizada la creación del Issuer, se compara el evento emitido por la función con el booleano “true” para comprobar si se ha realizado la operación con éxito.

El resultado de esta prueba es satisfactorio:

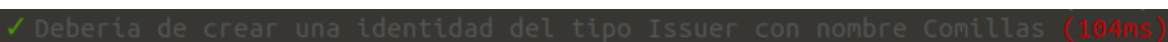


Figura 24 - Resultado del test de creación de un "Issuer"

5.8.4 PRUEBA DE CREACIÓN DE UN “CHECKER”

```
it("Debería de crear una identidad del tipo Checker con nombre BBVA", () =>
  Identity.deployed()
    .then(instance => instance.add("BBVA",3,{from:accounts[2]}))
    .then(result => {
      assert.equal(
        result.logs[0].args.created,
        true,
        "El usuario ha sido creado con éxito"
      );
    }));
```

Figura 25 - Código del test de creación de un "Checker"

Como se ve, se crea un Checker con la cuenta [2] de Ganache con el nombre “BBVA”. Una vez realizada la creación del Checker, se compara el evento emitido por la función con el booleano “true” para comprobar si se ha realizado la operación con éxito.

El resultado de esta prueba es satisfactorio:

```
✓ Debería de crear una identidad del tipo Checker con nombre BBVA (136ms)
```

Figura 26 - Resultado del test de creación de un "Checker"

5.8.5 PRUEBA DE CREACIÓN DE UNA CLAIM

```
it("Debería de crear una claim del tipo GITT (1) al usuario Alfonso teniendo Comillas como Issuer", () =>
  Identity.deployed()
    .then(instance => instance.addC(1,"Comillas","Alfonso",{from:accounts[0]}))
    .then(result => {
      assert.equal(
        result.logs[0].args.created,
        true,
        "La Claim se ha creado con éxito"
      );
      assert.equal(
        result.logs[0].args.num,
        1,
        "Se ha creado 1 claim"
      );
    }));
```

Figura 27 - Código del test de creación de una claim

Como se ve, se crea una claim, teniendo como Issuer a “Comillas” y como dueño a “Alfonso”, siendo, así mismo, una claim de para certificar que Alfonso tiene el título de “GITT”. Dicha operación se realiza desde la cuenta [0], cuenta que pertenece al usuario Alfonso. Se compara el evento emitido por la función con el booleano “true” para comprobar si se ha realizado la operación con éxito, además se compara que el número de claims creadas es “1”.

El resultado de esta prueba es satisfactorio:

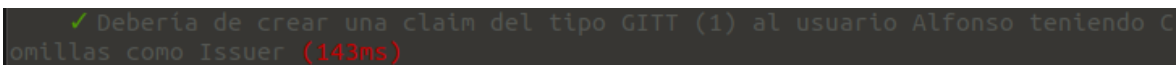


Figura 28 - Resultado del test de creación de una claim

5.8.6 PRUEBA DE VALIDACIÓN LA CLAIM

```
it("Debería de validar la claim creada por Alfonso", () =>
  Identity.deployed()
    .then(instance => instance.val(2,0,{from:accounts[1]}))
    .then(result => {
      assert.equal(
        result.logs[0].args.created,
        true,
        "La Claim se ha validado con éxito"
      );
    });
});
```

Figura 29 - Código del test de validación de una claim

Como se ve, se llama a la función val() que valida la claim mediante la opción “2”, que es la de validar, a la claim con id “0” desde la cuenta [1] a la que pertenece el Issuer “Comillas”, que es contra quién se quiere validar la claim. Una vez se valida la claim, se comprueba que se ha validado con éxito contra el booleano “true”.

El resultado de esta prueba es satisfactorio:

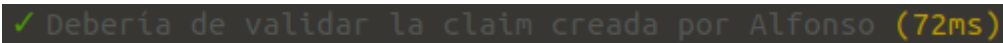


Figura 30 - Resultado del test de validación de una claim

5.8.7 PRUEBA DE CREAR UNA OFERTA DE TRABAJO

```
it("Debería de crear una oferta de trabajo con requisito GITT de software developer", () =>
  Identity.deployed()
    .then(instance => instance.add0(1,"Software Developer",{from:accounts[2]}))
    .then(result => {
      assert.equal(
        result.logs[0].args.created,
        true,
        "La Oferta se ha creado con éxito"
      );
      assert.equal(
        result.logs[0].args.num,
        1,
        "Se ha creado 1 oferta"
      );
    }
  ));
```

Figura 31 - Código del test de creación de una oferta de trabajo

Como se ve, se crea una oferta con título “Software Developer” que tenga como requisito un título de “GITT”. Dicha oferta se crea desde la cuenta [2] a la cual pertenece el Checker “BBVA”. Se compara el evento emitido por la función con el booleano “true” para comprobar que se ha creado la oferta con éxito.

El resultado de esta prueba es satisfactorio:

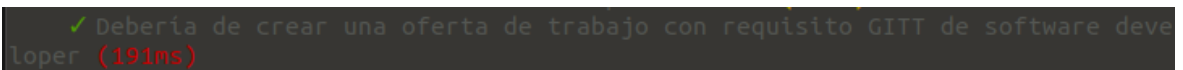


Figura 32 - Resultado del test de creación de una oferta de trabajo

5.8.8 PRUEBA DE APLICACIÓN A LA OFERTA DE TRABAJO

```
it("Debería de permitir aplicar a la oferta de trabajo de Software Developer", () =>
  Identity.deployed()
    .then(instance => instance.apply0(0,{from:accounts[0]}))
    .then(result => {
      assert.equal(
        result.logs[0].args.created,
        true,
        "Se ha aplicado con éxito"
      );
    }));
```

Figura 33 - Código del test de aplicación a una oferta de trabajo

Como se ve, se aplica a la oferta de trabajo con id “0” desde la cuenta [0], a la que pertenece el User “Alfonso”. Se compara el evento emitido por la función con el booleano “true” para comprobar que se ha aplicado a la oferta con éxito.

El resultado de esta prueba es satisfactorio:

```
✓ Debería de permitir aplicar a la oferta de trabajo de Software Developer (
79ms)
```

Figura 34 - Resultado del test de aplicación a una oferta de trabajo

5.8.9 PRUEBA DE DEVOLUCIÓN DE TODAS LA INFORMACIÓN GUARDADA

```
it("Debería de Devolver todo lo creado y probado antes", () =>
  Identity.deployed()
    .then(instance => instance.getIdentities({from:accounts[0]}))
    .then(result => {
      assert.equal(
        result.logs[0].args.created,
        true,
        "3 identidades + 1 claim creados"
      );
      assert.equal(
        result.logs[2].args.ownerType,
        1,
        "1 User creado"
      );
      assert.equal(
        result.logs[2].args.name,
        "Alfonso",
        "1 User creado de nombre Alfonso"
      );
      assert.equal(
        result.logs[2].args.numClaims,
        1,
        "1 claim creada por Alfonso"
      );
      assert.equal(
        result.logs[3].args.ownerType,
        2,
        "1 Issuer creado"
      );
      assert.equal(
        result.logs[3].args.name,
        "Comillas",
        "1 Issuer creado de nombre comillas"
      );
      assert.equal(
        result.logs[4].args.ownerType,
        3,
        "1 Checker creado"
      );
    });
```

Figura 35 - Código del test de devolución de toda la información guardada (1)

```
assert.equal(
  result.logs[4].args.name,
  "BBVA",
  "1 Checker creado con nombre BBVA"
);
assert.equal(
  result.logs[5].args.issuerName,
  "Comillas",
  "1 Claim contrastada con Comillas"
);
assert.equal(
  result.logs[5].args.ownerName,
  "Alfonso",
  "1 Claim creada por Alfonso"
);
assert.equal(
  result.logs[5].args.valid,
  true,
  "La Claim creada por Alfonso es valida"
);
assert.equal(
  result.logs[6].args.num,
  1,
  "El número de ofertas creadas es de 1"
);
assert.equal(
  result.logs[7].args.offerName,
  "Software Developer",
  "La oferta creada es de Software Developer"
);
assert.equal(
  result.logs[7].args.checkerName,
  "BBVA",
  "El creador de la oferta es BBVA"
);
assert.equal(
  result.logs[7].args.requirement,
  1,
  "El requisito para aplicar a la oferta es tener el título de GITT"
);
```

Figura 36 - Código del test de devolución de toda la información guardada (2)

```
assert.equal(  
  result.logs[7].args.usersInterested[0],  
  accounts[0],  
  "El user que ha aplicado se llama Alfonso"  
);  
});
```

Figura 37 - Código del test de devolución de toda la información guardada (3)

En esta prueba se obtiene la información creada que se ha almacenado en la cadena de bloques. Esta está compuesta de Identidades, claims y ofertas. Las comprobaciones que se realizan son las siguientes:

1. La cantidad de claims e identidades creados son 4.
2. La cantidad de User creados es 1.
3. El User creado se llama "Alfonso".
4. Hay 1 claim creada por el usuario "Alfonso".
5. La cantidad de Issuer creados es 1.
6. El Issuer creado se llama "Comillas".
7. La cantidad de Checker creados es 1.
8. El Checker creado se llama "BBVA".
9. La claim creada la contrasta el Issuer "Comillas".
10. La claim creada pertenece al User "Alfonso".
11. La claim creada por "Alfonso" está validada.
12. La cantidad de ofertas creadas es 1.
13. El título de la oferta es "Software Developer".
14. El creador de la oferta es el Checker "BBVA".
15. El requisito para aplicar a la oferta es tener el título "GITT".
16. El User que ha aplicado a la oferta es "Alfonso".

Si todas las condiciones se cumplen, la prueba saldrá con éxito:

✓ Debería de Devolver todo lo creado y probado antes (309ms)

Figura 38 - Resultado del test de devolución de toda la información guardada

5.8.10 PRUEBA CANTIDAD DE IDENTIDADES CREADAS

```
it("Debería devolver la suma de todas las identidades", () => {
  let num;
  Identity.deployed()
    .then(instance => {instance.getNumIdentities.call(accounts[0]);
    })
    .then(outNumIdentities => {
      num = outNumIdentities.toNumber();
    })
    .then(() => {
      assert.equal(
        num,
        3,
        "El número de identidades registradas es 3"
      );
    });
});
```

Figura 39 - Código de test de cantidad de identidades creadas

Como se puede apreciar, se llama a la función que devuelve el número de identidades, de “Identity.sol”. El resultado se compara con el número 3.

El resultado de esta prueba es satisfactorio:

✓ Debería devolver la suma de todas las identidades

Figura 40 - Resultado del test de cantidad de identidades creadas

6. ANÁLISIS DE RESULTADOS

El trabajo realizado a cumplido con todas las expectativas generadas, sobre el mismo, siendo 100% funcional y sirviendo para solventar los problemas presentados en el apartado **descripción del proyecto**.

Se ha conseguido, por tanto, que se puedan crear 3 tipos diferentes de identidades con las respectivas acciones esperadas y descritas en el apartado de **Sistema Desarrollado**. Esto es, los usuarios de la plataforma son capaces de identificarse como User, Issuer o Checker. Una vez se ha identificado, el usuario es capaz de:

5. Crear claims, respaldadas por un Issuer, y emplearlas para aplicar a ofertas de trabajo, creadas por los Checker, para tratar de optar a un puesto vacante, siempre y cuando la claim creada se encuentre validada y el User disponga de los requisitos, claims, necesarios para aplicar a la oferta.
 -
6. Validar o invalidar las claims creados por los usuarios contra tu institución, si eres un Issuer.
 -
7. Crear ofertas de trabajo que necesiten de unos requisitos, para poder aplicar a ellas, si eres un Checker.

Así, se ha creado un portal de empleo, totalmente funcionalidad, que es capaz de afrontar los retos del Internet del siglo XXI. Estos son:

1. Provee de la seguridad necesaria, gracias a los cifrados e inmutabilidad, características, de Blockchain.
 -
2. Proporciona identificación personal, e intransferible, a cada uno de los usuarios, ya que están enlazados con su cuenta, la cual es única e irrepetible.

-

La consecución de todos los objetivos, de manera satisfactoria, así como la realización del proyecto por un bajo coste podrían llevar al proyecto a ser mejorado, en un futuro, para su posible aplicación en el mundo real.

7. CONCLUSIONES Y TRABAJOS FUTUROS

En conclusión, a lo largo de este proyecto, se ha tratado de realizar un portal de comunicación entre el alumnado, la universidad y las empresas.

Con, se ha tratado de simular un entorno en el que un usuario tratase de aplicar a una oferta de empleo, creada por una empresa, y para la cual fuera necesaria la acreditación de un título académico.

En este punto apareció la figura de la entidad académica como garante de la veracidad de las titulaciones académicas, dando así validez a los títulos académicos y, permitiendo, la aplicación a la oferta de trabajo, por parte del alumno.

Sin embargo, la seguridad e inmutabilidad de los títulos no solo las han brindado las instituciones académicas ni la complejidad, o falta de complejidad, del código. Estas las han brindado la tecnología Blockchain, gracias a sus características, las cuales permitieron la creación de identidades no suplantables y confirió la seguridad de las comunicaciones, a través del sistema de cifrado asimétrico que confiere.

Por último, y aunque al proyecto se le ha tratado de dar la mayor amplitud posible, el portal está sujeto a futuras mejoras que podrían ser añadidas evolucionando algunos aspectos del proyecto, para mejorar su funcionalidad, con vistas al futuro. Los cambios a realizar serían los que se describen a continuación.

1. Integración de la plataforma en un entorno web con más funcionalidades, como sea el envío de emails tras el registro, o comunicación de aplicación a las ofertas a los correos corporativos de las empresas.
2. Integración en la red universitaria para facilitar el envío de peticiones de validación de claims a las universidades, así como facilitar la validación de las claims tras la creación de las mismas y no permitir usarlas hasta que no sean validadas.

3. Aumento de la cantidad de claims que se puedan permitir, así como, nombres, apellidos, etc. Que sean validables desde otras instituciones de carácter nacional, como el Ministerio del interior, para la creación de perfiles más completos.

Estos son algunos ejemplos de evoluciones que podrían convertir la plataforma en una mucho más completa y funcional, con el fin de implantar la plataforma de forma real, y no académicos y de investigación.

8. BIBLIOGRAFÍA

- [1] Página Oficial de Truffle. <https://truffleframework.com>.
- [2] Página Oficial de Ganache. <https://truffleframework.com/ganache>.
- [3] Página Oficial de Virtualbox. <https://www.virtualbox.org>.
- [4] Página Oficial de NPM. <https://www.npmjs.com>.
- [5] Página Oficial de Node. <https://nodejs.org/es>.
- [6] Página Oficial de Ethereum. <https://www.ethereum.org>.
- [7] Página Oficial de Trello. <https://trello.com>.
- [8] Página Oficial de Visual Studio Code. <https://code.visualstudio.com>.
- [9] Página Oficial de Ubuntu. <https://www.ubuntu.com>.
- [10] Página Oficial de Mocha. <https://mochajs.org>.
- [11] Página oficial de MetaMask. <https://metamask.io>.
- [12] Página Oficial de Solidity. <https://solidity-es.readthedocs.io/es/latest>.
- [13] Página Oficial de Web3js. <https://web3js.readthedocs.io/en/1.0>.
- [14] Página Oficial de W3Schools. <https://www.w3schools.com>.
- [15] Página Oficial de JQuery. <https://jquery.com>.
- [16] Página Oficial de Bootstrap. <https://getbootstrap.com>.

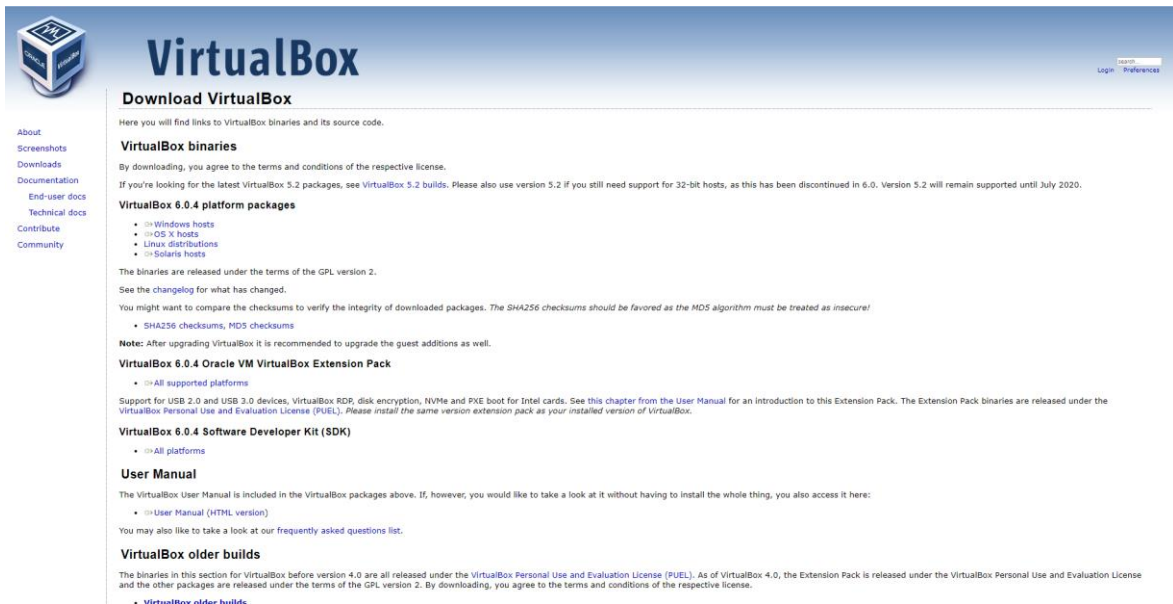
- [17] Ian Sommerville, Pearson, 2010. “Software Engineering 9”. <https://www.pearson.com/us/higher-education/product/Sommerville-Software-Engineering-9th-Edition/9780137035151.html>.
- [18] Página de la Wikipedia sobre Blockchain. <https://en.wikipedia.org/wiki/Blockchain>.
- [19] Página Oficial de Blockchain. <https://www.blockchain.com/es>.
- [20] Preukschat, Alexander. “Blockchain: la revolución industrial de internet”. https://books.google.es/books/about/Blockchain_la_revoluci%C3%B3n_industrial_de_internet?id=Lb7DDgAAQBAJ&source=kp_book_description&redir_esc=y
- [21] Pastor, Javier. “Qué es blockchain: la explicación definitiva para la tecnología más de moda”. <https://www.xataka.com/especiales/que-es-blockchain-la-explicacion-definitiva-para-la-tecnologia-mas-de-moda>

ANEXO A: GUÍA DE INSTALACIÓN

En el presente Anexo se tratará de realizar una explicación del entorno necesario a instalar para el desarrollo de la aplicación. Para ello se realizará la instalación de los programas descritos a continuación.

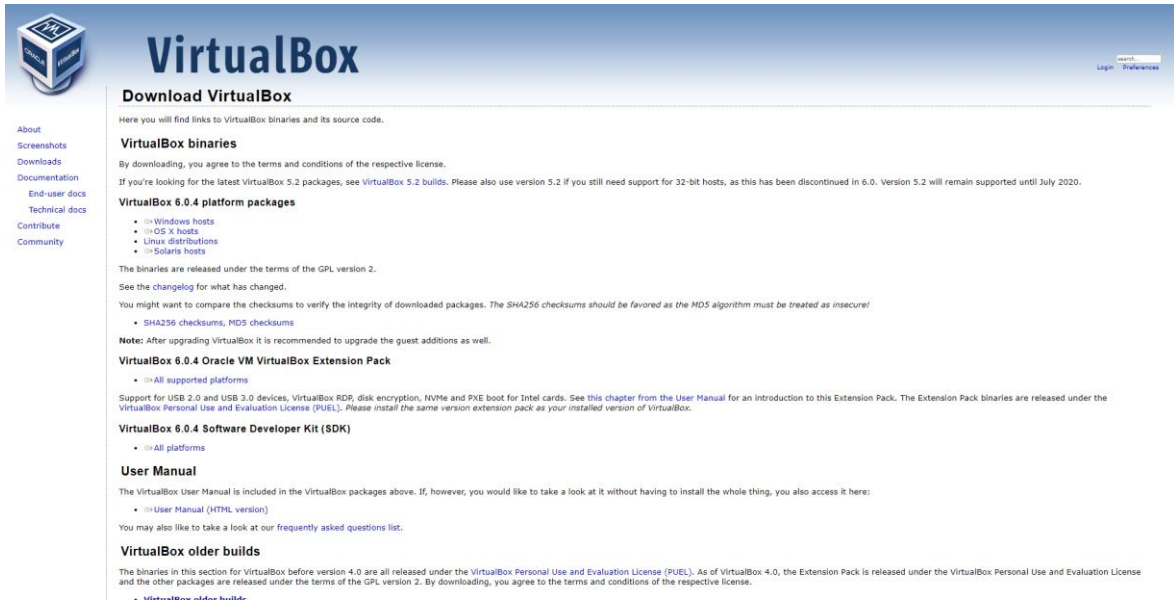
8.1 VIRTUALBOX

VirtualBox es el programa que permite la virtualización del sistema operativo Ubuntu, el cual se va a emplear en este proyecto. Para la instalación de VirtualBox, primero hay que acceder a su página web [3], una vez dentro, se presiona sobre descargas y le redireccionará a la siguiente página:



The screenshot shows the 'Download VirtualBox' page on the Oracle website. It features a navigation menu on the left with links for About, Screenshots, Downloads, Documentation, End-user docs, Technical docs, Contribute, and Community. The main content area includes a 'Download VirtualBox' heading, a 'VirtualBox binaries' section with a link to the latest version (6.0), and a 'VirtualBox 6.0.4 platform packages' section listing links for Windows, OS X, Linux, and Solaris hosts. There are also sections for the Oracle VM VirtualBox Extension Pack, the Software Developer Kit (SDK), and the User Manual. A footer note mentions older builds and their licensing.

Una vez se está en esta página, debe elegirse el package para el sistema operativo que se tiene, en este caso, se elegirá Windows:

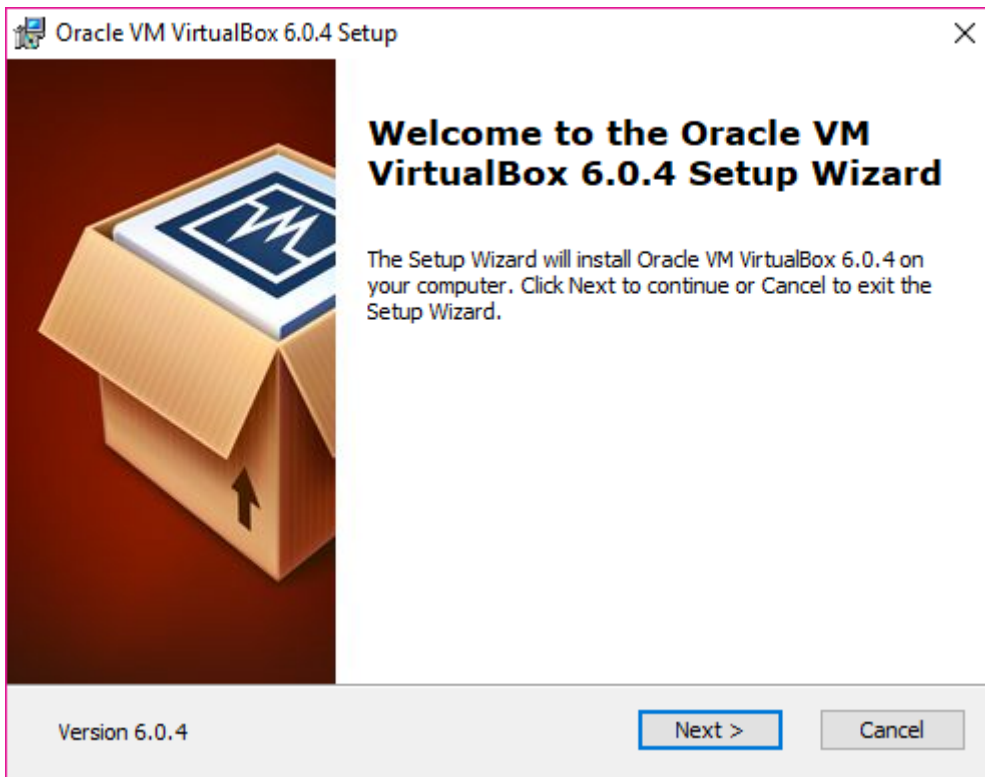


The screenshot shows the VirtualBox website's download page. The main heading is "Download VirtualBox". Below it, there are sections for "VirtualBox binaries", "VirtualBox 6.0.4 platform packages" (listing Windows, OS X, Linux, and Solaris hosts), "VirtualBox 6.0.4 Oracle VM VirtualBox Extension Pack", "VirtualBox 6.0.4 Software Developer Kit (SDK)", "User Manual", and "VirtualBox older builds". A sidebar on the left contains navigation links like "About", "Screenshots", "Downloads", "Documentation", "End-user docs", "Technical docs", "Contribute", and "Community".

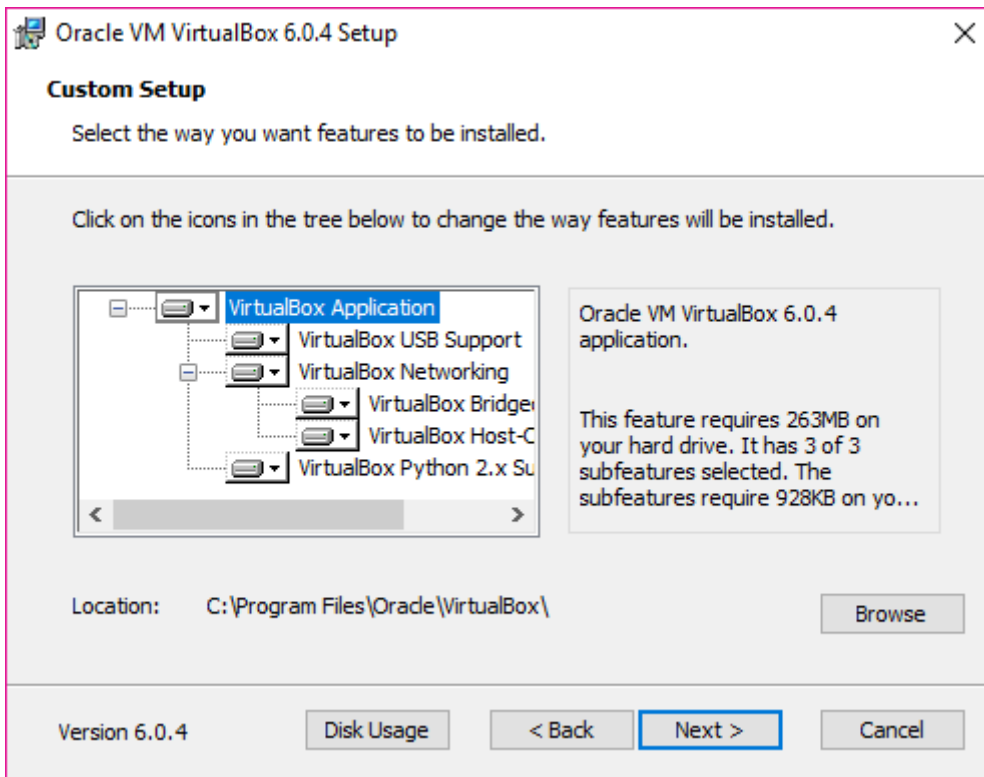
Una vez pulsado, comenzará una descarga, si emplea Google Chrome, para la descarga, se mostrará la siguiente barra de descargas:



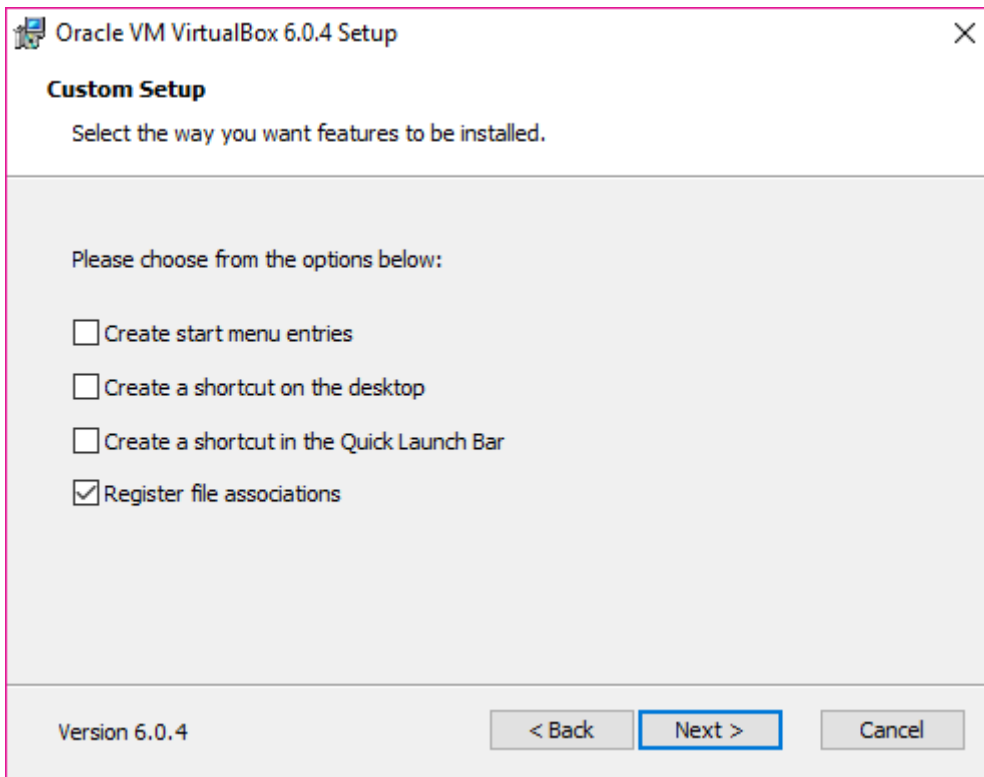
Una vez haya finalizado, hacer click sobre el programa, el cual abrirá el siguiente menú de instalación:



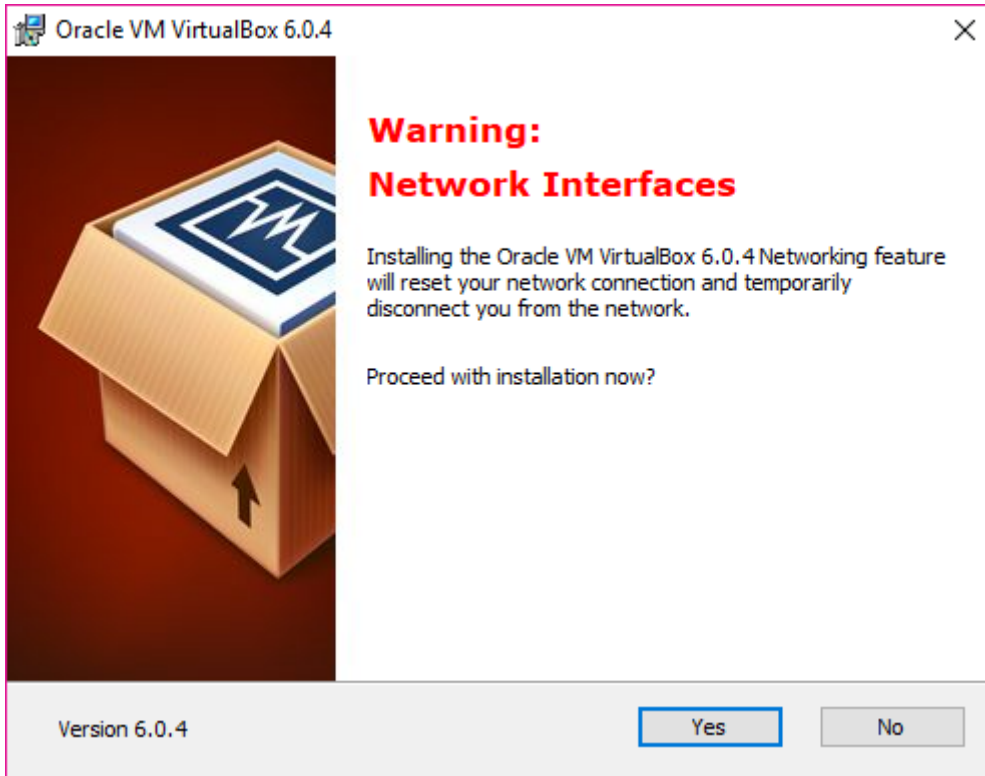
Hacer click sobre “Next>”:



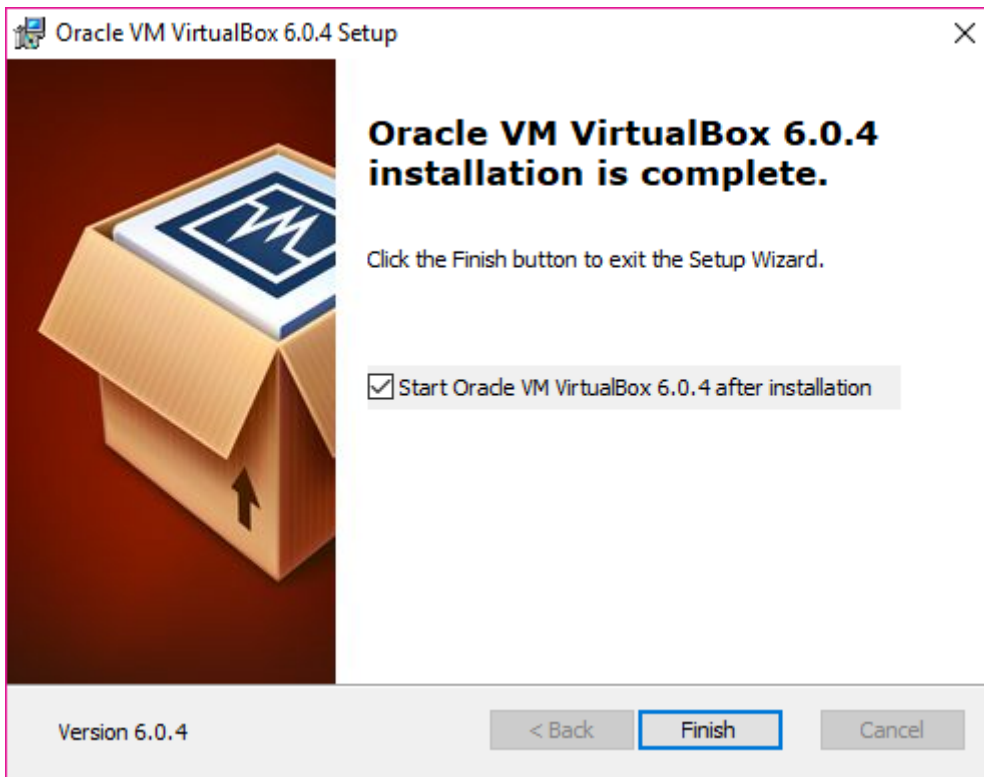
Elegir el tipo de instalación, en este caso se dejó el por defecto, y darle a “Next>”:



Elegir las opciones que se quiera para crear acceso directo en el escritorio, etc. En este caso se dejó por defecto, una vez elegidas las opciones pulsar sobre “Next>”:



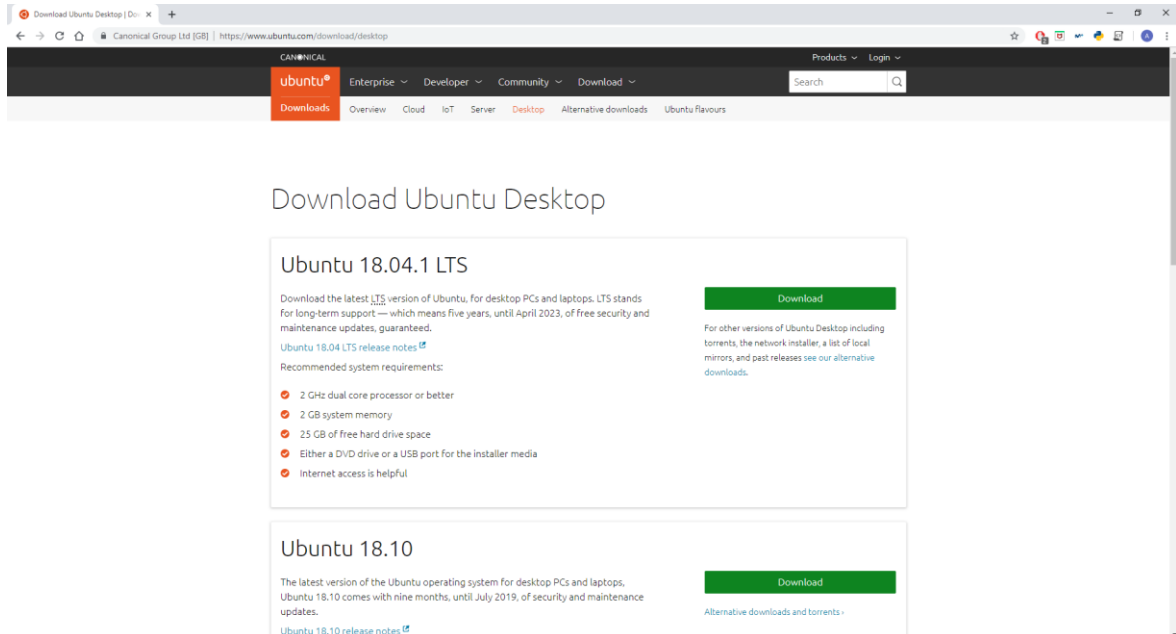
Pulsar sobre “Yes”:



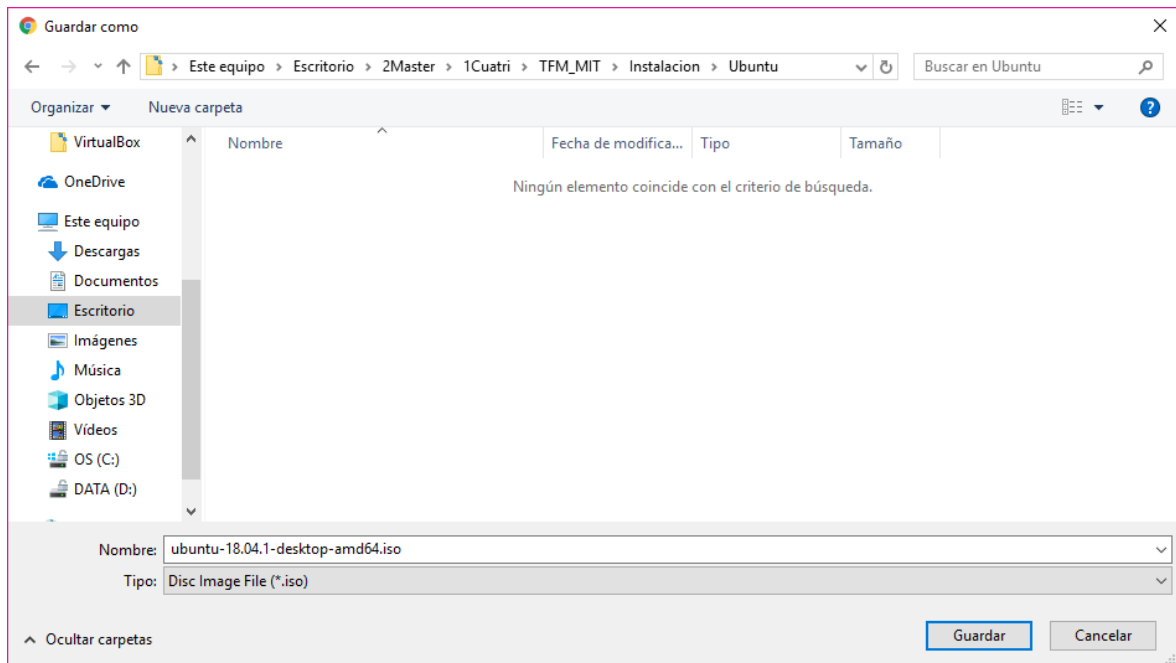
La instalación se ha finalizado de manera satisfactoria, ahora pulsar sobre “Finish” para poder continuar con la instalación de la máquina virtual.

8.2 DESCARGA DE UBUNTU E INSTALACIÓN DE LA MÁQUINA VIRTUAL

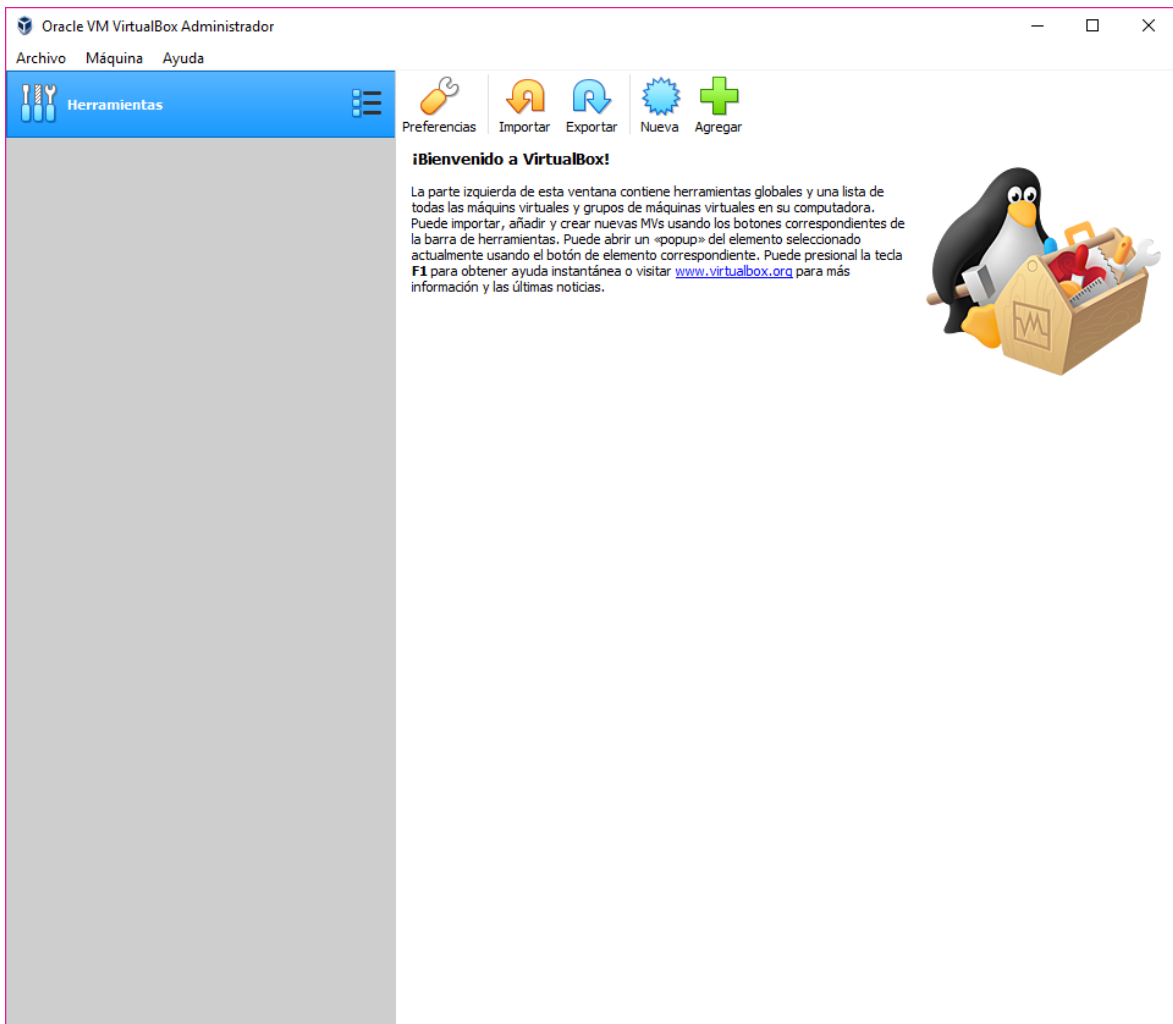
Una vez se ha descargado e instalado, de manera satisfactoria, VirtualBox, se procede a la descarga de Ubuntu. Para ello se entra en la página web oficial de Ubuntu ^[9] y se le da a descargar en la última versión, del mismo:



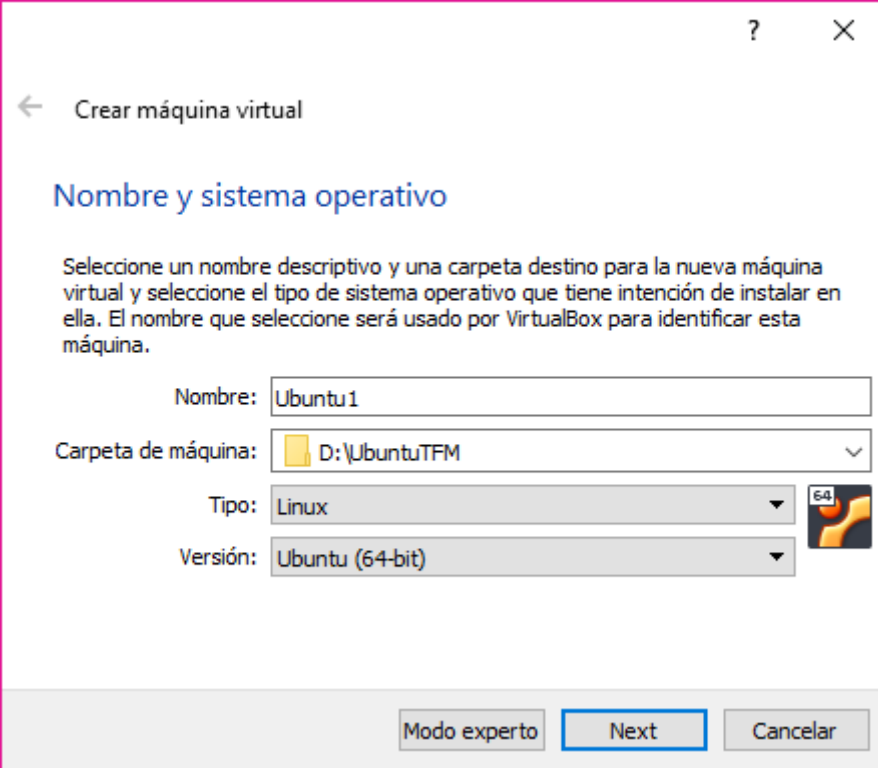
Una vez dado a descargar, se elige el directorio donde guardar la imagen y se guarda:



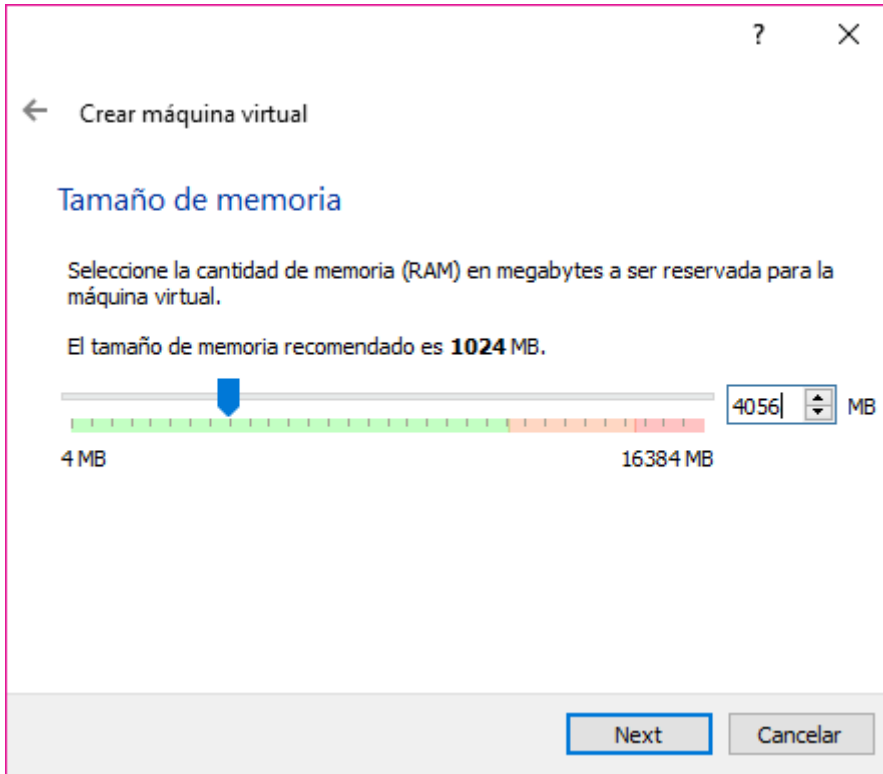
A continuación, se abre la ventana de inicio de VirtualBox:



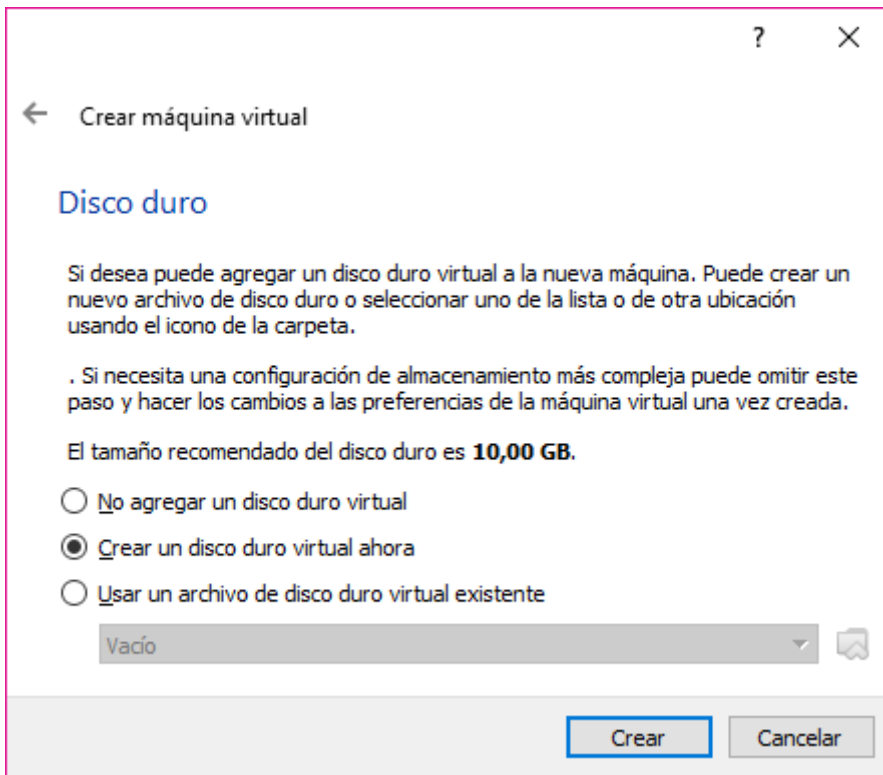
Se pulsa sobre la cruz verde que pone “Agregar” y se abrirá la siguiente ventana:



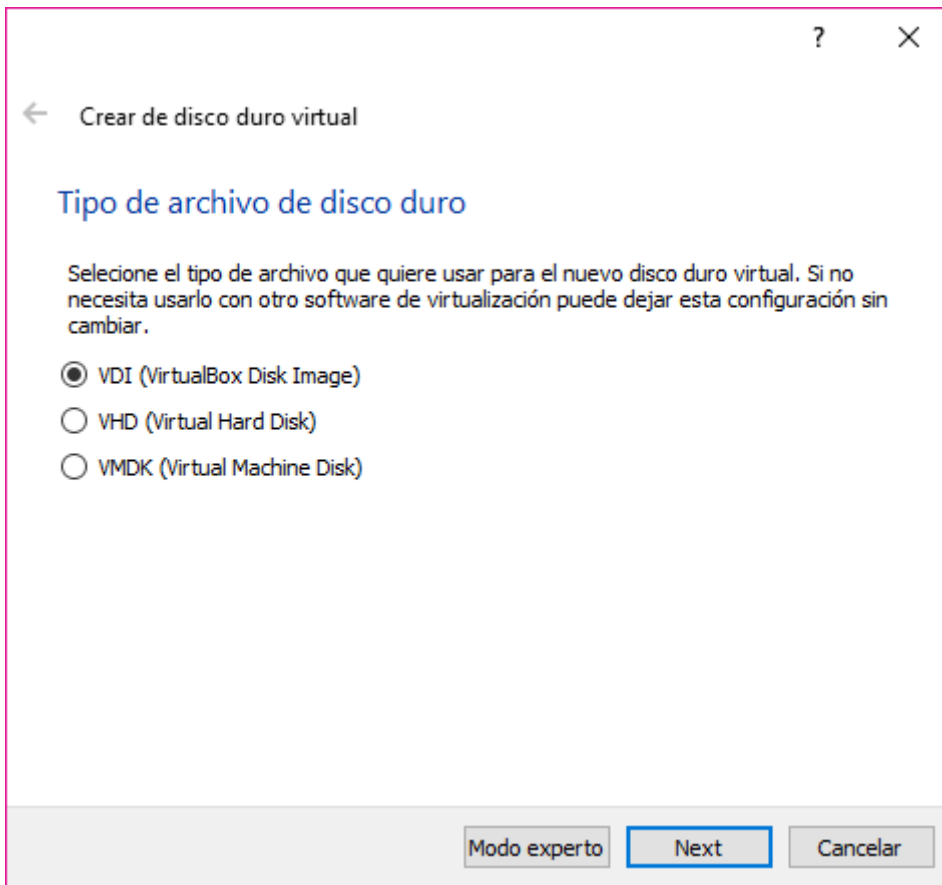
Aquí se debe de escribir un nombre para la máquina virtual, en este caso se dejó el por defecto, se elige la dirección de la carpeta donde se guardará la máquina virtual, en este caso se eligió la carpeta “UbuntuTFM” y el resto de las opciones se dejan como están. Se pulsa sobre “Next”:



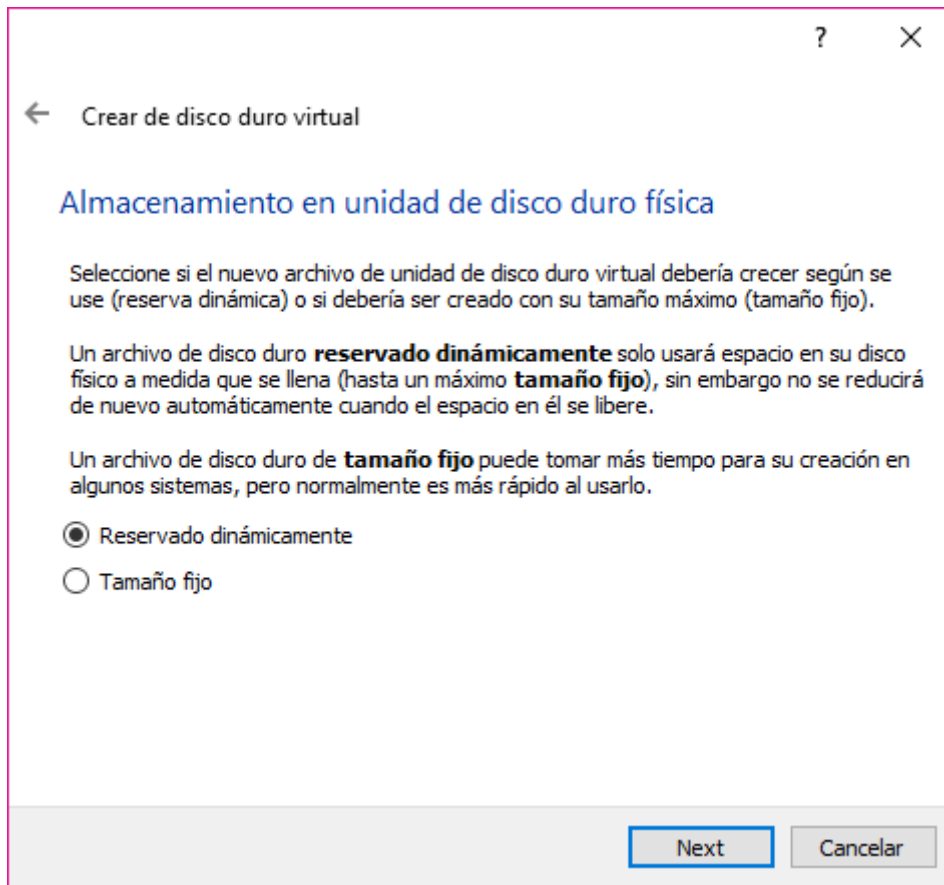
Se elige el tamaño de memoria RAM que se le quiere suministrar a la máquina virtual y se pulsa en “Next”:



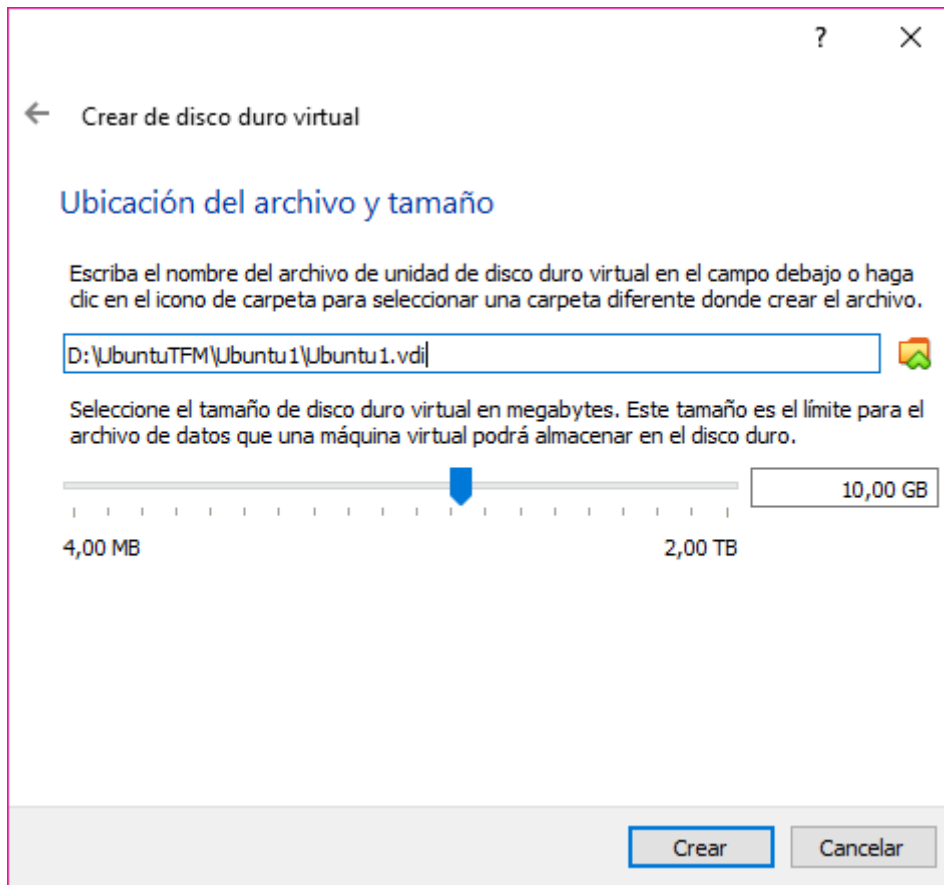
Se deja como está, ya que es la primera máquina creada y se pulsa en “Crear”:



Se elige el preferido, en este caso “VHI (VirtualBox Disk Image)” y se pulsa en “Next”:



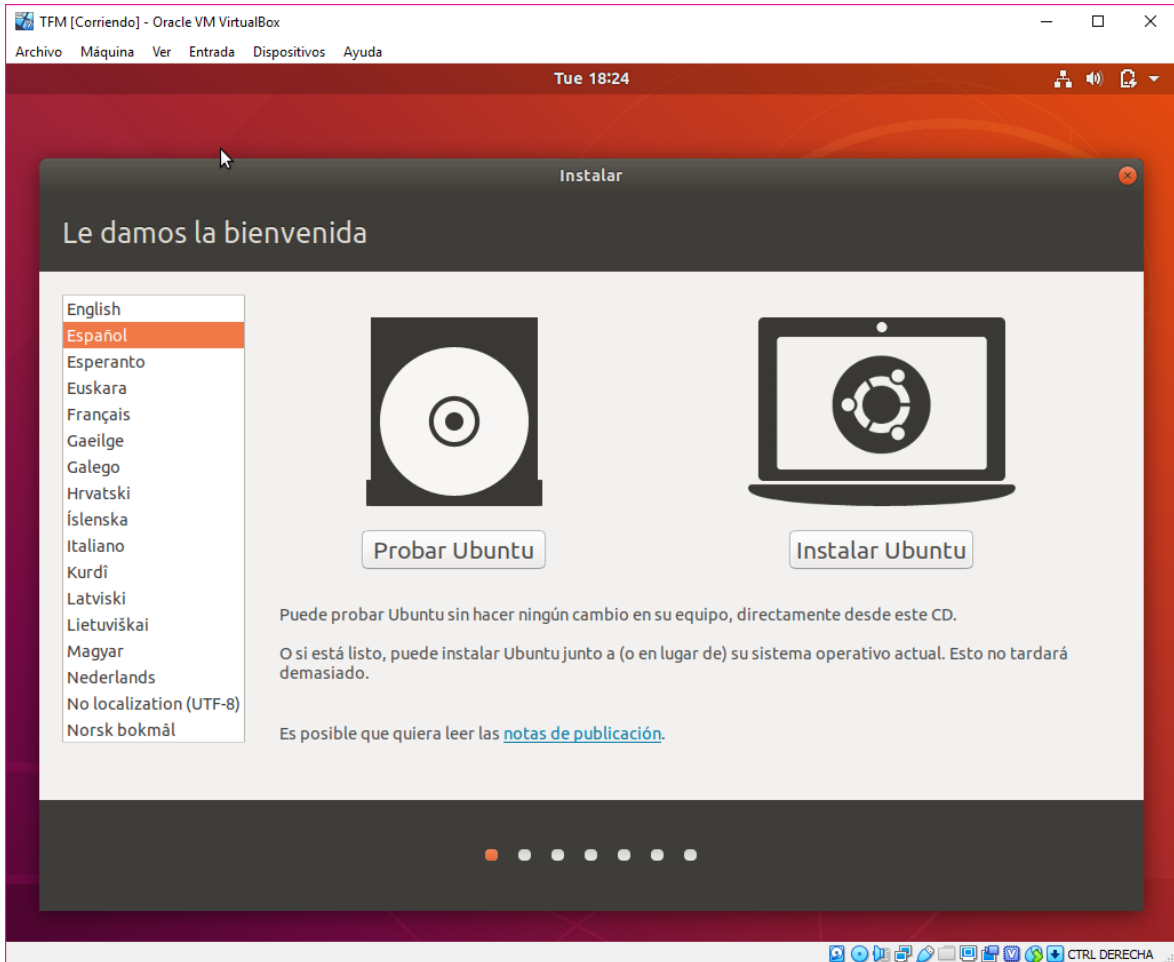
Dejar en “Reservado dinámicamente” y pulsar en “Next”:



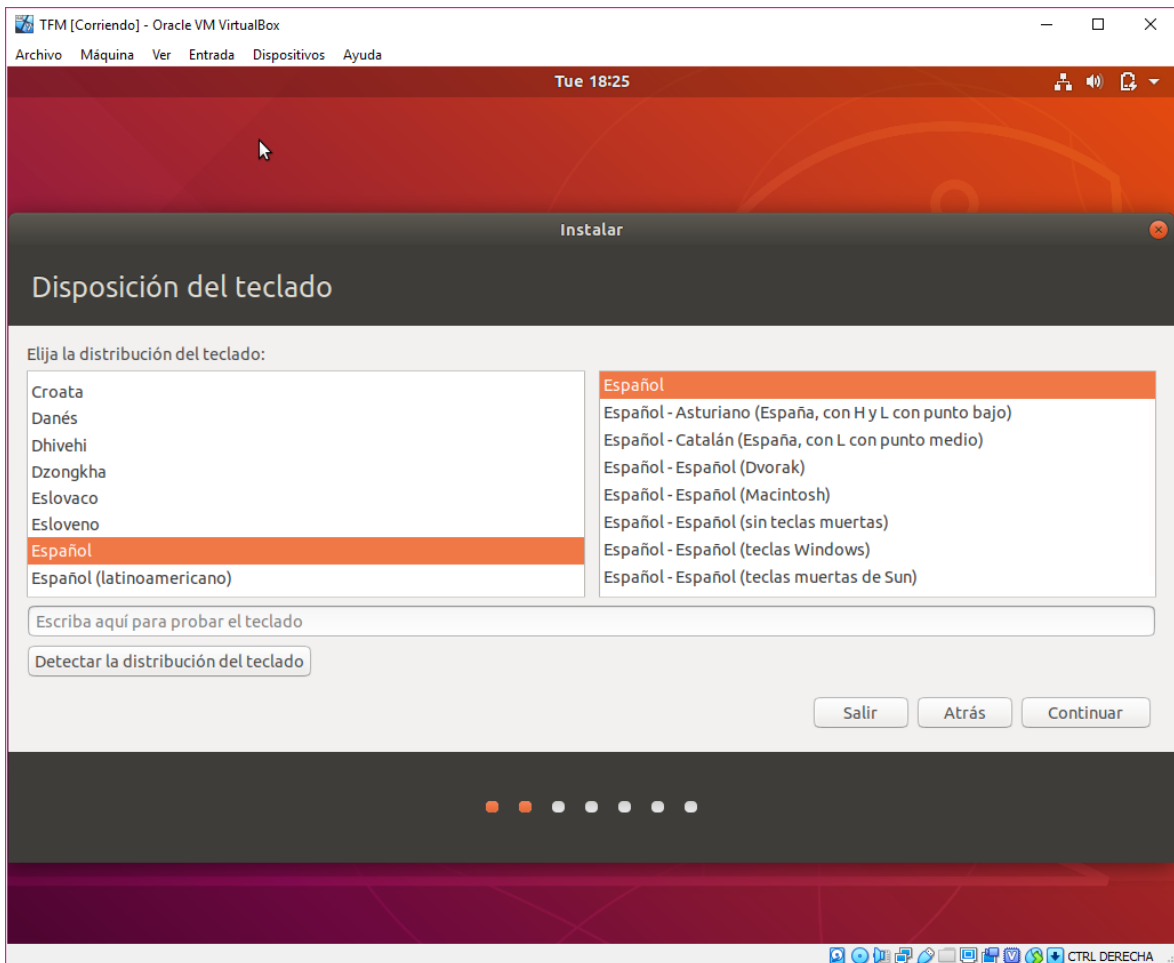
Se elige el tamaño de memoria física y la dirección de almacenamiento y se pulsa en “Crear”.

8.3 *INSTALACIÓN DE UBUNTU EN LA MÁQUINA VIRTUAL*

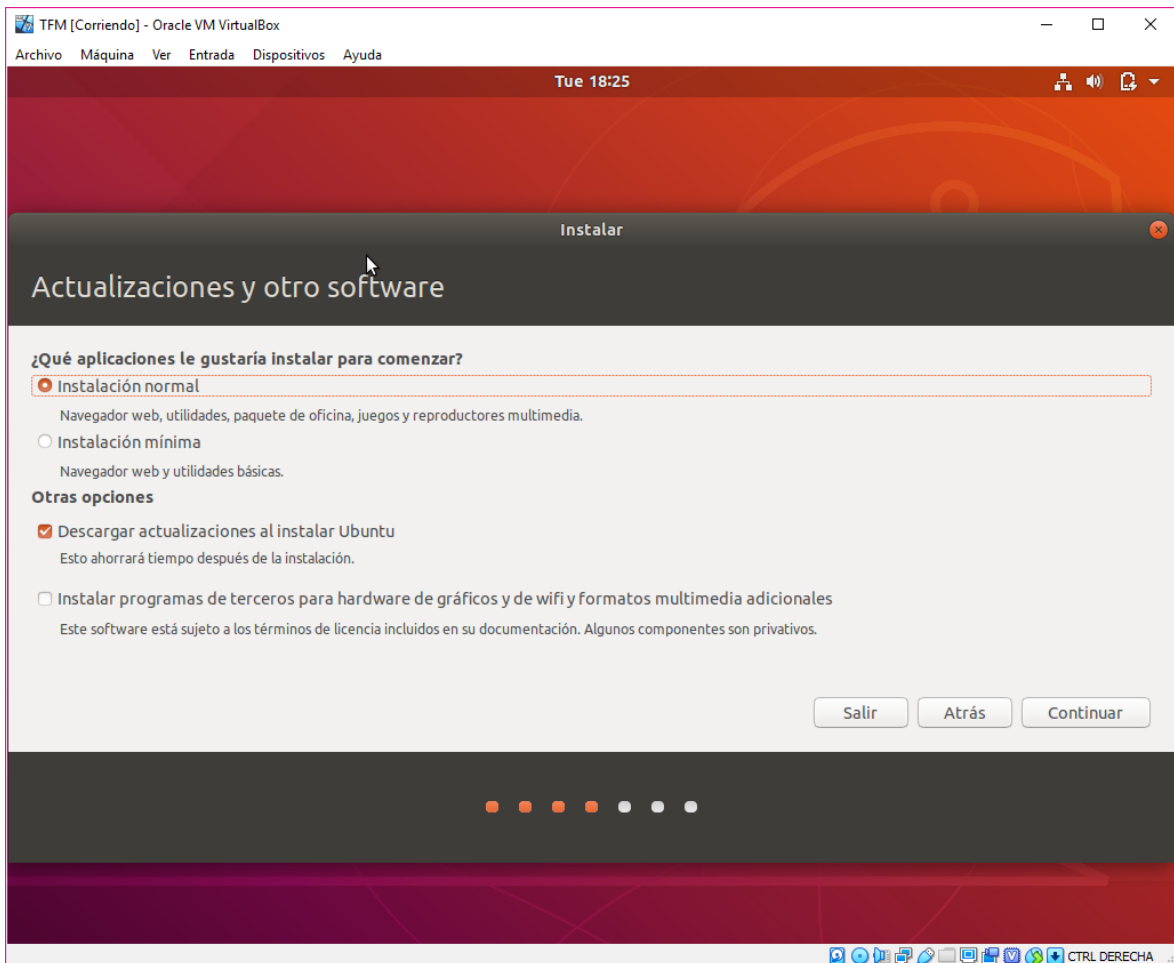
En este apartado se realizará la instalación del Sistema Operativo Ubuntu, en la máquina virtual creada. Para ello se inicia la máquina virtual y aparecerá la siguiente ventana:



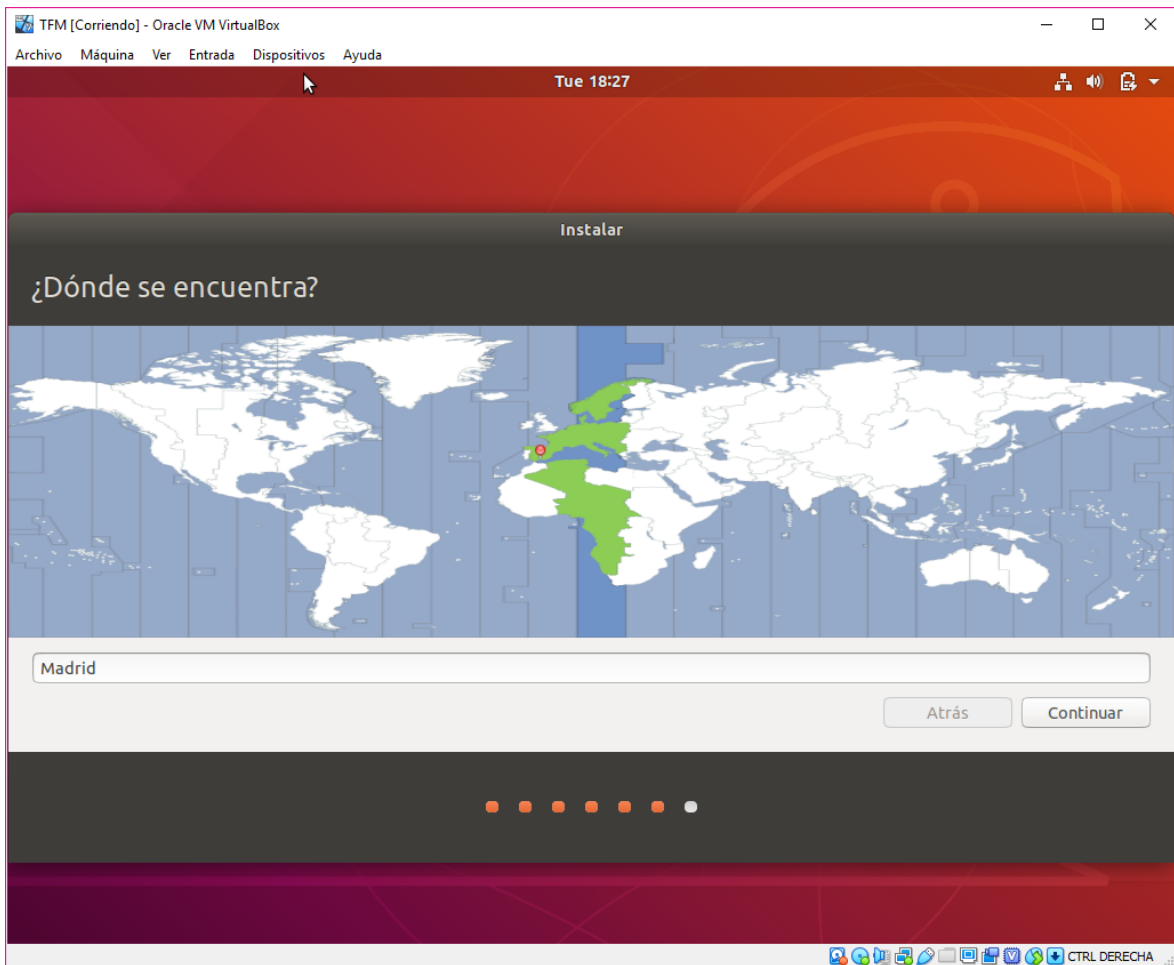
Se elige el idioma preferido, en este caso “Español”, y se pulsa sobre “Instalar Ubuntu”:



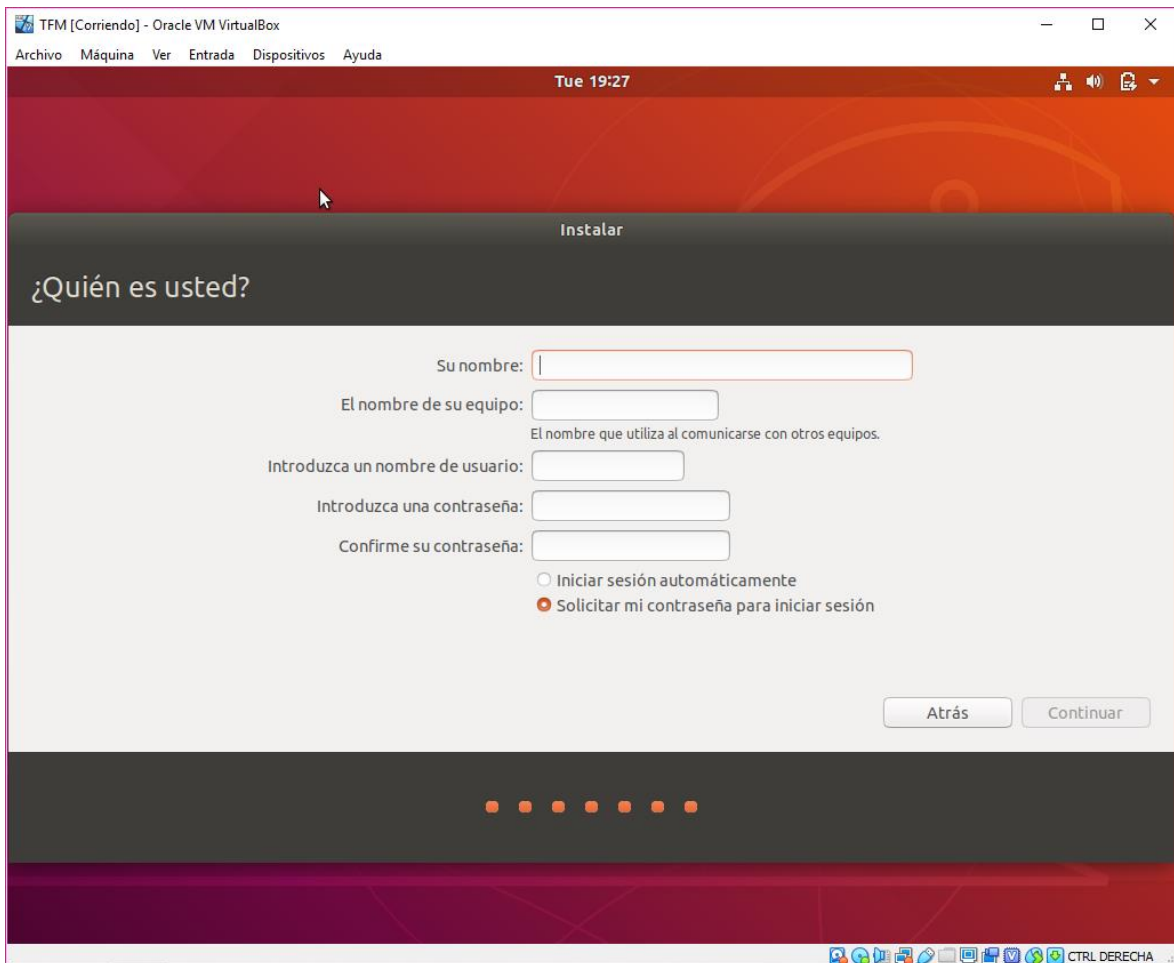
Se elige la disposición del teclado, en este caso “Español”, y se pulsa sobre “Continuar”:



Se elige el tipo de instalación que se quiere llevar a cabo, en este caso se empleará “instalación normal”, y se pulsa sobre “Continuar”:



Se elige la zona horaria, en este caso “Madrid”, y se pulsa en “Continuar”:



Finalmente, se rellenan los campos privados del usuario y se pulsa sobre “Continuar”. De esta manera se realiza la instalación completa de Ubuntu en la máquina virtual.

8.4 INSTALACIÓN DE NODE 8

Una vez se tiene el entorno preparado, hace falta comenzar a instalar las tecnologías necesarias, empezando por Node 8. Para ello se abre el terminal en Ubuntu y se actualiza el Ubuntu mediante el comando “sudo apt update”:

```
Archivo Editar Ver Buscar Terminal Ayuda
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

alfonso@alfonso-VirtualBox:~$ sudo apt update
[sudo] contraseña para alfonso:
```

A continuación se instala Node mediante el comando “sudo apt install nodejs”:

```
alfonso@alfonso-VirtualBox: ~
Archivo Editar Ver Buscar Terminal Ayuda
alfonso@alfonso-VirtualBox:~$ sudo apt install nodejs
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias
Leyendo la información de estado... Hecho
Se instalarán los siguientes paquetes adicionales:
 libc-ares2 libhttp-parser2.7.1 libuv1 nodejs-doc
Se instalarán los siguientes paquetes NUEVOS:
 libc-ares2 libhttp-parser2.7.1 libuv1 nodejs nodejs-doc
0 actualizados, 5 nuevos se instalarán, 0 para eliminar y 414 no actualizados.
Se necesita descargar 5.670 kB de archivos.
Se utilizarán 24,8 MB de espacio de disco adicional después de esta operación.
¿Desea continuar? [S/n] s
Des:1 http://es.archive.ubuntu.com/ubuntu bionic/main amd64 libuv1 amd64 1.18.0-3 [64,4 kB]
Des:2 http://es.archive.ubuntu.com/ubuntu bionic-updates/universe amd64 nodejs-doc all 8.10.0~dfsg-2ubuntu0.4 [752 kB]
Des:3 http://es.archive.ubuntu.com/ubuntu bionic/main amd64 libc-ares2 amd64 1.14.0-1 [37,1 kB]
Des:4 http://es.archive.ubuntu.com/ubuntu bionic/main amd64 libhttp-parser2.7.1 amd64 2.7.1-2 [20,6 kB]
Des:5 http://es.archive.ubuntu.com/ubuntu bionic-updates/universe amd64 nodejs amd64 8.10.0~dfsg-2ubuntu0.4 [4.796 kB]
Descargados 5.670 kB en 9s (646 kB/s)
Seleccionando el paquete libuv1:amd64 previamente no seleccionado.
```

Finalmente se comprueba la versión instalada mediante “nodejs -v”:

```
alfonso@alfonso-VirtualBox:~$ nodejs -v
v8.10.0
```

Se puede ver que es la versión “v8.10.0”.

8.5 INSTALACIÓN NPM

Una vez instalado Node, se procede a la instalación del gestor de archivos, NPM. Para ello, se escribe en el terminal de Ubuntu “sudo apt install npm”:

```
alfonso@alfonso-VirtualBox: ~
Archivo Editar Ver Buscar Terminal Ayuda
alfonso@alfonso-VirtualBox:~$ sudo apt install npm
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias
Leyendo la información de estado... Hecho
Se instalarán los siguientes paquetes adicionales:
build-essential cpp cpp-7 dpkg-dev fakeroot g++ g++-7 gcc gcc-7 gcc-7-base
gcc-8-base gyp javascript-common libalgorithm-diff-perl
libalgorithm-diff-xs-perl libalgorithm-merge-perl libasan4 libatomic1
libc-dev-bin libc6-dev libcc1-0 libcilkrts5 libdpkg-perl libfakeroot
libgcc-7-dev libgcc1 libgomp1 libitm1 libjs-async libjs-inherits
libjs-jquery libjs-node-uuid libjs-underscore liblsan0 libmpx2
libpython-stdlib libpython2.7 libpython2.7-minimal libpython2.7-stdlib
libquadmath0 libssl1.0-dev libssl1.0.0 libstdc++-7-dev libstdc++6 libtsan0
libubsan0 libuv1-dev linux-libc-dev make manpages-dev node-abbrev node-ansi
node-ansi-color-table node-archy node-async node-balanced-match
node-block-stream node-brace-expansion node-builtin-modules
node-combined-stream node-concat-map node-cookie-jar node-delayed-stream
node-forever-agent node-form-data node-fs.realpath node-fstream
node-fstream-ignore node-github-url-from-git node-glob node-graceful-fs
node-gyp node-hosted-git-info node-inflight node-inherits node-ini
node-is-builtin-module node-isexe node-json-stringify-safe node-lockfile
node-lru-cache node-mime node-minimatch node-mkdirp node-mute-stream
node-node-uuid node-nopt node-normalize-package-data node-npmlog node-once
node-osenv node-path-is-absolute node-pseudomap node-qs node-read
```

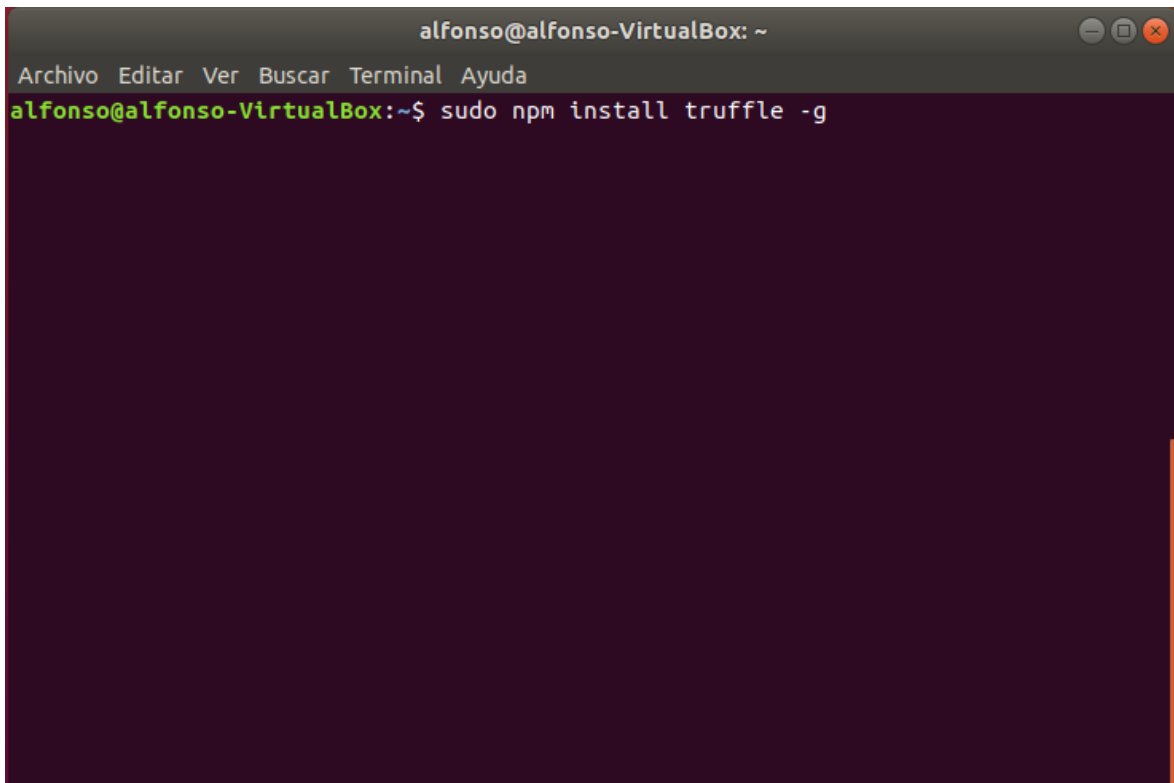
Finalmente se comprueba la versión instalada mediante “npm -v”:

```
alfonso@alfonso-VirtualBox:~$ npm -v
3.5.2
```

Se puede ver que la versión instalada es “3.5.2”.

8.6 INSTALACIÓN TRUFFLE

Una vez instalados Node y NPM, se procede a la instalación de Truffle. Este se instala mediante un terminal de Ubuntu mediante el comando “sudo npm install truffle -g”:



```
alfonso@alfonso-VirtualBox: ~  
Archivo Editar Ver Buscar Terminal Ayuda  
alfonso@alfonso-VirtualBox:~$ sudo npm install truffle -g
```

Una vez instalado, se comprueba la versión instalada mediante “truffle -v”:

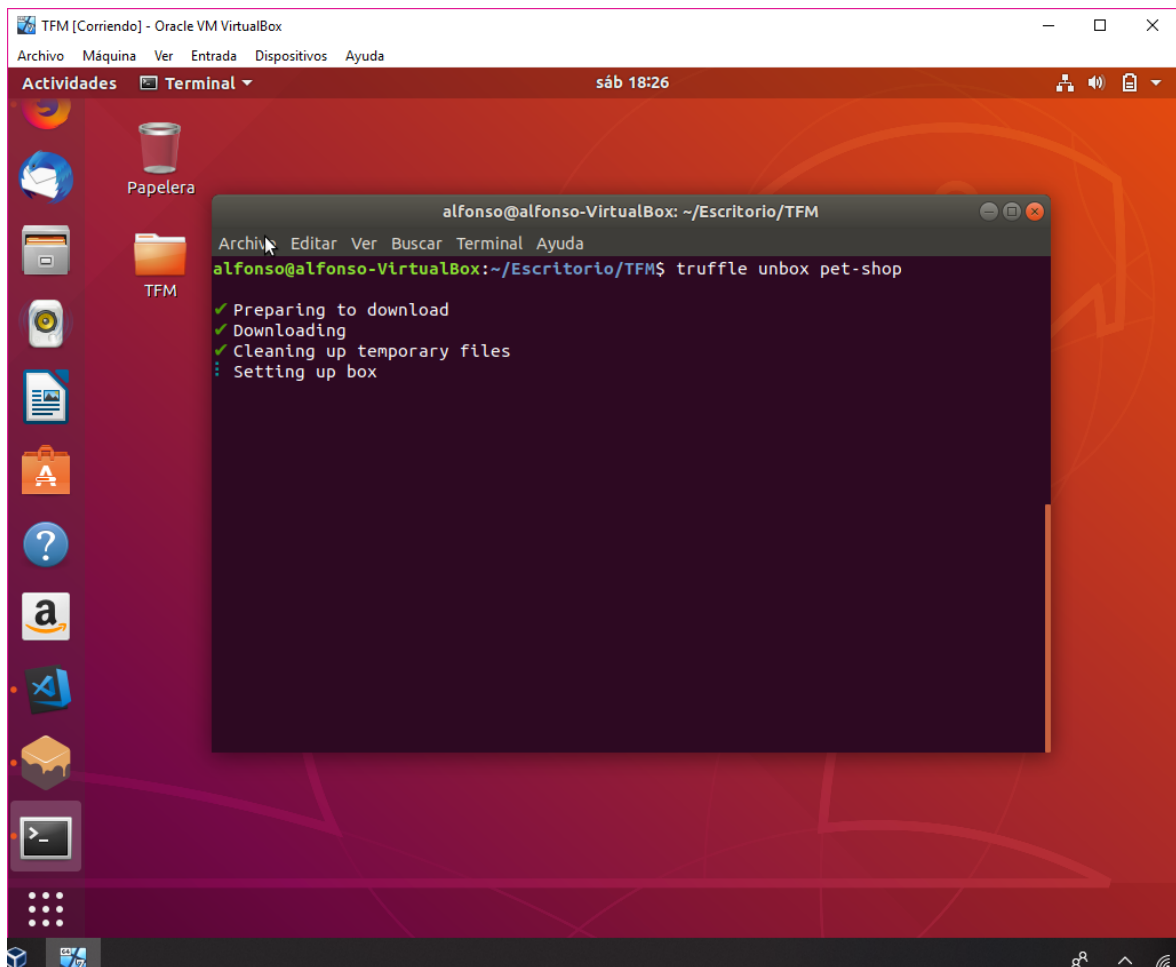


```
alfonso@alfonso-VirtualBox:~$ truffle -v  
Truffle v5.0.4 - a development framework for Ethereum  
  
Usage: truffle <command> [options]  
  
Commands:  
  build      Execute build pipeline (if configuration present)  
  compile    Compile contract source files  
  config     Set user-level configuration options  
  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
```

Se puede ver que la versión instalada es “v5.0.4”.

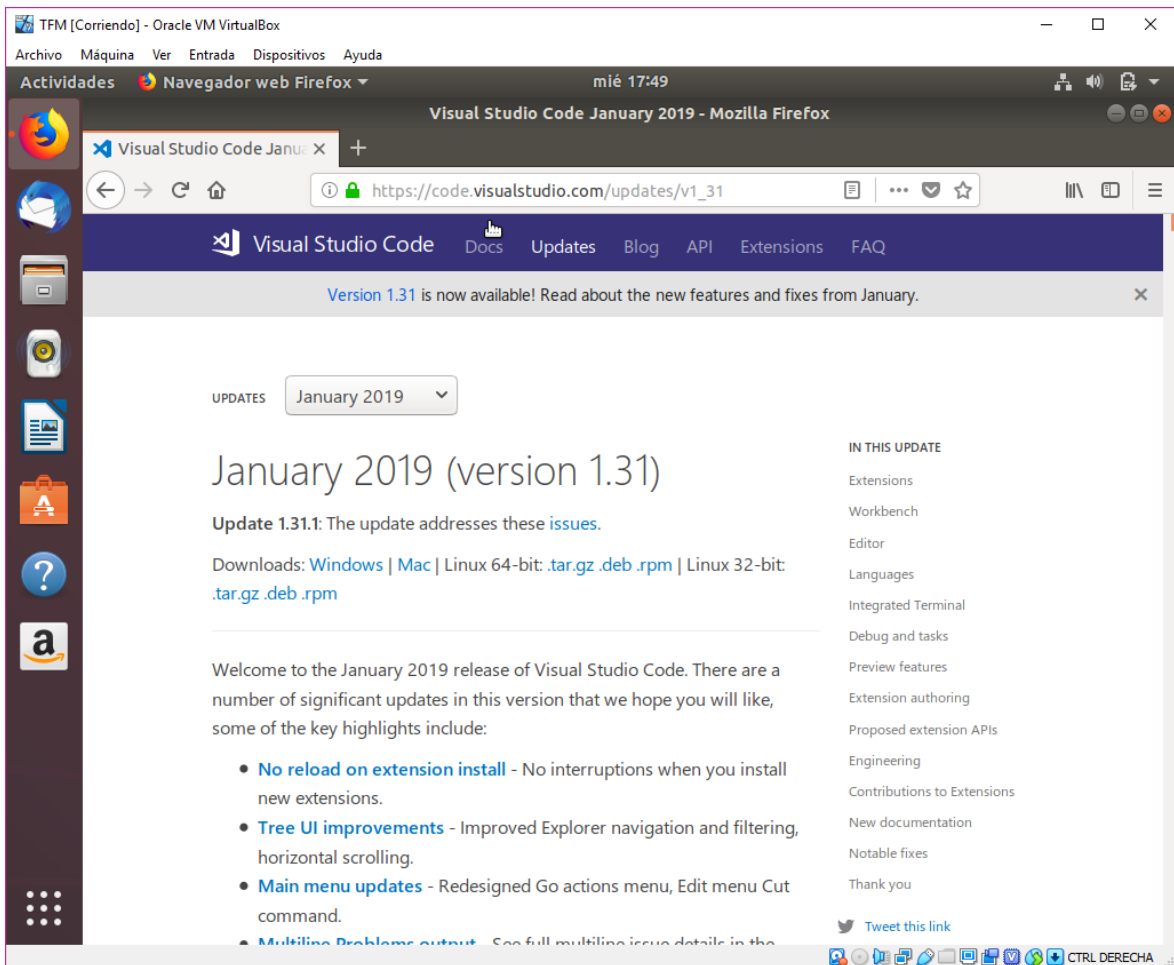
8.7 *INSTALACIÓN PROYECTO EN TRUFFLE*

Una vez instalado truffle, es necesario iniciar un proyecto para que se creen los archivos necesarios, de manera automática. Para ello, truffle suministra un proyecto de ejemplo, que se puede instalar mediante el comando “truffle unbox pet-shop”:

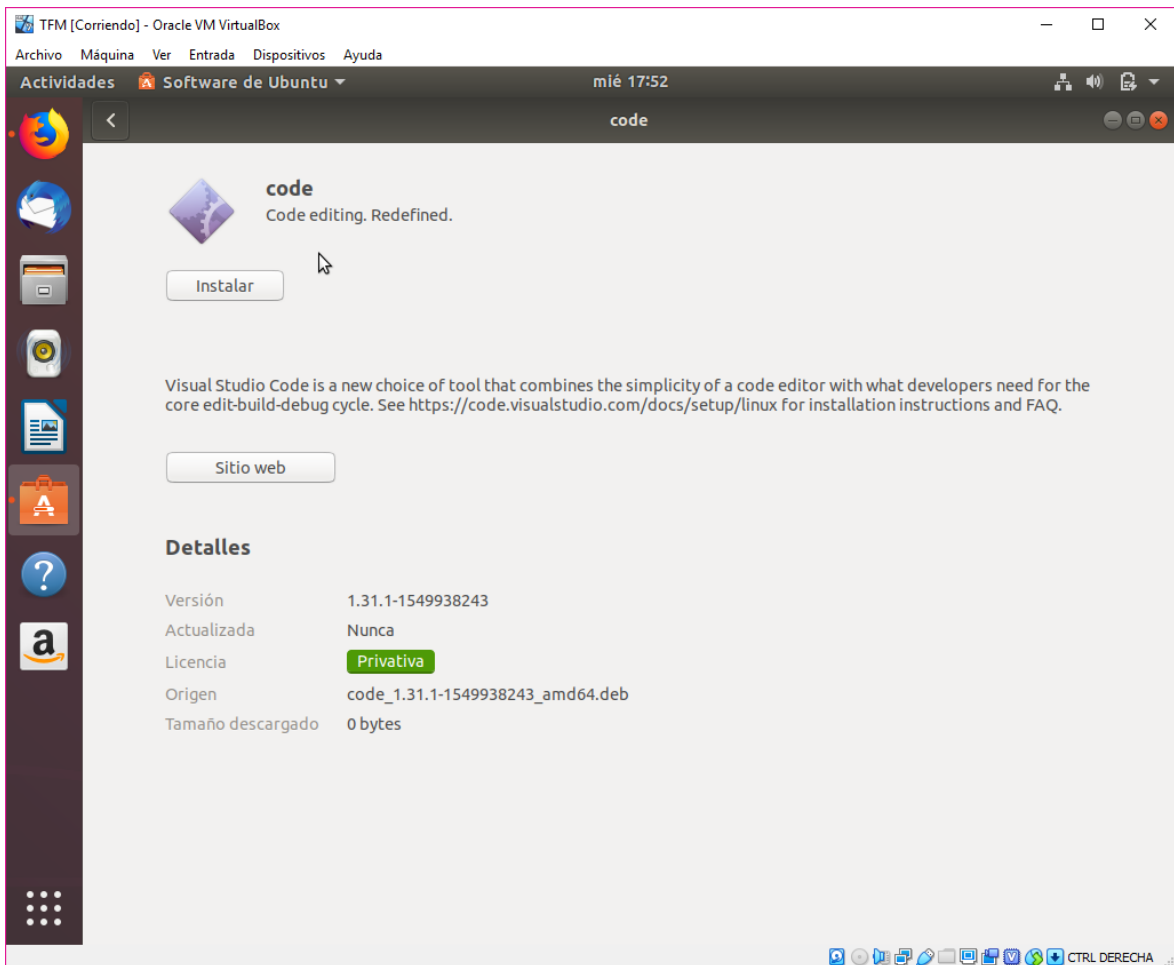


8.8 *INSTALACIÓN VISUAL STUDIO CODE*

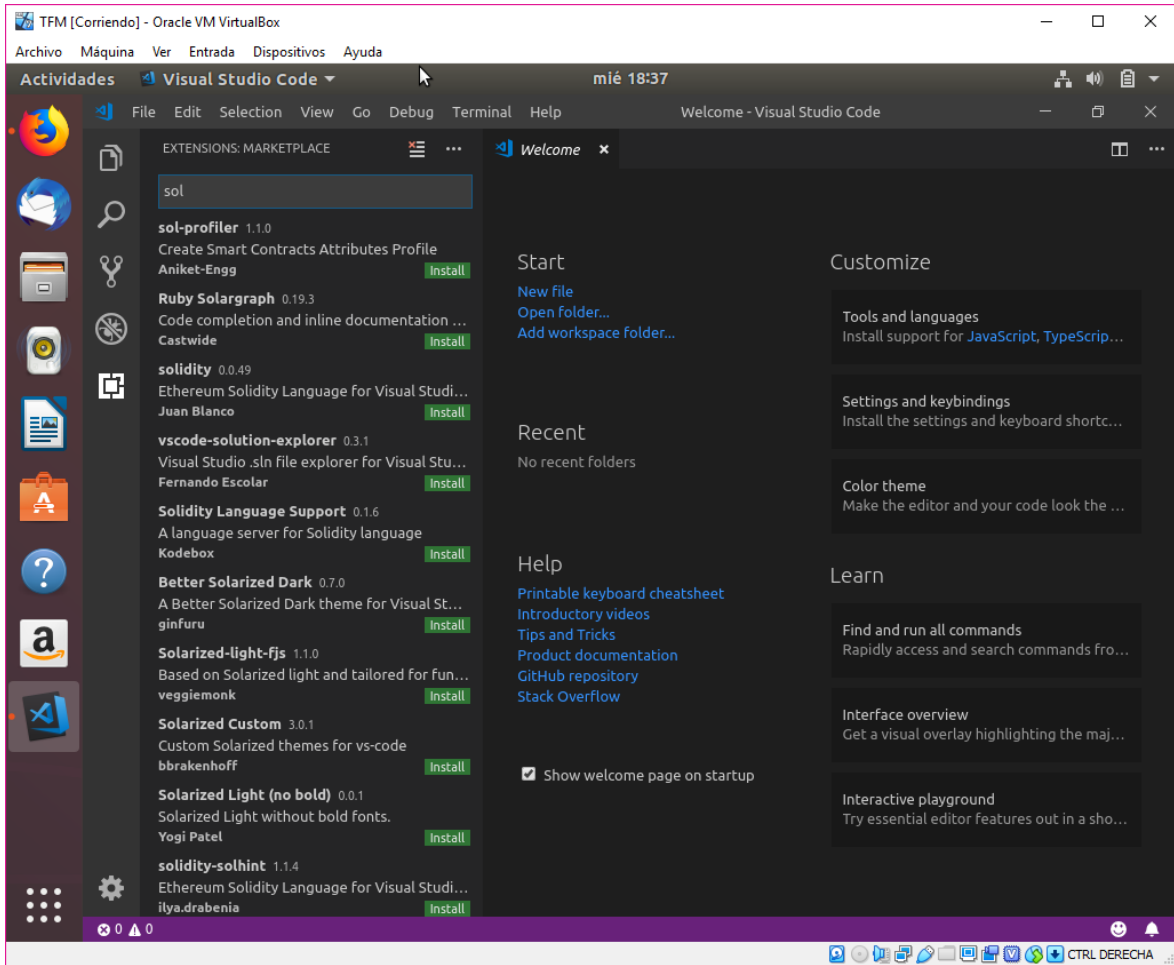
Una vez se ha realizado la instalación del proyecto, es necesaria la instalación del IDE que se va a emplear para la programación del proyecto. Para ello, se entra en la página oficial de Visual Studio Code ^[8] y se va al apartado de descargas:



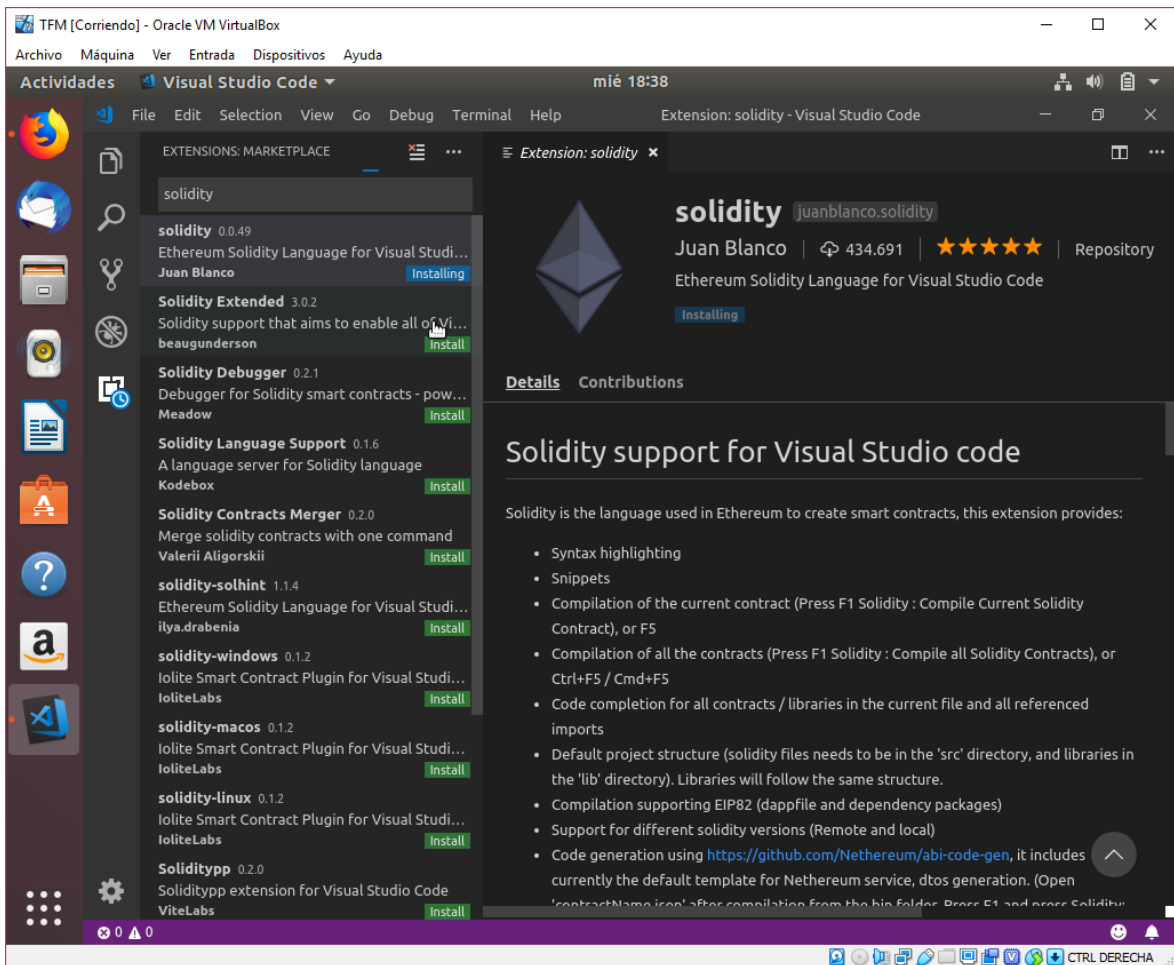
Se elige la última versión y se pulsa sobre “.tar.gz”. Una vez finalizada la descarga, se ejecuta y se abrirá la siguiente ventana:



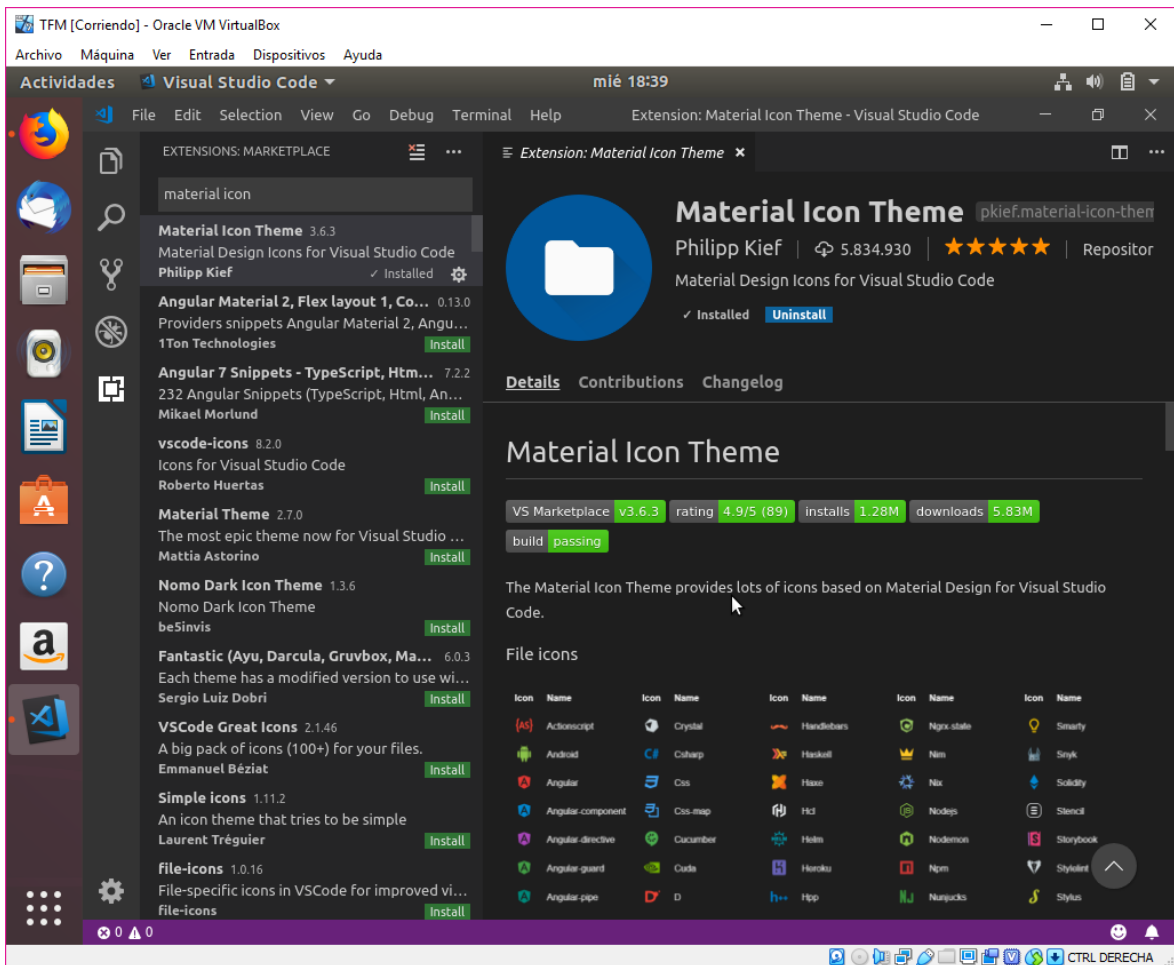
Pulsar sobre “Instalar” y se instalará de manera satisfactoria. Una vez se finalice la instalación se ejecuta Visual Studio Code y se pulsa sobre el apartado “Extensiones” para instalar lo necesario para que ayude con la programación de Solidity. Se busca la extensión “Solidity”:



Y se pulsa sobre “Install”:



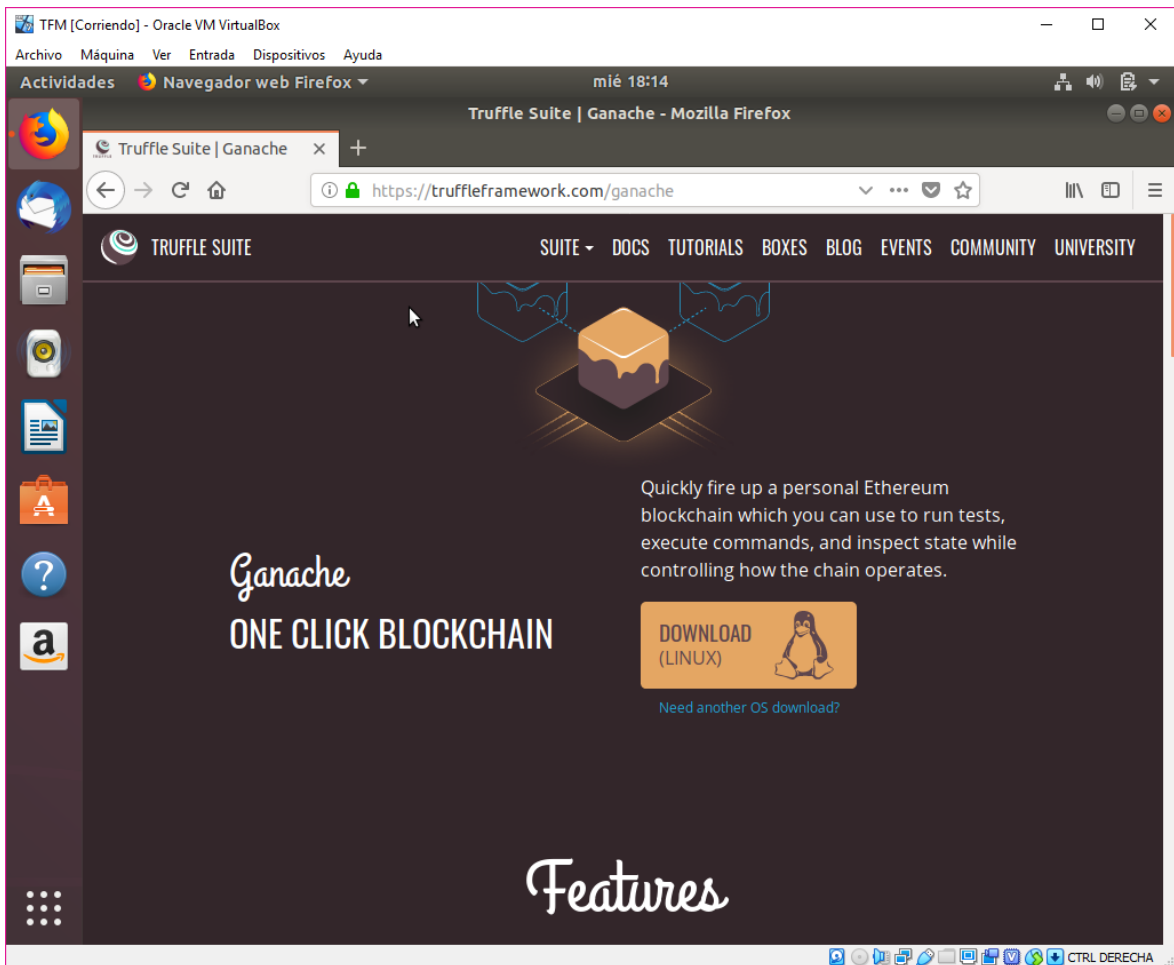
Una vez instalado, se busca la extensión “Material Icon Theme” y se pulsa sobre “Install”:



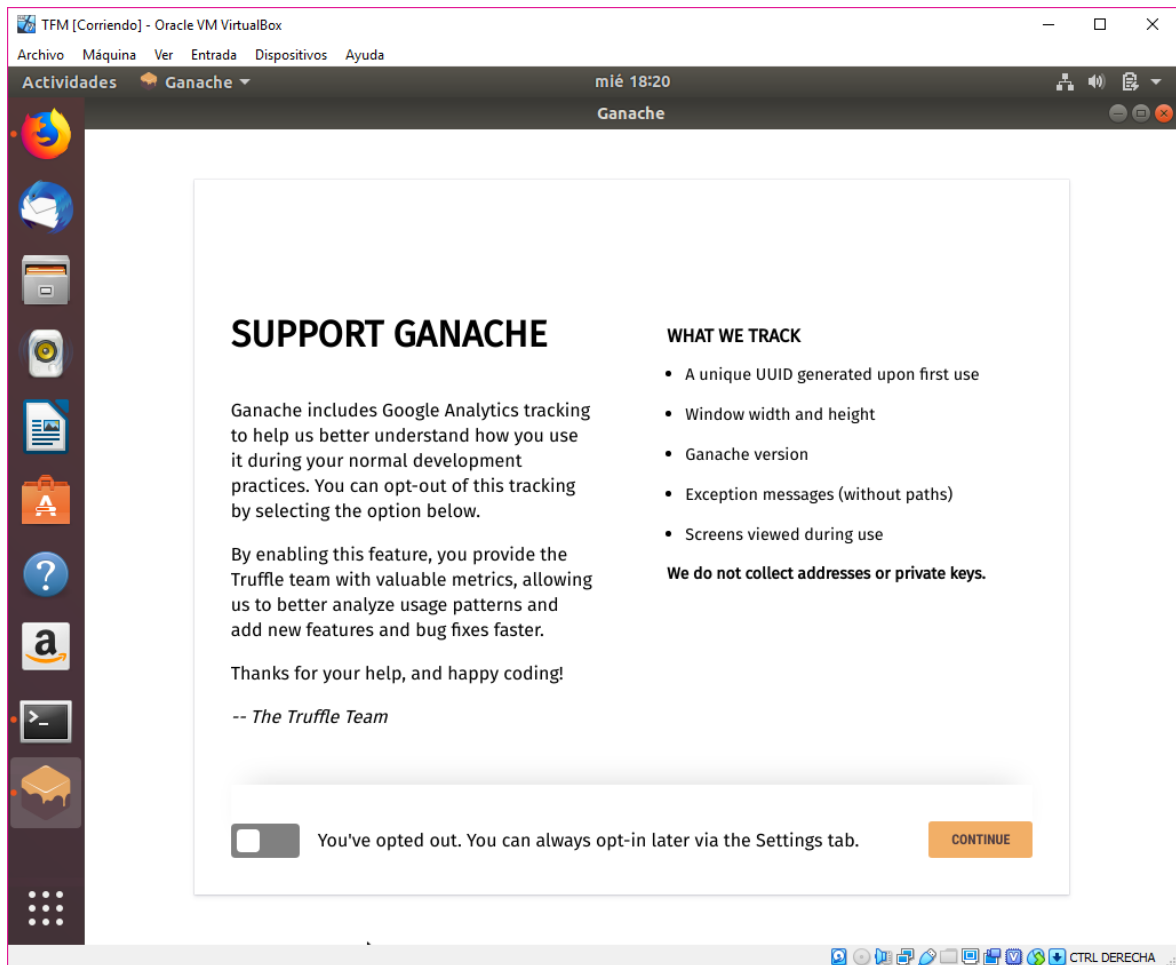
Esto permitirá realizar la programación de una manera más sencilla e intuitiva.

8.9 *INSTALACIÓN GANACHE*

Una vez instalado todo lo necesario para la programación, es necesario instalar la red privada de Ethereum para las pruebas. Para ello es necesario entrar en la página web oficial de Ganache [2] :



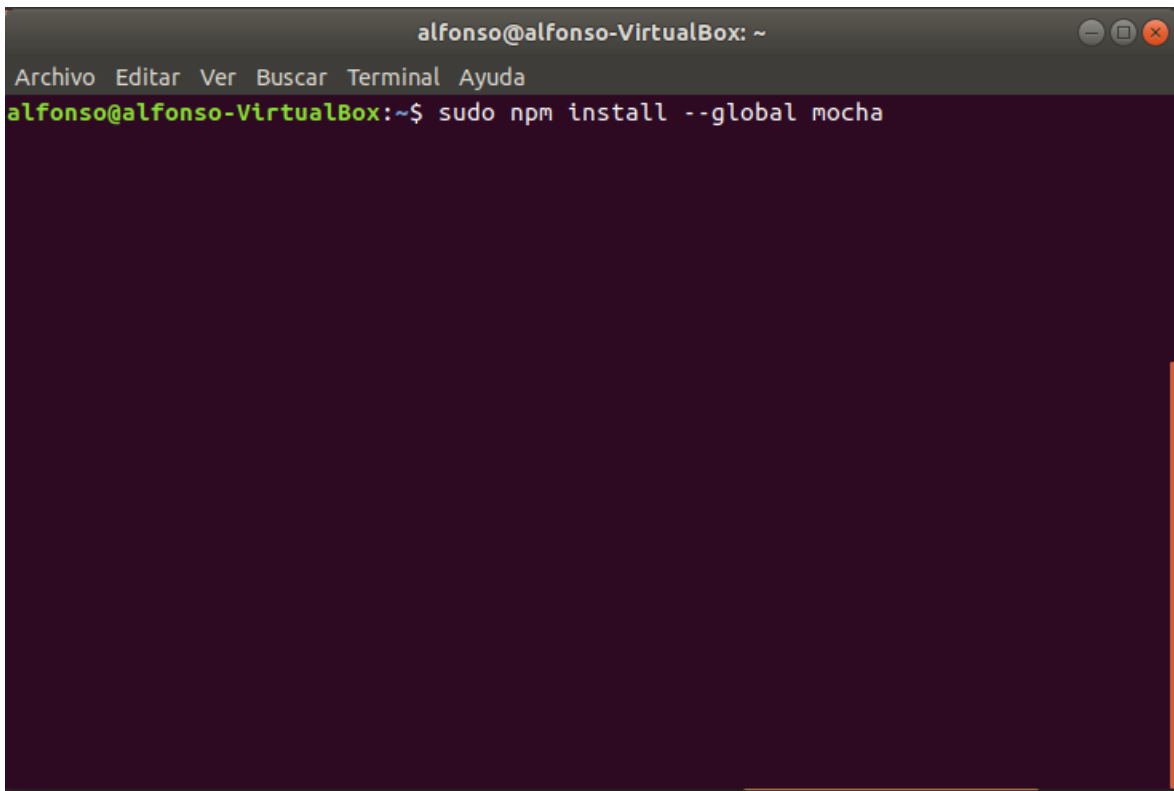
Pulsar sobre “DOWNLOAD”:



Una vez finalizada la descarga, se ejecuta el programa y se da a continuar. La instalación finalizará de manera automática.

8.10 INSTALACIÓN DE MOCHA

Una vez todo esté listo, será necesario instalar Mocha para las pruebas sobre el proyecto. Para ello debe abrirse un terminal de Ubuntu y ejecutar el comando “sudo npm install – global mocha”:

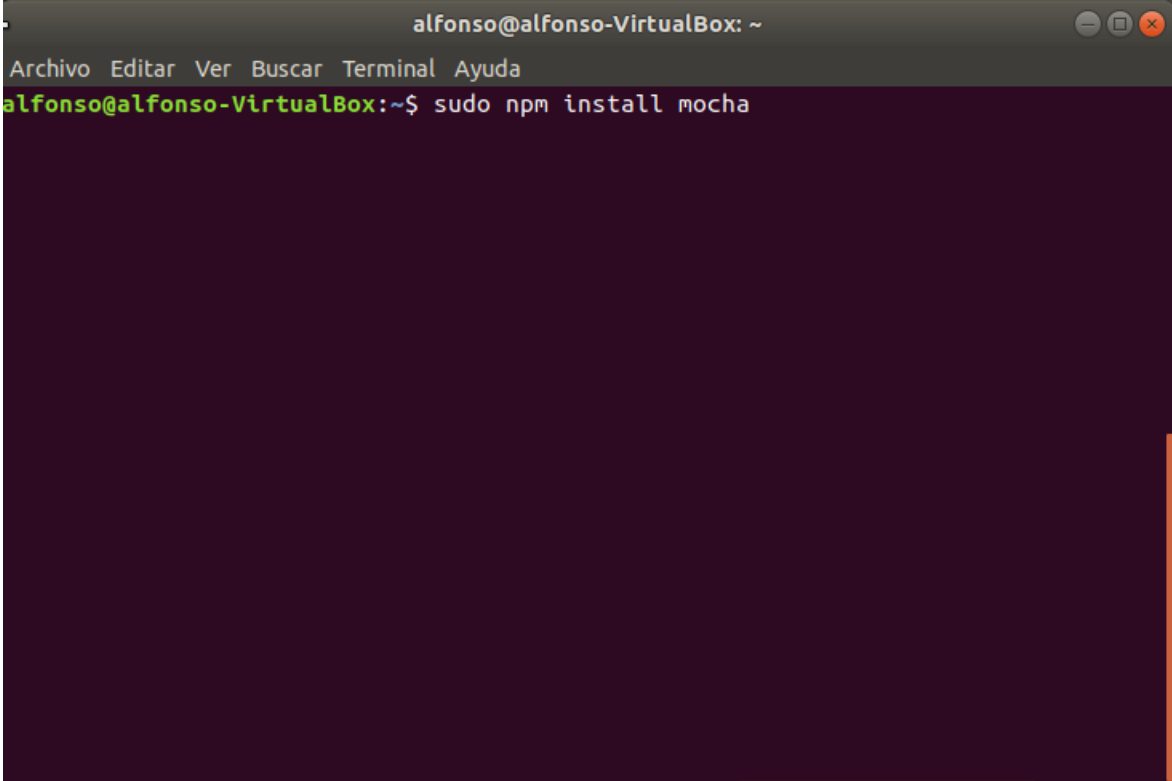


```
alfonso@alfonso-VirtualBox: ~  
Archivo Editar Ver Buscar Terminal Ayuda  
alfonso@alfonso-VirtualBox:~$ sudo npm install --global mocha
```

A continuación se ejecuta el comando “sudo npm install –save-dev mocha”:

```
alfonso@alfonso-VirtualBox: ~  
Archivo Editar Ver Buscar Terminal Ayuda  
├── p-finally@1.0.0  
├── signal-exit@3.0.2  
├── strip-eof@1.0.0  
├── lcid@2.0.0  
├── invert-kv@2.0.0  
├── mem@4.1.0  
├── map-age-cleaner@0.1.3  
├── p-defer@1.0.0  
├── mimic-fn@1.2.0  
├── p-is-promise@2.0.0  
├── require-directory@2.1.1  
├── require-main-filename@1.0.1  
├── set-blocking@2.0.0  
├── which-module@2.0.0  
├── y18n@4.0.0  
├── yargs-parser@11.1.1  
├── camelcase@5.0.0  
├── yargs-unparser@1.5.0  
├── flat@4.1.0  
├── is-buffer@2.0.3  
└── lodash@4.17.11  
  
alfonso@alfonso-VirtualBox:~$ sudo npm install --save-dev mocha
```

Finalmente se instala mocha mediante el comando “sudo npm install mocha”:



```
alfonso@alfonso-VirtualBox: ~  
Archivo Editar Ver Buscar Terminal Ayuda  
alfonso@alfonso-VirtualBox:~$ sudo npm install mocha
```

Una vez realizado todo este proceso, el entorno ya estará listo para la programación del proyecto.

ANEXO B: MANUAL DE USUARIO

En el presente anexo tratará de realizarse una explicación, lo más detallada posible, del funcionamiento de la plataforma.

8.1 PUESTA EN MARCHA Y VISTA INICIAL

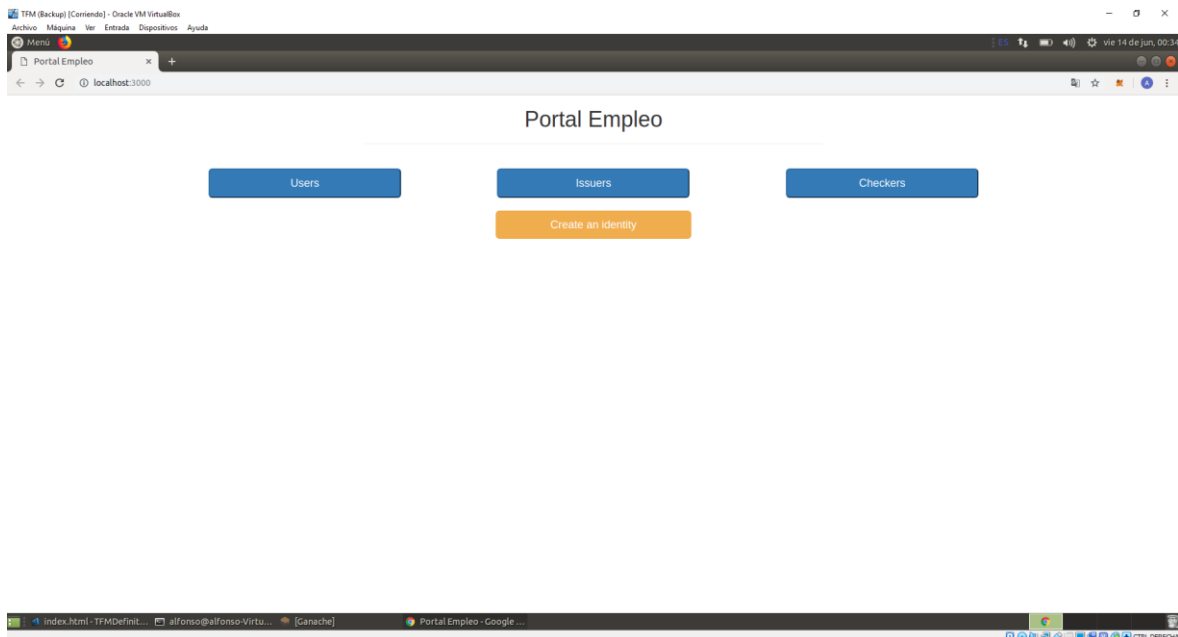
Para el lanzamiento de la aplicación, primero es necesario escribir el código siguiente en la terminal:

```
alfonso@alfonso-VirtualBox:~/Escritorio/TFMDefinitivo$ truffle migrate --reset
```

De este modo se realizará la migración de los contratos a la red Ethereum, a la que se esté conectados, para que puedan ser llamados desde el servidor. A continuación, se arrancará el servidor a través del siguiente comando:

```
alfonso@alfonso-VirtualBox:~/Escritorio/TFMDefinitivo$ npm run dev
```

Una vez arrancado el servidor, se abrirá una ventana con, del explorador predeterminado, mostrando el siguiente contenido:

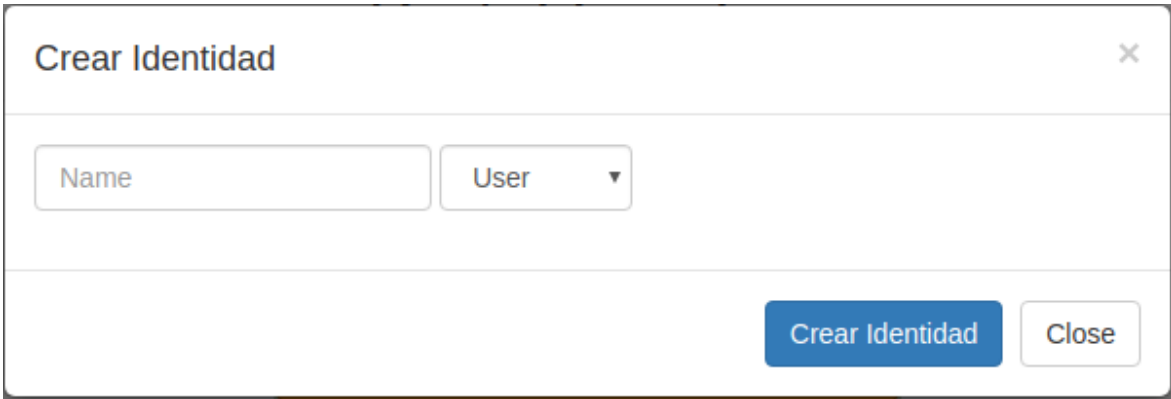


Como se puede observar, hay cuatro botones:

- Users: que será el apartado donde se mostrarán los Users.
- Issuers: que será el apartado donde se mostrarán los Issuers.
- Checkers: que será el apartado donde se mostrarán los Checkers.
- Create an identity: botón que abrirá un modal para crear una identidad.

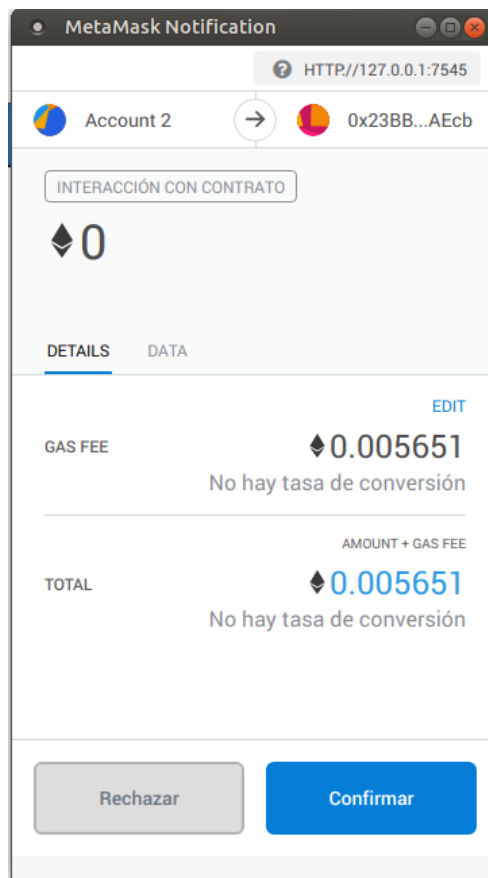
8.2 CREAR UN ISSUER

El primer paso, por lo tanto, será crear una identidad del tipo Issuer. Para ello, pulsaremos sobre “Create an identity” y se abrirá la siguiente ventana:

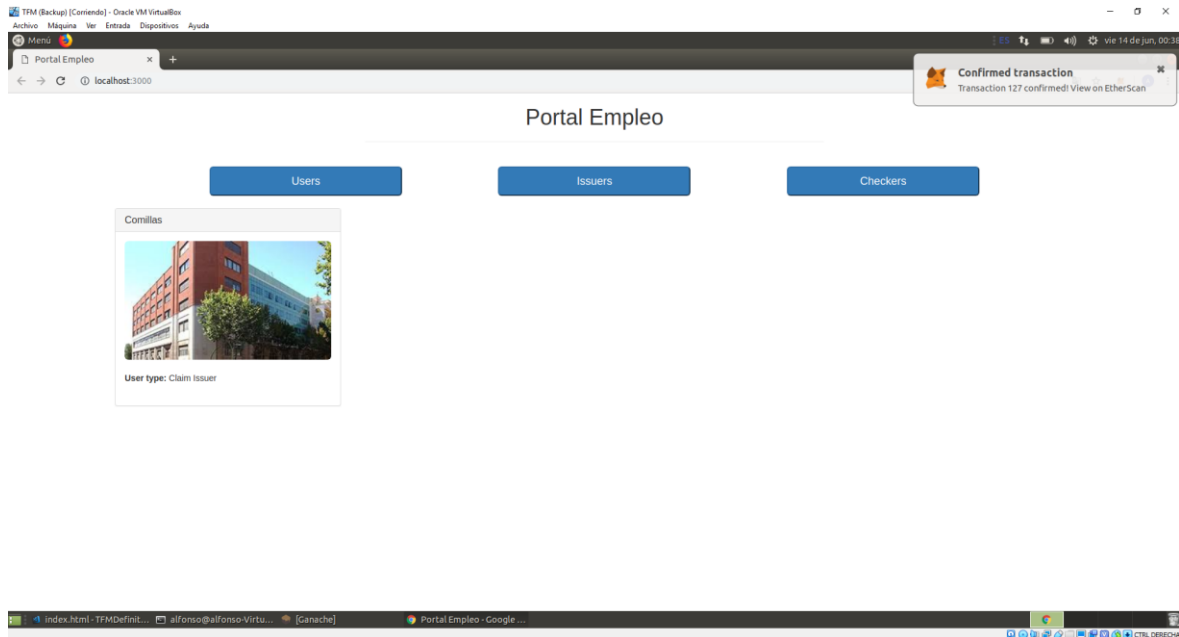


Sobre esta identidad se rellena con el nombre “Comillas” y el tipo “Issuer” y se pulsa sobre “Crear una Identidad”.

Se confirma la transacción de MetaMask:

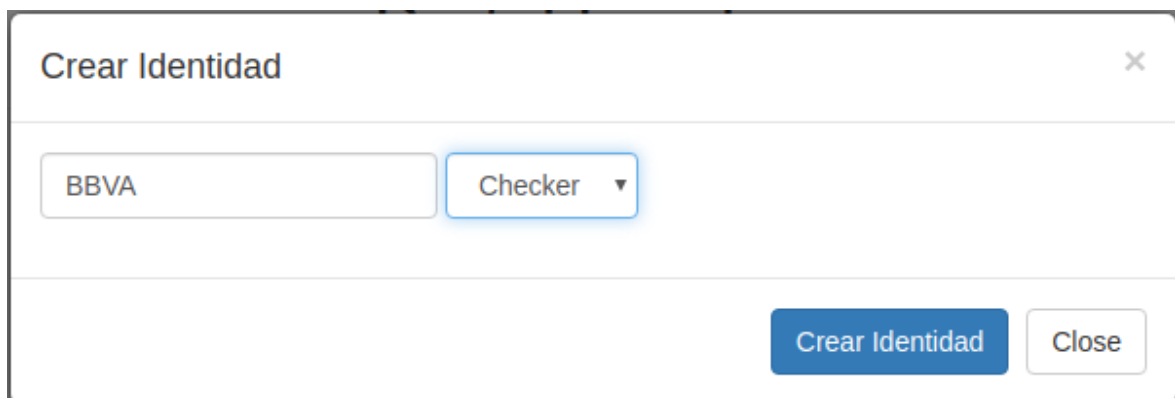


La página, por tanto, volverá a cargarse mostrando las identidades creadas, la cual es únicamente el Issuer “Comillas”:

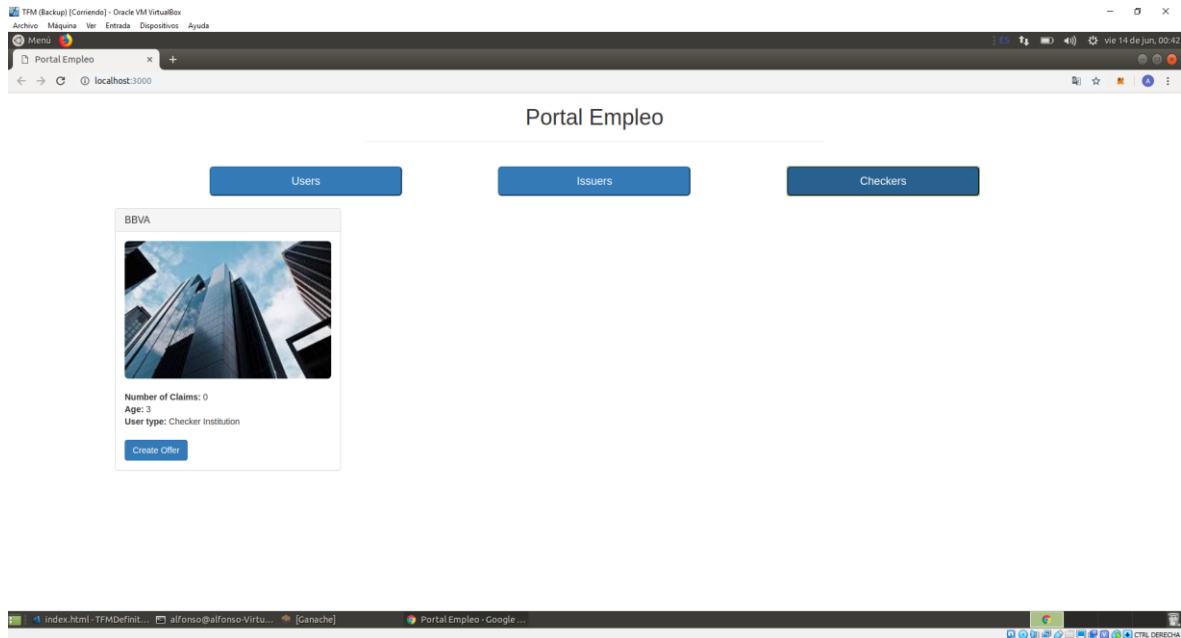


8.3 *CREAR UN CHECKER*

Una vez que el Issuer ha sido creado, se procede a la creación de un Checker. Para ello, se pulsa sobre MetaMask y se cambia de cuenta. Una vez se ha cambiado de cuenta, se recarga la página y se pulsa sobre “Create an identity” de nuevo. Se desplegará un nuevo modal en el que se ingresen los siguientes valores:



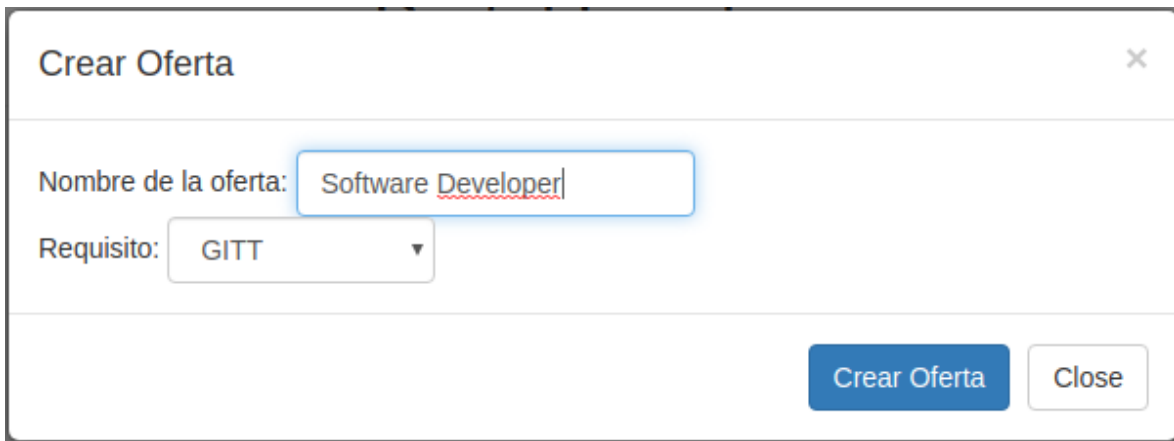
Se pulsa sobre “Crear Identidad” y el resultado es el siguiente:



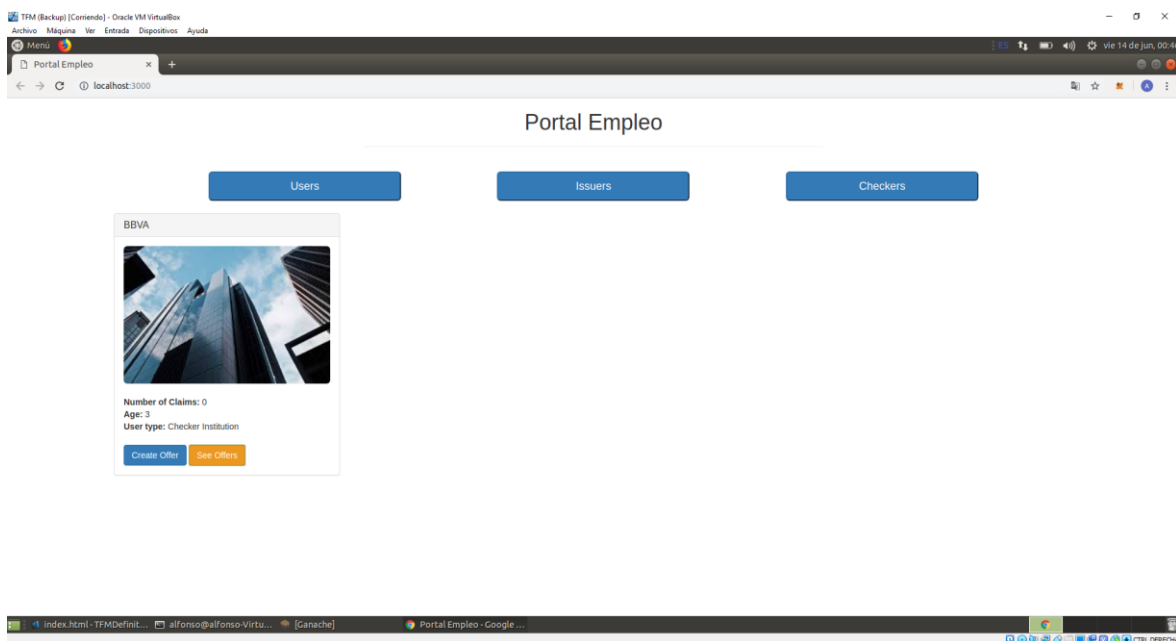
Si se pulsase sobre el botón “Issuers” se vería el Issuer “Comillas”, y al pulsar sobre “Checkers” se ve el Checker “BBVA”.

8.4 CREACIÓN DE UNA OFERTA

El siguiente paso es la creación de una oferta de trabajo para que el futuro alumno pueda aplicar. Para ello se pulsa sobre “Create Offer” en el Checker “BBVA” y se rellena, el modal emergente, con los siguientes campos:



En especial, esta oferta de trabajo se llamará “Software Developer” y tendrá como requisito el título de “GIT”. Se pulsa sobre “Crear Oferta” y el resultado es el siguiente:



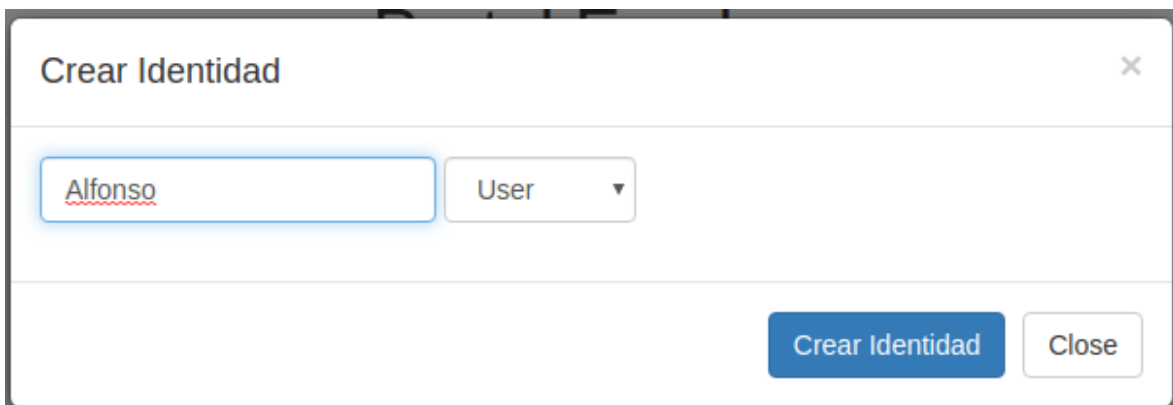
Se deshabilita un botón llamado “See Offers” en el Checker “BBVA”, el cual, al pulsar sobre él se muestra la siguiente información:



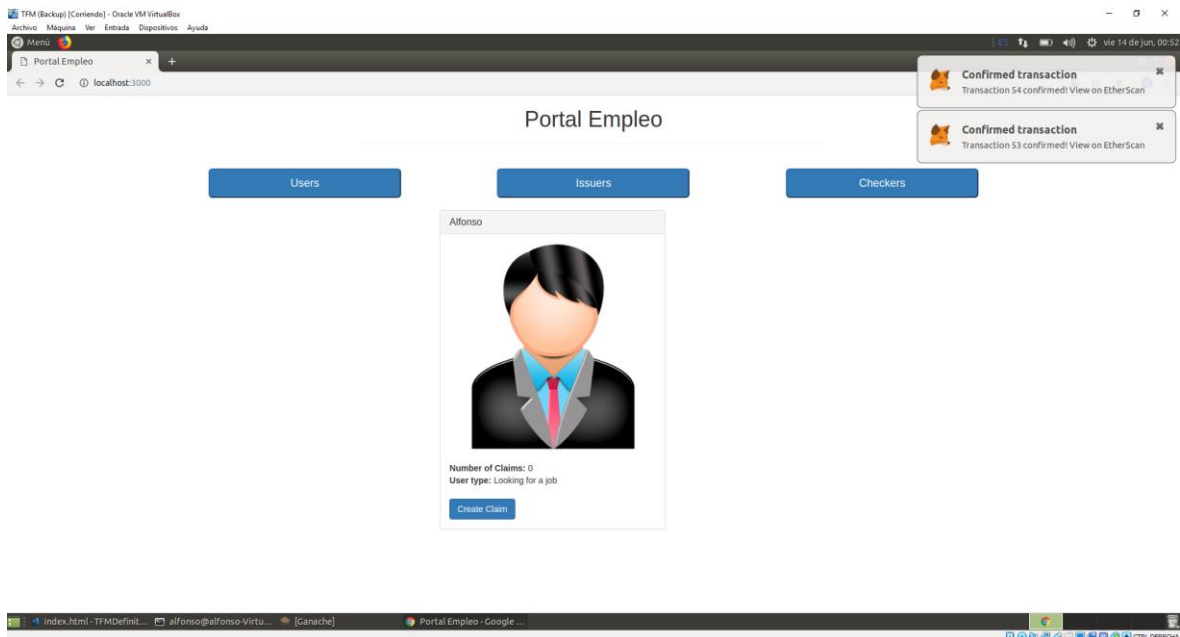
Esto es que hay una oferta con el nombre “Software Developer”, con el requisito “GITT” y a la que nadie ha aplicado, por el momento.

8.5 CREAR UN USER

La última identidad que se va a crear es un User. Como para la creación del Checker, será necesario cambiar de cuenta a través de MetaMask. Una vez realizado el cambio, será necesario volver a pulsar sobre “Create an identity” y en el modal emergente introducir la siguiente información:



Esto creará una identidad del tipo “User” con nombre “Alfonso”. Pulsar sobre “Crear Identidad” y aceptar la transacción de MetaMask. El resultado será el siguiente:



Se ha creado una identidad del tipo User y con nombre “Alfonso”. Igual que en el caso de la creación del Checker, si se pulsa sobre “Issuers” aparecerá el Issuer con nombre “Comillas” y si se pulsa sobre “Checkers” aparecerá el Checker con nombre “BBVA”.

8.6 CREACIÓN DE UNA CLAIM

El siguiente paso será la creación de una claim. Para ello se pulsa sobre el botón “Create claim” y se abrirá el siguiente modal:

En el primer desplegable se elige el título que se quiere certificar y, en el segundo, el Issuer, de todos los Issuers creados, contra el que se quiere validar. En este caso se deja todo igual y se pulsa sobre “Crear Claim” y se acepta la transacción de MetaMask. El resultado es el siguiente:



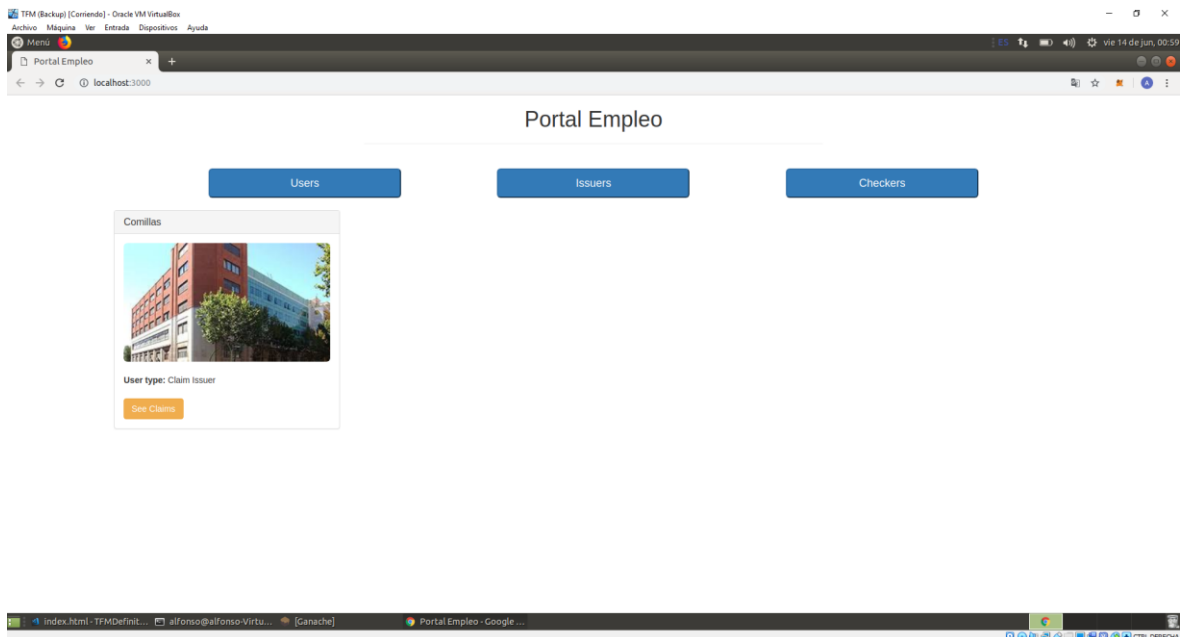
Se ha desbloqueado un botón que pone “See Claims” el cual si es pulsado aparece el siguiente modal:



Esto suministra la información de que el usuario tiene una Claim creada con el nombre “GITT”, nombre del título, contra el Issuer “Comillas” y que la Claim aún no ha sido validada.

8.7 VALIDACIÓN DE LA CLAIM

Para poder aplicar a la oferta será necesario la validación del claim, por parte del Issuer “Comillas”. Para ello se cambia, a través del MetaMask, a la cuenta que maneja al Issuer “Comillas”.

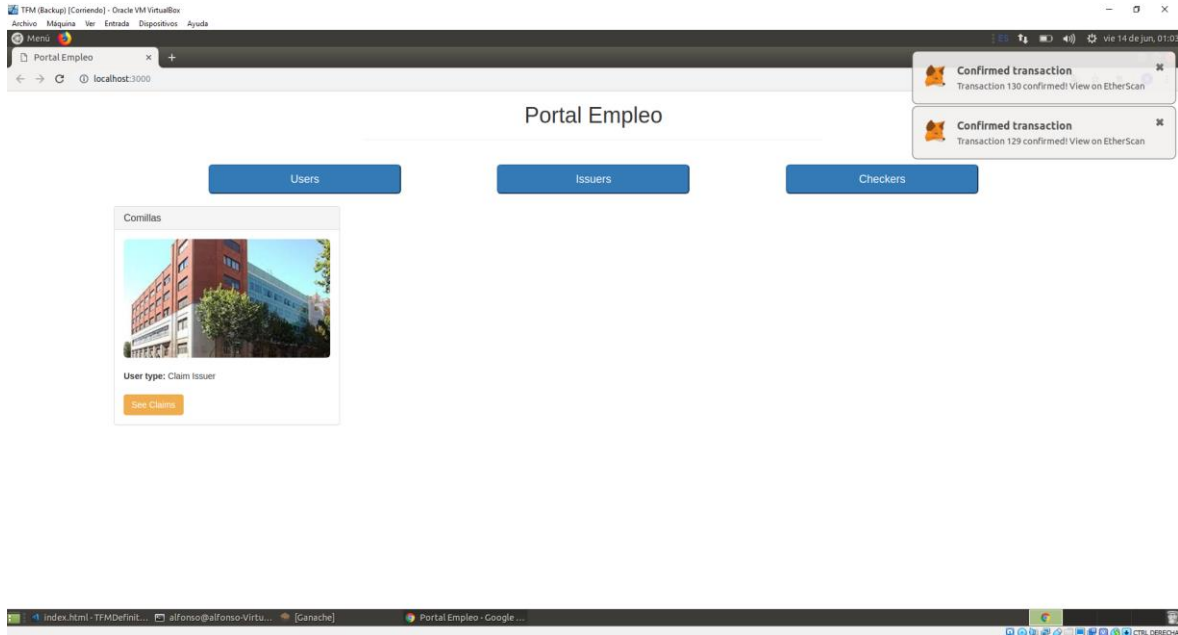


Se pulsa sobre “See Claims” y se abre el siguiente modal:



Esta ventana dice que hay una claim de nombre “GITT”, nombre del título, con nombre del creador “Alfonso” y que está invalidada. Por lo tanto, para validarla, será necesario pulsar

sobre “Validar”, ya que está escogida en el select de la izquierda de “Validar”. Una vez pulsado en “Validar” se acepta la transacción de MetaMask y se verá lo siguiente:



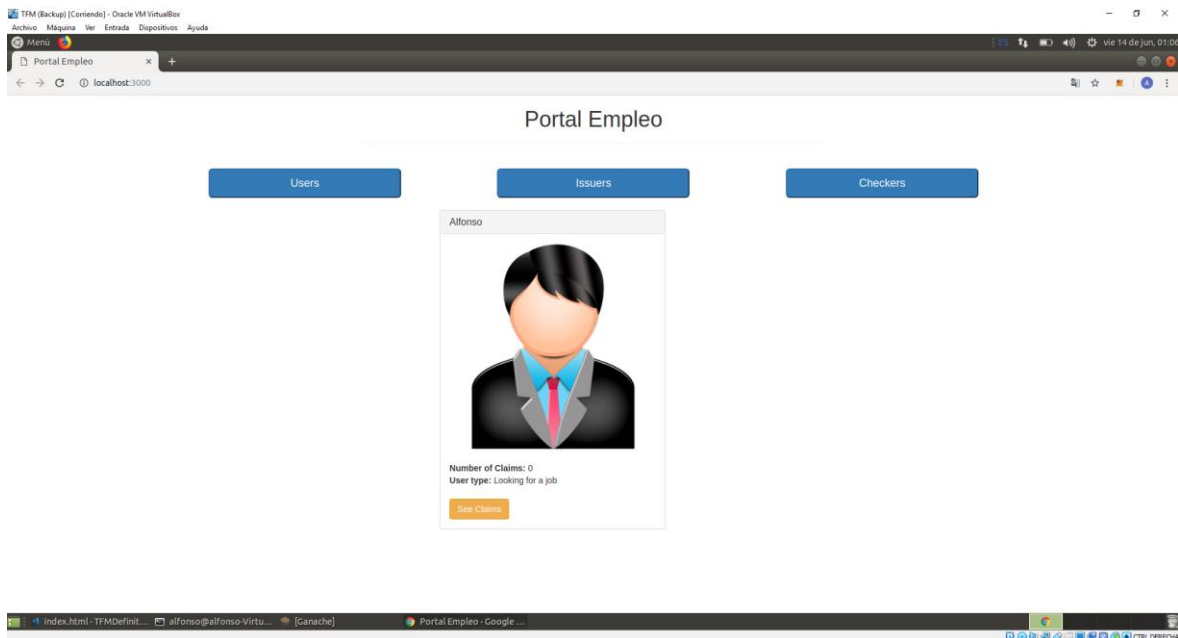
Se pulsa sobre “See Claims” y se verá el siguiente modal:



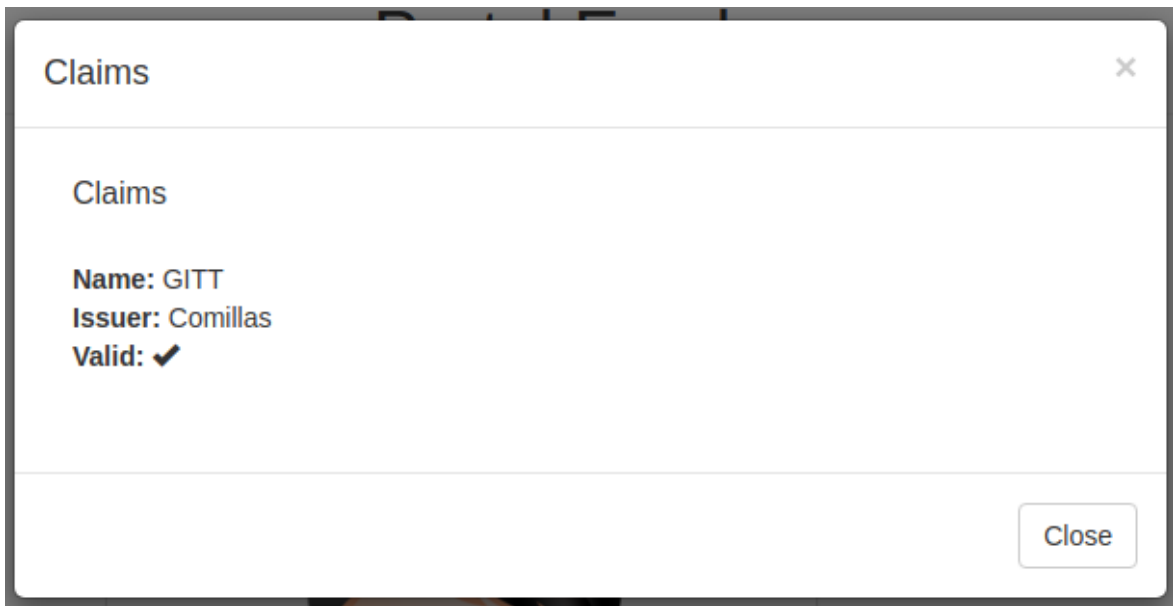
Esto dice que la claim “GITT” con dueño “Alfonso” está ahora validada. Aún así podría invalidarse en cualquier momento escogiéndola del select de la izquierda de “Invalidar” y pulsando sobre “Invalidar”.

8.8 APLICAR A UNA OFERTA

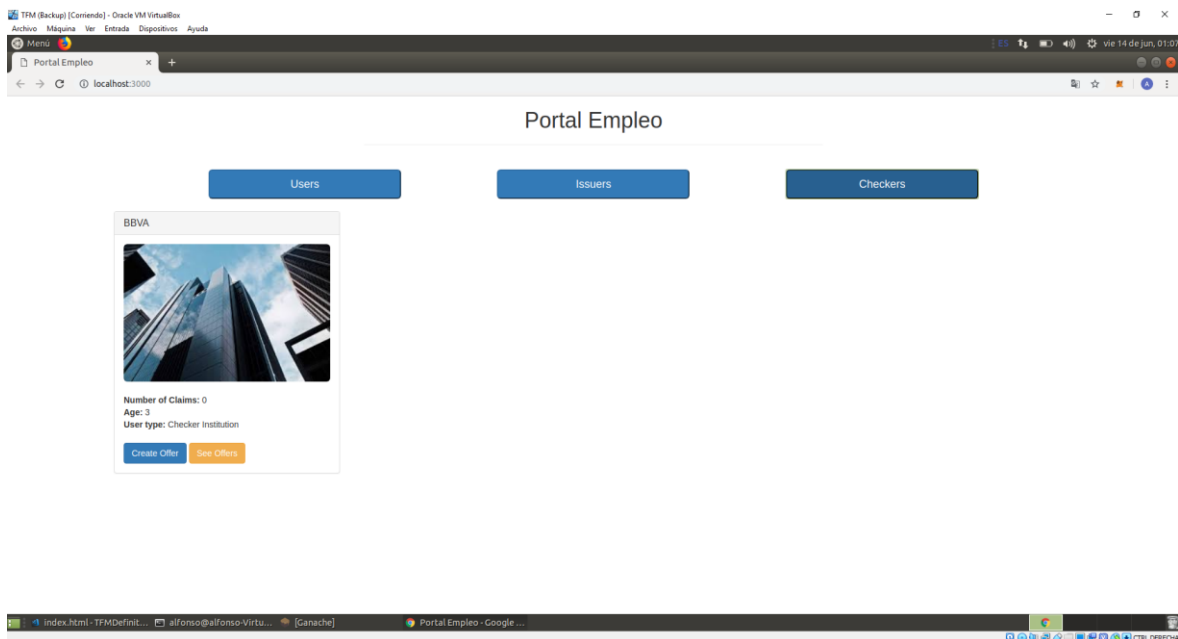
Por último, se cambia, a través de MetaMask, a la cuenta a la que pertenece el User “Alfonso”. Aparecerá la siguiente página:



Se pulsa sobre “See Claims” y se comprueba que la claim está validada:



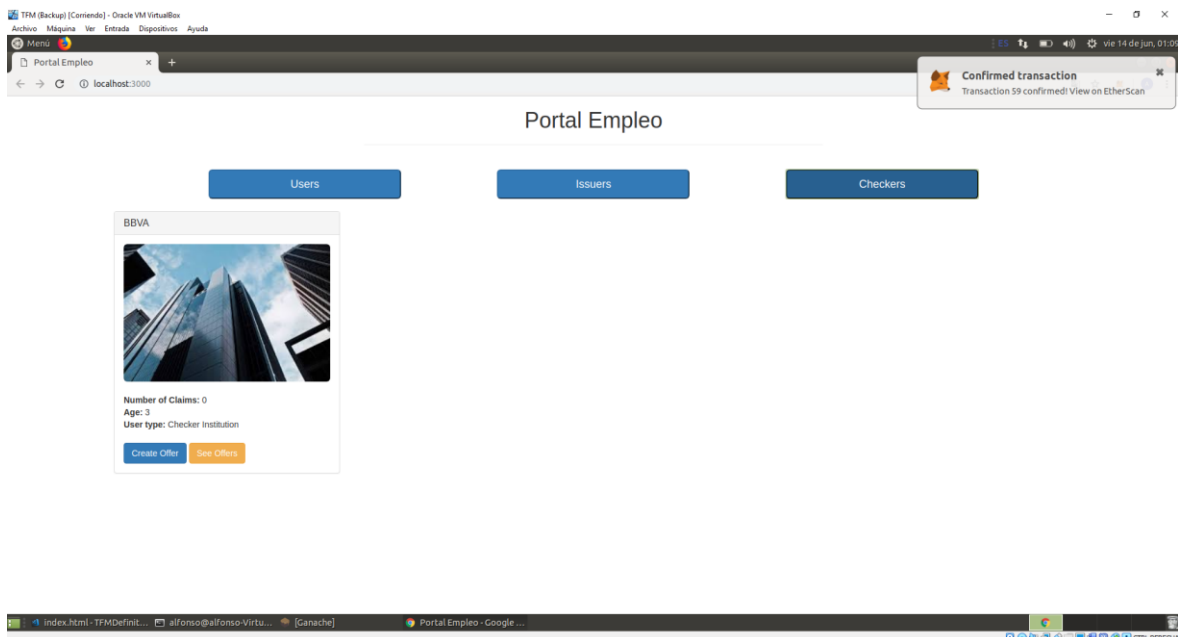
Como se puede ver, la claim es válida. Se cierra el modal y se pulsa sobre “Checkers”. Se verá lo siguiente:



Por lo tanto, se muestran los Checkers creados. Se pulsa sobre “See Offers” y se abre el siguiente modal:



Se puede observar que hay una oferta que tiene el mismo requisito que título validado tiene el User “Alfonso” ya que permite aplicar a ella. Se pulsa sobre “Aplicar” y se acepta la transacción de MetaMask. Se verá la siguiente ventana:



Se pulsa sobre “See Offers” y se mostrará el siguiente modal:



Ahora se puede ver que en la oferta “Software Developer” con requisito “GITT”, ha aplicado el usuario “Alfonso”.