



UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática



TFG del Grado en Ingeniería Informática

**Aplicación de tecnología
Blockchain a una cadena de
distribución de productos
Documentación Técnica**



Presentado por Eduardo Rodríguez Soto
en Universidad de Burgos — 17 de enero
de 2020

Tutor académico: Ángel Arroyo Puente
Tutor empresarial en HP: Pablo Tejedor
García

Índice general

Índice general	I
Índice de figuras	III
Índice de tablas	v
Apéndice A Plan de Proyecto Software	1
A.1. Introducción	1
A.2. Planificación temporal	1
A.3. Estudio de viabilidad	11
Apéndice B Especificación de Requisitos	15
B.1. Introducción	15
B.2. Objetivos generales	15
B.3. Catalogo de requisitos	15
B.4. Especificación de requisitos	17
Apéndice C Especificación de diseño	23
C.1. Introducción	23
C.2. Diseño de datos	23
C.3. Diseño procedimental	27
C.4. Diseño arquitectónico	29
Apéndice D Documentación técnica de programación	31
D.1. Introducción	31
D.2. Estructura de directorios	31
D.3. Manual del programador	32

D.4. Compilación, instalación y ejecución del proyecto	36
Apéndice E Documentación de usuario	39
E.1. Introducción	39
E.2. Requisitos de usuarios	39
E.3. Instalación	39
E.4. Manual del usuario	40
Bibliografía	51

Índice de figuras

A.1. Gráfico de las horas empleadas en el proyecto	8
A.2. Diagrama de Gantt	10
B.1. Diagrama de casos de uso del administrador	17
B.2. Diagrama de casos de uso del usuario	18
C.1. Modelo de datos	24
C.2. Tabla de usuarios	24
C.3. Tabla de productos	26
C.4. Diagrama de flujo de inicio de sesión	27
C.5. Tabla de productos	28
C.6. Mensaje correo no registrado en la base de datos	29
C.7. Mensaje de usuario creado con éxito	29
C.8. Mensaje de confirmar proceso	29
D.1. Confirmar el contrato	34
D.2. Conexión pendiente	34
D.3. Confirmar el contrato	34
D.4. Conexión pendiente	34
D.5. Conexión confirmada	34
D.6. Ejecución de XAMPP automáticamente	37
D.7. Ejecución de XAMPP manualmente	37
E.1. Inicio de sesión	40
E.2. Registrar nueva cuenta	41
E.3. Recuperar contraseña	42
E.4. Administrador de usuarios	43
E.5. Editar usuario desde administrador	43

E.6. Mensaje confirmación	44
E.7. Opciones a realizar por usuario	45
E.8. Conexion Metamask	45
E.9. Conexion proyecto metamask	45
E.10.Añadir producto	46
E.11.Confirmar el contrato	47
E.12.Conexión pendiente	47
E.13.Conexión confirmada	47
E.14.Información del contrato creado vistos por Etherscan	48
E.15.Datos proporcionados por Etherscan	48
E.16.Visualización del último producto	49
E.17.Visualización de todos los productos realizados	49

Índice de tablas

A.1. Tiempo empleado en el proyecto	7
A.2. realizadas en cada sprint	8
A.3. Tareas realizadas en cada sprint	9
A.4. Gastos de la seguridad social	11
A.5. Coste del empleado	12
A.6. Coste del empleado	13
B.1. RF 1: Crear usuarios	18
B.2. RF 2: Actualizar usuarios	19
B.3. RF 3: Eliminar usuarios	19
B.4. RF 4: Comprobar captha	20
B.5. RF 5: Conexión metamask	20
B.6. RF 6: Creación del <i>smart contracts</i>	21
B.7. RF 7: Consultar productos	21
B.8. RF 8: Salir de la aplicación	22

Apéndice A

Plan de Proyecto Software

A.1. Introducción

Cuando queremos hablar de la realización de proyecto, nos encontramos con una fase primordial, que es la planificación. En esta etapa se estiman los tiempos y el presupuesto del proyecto.

Para hablar de estas estimaciones, se debe tener en cuenta las partes que realizan el proyecto estén de acuerdo con todas las ideas planteadas. Se dividirán en dos partes:

- *Planificación temporal*: hablaremos de todas las reuniones realizadas y además comprobaremos si los tiempos de la realización del proyecto son óptimos.
- *Estudio de viabilidad*: intentaremos decidir si el proyecto fue bien planteado o falló algo en la ejecución, dentro de este estudio, tendremos dos tipos de viabilidad:
 - a *Económica*: explicaremos los costes y posibles beneficios del proyecto.
 - b *Legal*: estudiaremos las leyes que tuvieran que ver con nuestro proyecto.

A.2. Planificación temporal

Como hablamos en la memoria, utilizaremos una metodología Scrum. Las características de este método son:

- Utilizaremos una estrategia de sprint para esta metodología, dividiremos el trabajo en intervalos de un tiempo y así observaremos su desarrollo.
- Con cada sprint se planifican las tareas a realizar y el tiempo en el que desarrollaremos el trabajo. En general las reuniones fueron cada dos semanas.
- Al finalizar el sprint, realizaremos una reunión mediante videollamada o asistiendo personalmente a León.

El inicio del proyecto comienza con una reunión con el tutor Pablo Tejedor García mediante una llamada telefónica, en ella hablamos un poco acerca del porqué de la elección de este trabajo y cuales serán los objetivos del proyecto y las opciones que tendremos para realizarlas.

El tutor de HP, me comentó que es un proyecto de investigación, con una tecnológica nunca utilizada anteriormente en dicha empresa, el mundo Blockchain.

Sprint 1 (24/10/18 - 16/11/18)

En la primera reunión realizada hablamos de los objetivos y la realización de la siguiente tarea:

- Lectura del libro “*Scrum desde las trincheras*” [1].
- Realización del curso “CryptoZombies”.

Se calculan 30 horas de trabajo.

Sprint 2 (16/11/18 - 15/01/19)

Primeramente hablamos de la conclusión del sprint anterior y cómo fue la primera toma de contacto con un lenguaje nunca usado anteriormente.

Para esta segunda reunión se propusieron las siguientes tareas:

- Resumen del libro leído en el sprint anterior.
- Creación de un *smart contract*, con cierta funcionalidad en modo local.

- Realización de programas en Visual Studio, para la realización de pequeños archivos.
- Investigación de Truffle como entorno, ya que es una plataforma que nos permitirá desarrollar *smart contracts* y desplegarlos para poder consumirlos desde un frontEnd.

Se calculan 45 horas de trabajo.

Sprint 3 (15/01/19 - 12/02/19)

Para esta nueva tarea se proponen las siguientes tareas:

- Documentación somera de Truffle. Necesitamos una pequeña guía de que es, cómo instalarlo y qué funcionalidades proporciona Truffle para la memoria del Proyecto.
- Desarrollo de un *smart contract* sencillo con la plataforma Truffle. Un pequeño *smart contract* funcional con el que podamos empezar a trabajar.
- Desarrollo de una pequeña aplicación de escritorio o web (a decisión del alumno) en Visual Studio 2017 que disponga de una sola pantalla y elementos para interactuar con la capa de abajo donde se situará el *smart contract* desplegado.

Se calculan 40 horas de trabajo.

Sprint 4 (12/02/19 - 28/02/19)

Para la cuarta reunión en el proyecto se habló de la sesión anterior, en la cual comprobamos el guión de Truffle. Realizamos la ejecución de los *smart contract* realizados en el lenguaje Solidity y visualización de los proyectos creados en la aplicación Visual Studio 2017.

Las tareas a realizar para la siguiente sesión serán:

- Realización de un guión con la información a poner en la memoria.
- Investigación sobre web3.js (conexión entre Visual Studio y Ethereum).
- Creación de página de autobuses para realizar un simulacro de la página final, pero con modo local (Ganache).

Se calculan 52 horas de trabajo.

Sprint 5 (28/02/19 - 29/03/19)

Comentamos los problemas que surgieron en la sprint 4.

El principal quebradero de cabeza del sprint anterior fue la conexión entre web3.js, no sabíamos dar con el error ya que cuando probamos la aplicación nueva de autobuses con Ganache (modo Local) la conexión era perfecta y realizaba el cambio de divisa sin ningún problema.

Intentamos buscar alternativas cuando teníamos el error con web3.js (en modo red privada global o red principal Ethereum). Me propuso realizar la aplicación olvidándonos de la conexión con Ethereum y realizar una base de datos para poder mostrar todo desde ahí.

Pero seguimos indagando e intentamos realizarlo, pero con otras propuestas de editores de código.

- Realizar conexión entre metamask y un editor de código, con web3.
- Desarrollar el guión creado del sprint anterior.

Se calculan 25 horas de trabajo.

Sprint 6 (29/03/19 - 08/04/19)

Esta reunión se realizó en León de forma presencial y en ella comentamos el guion y la posibilidad de modificar algún punto para la mayor compresión de esta nueva tecnología.

Expliqué a Pablo el editor nuevo usado para la conexión de Ethereum y nuestro editor, y conseguimos la conexión entre ambos medios; pudiendo así realizar los *smart contract*, tanto en red local, privada y global (esto lo conseguimos en Visual Studio Code).

Una vez teníamos la conexión realizada ya teníamos una gran parte del objetivo del proyecto realizado.

Los siguientes objetivos a tratar para hablar de ellos en el siguiente sprint fueron:

- Creación de la pagina web, con las diferentes pantallas y la conexión entre ellas que fuera correctas.

- Buscar una base de datos para nuestro proyecto.
- Documentar la memoria.

Se calculan 60 horas de trabajo.

Sprint 7 (08/04/19 - 02/05/19)

Comentamos las páginas creadas de nuestra web, comprobamos la funcionalidad con metamask y este con Ethereum. La conexión era buena, ahora queda realizar los css y dejar la página armonizar todas las páginas.

La base de datos que realizamos es la de MySql, se encarga de almacenar a los usuarios y los datos enviados a la red Ethereum.

Para el siguiente sprint los objetivos serían:

- Armonización de la página web en todas sus páginas.
- Creación de página de Administrador que pueda manejar a todos los usuarios que tenemos registrados.
- Poder seleccionar un avatar cuando creamos usuario.

Se calculan 27 horas de trabajo.

Parón de los Sprint (02/05/19 - 23/10/19)

En esta reunión comenté la idea de que me iba de viaje a Estados Unidos y la fecha de ir no sería posible la entrega del Tfg de modo presencia, y que era demasiado tarde para poder cambiarlo a modo on-line, les hable de mi idea de continuar con el mismo Tfg para entregar en la primera convocatoria del curso 2019/2020.

Lo que supuso que en este parón hasta volver de Estados Unidos no realizaremos ningún Sprint ni hubiera avances en el desarrollo del proyecto.

Sprint 8 (23/10/19 - 18/11/19)

En esta reunión hablamos sobre pequeñas funcionalidades de nuestra página que se podrían añadir:

- Introducción de un código *captcha* cuando se registra el usuario.

- Creación de una nueva página: ¿Has olvidado la contraseña?
- Modificación de la visualización de todos los productos pedidos.
- Comenzar con el apartado de los anexos, explicación de cada sprint.

También nos dimos cuenta de que la red Ropsten se nos estaba quedando sin Ether y pensamos en suprimir el dinero cada vez que se realiza el *smart contract* y dejar solamente el gas de cada transacción.

Se calculan 45 horas de trabajo.

Sprint 9 (18/11/19 - 12/12/19)

Primeramente hablamos de la reunión anterior y comentamos la dificultad encontrada en realizar la función del *captha* y ¿Has olvidado la contraseña?, ya que al ser en modo local y no tener un dominio, tendríamos que realizarlo de otras maneras.

El objetivo para la décima reunión es:

- Mejorar los apartados resumen, introducción y concretos teóricos de la memoria.
- Investigar la opción de realizar el captha y ¿has olvidado la contraseña? mediante otro método.
- Continuar con los anexos.

Se calculan 37 horas de trabajo.

Sprint 10 (12/12/19 - 20/12/2019)

En la décima reunión hablamos de lo siguiente:

- Finalizar de la página Web.
- Intentar crear un ejecutable para nuestra aplicación.
- Finalizar la memoria y anexos.

Se calculan 55 horas de trabajo.

Sprint 11 (20/12/2019 - entrega proyecto)

En la última reunión hablamos de lo siguiente:

- Comprobar todas la funcionalidad de la web y la conexión con la red Ethereum.
- Decisión de realizar la entrega desde la cuenta Localhost.
- Revisar la memoria y anexos.
- Descargar todos los documentos para visualizar la información y ver que es correcta. Así evitaremos que se pueda encontrar imágenes cortadas o textos mal indexados.

Resumen

En la siguiente tabla A.1, veremos la distribución del tiempo en las diferentes tareas que se realizarán para la elaboración de este proyecto, cada tarea se expresa en horas:

Tarea a realizar	Tiempo(horas)
Programación	150
Investigación	105
Documentación	90
Revisión y corrección de errores	54
Reuniones	17
Total	416

Tabla A.1: Tiempo empleado en el proyecto.

En el siguiente gráfico A.1, observaremos el tiempo empleado estructurado en el desarrollo del proyecto.

A continuación explicaremos, el proceso de los sprint con una breve explicación de las tareas que se realizan en cada una de ellas A.2 ??

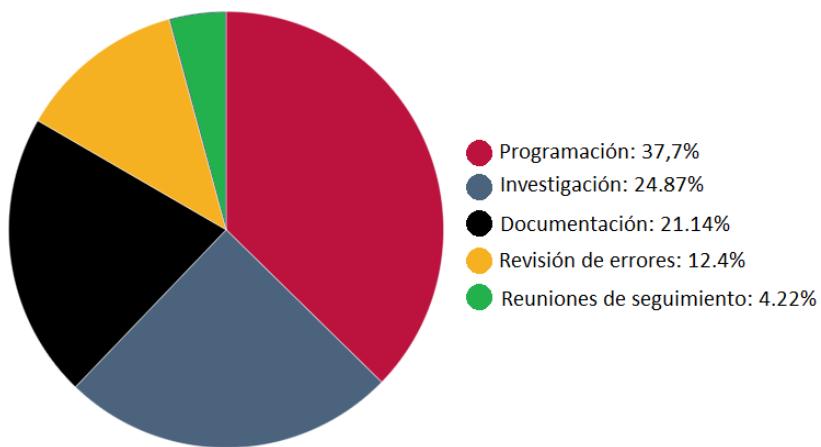


Figura A.1: Gráfico de las horas empleadas en el proyecto

Sprint-horas	Fechas de las tareas	Concepto
Sprint 1: 30 horas	24/10/18 24/10/18	
	Tarea 1	Conocer al tutor empresarial y comentar los objetivos y decisión de la elección de este TFG.
	Tarea 2	Lectura del libro "Scrum desde las trincheras".
	Tarea 3	Realización del curso "CryptoZombies".
Sprint 2: 45 horas	16/11/18 15/01/19	
	Tarea 1	Puesta en común de las tareas del Sprint anterior.
	Tarea 2	Creación de un smart contract, con cierta funcionalidad en modo local
	Tarea 3	Comienzo con visual studio y con herramienta Truffle
Sprint 3: 40 horas	15/01/19 12/02/19	
	Tarea 1	Documentación de truffle para creación de smart contract
	Tarea 2	Creacion de programas Visual Studio y comprobar su conexión con ethereum
Sprint 4: 52 horas	12/02/19 28/02/19	
	Tarea 1	Creación de guión para la futura memoria.
	Tarea 2	Investigación de Web3.js, con creación de página para ver su funcionamiento.

Tabla A.2: realizadas en cada sprint

Sprint- horas	Fechas de las tareas	Concepto
Sprint 5: 25 horas	28/02/19 29/03/19	
	Tarea 1	Desarrollo del guión del Sprint 4.
	Tarea 2	Comprender y enlazar la herramienta Metamask con web3
Sprint 6: 60 horas	29/03/19 08/04/19	
	Tarea 1	Comienzo de la página web final.
	Tarea 2	Buscar una base de datos para almacenar usuarios.
	Tarea 3	Documentar la memoria
Sprint 7: 27 horas	08/04/19 02/05/19	
	Tarea 1	Creación de página de Administrador
	Tarea 2	Crear avatar para los usuarios.
	Tarea 3	Armonizar la página web
Sprint pausa	02/05/19 23/10/19	
		Parón por motivo de trabajo en el extranjero
Sprint 8: 45 horas	23/10/19 18/11/19	
	Tarea 1	Introducción de un código captha y página ¿has olvidado contraseña?
	Tarea 2	Modificación de alguna página.
Sprint 9: 37 horas	18/11/19 12/12/19	
	Tarea 1	Mejorar memoria y continuar anexos.
	Tarea 2	Insertar captha mediante otro método.
Sprint 10: 55 horas	12/12/19 20/12/19	
	Tarea 1	Finalizar de la página Web.
	Tarea 2	Investigar como crear un ejecutable.
Sprint 11:	20/12/19 entrega	
	Tarea 1	Comprobar todas la funcionalidad de la web y la conexión con la red Ethereum.
	Tarea2	Entrega del proyecto desde Localhost.

Tabla A.3: Tareas realizadas en cada sprint

El el siguiente gráfico A.2 se podrá observar, el Diagrama de Gantt [2] es una herramienta gráfica cuyo objetivo es exponer el tiempo de dedicación previsto para diferentes tareas o actividades a lo largo de un tiempo total determinado. El diagrama se ha creado gracias a las herramientas que ofrece la siguiente pagina web [3]

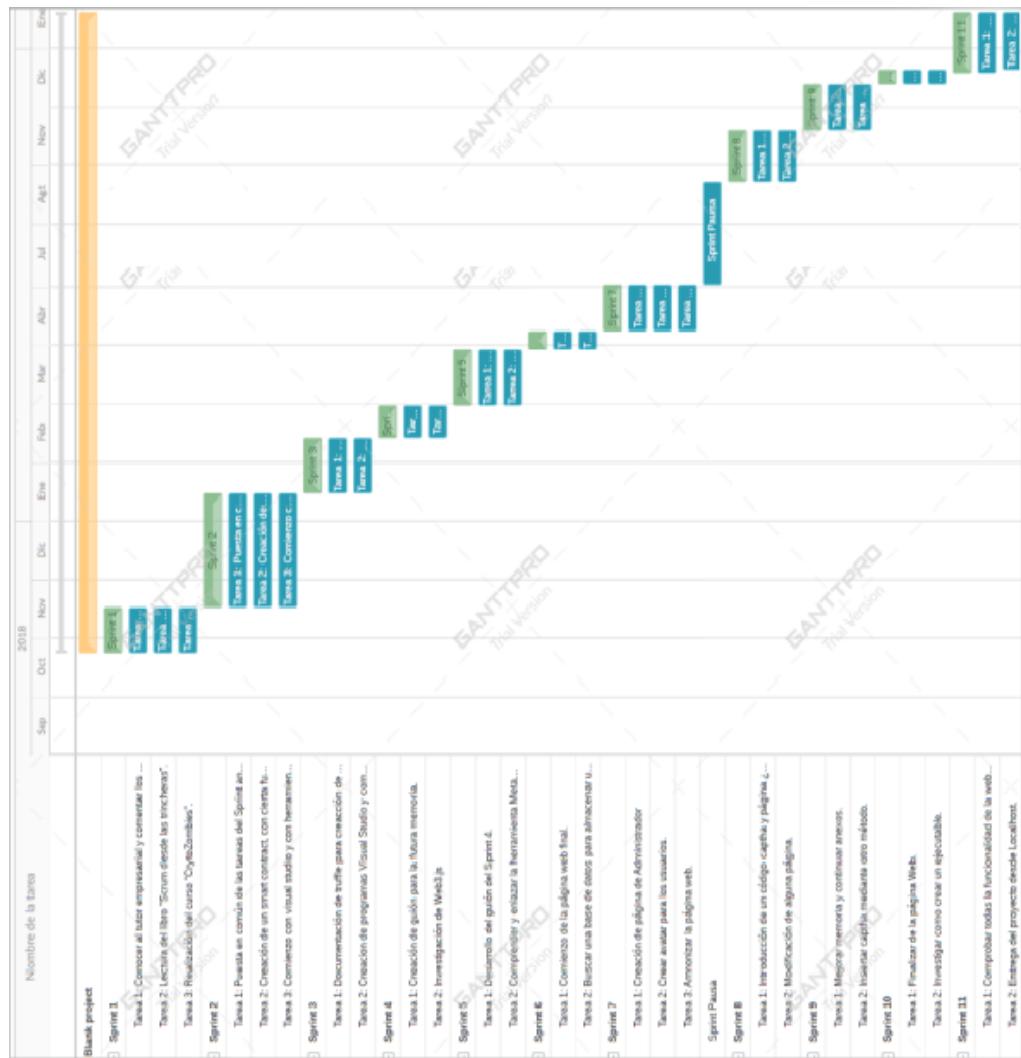


Figura A.2: Diagrama de Gantt

A.3. Estudio de viabilidad

En este apartado nos dispondremos a analizar los puntos de viabilidad económica y legal, con esto podremos observar si es beneficioso o no realizar el proyecto.

Viabilidad económica

En el primer subapartado, expondremos costes y beneficios que tendría la realización de este proyecto en el entorno empresarial.

Coste de personal

La realización de este trabajo ha sido llevada acabo por un desarrollador junior durante 8 meses, con base en estos datos calcularemos los siguientes parámetros:

Se estima que el sueldo medio de un programador junior en España esta entre los 14.000 y los 16.000 euros brutos anuales. Para este calculo tomaremos la cifra de 14.000 euros , a este coste tendremos que añadir la parte proporcional a la seguridad social, la cual corre a cargo de la empresa, que es el 23.6 % del sueldo del empleado [4].

La estimación media para la realización del proyecto es de 13 horas semanales, compatibilizando el tiempo con la finalización del grado en ingeniería informática y realización de trabajo externo. Lo que hace un total de 416 horas empleadas en la elaboración del proyecto (8 meses * 4 semanas/mes * 13 horas/semana).

Concepto	Trabajador	Empresa
Contingencias comunes	4,7 %	23,6 %
Desempleo	1.55 %	5,5 %
Formación profesional	0,1 %	0,6 %
Garantía social	0	0,2 %
TOTAL	6.35 %	29.9 %

Tabla A.4: Gastos de la seguridad social

A continuación calculamos el precio hora que cobraremos, para luego calcular el salario con la seguridad social.

Dentro de estos costes tendremos que añadir el coste del tutor.

Concepto	Coste (Euros)
Salario anual	14.000
Salario mensual	1166.66
Salario a la hora	7.29
Suelo sin seguridad social	3033.33
Sueldo con seguridad social (23.6 %)	3749.19

Tabla A.5: Coste del empleado

Coste informático

Dentro de esta rama, los podremos dividir en dos clases que serán las siguientes:

Coste de hardware Para la realización de este proyecto, únicamente se tendrá en cuenta el coste del ordenador utilizado, el coste del ordenador ronda entre los 700 y los 900 euros. El ordenador con el cual se ha realizado el proyecto tiene una antigüedad de 5 años y se estima un uso de vida total de 8 años y tiene un valor de 750 euros. El tiempo de empleo para este trabajo ha sido de 8 meses, por lo tanto el coste amortizado de estos meses será de 116.69 euros.

También se podrían sumar otros gastos como son luz, Internet, material de oficina pero son despreciables y por eso no se incluyen en este calculo.

Coste de software Para la realización del proyecto, todos los programas, bibliotecas y herramientas empleadas son de software libre y gratuito. En caso de haber tenido alguna versión de pago, empleábamos la version anterior o su versión de prueba.

Coste de total

Para concluir, tendremos los siguientes costes, durante el tiempo dedicado a este proyecto, salario del empleado con la seguridad social incluida y costes informáticos.

Concepto	Coste (Euros)
Salario total	3749.19
Costes informáticos	116.69
TOTAL	3862.88

Tabla A.6: Coste del empleado

El precio total del proyecto serían 3862.88 euros, esto teniendo en cuenta, que faltaría indicar de sumar el salario del tutor.

Beneficio

En cuanto a los beneficios serían inexistentes, ya que serían indirectos y se plantearía como una aplicación de uso propio para HP, con ello ayudaría en el desarrollo de la tecnología Blockchain, teniendo en cuenta en el auge de esta herramienta cada vez más usada en todos las industrias podrá ser adaptado para futuras operaciones con diferentes clientes. Así los beneficios serían 100 % indirectos.

Viabilidad legal

En este último apartado expondremos la viabilidad en el ámbito legal, hablaremos sobre las licencias software que empleamos en el desarrollo del mismo.

Se define una licencia software [5] como un contrato mediante el cual una persona recibe de otra el derecho de uso, copia, distribución, estudio y modificación de varios de sus bienes, (normalmente de carácter intelectual), pudiéndose realizar un pago de una determinada cantidad por el uso de los mismos.

Los programas y herramientas utilizadas durante el desarrollo de este proyecto han sido de licencia abierta o de uso libre y gratuito (Solidity, HTML, PHP, Remix, Ganache ...). Esto es gracias a la licencia MIT (*Massachusetts Institute of Technology*) [6], la cual da libertad para poder usar el software libremente y con ello exime de responsabilidad a la parte creadora de las herramientas.

El proyecto se ha realizado en colaboración con HP SCDS, por esta razón el software resultante está bajo licencia copyright de la empresa. Asimismo, el código fuente será propiedad de HP SCDS y de la universidad de Burgos y podrá estar también bajo licencia copyright.

Apéndice B

Especificación de Requisitos

B.1. Introducción

En este apartado realizaremos un estudio de requisitos y objetivos que definirán el comportamiento del proyecto actual. También hablaremos de los diferentes casos de uso y los requisitos funcionales de la herramienta en cuestión.

B.2. Objetivos generales

Los objetivos generales que se quieren alcanzar al realizar este proyecto son los siguientes:

- Implementar una página web para la realización de *smart contracts*, en diferentes redes y con diferentes usuarios.
- Investigar las posibilidades y capacidades de conexión de la red ethereum y solidity para desarrollar *smart contracts*.

B.3. Catalogo de requisitos

En este apartado mencionaremos los requisitos, divididos en funcionales y no funcionales.

Requisitos funcionales

RNF-1 Crear usuarios : crear un sistema de gestión de usuarios y permisos para, así, garantizar la seguridad de la aplicación.

RNF-2 Actualizar usuario : comprobación que permita la verificación de si el usuario o la contraseña fueron modificadas y desde qué cuenta realizó los cambios.

RNF-3 Eliminar usuario : revisión para saber si el usuario fue borrado desde su propia cuenta o desde la cuenta de administrador.

RNF-4 Comprobar captha : verificar el botón si pasa el proceso de no soy un robot.

RNF-5 Conexión con metamask : al añadir producto, comprobar que pregunta si queremos conectar con dicha aplicación.

RNF-6 Creación del *smart contracts* : confirmar que el contrato fue creado desde la página web a una red Ethereum mediante metamask.

RNF-7 Consultar productos : podremos consultar todos los productos, tanto el ultimo añadido o todos en forma de tabla. También se incluye la hora de la creación del contrato.

RNF-8 Salir de la aplicación : funcionalidad que permita desconectar al usuario de la página web mediante un botón.

Requisitos no funcionales

RNF-1 Usabilidad : la aplicación será intuitiva y fácil de utilizar, así como tener una estructura clara para qué cualquier persona pueda comprenderla sin necesidad de ninguna formación.

RNF-2 Eficiencia : la aplicación funcionará de una forma fluida, sin que la interfaz gráfica quede bloqueada.

RNF-3 Escalabilidad : estará siempre abierta a la incorporación de nuevas funciones o programas.

RNF-4 Autonomía : deberá realizar actualizaciones para no perder información y esta siempre intente ser la más fiable posible.

B.4. Especificación de requisitos

En esta sección podremos visualizar los casos de uso de los requisitos funcionales anteriormente explicados y también detallaremos todos los casos de uso mediante tablas.

Diagramas de casos de uso

En las siguientes figuras podremos observar los casos de uso que corresponden a nuestra aplicación, en la cual se podrán observar dos actores (administrador y usuario). Los UML los hemos realizado en la siguiente página web [7]

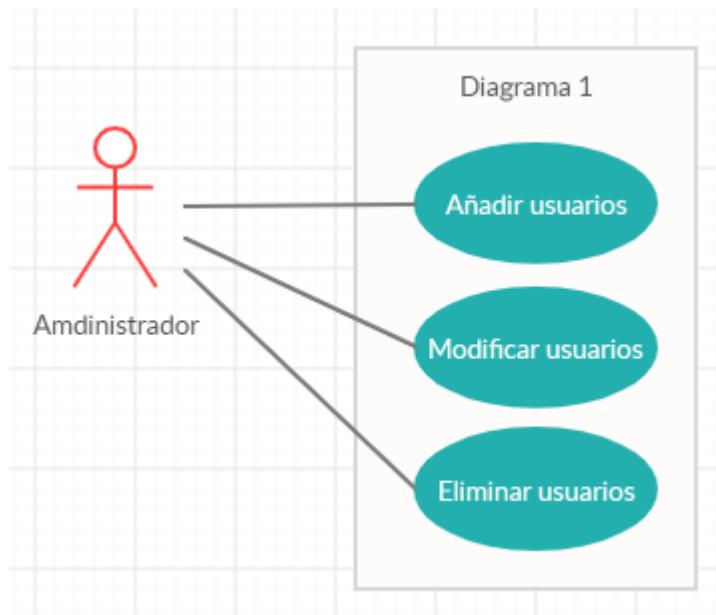


Figura B.1: Diagrama de casos de uso del administrador

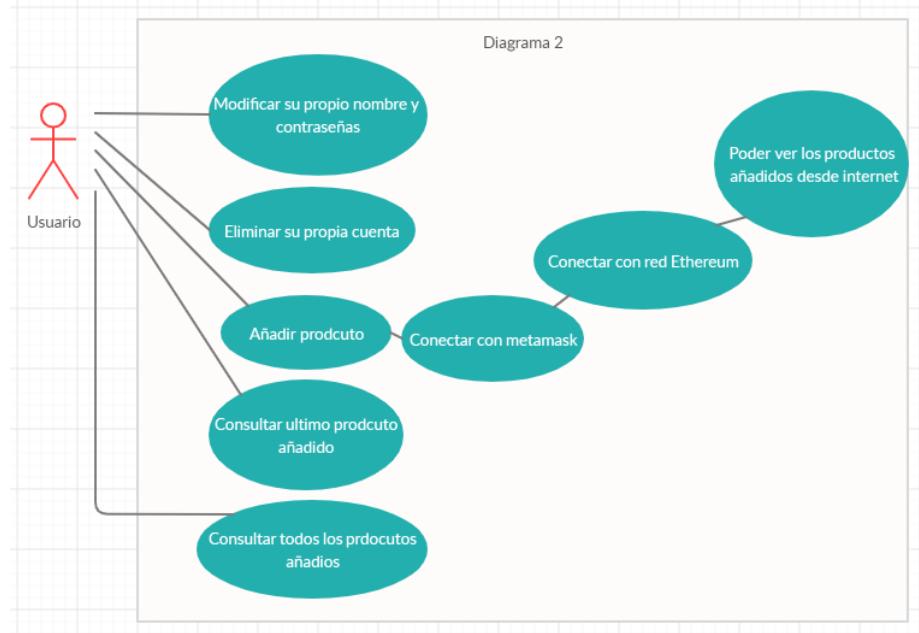


Figura B.2: Diagrama de casos de uso del usuario

Especificación de los casos de uso

A continuación detallaremos mediante tablas los casos de uso:

RF 1	Crear usuarios
Descripción	Registro en la base de datos de nuevos usuarios.
Pre-condición	Un usuario deberá introducir un usuario, correo y una contraseña, con el objetivo de poder registrarse en la base de datos.
Acción	Si el correo ya esta en uso, las contraseñas no coinciden o no marcamos la opción “no soy un robot”, tendremos que repetir el proceso de registrarse.
Post-condición	Si el correo y la contraseña son correctos, nos mostrará la siguiente pantalla de la página Web.
Importancia	Alta

Tabla B.1: RF 1: Crear usuarios

RF 2	Actualizar usuario
Descripción	Dependiendo de si nos encontramos en un usuario o desde el administrador
Pre-condición	Haber iniciado sesión previamente (desde administrador o de usuario).
Acción	Podremos modificar el usuario y la contraseña de un usuario concreto, o, si estamos en el administrador, podremos acceder a modificar todos los usuarios.
Post-condición	Si todo fue correcto, los datos se modificarán en nuestra base de datos
Importancia	Alta

Tabla B.2: RF 2: Actualizar usuarios

RF 3	Eliminar usuario
Descripción	Dependiendo de si nos encontramos en un usuario o desde el administrador, podremos ver o uno o a todos.
Pre-condición	Haber iniciado sesión previamente (desde administrador o de usuario).
Acción	Podremos eliminar el usuario de nuestra cuenta, ,o si estamos en el administrador, podremos acceder a eliminar los usuarios que deseemos.
Post-condición	Si todo fue correcto, los datos se eliminarán en nuestra base de datos, también eliminaremos todos los productos que tiene adquiridos previamente.
Importancia	Alta

Tabla B.3: RF 3: Eliminar usuarios

RF 4	Comprobación captha
Descripción	Cuando registramos un nuevo usuario, el ultimo apartado, una vez introducidos todos los campos, es demostrar que no se es un robot.
Pre-condición	Ninguna
Acción	Pulsar el botón no soy un robot, y seleccionar los campos que nos pida en cada momento.
Post-condición	Si todo fue correcto, nos creará la nueva cuenta.
Importancia	Alta

Tabla B.4: RF 4: Comprobar captha

RF 5	Conexión con metamask
Descripción	Realizar contrato desde la página web con la red Ethereum.
Pre-condición	Haber iniciado sesión como usuario.
Acción	Creación de un nuevo contrato cuando pulsamos “Añadir Producto”. Si no tenemos Metamask conectado, nos saldrá una pantalla para iniciar sesión y crear el primer <i>smart contracts</i> vacío.
Post-condición	Añadir todos los campos vacíos que se encuentran en la página web para la creación del <i>smart contracts</i> .
Importancia	Alta

Tabla B.5: RF 5: Conexión metamask

RF 6	Creación del <i>smart contracts</i>
Descripción	Realizar contrato desde la página web con la red Ethereum.
Pre-condición	Haber iniciado sesión y estar en la página Añadir Producto.
Acción	Pulsaremos en “Realizar contrato” y nos saltará una notificación en metamask para confirmar o rechazar el contrato a realizar
Pos-condición	Pulsar “confirmar” y tras una espera nos confirmará si se realizó correctamente pudiendo ver el contrato tanto en la red privada de Ropsten o consultando los productos añadidos de la página web.
Importancia	Alta

Tabla B.6: RF 6: Creación del *smart contracts*

RF 7	Consultar productos
Descripción	Posibilidad de informarse de los productos que un usuario ha registrado en la red Ethereum.
Pre-condición	Haber iniciado sesión y haber añadido un producto anteriormente (si no la página nos saldrá vacía).
Acción	Ir a la página de consulta de productos para poder observar los productos añadidos.
Post-condición	Podremos ver los productos que añadió cada usuario, solo podrán verse los productos del usuario registrado.
Importancia	Alta

Tabla B.7: RF 7: Consultar productos

RF 8	Salir de la aplicación
Descripción	Cerrar sesión de nuestra página web.
Pre-condición	Haber iniciado sesión.
Acción	Pulsar el botón de cerrar sesión, ubicado en su propio avatar.
Post-condición	Al cerrar sesión se desconectará de la pagina web.
Importancia	Alta

Tabla B.8: RF 8: Salir de la aplicación

Apéndice C

Especificación de diseño

C.1. Introducción

En este apartado de los anexos, explicaremos la forma que se resolvieron diferentes especificaciones y los casos de uso mencionados en el apartado anterior. Se intentarán analizar el porqué de las tomas de decisiones. Los diseños se dividen en:

- Diseño de datos.
- Diseño procedimental.
- Diseño arquitectónico.
- Diseño de interfaz.

C.2. Diseño de datos

En este proyecto, los datos que manejaremos serán la creación de usuarios y los contratos que estos creen posteriormente.

Estos datos se almacenarán en una base de datos, aquellos contendrán el id de usuario y este irá relacionado con el resto de elementos de la tabla. Estos elementos les comentaremos a continuación cuando hablamos de la base de datos.

Modelo de datos

En la siguiente figura C.1 se mostrará el modelo de datos.

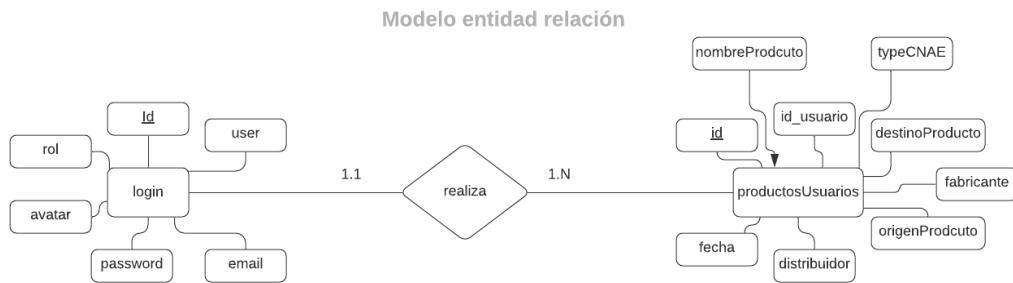


Figura C.1: Modelo de datos

XAMPP

Utilizaremos una base de datos MySql para el almacenamiento de la información de los datos de nuestra aplicación. La base de datos está compuesta por dos tablas:

Usuarios: en esta tabla almacenaremos la información relacionada con la creación de nuevos usuarios registrados previamente. Esta formado por siete campos, que son los siguientes:

#	Nombre	Tipo
1	id	int(11)
2	user	varchar(200)
3	password	varchar(200)
4	email	varchar(200)
5	pasadmin	varchar(200)
6	rol	int(3)
7	avatar	text

Figura C.2: Tabla de usuarios

- *ID*: es la clave primaria (PK) y almacenará un valor único por cada usuario. Este valor es autoincremental.
- *User*: donde se guarda el valor del usuario registrado.
- *Password*: guardaremos la contraseña que indicó el usuario, esta a su vez se guardará de forma codificada mediante sha-1.
- *Email*: en este campo guardaremos el correo del usuario, el cual será un campo único y, en caso de introducir el mismo, no nos dejará guardarla de nuevo.
- *Pasaadmin*: campo diseñado para guardar la contraseña en caso de ser los administradores de la página. Este campo actualmente no está en uso ya que descartamos la opción de poder crear administradores nuevos.
- *Rol*: tendrá dos posibles valores, dependiendo de si somos los administradores o, por el contrario, somos los usuarios de la página. Los valores serán 1 o 2 respectivamente.
- *Avatar*: en el último campo tendremos la foto predeterminada del usuario en caso de que haya seleccionado foto a la hora de registrarse, si no está, será una foto neutra.

Productos de los usuarios: en esta nueva tabla se almacenarán todos los productos que el usuario, previamente registrado, ha realizado en la red de Ethereum. Hay un total de diez campos y son los siguientes:

- *Id*: es la clave primaria de la tabla (PK), el valor es auto incremental haciendo que cada producto posea un identificador único.
- *Id usuario*: es la clave foránea, la cual se relaciona con la clave primaria de la tabla anterior (usuarios). Gracias a esto tendremos siempre relacionada la tabla usuarios con la de productos, gracias al Id del usuario.
- *Fabricante*: indica a quién estamos pidiendo el producto cada vez. Se trata de un campo de caracteres.
- *OrigenProducto*: nos referimos de dónde saldrá el paquete en su origen, es un campo de caracteres.

#	Nombre	Tipo
1	id 	<code>int(11)</code>
2	id_usuario 	<code>int(11)</code>
3	fabricante	<code>varchar(200)</code>
4	origenProducto	<code>varchar(200)</code>
5	distribuidor	<code>varchar(200)</code>
6	cantidad	<code>int(11)</code>
7	nombreProducto	<code>varchar(200)</code>
8	typeCNAE	<code>int(11)</code>
9	fecha	<code>timestamp</code>
10	destinoProducto	<code>varchar(200)</code>

Figura C.3: Tabla de productos

- *Distribuidor*: indicaremos quién es el encargado del reparto en cada producto registrado, este campo es de caracteres.
- *Cantidad*: hará referencia al número que queremos adquirir para el pedido de ese producto, este campo es numérico.
- *NombreProducto*: nos referimos al nombre concreto del producto que deseamos adquirir, este campo será de caracteres.
- *TypeCNAE*: indicaremos mediante un valor numérico la Clasificación Nacional de Actividades Económicas (CNAE).

- *Fecha*: recogeremos la hora en la que se realizó el contrato en nuestra red, este tiempo puede variar unos segundos la red Ethereum.
- *DestinoProdcuto*: es lo mismo que el *origenProdcuto*, pero esta vez tendremos que indicar el punto final del paquete, es un campo de caracteres.

C.3. Diseño procedimental

En este punto, explicaremos el comportamiento de la aplicación y veremos los procedimientos o funcionalidades que existen en el proyecto. En bases generales esto intenta ser lo más intuitivo y eficaz para que lo pueda usar todo tipo de usuarios.

Contamos con una base de datos ya creada, la cual contará con las dos tablas previamente explicadas, dichas tablas no se podrán modificar.

Tendremos dos tipos de entrada en la página web, podrán ser mediante administrador, o bien, mediante usuario.

La siguiente imagen muestra el diagrama de flujo C.4 el inicio de sesión de la página web:

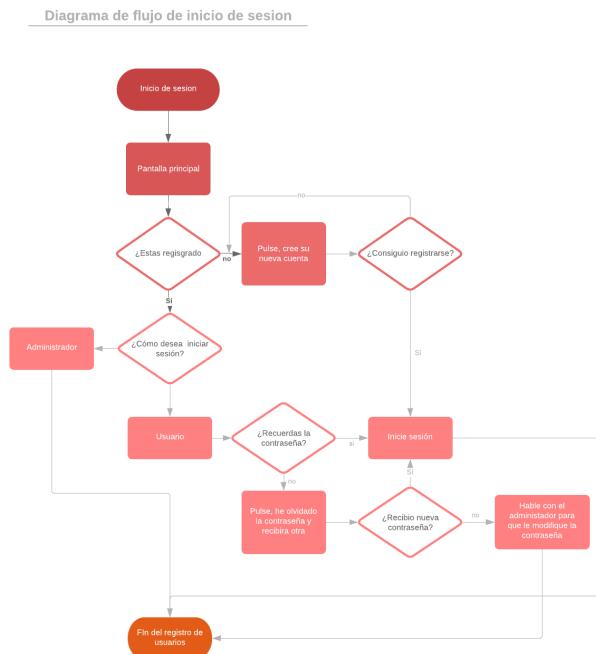


Figura C.4: Diagrama de flujo de inicio de sesión

En la siguiente imagen mostraremos el funcionamiento de la página una vez, hemos registrado sesión, dependiendo de si estamos en modo administrador o modo usuario [C.5](#).

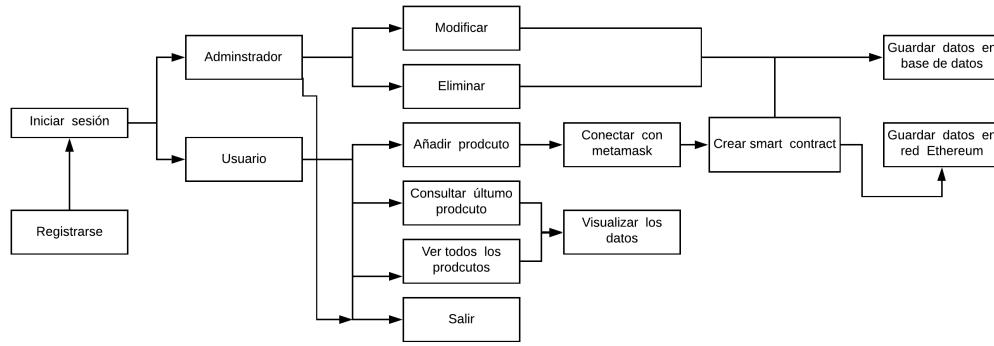


Figura C.5: Tabla de productos

A continuación, explicaremos las funciones que coinciden del usuario y del administrador:

1. Modificar:

- *Usuario*: podrá cambiar tanto el nombre y la contraseña de su propio usuario.
- *Administrador*: tendrá acceso a cambiar el nombre y contraseña de todos los usuarios registrados en la base de datos, también podrá acceder a ver el id de los usuarios.

2. Eliminar:

- *Usuario*: dispondrá de una opción para eliminar su propia cuenta y todos los productos asociados a él.
- *Administrador*: tendrá acceso a visualizar todos los usuarios registrados en la base de datos y podrá eliminar los que desee, esto también elimina todos los productos asociados previamente.

3. Salir:

- *Usuario y administrador*: podremos salir una vez estemos registrado en la página web.

C.4. Diseño arquitectónico

Para la creación de la interfaz de usuario, los requisitos que deseábamos cumplir eran facilidad y sencillez a la hora de usar la página web. La interfaz es clara y limpia, con esto se consigue facilitar la interacción con el usuario. Esto se logra gracias a las páginas de estilo (css) y utilizando los *bootstrap* que resulta en que la página quede con buen aspecto visual.

En el apéndice E.4 realizaremos la explicación del *front-end* de la aplicación final, para que el usuario pueda tener una guía explicativa.

Con el objetivo de ayudar al usuario, hemos incorporado mensajes de aviso, tanto si son de error C.6, si son informativos C.7 o son preguntas de confirmación C.8.

localhost dice
El usuario no está registrado, regístrate para iniciar sesión.

localhost dice
Usuario fue registrado con éxito

Aceptar

Aceptar

Figura C.6: Mensaje correo no registrado en la base de datos

Figura C.7: Mensaje de usuario creado con éxito

localhost dice
¿Estás seguro de que deseas borrar este usuario?

Aceptar

Cancelar

Figura C.8: Mensaje de confirmar proceso

Apéndice D

Documentación técnica de programación

D.1. Introducción

La documentación técnica de programación describirá la estructura de los programas empleados en el proyecto, el entorno de desarrollo, la instalación y ejecución de los programas.

D.2. Estructura de directorios

Dentro de GitLab tendremos programas y códigos que nos resultaran ajenos para el correcto funcionamiento, estos programas son pruebas realizadas anteriormente (tutoriales, código de pruebas ...).

Para el desarrollo del proyecto ¹ ², se utiliza los siguientes ramas:

- *Proyecto*: en la raíz \xampp\htdocs\proyectoFinal tendremos todo el contenido del código de nuestro proyecto, archivos: PHP, CSS, Bootstrap ... y también el enlace de conexión con metamask.
- *Documentación*: se encontrará en la carpeta Latex y contará con la documentación (memoria y anexos) para la compresión del funcionamiento de la aplicación, también cuenta con un guion de la instalación

¹Repositorio GitLab: <https://gitlab.com/HP-SCDS/Observatorio/2018-2019/ubu-bc4distribution/tree/paginaWeb>

²Repositorio GitHub: <https://github.com/ers0026/ProyectoFinal>

APÉNDICE D. DOCUMENTACIÓN TÉCNICA DE PROGRAMACIÓN

de truffle, el cual no hace falta para el funcionamiento pero creemos que un buena herramienta para el funcionamiento del proyecto). También se dispondrán los pdf en la rama principal para su fácil acceso.

D.3. Manual del programador

En esta sección hablaremos de la instalación de los programas necesarios para el funcionamiento de la página web y su conexión con la red Ethereum.

■ Metamask

- Primeramente realizaremos la instalación en el navegador que deseemos desde la página³ (en nuestro caso lo hemos realizado desde Google Chrome).
- Ahora nos redireccionará y nos aparecerá un botón “Añadir a Chrome”.
- A continuación nos saldrá una pantalla emergente: “Añadir extensión”.
- Esto instalará en la barra de extensiones el símbolo de metamask, un zorro.

Después de estos pasos, tendremos una serie de pasos a realizar para importar la cuenta:

- Primera imagen indica el comienzo de metamask **D.1**, pulsaremos “Get Started” para continuar.
- Seguidamente tendremos dos caminos posibles **D.2**:
 1. Continuar con una cuenta ya existente.
 2. Crear una nueva cuenta.

En este caso iremos por el camino de “Importar Wallet” ya que las operaciones las realizaremos desde la misma cuenta, si deseáramos cambiar de cuenta tendríamos que realizar cambios en el código.

- A continuación, nos pedirá los permisos para analizar los movimientos que realizamos, pulsaremos “I agree”**D.3**
- Esta es la pantalla más importante, ya que tendremos que tener la cadena de palabras *nemonic* que nos dieron al crear la cuenta

³<https://metamask.io/>

por primera vez y al iniciar en otro ordenador tendremos que introducir nuevamente una contraseña para entrar futuras veces (pondremos la misma contraseña que tenemos en otros navegadores o equipos) para así tener acceso siempre a nuestro metamask y aceptaremos los términos de uso. Una vez todo esto realizado pulsaremos “importar” D.4

- Nos mostrará un mensaje de registro completado con éxito D.5 y tendremos que clic en “All done”

APÉNDICE D. DOCUMENTACIÓN TÉCNICA DE PROGRAMACIÓN

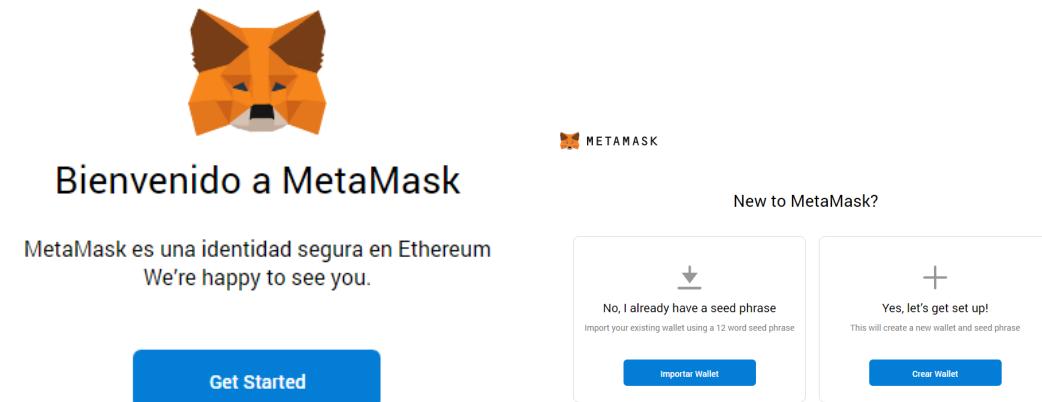


Figura D.1: Confirmar el contrato

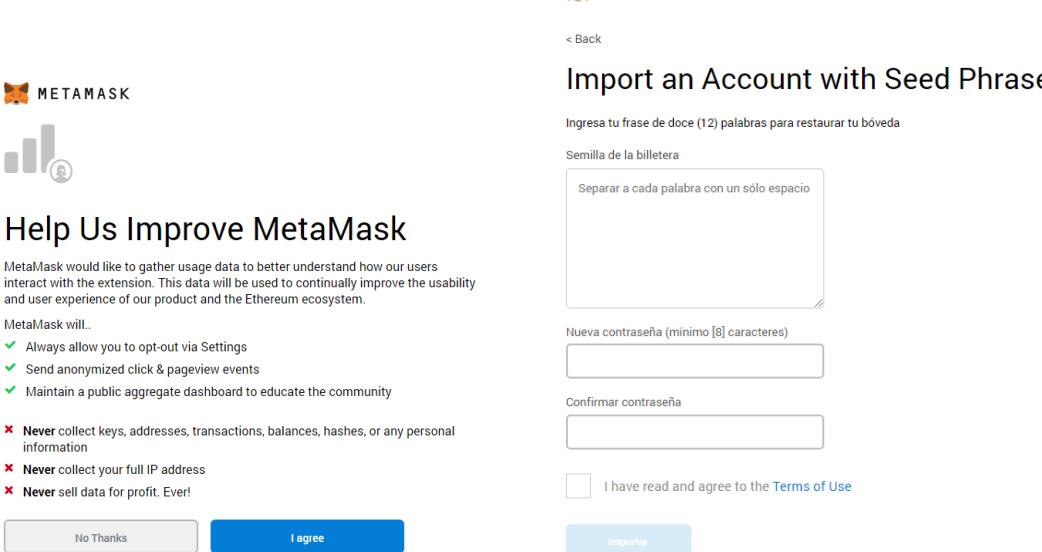


Figura D.2: Conexión pendiente

Figura D.3: Confirmar el contrato

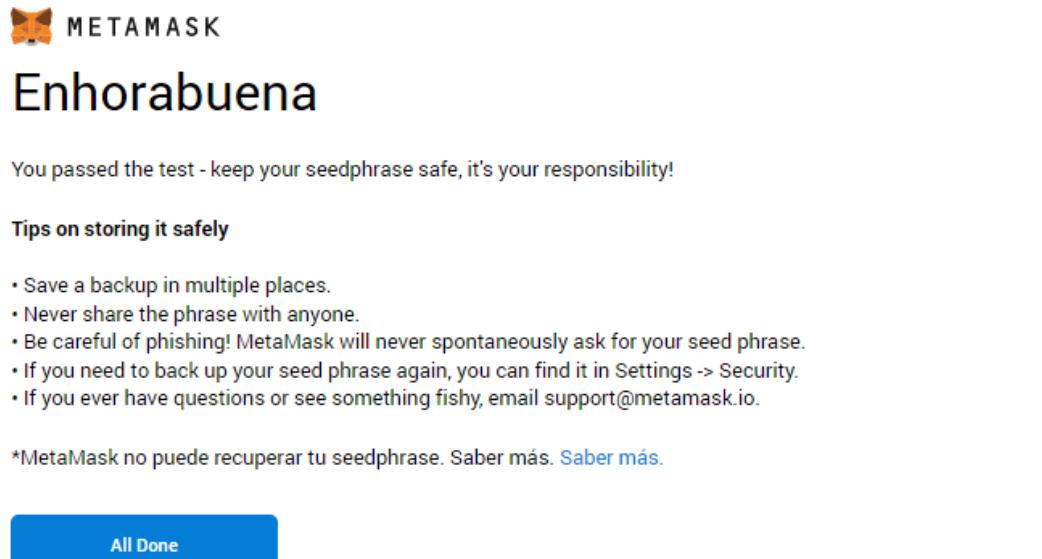


Figura D.4: Conexión pendiente

■ XAMPP

Para descargar el programa XAMPP, tendremos que seguir los siguientes pasos:

1. Iremos a la página⁴.
2. Elegiremos la versión en la que ejecutaremos XAMPP (Windows, Linux o iOS).
3. Una vez descargado, buscaremos el archivo “xampp-win32-7.2.4-0-VC15-installer” y ejecutaremos el archivo.
4. Cuando se abra la primera pantalla, haremos clic en *Yes*.
5. A continuación seleccionaremos las funciones XAMPP que deseamos instalar, en nuestro caso seleccionamos todas ellas, será la forma predeterminada y pulsaremos “*Next*”.
6. Seleccionaremos la ubicación donde deseemos instalar el programa. Una vez ubicado pulsaremos “*Next*”.
7. En la penúltima pantalla desmarcaremos la opción de “textitLearn more about Bitnami for XAMPP” “Aprender más sobre Bitnami” y clicamos “*Next*”.
8. Con esto empezará a instalar XAMPP en la carpeta seleccionada. Pulsaremos *Finish*, esto cerrará la ventana y abrirá el panel de control de XAMPP y aquí accedemos a los servidores. Antes de ello, tendremos que seleccionar el idioma que deseemos.

■ Visual Studio Code

1. Nos dirigiremos a la página⁵ y seleccionaremos para qué sistema deseamos el programa: Windows, Linux, macOS.
2. Una vez descargado, se abrirá un asistente de instalación y tendremos que aceptar el acuerdo de licencia y pulsaremos “siguiente”.
3. Elegiremos la carpeta donde instalará el programa.
4. La siguiente pantalla nos preguntara sobre las tareas adicionales que queremos realizar, una vez seleccionadas pulsaremos siguiente.
5. La penúltima pantalla, clicaremos sobre “instalar”. Una vez finalizado nos preguntará si deseamos ejecutarle por primera vez y pulsaremos “finalizar”.

⁴<https://www.apachefriends.org/index.html>

⁵<https://code.visualstudio.com/>

APÉNDICE D. DOCUMENTACIÓN TÉCNICA DE PROGRAMACIÓN

6. Ahora, cada vez que queremos iniciar el programa, sólo tendremos que pulsar sobre el ícono creado de Visual Studio Code para iniciar el programa.

D.4. Compilación, instalación y ejecución del proyecto

Descarga del proyecto

En este apartado explicáramos los pasos a realizar para poder tener la aplicación web en nuestro ordenador y poder coger la base de datos y poder operar con ella directamente.

Para la obtención del código y los recursos del proyecto, tendremos que dirigirnos al repositorio de GitLab desde el siguiente enlace⁶ ⁷.

Tendremos que descargar la carpeta XAMPP, primeramente habrá que descomprimirlo y guardarlo directamente en (C:), seguidamente dentro de la carpeta XAMPP encontraremos diferentes carpetas y documentos:

1. En dicha carpeta buscaremos el archivo **xampp_start.exe** ^{D.6}, clicaremos sobre ese ejecutable, a continuación daremos los permisos pertinentes para poder ejecutar el archivo.
2. Con esto se activaran los módulos de Apache y MySQL. Para cerciorarnos que están activados tendremos que ejecutar **xampp_control.exe** ^{D.6} y ejecutar desde los comandos *START* de Apache y MySQL que tenemos en fosforito en la imagen ^{D.7}.

Así ya tendremos nuestra base de datos local activada y tendremos que dirigirnos a nuestro navegador para poner en nuestro navegador: local-host/proyectoFinal (proyectoFinal, ya que es así como se llama la carpeta donde tenemos el código de la página web).

Una vez estemos visualizando la página web en el modo local, nos servirá de ayuda el punto Interfaz gráfica *Front-End* ^{E.4}

Una vez descargado el archivo y tengamos acceso a la página web tendremos que instalar Metamask ^{D.3}

⁶Repositorio GitLab: <https://gitlab.com/HP-SCDS/Observatorio/2018-2019/ubu-bc4distribution/tree/paginaWeb>

⁷Repositorio GitHub: <https://github.com/ers0026/ProyectoFinal>

D.4. COMPILACIÓN, INSTALACIÓN Y EJECUCIÓN DEL PROYECTO³⁷

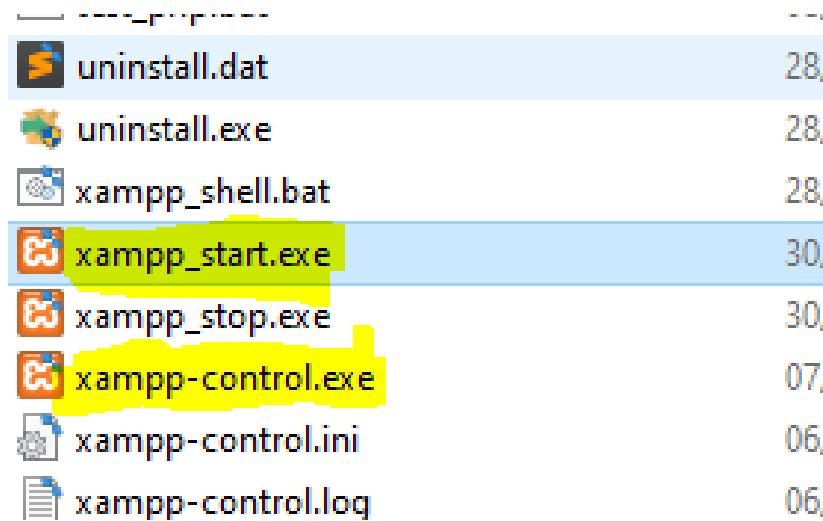


Figura D.6: Ejecución de XAMPP automáticamente

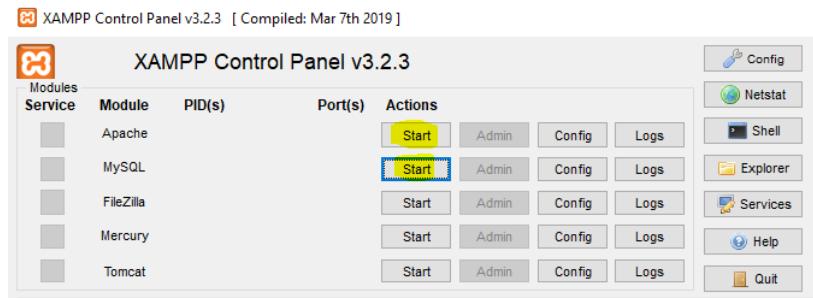


Figura D.7: Ejecución de XAMPP manualmente

Metamask: tendremos que estar registrados previamente o, si no, al añadir producto nos saldrá una pantalla para introducir la contraseña⁸ y a continuación nos desplegará el primer contrato. Para comenzar a funcionar tendremos que seleccionar la Red privada Ropsten.

⁸La contraseña la tendremos en un archivo en el GitLab, el archivo se llama mnemonic metamask.txt

Apéndice E

Documentación de usuario

E.1. Introducción

En el último apéndice, indicaremos lo que necesitarán los usuarios de nuestra aplicación.

E.2. Requisitos de usuarios

Enumeraremos los requisitos que se tienen que tener para poder usar la aplicación:

- Si fuera la primera vez que vamos trabajar con esta página, tendremos que descargarnos la base de datos y el código de la página del repositorio GitLab.
- La extensión de metamask, en el navegador en el que despleguemos la pagina web.
- Conexión a Internet, ya que la página web el código cuenta con iconos que cogemos directamente de internet.

E.3. Instalación

Únicamente tendremos que descargar la extensión de metamask y la base de datos y código del repositorio GitLab [D.4](#).

E.4. Manual del usuario

Este manual va a constar la parte del administrador y usuario, el administrador podrá controlar a los usuarios y los usuarios podrán crear añadir nuevos productos y visualizar los productos que el mismo realizo.

Interfaz gráfica *Front-End*

En este apartado mostraremos el aspecto final de nuestra aplicación.

En este apartado explicaremos las diferentes pantallas de nuestra página web, y como consigue realizar la conexión con el método de pago mediante Metamask, y comprobar mediante la red Ethereum que el contrato se creó correctamente.

Inicio de sesión [E.1](#): la primera pantalla, nos resultará familiar de otras páginas web, hemos intentado que la página sea concisa, fácil e intuitiva.

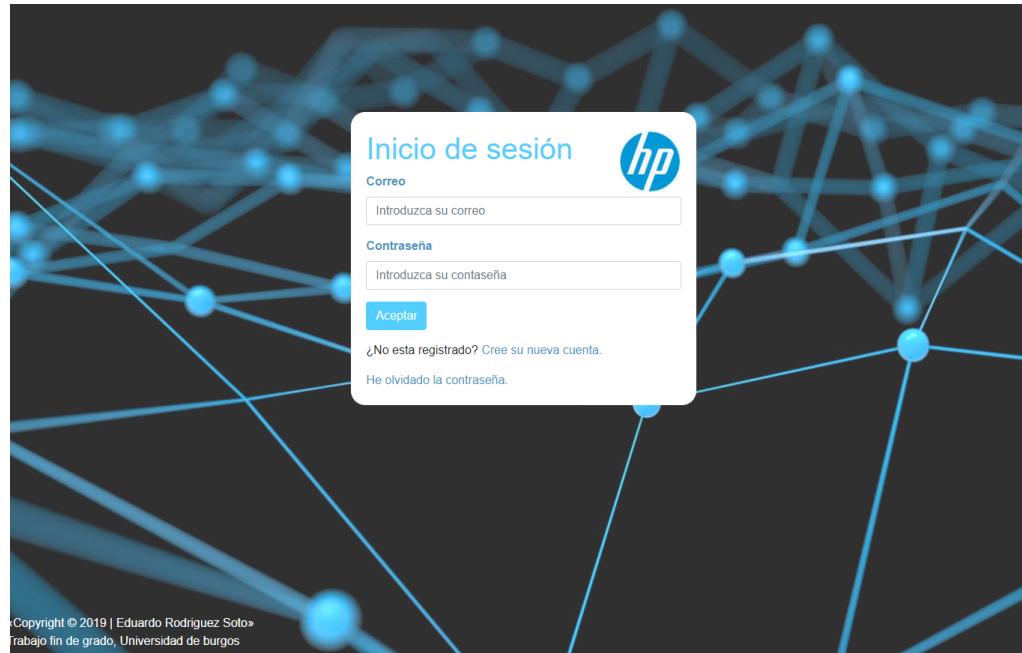


Figura E.1: Inicio de sesión

Tendremos que introducir el correo y la contraseña que previamente hayamos indicados a la hora de registrarnos y que tendremos en la misma

página en un enlace “¿No esta registrado? Cree su nueva cuenta”. En cualquier caso, si introducimos mal los datos, nos saltará mensaje de error y nos pedirá volver a iniciar sesión, teniendo la opción de poder enviarnos un correo de recordatorio de la contraseña al correo con el que nos registramos.

Crear nueva cuenta [E.2](#): al igual que muchas páginas web, hemos creado la pantalla de registro (a esta página llegaremos desde la pantalla de inicio de sesión en la cual tendremos un apartado, “¿No esta registrado? Cree su nueva cuenta”). Esta pantalla se compondrá de los campos: nombre, avatar, correo, contraseña, repetir contraseña y comprobación de que no eres un robot.

Todos los campos serán obligatorios, excepto el avatar, que en caso de no seleccionar ninguno, se le asignará el que tenemos por defecto.

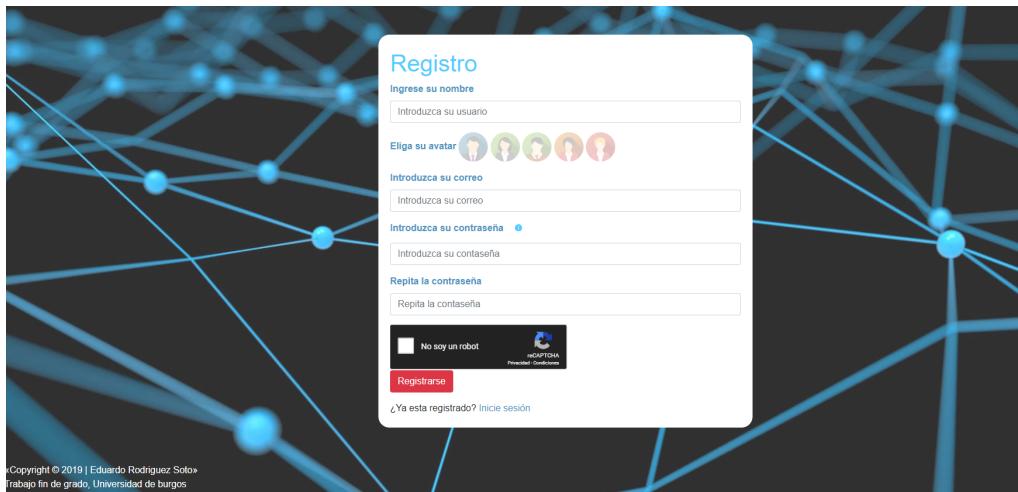


Figura E.2: Registrar nueva cuenta

La contraseña tendrá que tener un formato típico (incluyendo mayúscula, minúscula y un número, y tendrá que ser compuesta de una cadena entre 8 y 12 elementos). En el caso de no introducir bien ambas contraseñas nos mostrará un error y nos llevará a la misma pantalla nuevamente, para volver a realizar el registro del usuario. En el caso de meter un correo ya existente a la hora de registrarse nos informará que ese correo ya está registrado.

Olvido su contraseña [E.3](#) suponemos que todo el mundo comete errores, y uno de los mas comunes es poder olvidar la contraseña, por ello hemos

creado desde la pantalla de inicio de sesión, la opción de “He olvidado la contraseña”, pinchando en el enlace nos redirige a una página, la cual nos pedirá el correo del usuario, para así poder enviar un mensaje a su correo. Este nos dará una contraseña temporal para que el usuario pueda entrar en la cuenta y después modificar la contraseña, dependiendo si él quiere o no, en caso que el correo no esté en la base de datos, nos dará un mensaje de error de que ese usuario no está registrado¹.



Figura E.3: Recuperar contraseña

Página del administrador [E.4](#): una vez reconocemos todos los enlaces de la pantalla de inicio, es hora de poder entrar en la sesión. Tendremos dos maneras de entrar: mediante el registro en la aplicación o por el uso de administrador, el cual será (admin/admin). Tendremos una pantalla como la siguiente:

¹Esta página al trabajar en modo LocalHost, no estará habilitada, pero esta programada para funcionar cuando dispongamos de un dominio

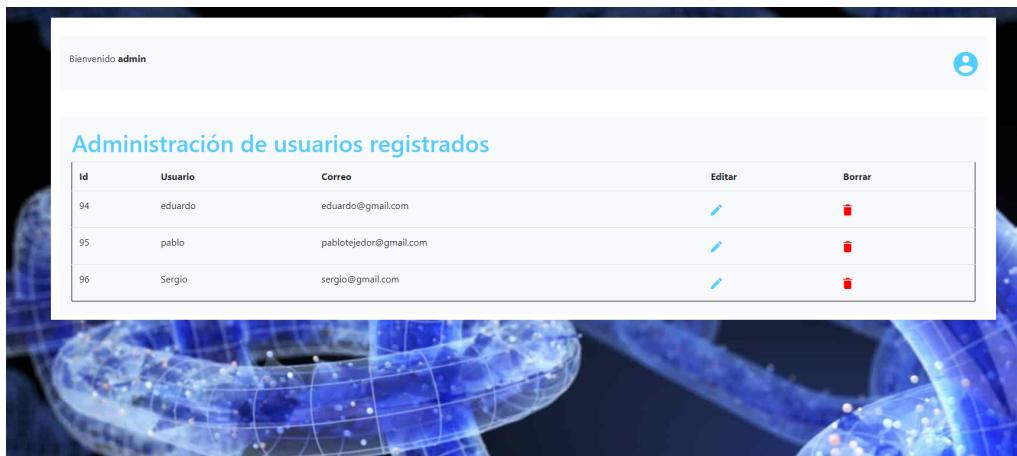


Figura E.4: Administrador de usuarios

El administrador será el responsable de poder gestionar y editar o borrar a los usuarios.

Administrador: *Editar* [E.5](#): en la cabecera nos informará mediante un mensaje de “Bienvenido ...” el usuario con el que se encuentra en cada momento. En la parte derecha tendremos un ícono del avatar y en esta pantalla nos permitirá cerrar sesión.

Si pulsamos en el lápiz azul de la columna editar de la fotografía [E.4](#) nos llevará a la siguiente pantalla:

Editar datos del usuario: eduardo

Id: 94
Usuario: eduardo
Correo: eduardo@gmail.com
Contraseña
Introduce nueva contraseña
Repetir contraseña
Repite la nueva contraseña

Guardar Volver

Figura E.5: Editar usuario desde administrador

Nos mostrará los campos del usuario, pero únicamente podremos cambiar usuario y la contraseña, y nunca estos campos podrán quedar vacíos si hemos introducido algún valor anteriormente.

Cuando guardemos nos indicará si todo fue correcto, y, en caso de haber puesto las contraseñas incorrectas o el usuario vacío, nos mostrará un mensaje de error y tendremos que volver a realizar la misma operación, esta vez teniendo más cuidado y poniendo todo correctamente.

Administrador: Borrar [E.6](#): así mismo, si queremos borrar tendremos que pulsar el ícono de la papelera de la imagen [E.4](#), con esta acción nos mostrará un mensaje indicando si deseamos realizar la acción o no.

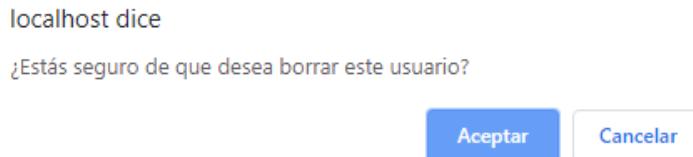


Figura E.6: Mensaje confirmación

Opciones [E.7](#): una vez quedan explicadas las pantallas nos centramos en las que se involucran con el usuario de forma directa en su sesión.

Tendremos varias opciones en la pantalla de selección, los cuales explicaremos a continuación:

- Añadir producto.
- Consultar el ultimo producto.
- Consultar todos los productos.
- Modificar datos (desde el avatar del usuario).

Añadir producto : cuando pulsamos sobre la opción de añadir producto, esta automáticamente nos conectará con Metamask, pidiendo la contraseña [E.8](#) (en caso de no estar logeados) o directamente nos dará la opción de conectar el proyecto con la cuenta que tengamos asociada de Metamask actualmente [E.9](#).

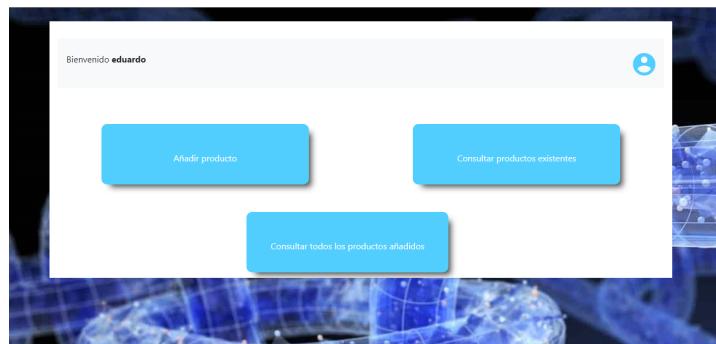


Figura E.7: Opciones a realizar por usuario

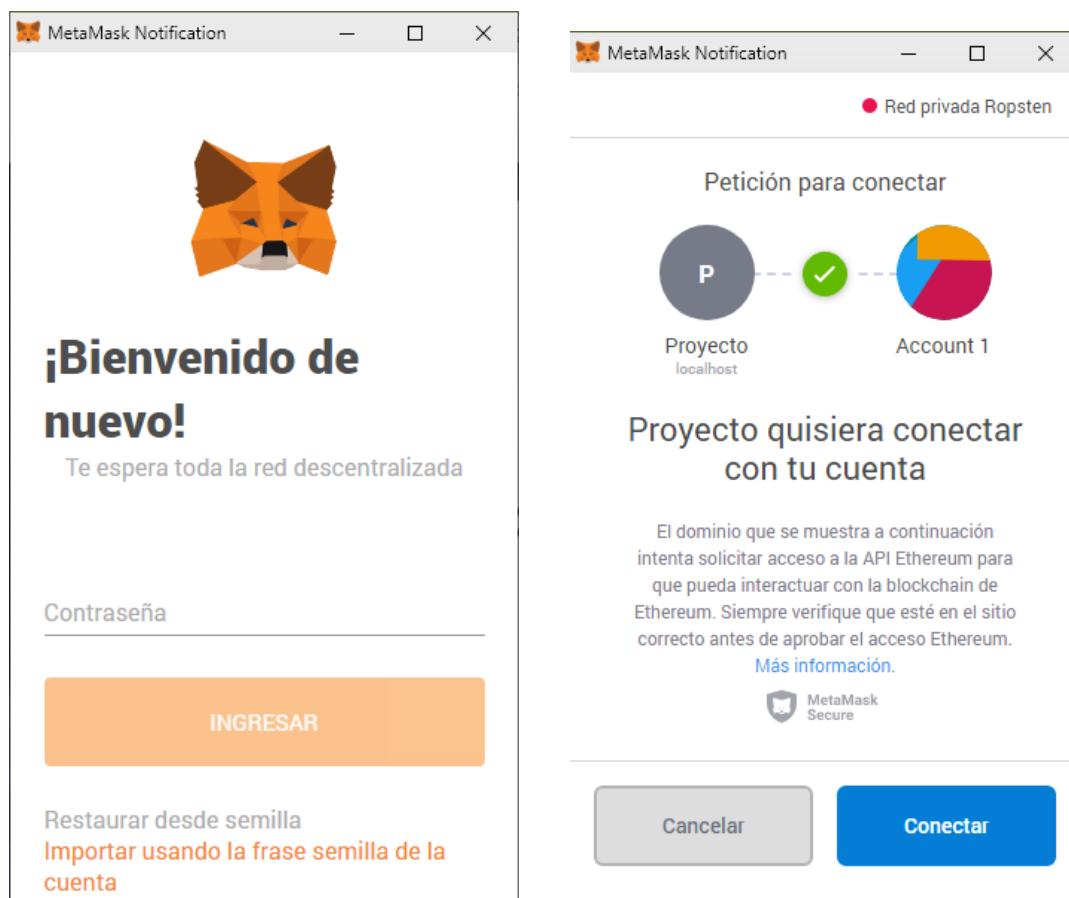


Figura E.8: Conexion Metamask

Figura E.9: Conexion proyecto metamask

Una vez hemos completado el acceso y la conexión a metamask, tendremos

que rellenar el producto que deseemos añadir **E.10**, que serán los datos que almacenaremos en nuestro smart contract en la red Ethereum.

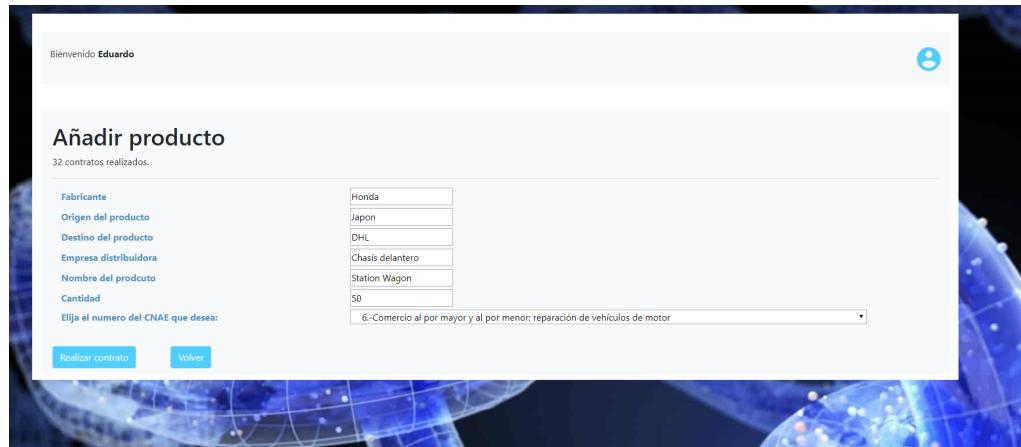


Figura E.10: Añadir producto

Una vez llenados los datos en nuestra pantalla, pulsaremos el botón de realizar contrato, esto nos mostrará nuevas pantallas que abrirá el conector de metamask para poder realizar el contrato **E.11**, son las siguientes:

Cuando el usuario realiza la acción de añadir un nuevo producto, antes de guardar todos los datos, primero se tiene que conectar con la red Ethereum, y hasta que no se crea el *smart contract*, estos datos no quedarán debidamente guardados en la base de datos.

Una vez que se haya confirmado la transacción **E.13** en metamask, los datos se habrán guardado tanto en nuestra base de datos como en la red Ethereum de la red privada Ropsten. Si por un causal hubiera fallo de red o no tuviéramos conexión a la red, estos datos se perderían y tendríamos que realizar el contrato nuevamente para guardarlos.

Para poder ver dicha transacción existen dos opciones: en la red Ethereum o bien mediante la nuestra página web.

En la red tendremos varias formas de llegar, pulsando directamente cuando nos sale el aviso de la **E.13**, también se puede llegar pulsando en el icono de Metamask y la transacción que deseemos pulsar sobre la fecha ver en Etherscan. Una vez pulsado nos redirige a una pagina, llamada ropsten.etherscan.io y el código de la transacción según cuál sea la que estamos trabajando.

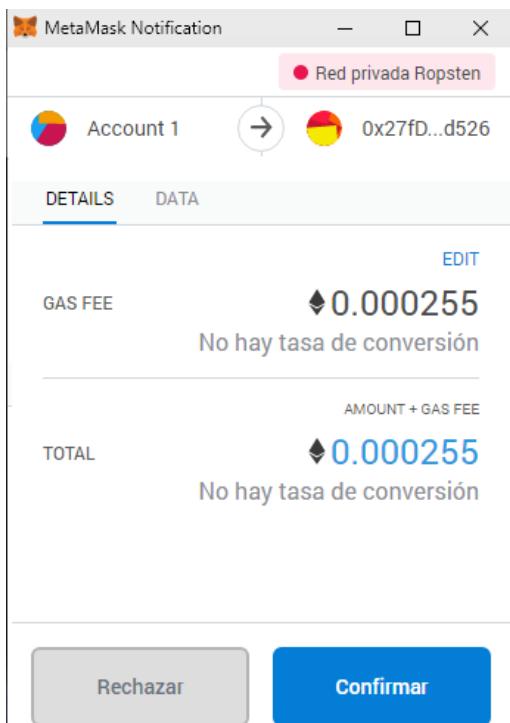


Figura E.11: Confirmar el contrato

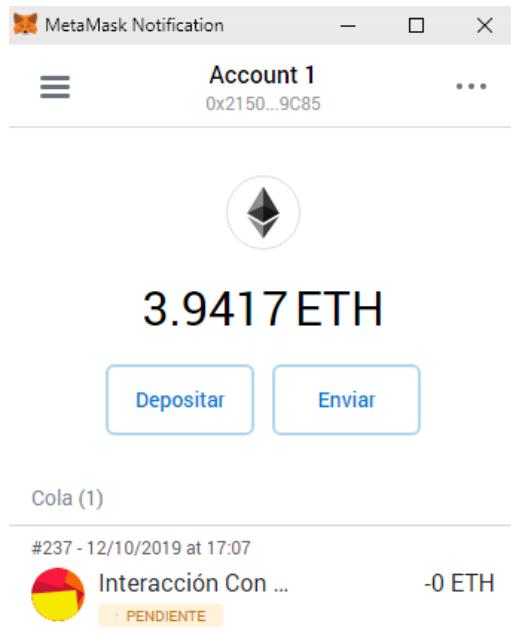


Figura E.12: Conexión pendiente

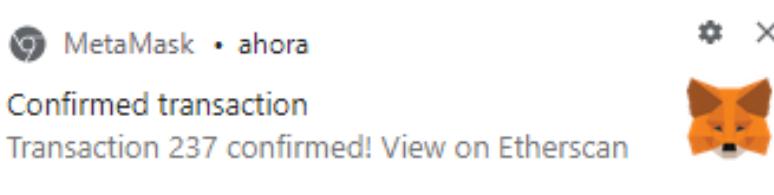


Figura E.13: Conexión confirmada

En esta nueva pagina podremos ver toda la información del contrato mediante dos pestañas encontradas en la parte superior de la página:

- **Overview E.14:** el estado y la hora de la transacción, la cuenta desde la que se hizo y a la cuenta que se realiza la transacción, el gas y el coste de ether.
- **Event Logs E.15:** en la cual podremos ver los datos que hemos introducido en nuestra pantalla de añadir producto(al principio nos

saldrán todas los datos en formato hexadécimal y convertirlo a formato text o formato numérico según el datos que estemos buscando).

Figura E.14: Información del contrato creado vistos por Etherscan

Figura E.15: Datos proporcionados por Etherscan

Una vez vistos todos estos datos en cuanto a la conexión de la red con Metamask y, esta a su vez, con la red Ropsten y Etherscan, ya tendríamos nuestro contrato ubicado en la red en todo momento, siempre que tengamos el enlace o bien la aplicación de metamask en la transacción que deseemos saber los datos.

Consultar todas las transacciones y la última transacción [E.16](#): en el apartado anterior se explicó como observar la transacción que se realizó en un momento dado gracias a la red Etherscan. La otra forma de poder observarlas es dar a “volver”, una vez confirmada la transacción. Lo que intentamos realizar con la creación de estos dos botones es que el usuario de la página no tenga que perder el tiempo entrando en esta red y poder buscarlo de forma más eficaz.

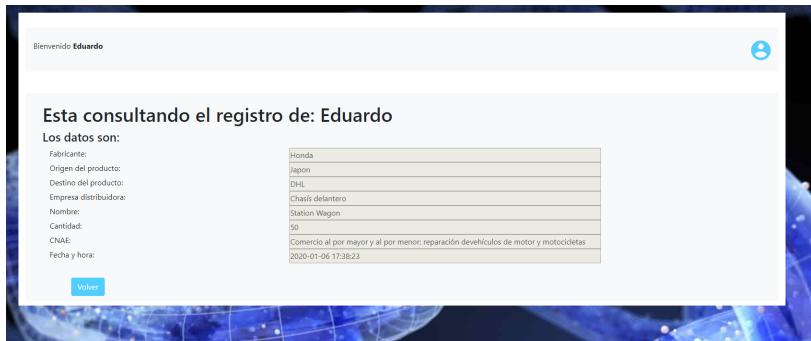


Figura E.16: Visualización del último producto

Figura E.17: Visualización de todos los productos realizados

Las transacciones que realicemos nunca se podrán modificar ni borrar.

Cierto es que en nuestra página (hemos decidido que mientras el usuario tenga una cuenta. Exista la transacción, pero si borramos al usuario, estaremos eliminado todos los contratos que creó de nuestra base de datos) pero en la red de Etherscan se podrán seguir visualizando de forma permanente, tal y como se explicó anteriormente.

Bibliografía

- [1] Henrik Kniberg, <http://www.proyctalis.com/wp-content/uploads/2008/02/scrum-y-xp-desde-las-trincheras.pdf>, 2019.
- [2] Diagrama de Gantt, https://es.wikipedia.org/wiki/Diagrama_de_Gantt, 2019.
- [3] Creación del diagrama de Gantt, <https://app.ganttpro.com/>, 2019.
- [4] Cotización seguridad social y salario medio, <http://www.seg-social.es/wps/portal/wss/internet/Trabajadores/CotizacionRecaudacionTrabajadores/36537>, 2019.
- [5] Definición de licencia, <https://es.wikipedia.org/wiki/Licencia>, 2019.
- [6] Licencia MIT, <https://es.wikipedia.org/wiki/Licencia/MIT>, 2019.
- [7] Diagrama de datos, <https://app.creately.com/diagram/pZV1YnnLlgd/edit>, 2019.