
# Universidad Valladolid

Escuela de Ingeniería Informática
TRABAJO FIN DE GRADO

Grado en Ingeniería Informática
Mención Ingeniería de software

# ContractMe: Aplicación Móvil Basada en Blockchain que Simplifica la Contratación y Verificación a Través de Contratos Inteligentes.

Alumno: **D. Pablo Gutiérrez Martínez**
Tutor Académico: **D. Jesús María Vegas Hernández**
Tutores de empresa: **D. Jorge Aparicio Bermejo**

# Agradecimientos

Quiero expresar mi más sincero agradecimiento a todas las personas que han contribuido al desarrollo y finalización de este Trabajo de Fin de Grado. A mi tutor académico, Jesús María Vegas Hernández, por su inestimable guía, apoyo y paciencia a lo largo de todo el proceso; a mis tutores de empresa, Julia Zuara Jiménez y Jorge Aparicio Bermejo, por sus valiosas orientaciones, conocimientos y apoyo práctico que han sido esenciales para llevar a cabo este proyecto; a mis profesores, cuyas enseñanzas y dedicación han sido fundamentales para mi formación; a mis compañeros de clase, por los momentos de colaboración y aprendizaje compartido; y a mi familia y amigos, por su constante apoyo y comprensión. Sin la ayuda y el apoyo de todos ellos, este logro no habría sido posible.

# Resumen

Este proyecto consiste en el desarrollo de una aplicación móvil Android orientada a la verificación y firma de contratos utilizando la tecnología Blockchain.

El principal motivo por el que se realiza este proyecto es para fomentar prácticas de contratación legales y transparentes, dando a conocer todos los detalles acerca de la creación y firma de un contrato, incluyendo quien es el creador de este, cuáles son sus objetivos, quien es el firmante o la remuneración. Todo ello se va a desarrollar sobre la red local de Ganache con la intención de alcanzar una buena eficiencia y hacer el desarrollo de manera segura y totalmente controlada.

En este documento se incluye la planificación del proyecto, que está basada en metodologías ágiles, concretamente Scrum.

Además, se presenta todo el proceso de desarrollo desde el análisis de requisitos, pasando por el diseño de la aplicación y la implementación de la misma. Todo ello utilizando el lenguaje Solidity para la creación de contratos inteligentes sobre una red de Blockchain y React Native para desarrollar el front-end.

# Abstract

This project involves the development of an Android mobile application aimed at the verification and signing of contracts using Blockchain technology.

The main reason for undertaking this project is to promote legal and transparent contracting practices, providing all details regarding the creation and signing of a contract, including who the creator is, what its objectives are, who the signatory is, and the remuneration involved. All of this will be developed on the local Ganache network with the aim of achieving good efficiency and conducting development in a secure and fully controlled environment.

This document includes the project planning, which is based on agile methodologies, specifically Scrum.

Additionally, it presents the entire development process from requirements analysis, through application design, to implementation. All of this will be done using the Solidity language for creating smart contracts on a Blockchain network and React Native for developing the front-end.

# Tabla de contenidos


**1 INTRODUCCIÓN ..... 1**

1.1 CONTEXTO ..... 1
1.2 MOTIVACIÓN ..... 1
1.3 OBJETIVOS ACADÉMICOS ..... 2
1.4 OBJETIVOS PERSONALES ..... 2
1.5 ESTRUCTURA DE LA MEMORIA ..... 3

**2 ESTADO DEL ARTE ..... 5**

2.1 FUNDAMENTOS DE LA BLOCKCHAIN ..... 5
2.2 INTRODUCCIÓN A LA BLOCKCHAIN ..... 5
2.3 TIPOS DE REDES ..... 6
2.4 ARQUITECTURA DE UNA RED BLOCKCHAIN ..... 6
2.5 TECNOLOGÍAS BLOCKCHAIN UTILIZADAS ..... 7
*2.5.1 Ethereum ..... 7*
*2.5.2 Solidity ..... 8*
*2.5.3 Truffle ..... 9*
*2.5.4 Ganache ..... 9*

**3 PLANIFICACIÓN DEL PROYECTO ..... 10**

3.1 INTRODUCCIÓN ..... 10
3.2 METODOLOGÍA UTILIZADA ..... 10
*3.2.1 Scrum ..... 10*
*3.2.2 Organización del equipo ..... 10*
*3.2.3 Eventos ..... 11*
*3.2.4 Artefactos ..... 11*
3.3 PLANIFICACIÓN ..... 12
*3.3.1 Planificación de los Sprint ..... 12*
*3.3.2 Análisis de riesgos ..... 13*
3.4 ANÁLISIS DE COSTES ..... 19
*3.4.1 Costes de mano de obra ..... 19*
*3.4.2 Costes de hardware ..... 20*
*3.4.3 Costes de software ..... 20*
*3.4.4 Costes totales ..... 21*

**4 ANÁLISIS ..... 22**

4.1 USUARIOS OBJETIVOS ..... 22
4.2 ANÁLISIS DE REQUISITOS ..... 22
*4.2.1 Requisitos Funcionales ..... 22*
*4.2.2 Requisitos no funcionales ..... 24*
*4.2.3 Reglas de negocio ..... 24*
4.3 CASOS DE USO ..... 25
*4.3.1 Descripción de los casos de uso ..... 25*
4.4 MODELO DE DOMINIO ..... 32
*4.4.1 Contrato ..... 32*
*4.4.2 Usuario ..... 33*
*4.4.3 Contratista ..... 33*
*4.4.4 Solicitante ..... 33*
*4.4.5 Firmante ..... 33*
*4.4.6 Evento ..... 33*
*4.4.7 Creación ..... 33*
*4.4.8 Modificación ..... 33*
*4.4.9 Firma ..... 34*

**5 DISEÑO ..... 36**

5.1 PATRÓN MVVM ..... 36
*5.1.1 Fundamento teórico ..... 36*
*5.1.2 Adaptación del patrón MVVM ..... 36*
*5.1.3 Ventajas de usar MVVM ..... 37*
5.2 DISEÑO MÓVIL ..... 37
*5.2.1 Diseño basado en componentes ..... 37*
*5.2.2 Componentes del proyecto ..... 38*
5.3 INTERFAZ DE USUARIO ..... 39

**6 IMPLEMENTACIÓN ..... 43**

6.1 HERRAMIENTAS DE DESARROLLO ..... 43
6.2 TECNOLOGÍAS UTILIZADAS ..... 43
*6.2.1 React Native ..... 43*
*6.2.2 Node ..... 44*
*6.2.3 Npm ..... 44*
*6.2.4 Web3.js ..... 44*
*6.2.5 Astah ..... 44*
*6.2.6 GitLab ..... 45*
*6.2.7 Expo y Expo Go ..... 45*
6.3 IMPLEMENTACION EN LA RED BLOCKCHAIN ..... 45
*6.3.1 Implementación del Smart Contract ..... 45*
6.4 IMPLEMENTACIÓN DEL CLIENTE ..... 48
*6.4.1 Estructura general ..... 48*
*6.4.2 Navegabilidad de la aplicación ..... 49*
*6.4.3 Conexión con Blockchain ..... 50*
*6.4.4 Registro e inicio de sesión ..... 51*
*6.4.5 Creación y modificación de contratos ..... 52*
*6.4.6 Firma de contratos ..... 53*

**7 PRUEBAS ..... 56**

7.1 PRUEBAS SOBRE EL INICIO DE SESIÓN Y REGISTRO DE USUARIOS ..... 56
7.2 PRUEBAS SOBRE LA CREACIÓN Y MODIFICACIÓN DE CONTRATOS ..... 58
7.3 PRUEBAS SOBRE LA FIRMA DE CONTRATOS ..... 59

**8 CONCLUSIONES Y TRABAJO FUTURO ..... 63**

8.1 CONCLUSIONES ..... 63
8.2 TRABAJO FUTURO ..... 63

**9 BIBLIOGRAFÍA ..... 65**

**10 ANEXO ..... 67**

10.1 GUÍA DE INSTALACIÓN Y USO ..... 67
*10.1.1 INSTALACIÓN DE NODE, NPM Y TRUFFLE ..... 67*
*10.1.2 INSTALACIÓN Y CONFIGURACIÓN DE GANACHE ..... 68*
*10.1.3 COMPILACIÓN Y DESPLIEGUE DEL SMART CONTRACT ..... 69*
*10.1.4 EJECUCIÓN DE LA APLICACIÓN ..... 69*

# Índice de ilustraciones

ILUSTRACIÓN 1 ESTRUCTURA POR CAPAS DE LA TECNOLOGÍA BLOCKCHAIN [4] ..... 6
ILUSTRACIÓN 2 LOGO DE ETHEREUM [6] ..... 8
ILUSTRACIÓN 3 LOGO DE SOLIDITY [7] ..... 8
ILUSTRACIÓN 4 LOGO DE TRUFFLE [8] ..... 9
ILUSTRACIÓN 5 LOGO DE GANACHE [8] ..... 9
ILUSTRACIÓN 6 ESQUEMA METODOLOGÍA SCRUM [11] ..... 12
ILUSTRACIÓN 7 DIAGRAMA DE CASOS DE USO ..... 25
ILUSTRACIÓN 8 MODELO DE DOMINIO ..... 32
ILUSTRACIÓN 9 ESQUEMA DEL PATRÓN MVVM [15] ..... 36
ILUSTRACIÓN 10 DIAGRAMA DE COMPONENTES ..... 38
ILUSTRACIÓN 11 INTERFACES DE PANTALLA INICIAL, DE REGISTRO Y DE INICIO DE SESIÓN ..... 39
ILUSTRACIÓN 12 INTERFAZ DE PANTALLA PRINCIPAL ..... 39
ILUSTRACIÓN 13 INTERFAZ DE CREACIÓN DE UN CONTRATO ..... 40
ILUSTRACIÓN 14 INTERFAZ DE MODIFICACIÓN DE UN CONTRATO ..... 41
ILUSTRACIÓN 15 ESTRUCTURAS Y MAPEOS DEL SMART CONTRACT ..... 46
ILUSTRACIÓN 16 EVENTOS DEL SMART CONTRACT ..... 47
ILUSTRACIÓN 17 IMPLEMENTACIÓN DE STACK.NAGIVATOR Y STACK.SCREEN ..... 49
ILUSTRACIÓN 18 CONEXIÓN CON BLOCKCHAIN ..... 50
ILUSTRACIÓN 19 OBTENCIÓN DEL SMART CONTRACT ..... 50
ILUSTRACIÓN 20 MÉTODO "HANDLEREGISTER" ..... 51
ILUSTRACIÓN 21 MÉTODO "HANDLELOGIN" ..... 52
ILUSTRACIÓN 22 LLAMADA AL MÉTODO "CREARCONTRATO" ..... 52
ILUSTRACIÓN 23 LLAMADA AL MÉTODO "MODIFICARCONTRATO" ..... 53
ILUSTRACIÓN 24 LLAMADA AL MÉTODO "SOLICITARFIRMA" ..... 53
ILUSTRACIÓN 25 LLAMADA AL MÉTODO "FIRMARCONTRATO" ..... 53
ILUSTRACIÓN 26 LLAMADA AL MÉTODO "RECHAZARFIRMA" ..... 54
ILUSTRACIÓN 27 CONFIGURACIÓN DE GANACHE ..... 68
ILUSTRACIÓN 28 CONFIGURACIÓN EN TRUFFLE-CONFIG.JS ..... 68
ILUSTRACIÓN 29 TRANSACCIÓN EN GANACHE ..... 69
ILUSTRACIÓN 30 CÓDIGO QR GENERADO POR EXPO ..... 70

# Índice de tablas

TABLA 1 PLANIFICACIÓN POR SPRINT ..... 13
TABLA 2 LEYENDA CÁLCULO MATRIZ DE RIESGOS [12] ..... 14
TABLA 3 MATRIZ DE RIESGOS ..... 15
TABLA 4 R1- DESARROLLADOR NO DISPONIBLE ..... 15
TABLA 5 R2-DIFICULTADES EN LA FORMACIÓN ..... 16
TABLA 6 R3- PÉRDIDA DE LOS ENTREGABLES ..... 16
TABLA 7 R4-IMPOSIBILIDAD PARA REALIZAR ALGUNA REUNIÓN ..... 17
TABLA 8 R5-NO DISPONIBILIDAD DE ALGUNO DE LOS TUTORES ..... 17
TABLA 9 R6-REQUISITOS MUY CAMBIANTES ..... 18
TABLA 10 R7-PLANIFICACIÓN INCORRECTA ..... 18
TABLA 11 R8-FALLOS EN LA CODIFICACIÓN ..... 19
TABLA 12 R9-FALTA DE RIESGOS ..... 19
TABLA 13 COSTES DE HARDWARE ..... 20
TABLA 14 COSTES DE SOFTWARE ..... 21
TABLA 15 COSTES TOTALES ..... 21
TABLA 16 REQUISITOS FUNCIONALES ..... 23
TABLA 17 REQUISITOS NO FUNCIONALES ..... 24
TABLA 18 REGLAS DE NEGOCIO ..... 24
TABLA 19 CU1-REGISTRO ..... 26
TABLA 20 CU2-IDENTIFICACIÓN ..... 26
TABLA 21 CU3-CONRTRATOS PROPIOS ..... 27
TABLA 22 CU4-CONTRATOS DISPONIBLES ..... 27
TABLA 23 CU5-CREAR CONTRATO ..... 28
TABLA 24 CU6-MODIFICAR CONTRATO ..... 28
TABLA 25 CU7-CONTRATOS FIRMADOS ..... 29
TABLA 26 CU8-SOLICITAR FIRMA ..... 29
TABLA 27 CU9-ACEPTAR FIRMA ..... 30
TABLA 28 CU10-REALIZAR PAGO ..... 30
TABLA 29 CU11-DETALLES CONTRATO ..... 31
TABLA 30 CU12-VER INFORMACIÓN DE LA APLICACIÓN ..... 31
TABLA 31 PIR1- REGISTRO DE UN NUEVO USUARIO CORRECTAMENTE ..... 56
TABLA 32 PIR2 - INICIO DE SESIÓN EXITOSO ..... 56
TABLA 33 PIR3 - DNI INVÁLIDO ..... 56
TABLA 34 PIR4 - USUARIO YA EXISTENTE ..... 57
TABLA 35 PIR5 - CONTRASEÑA INCORRECTA ..... 57
TABLA 36 PIR6 - ALGÚN CAMPO VACÍO ..... 57
TABLA 37 PIR7 - USUARIO INEXISTENTE ..... 57
TABLA 38 PCM1 - CREAR NUEVO CONTRATO CORRECTAMENTE ..... 58
TABLA 39 PCM2 - MODIFICAR CONTRATO CORRECTAMENTE ..... 58
TABLA 40 PCM3 - ALGÚN CAMPO VACÍO EN LA CREACIÓN DE UN CONTRATO ..... 58
TABLA 41 PCM4 - FECHA DE FINALIZACIÓN ANTERIOR A LA FECHA DE INICIO ..... 58
TABLA 42 PCM5 - ERROR EN EL PRECIO INTRODUCIDO ..... 59
TABLA 43 PCM6 - PRECIO INTRODUCIDO MENOR O IGUAL QUE 0 ..... 59
TABLA 44 PCM7 - MODIFICACIÓN DE CONTRATO POR PROPIETARIO INCORRECTO ..... 59
TABLA 45 PF1 - SOLICITUD DE FIRMA CORRECTA ..... 59
TABLA 46 PF2 - SOLICITUD INVÁLIDA POR ESTADO “EXPIRADO” ..... 60
TABLA 47 PF3 - SOLICITUD INVÁLIDA POR ESTADO “SOLICITADO” ..... 60
TABLA 48 PF4 - ACEPTAR SOLICITUD CORRECTAMENTE ..... 60
TABLA 49 PF5 -RECHAZAR SOLICITUD CORRECTAMENTE ..... 60
TABLA 50 PF6 - FONDOS INSUFICIENTES ..... 61

# 1 INTRODUCCIÓN

## 1.1 CONTEXTO

En el mundo actual, el proceso de contratación enfrenta múltiples desafíos, como la falta de transparencia, la discriminación y la ineficiencia en la gestión de datos. Estos problemas se ven afectados por sistemas centralizados que pueden ser propensos a la manipulación y el error humano.

Una solución prometedora es la adopción de una tecnología descentralizada, como blockchain. Esta tecnología ofrece un registro inmutable y transparente de las transacciones y las interacciones, lo que puede mejorar significativamente la confianza y la equidad en los procesos de contratación. Además, la descentralización puede facilitar una mejor protección de los datos personales y proporcionar una forma más eficiente y segura de validar las credenciales y las habilidades de los candidatos [1].

La implementación de la tecnología descentralizada en los procesos de contratación no solo aborda los problemas existentes, sino que también introduce nuevas posibilidades y ventajas. Por ejemplo, la naturaleza distribuida de la blockchain permite la creación de un ecosistema de contratación donde los registros de empleo y las habilidades de los candidatos pueden ser verificados de manera independiente, sin la necesidad de intermediarios. Esto no solo reduce los costos asociados con la verificación de antecedentes, sino que también acelera el proceso de contratación.

Además, la descentralización puede democratizar el acceso al empleo. Al eliminar barreras geográficas y burocráticas, los candidatos pueden competir en igualdad de condiciones por oportunidades laborales. Esto es particularmente valioso en un mercado laboral cada vez más globalizado, donde la diversidad y la inclusión se están convirtiendo en prioridades para las organizaciones que buscan innovar y expandirse.

Por otro lado, la adopción de esta tecnología también plantea desafíos, como la necesidad de una infraestructura tecnológica robusta y la superación de la curva de aprendizaje asociada con nuevas herramientas digitales. Además, existen preocupaciones sobre la privacidad y la seguridad de los datos, aunque la blockchain está diseñada para ser segura, su adopción masiva requiere garantizar la protección de la información personal contra el acceso no autorizado o el uso indebido.

## 1.2 MOTIVACIÓN

La motivación principal de este proyecto es el desarrollo de una aplicación móvil descentralizada utilizando la tecnología blockchain para facilitar la firma y gestión de contratos. A través de este sistema se busca ofrecer una herramienta que no solo sea segura y confiable por su naturaleza inmutable, sino que también sea accesible y fácil de usar para cualquier usuario con un dispositivo móvil.

Además, la aplicación permite a los usuarios gestionar sus acuerdos legales de manera autónoma, sin depender de terceros, lo que refleja un compromiso con la autonomía y la eficiencia, todo ello basándose en un diseño intuitivo y una interfaz amigable buscando simplificar la complejidad de los contratos inteligentes.

Por otro lado, lo que se busca al utilizar la tecnología blockchain es la creación de registros inmutables y totalmente seguros cuya modificación sea prácticamente imposible con la intención de eliminar el fraude de la información que se registre en cada contrato.

De esta manera se obtiene un ecosistema fiable en el que toda la información acerca de los contratos es transparente y se puede verificar mediante bloques la integridad de dichos contratos, proporcionando la máxima fiabilidad posible tanto al usuario que crea el contrato como al usuario que lo firma.

Por un lado, es posible crear un contrato indicando fecha de inicio, fecha de finalización y precio, además de la descripción asociada y por otro lado es posible consultar todos los contratos tanto activos, como inactivos y firmar aquellos contratos en los cuales sea posible la firma.

Por último, cabe destacar que toda la información se maneja de forma encriptada para favorecer a la privacidad de los usuarios.

## 1.3 OBJETIVOS ACADÉMICOS

A continuación, se van a enumerar los objetivos que se van a cubrir en este proyecto:

- Realizar una investigación y análisis de las principales tecnologías blockchain con el fin de profundizar en su utilidad y posible desarrollo.
- Desarrollar una aplicación móvil funcional que permita a los usuarios interactuar con los contratos inteligentes.
- Implementar una interfaz intuitiva y minimalista con el objetivo de facilitar a los usuarios el uso de la aplicación y favorecer a una experiencia de usuario satisfactoria.
- Desplegar los contratos en una red blockchain de pruebas controlada.
- Dar paso a posibles futuras investigaciones a partir del trabajo realizado.

## 1.4 OBJETIVOS PERSONALES

Los objetivos personales del autor a raíz de la realización de este trabajo son los siguientes:

- Comprender el funcionamiento de la tecnología blockchain.
- Profundizar en el uso de contratos inteligentes utilizando Solidity.
- Obtener experiencia tanto en la implementación como en el despliegue de una aplicación blockchain descentralizada.

- Investigar acerca del uso de aplicaciones descentralizadas en entornos reales.

- Poner en prácticas metodologías ágiles en la planificación y desarrollo del proyecto.

## 1.5 ESTRUCTURA DE LA MEMORIA

A continuación, se presenta la estructura que va a tener la memoria por capítulos:

- **Capítulo 2 - Estado del Arte:** En este capítulo se va a cubrir la investigación previa realizada acerca de blockchain, incluyendo el fundamento teórico y cuáles son las tecnologías y herramientas que se van a utilizar dentro del proyecto

- **Capítulo 3- Planificación:** En este capítulo se pretende cubrir el proceso de planificación, incluyendo la metodología que se ha seguido, así como el análisis de los riesgos y como se ha desarrollado el proyecto con respecto a lo planificado.

- **Capítulo 4- Análisis:** En este capítulo se incluirán el análisis de requisitos, los casos de uso y los modelos iniciales de la aplicación.

- **Capítulo 5- Diseño:** En este capítulo se incluirán el diseño de la aplicación y los prototipos de las interfaces.

- **Capítulo 6- Implementación:** En este capítulo se incluirán la implementación y las tecnologías utilizadas en la aplicación.

- **Capítulo 7- Pruebas:** En este capítulo se presentan las pruebas realizadas para verificar el correcto funcionamiento de la aplicación.

- **Capítulo 8- Conclusiones y trabajo futuro:** En este capítulo se explicarán las conclusiones del proyecto, además del trabajo futuro que se podría realizar.

Por último, en la parte final de la memoria se incluirá un **anexo** con un manual de instalación y uso de la aplicación.

# 2 ESTADO DEL ARTE

## 2.1 FUNDAMENTOS DE LA BLOCKCHAIN

En este segundo capítulo se va a presentar el marco teórico detrás de la tecnología blockchain, incluyendo una introducción acerca de la misma. Además, se presentarán los principios y la arquitectura de blockchain y se entrará en detalle acerca de las tecnologías utilizadas incluyendo explicaciones de aquellas en las que está basado el proyecto.

## 2.2 INTRODUCCIÓN A LA BLOCKCHAIN

Los inicios de la tecnología blockchain se remontan a 1991, por parte de los investigadores Stuart.Haber y W.Scott Stornetta, quienes introdujeron una solución para que los documentos digitales firmados en una fecha concreta no pudieran ser falsificados.

Tras esto, muchos otros científicos y entusiastas de la criptografía siguieron investigando, lo que en 2008 llevó a la creación de la primera criptomoneda impulsada por la tecnología blockchain conocida actualmente como bitcoin.

La tecnología blockchain, también conocida como cadena de bloques, es un tipo especial de base de datos o libro de registro digital descentralizado. De manera que los datos en la blockchain se organizan en bloques ordenados cronológicamente y protegidos por criptografía, por lo que cada bloque en la cadena contiene una lista de transacciones, y cada vez que se realiza una nueva transacción, se registra y se añade a la cadena de bloques del participante [2].

Esta tecnología se basa en el concepto de la **descentralización**, que se trata de ofrecer a los usuarios la posibilidad de controlar y tomar sus propias decisiones dentro de una red sin la necesidad de que exista una única entidad intermediario, como puede ser un gobierno o una corporación. De esta manera las transacciones son verificadas y registradas por una red distribuida de computadoras que trabajan juntas para mantener la integridad.

El funcionamiento general de este tipo de tecnología se inicia cuando se realiza una transacción, esta transacción se registra como un "bloque" de datos que contiene la información relevante, posteriormente cada bloque se añade a una red de bloques donde cada uno está conectado al anterior y al posterior. Estos bloques confirman el tiempo exacto y la secuencia de las transacciones, ya que cada uno de ellos cuentan con un refuerzo de verificación del bloque anterior que hace que la cadena no se pueda manipular por un sujeto malicioso, ofreciendo un sistema totalmente **inmutable** y confiable por quienes estén utilizando la red.

Por último, hay que mencionar la **criptografía** y los **mecanismos de consenso** como puntos clave dentro del ecosistema blockchain. Por un lado, la criptografía hace que la blockchain mantenga un registro de transacciones seguro, transparente y resistente a las manipulaciones. Por otro lado, los mecanismos de consenso se utilizan para que los usuarios puedan coordinarse en un entorno distribuido, de manera que todos los agentes que participan en el sistema acepten una única fuente de verdad.

## 2.3 TIPOS DE REDES

Existen diferentes tipos de redes blockchain, por un lado, nos encontramos con las **redes públicas**, por otra lado las **redes privadas** y en última instancia podemos hablar de **redes autorizadas** y **redes de consorcio**.

En cuanto a las **redes públicas** son aquellas a las que cualquiera puede unirse y participar, como bitcoin. Las desventajas son que requiere una gran potencia computacional, existe poca privacidad para las transacciones y la seguridad es débil [3].

En cuanto a las **redes privadas** podemos decir que son similares a las redes públicas, sin embargo, estas últimas son gestionadas por una organización, de manera que es dicha organización la que controla quien puede participar, ejecutar un protocolo de consenso y mantener el libro mayor compartido. Dependiendo del caso de uso, esto se traduce en un aumento significativo de la confianza entre los participantes [3].

En cuanto a las **redes autorizadas**, normalmente hablamos de una red blockchain privada con permisos establecidos, de manera que se imponen restricciones sobre quién puede participar en la red y en que transacciones a través de una invitación o permisos para unirse [3].

Por último, las **redes de consorcio** son aquellas en las que diferentes organizaciones comparten las responsabilidades de mantener una red blockchain. Hablamos de organizaciones preseleccionadas que determinan quién puede enviar transacciones o acceder a los datos. Una blockchain de consorcio es ideal para empresas en las que todos los participantes necesitan permisos y tienen una responsabilidad compartida [3].

## 2.4 ARQUITECTURA DE UNA RED BLOCKCHAIN

En una red distribuida de tipo blockchain, cada participante en la red mantiene, autoriza y actualiza nuevas entradas en una colección de bloques con transacciones en un orden específico que representan la estructura de la tecnología blockchain. Estas listas pueden guardarse como un archivo plano (en formato txt) o una simple base de datos, adoptando formas públicas, privadas o de consorcio [4]. Entrando más en detalle con la arquitectura dentro de una red blockchain encontramos 5 capas diferentes.

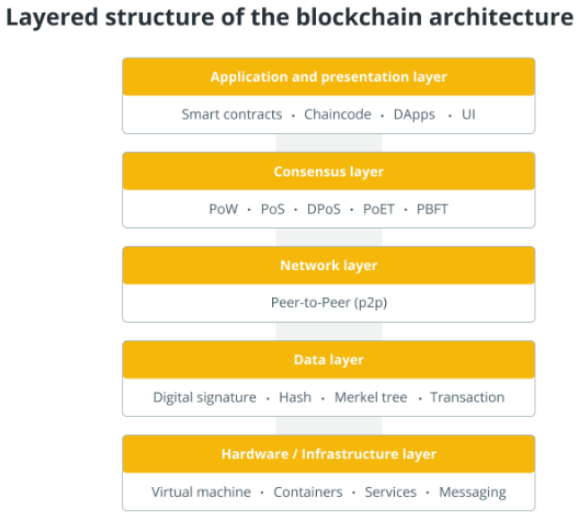


*Ilustración 1 Estructura por capas de la tecnología blockchain [4]*

La **capa de hardware** es la más importante de la blockchain ya que los nodos que realizan transacciones contables están alojados aquí. Las redes entre pares constituyen la base de las plataformas blockchain y los nodos de estas redes se conectan para que varios dispositivos puedan intercambiar y compartir datos entre sí. Durante las transacciones, cada nodo de la red puede realizar un seguimiento de datos aleatorio.

Como se ha mencionado anteriormente (ver 2.1), una característica distintiva de las redes blockchain es que la información sobre un bloque recién extraído se actualiza simultáneamente para todos los nodos, independientemente del grupo de nodos que lo compruebe. De esta manera, nos encontramos con nodos de validación que funcionan como servidor y nodos de actualización funcionan como cliente.

La **capa de datos** de la blockchain consta de dos elementos fundamentales: una lista enlazada donde se guardan los datos de forma encadenada y punteros utilizados para que cada bloque haga referencia a otro bloque dentro de la cadena. Para garantizar la seguridad, integridad e inmutabilidad del sistema, cada bloque contiene detalles como la dirección raíz del árbol de Merkle, el hash del bloque anterior o la fecha de creación del bloque.

Además, cabe destacar que se utiliza una clave privada para firmar las transacciones, y cualquiera que tenga la clave pública puede verificar al firmante.

La **capa de red** es la capa de distribución donde se produce la creación y la adición de bloques además de la comunicación entre nodos. También es responsable de administrar y mantener la infraestructura de la red de la blockchain. Los nodos y mineros usan la capacidad de cálculo para resolver problemas matemáticos complejos en esta capa para transmitir datos transaccionales de forma coherente a través de la red entre pares y validar transacciones.

La **capa de consenso** es donde se establecen los mecanismos de consenso entre nodos y es responsable de la funcionalidad más crítica, de manera que a través de algoritmos y protocolos se garantiza que todos los nodos de la red puedan verificar y acordar datos de las transacciones de manera descentralizada y sin necesidad de una autoridad central. En cada instante que ocurra una transacción varios nodos controlan los datos de la transacción y todos los demás nodos deben autenticar dicha transacción.

En la **capa de aplicación** se encuentran los programas de usuario finales que se utilizan para comunicarse con la red blockchain, incluyendo los contratos inteligentes o las interfaces de aplicaciones (API). Esta capa la red blockchain sirve como una tecnología back-end para crear aplicaciones descentralizadas (Dapps) que estarán alojadas en una capa intermedia de ejecución.

## 2.5 TECNOLOGÍAS BLOCKCHAIN UTILIZADAS

Una vez se ha introducido el marco teórico detrás de la tecnología blockchain lo siguiente que se va a abordar son las tecnologías y herramientas utilizadas en este proyecto en concreto.

### 2.5.1 Ethereum

Ethereum es una red de computadoras de todo el mundo que siguen un conjunto de normas conocidas como el protocolo Ethereum. La red Ethereum actúa como la base de comunidades, aplicaciones, organizaciones y activos digitales que cualquiera puede construir y utilizar. [5].

El Ether (ETH) es la criptomoneda que se utiliza dentro de la plataforma como método de pago. Cuando se envía ETH se paga una tarifa como incentivo para que un productor de bloques procese

y verifique lo que se está tratando de hacer, de esta manera se asegura que nadie hace trampas. Además, los validadores reciben pequeñas cantidades de ETH como incentivo. Por último, cabe destacar que todas estas transacciones son orquestadas gracias a los **contratos inteligentes**.

Las principales ventajas que ofrece Ethereum son la capacidad para ejecutar contratos inteligentes, la facilidad de verificación a la hora de realizar operaciones y la amplia libertad para todo tipo de usuarios, incluyendo desarrolladores que deseen construir aplicaciones descentralizadas o usuarios sin conocimientos técnicos que quieran aprovechar las características que ofrece la red. Todos estos son motivos suficientes para elegir Ethereum como ecosistema central en el que se basará el proyecto.

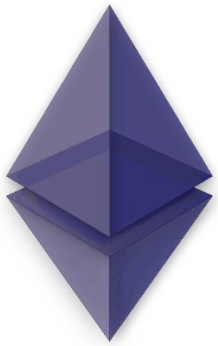

*Ilustración 2 Logo de Ethereum [6]*

### 2.5.2 Solidity

Solidity es un lenguaje de alto nivel orientado a contratos. Su sintaxis es similar a la de JavaScript y está enfocado específicamente a la Máquina Virtual de Etehreum (EVM). Está tipado de manera estática y acepta, entre otras cosas, herencias, librerías y tipos complejos definidos por el usuario [7].

A pesar de que existen otros lenguajes que ofrecen mayor seguridad, como Vyper, se ha decidido utilizar Solidity debido a que este último cuenta con una mayor comunidad de desarrollo, además de una mayor compatibilidad a la hora de desarrollar aplicaciones descentralizadas, incluyendo multitud de herramientas, bibliotecas y frameworks. Todo esto junto con la similitud a otros lenguajes conocido como JavaScript hace que la elección de Solidity como lenguaje para crear los contratos inteligentes sea la más adecuada.

*Ilustración 3 Logo de Solidity [7]*

### 2.5.3 Truffle

Truffle es un entorno de desarrollo de clase mundial, un marco de pruebas y una canalización de activos para blockchains que utilizan la máquina virtual Ethereum (EVM), con el objetivo de facilitar la vida al desarrollador [8].

Las principales características y funcionalidades que ofrece son la compilación, vinculación y gestión bancaria de contratos inteligentes, mecanismos de depuración avanzados con puntos de interrupción, análisis de variables y funcionalidad de pasos, una consola interactiva para la comunicación directa con el contrato y la posibilidad de migración de contratos inteligentes a un gran número de redes tanto públicas como privadas. Todas estas características hacen que Truffle sea un entorno idóneo para la creación, compilación y despliegue de contratos inteligentes.

*Ilustración 4 Logo de Truffle [8]*

### 2.5.4 Ganache

Ganache es una blockchain personal para el desarrollo rápido de aplicaciones distribuidas de Ethereum. Permite desarrollar, implementar y probar aplicaciones descentralizadas (dApps) en un entorno seguro y determinista. Además, cuenta con dos versiones: Interfaz de Usuario e interfaz por la línea de comandos [8].

Entre las funcionalidades que ofrece Ganache encontramos la posibilidad de crear una red local o privada, además de la creación de cuentas precargadas con un numero personalizado de Ether y un registro de eventos. Todas estas funcionalidades son muy útiles para desarrollar una aplicación basada en blockchain de manera sencilla y entendiendo en todo momento como funcionan cada una de las interacciones que se van a realizar en la red.

Aunque sería posible utilizar otras redes públicas como por ejemplo Infura, se ha decido utilizar Ganache debido a que al ser una red local ofrece una mayor transparencia y facilita el control tanto de usuarios como de transacciones. De esta manera es posible crear un entorno completamente supervisado sobre un concepto tan complejo como es blockchain.

*Ilustración 5 Logo de Ganache [8]*

# 3 PLANIFICACIÓN DEL PROYECTO

## 3.1 INTRODUCCIÓN

Durante este capítulo se va a presentar la planificación del proyecto, describiendo que metodologías ágiles se han utilizado y como se ha realizado la planificación a lo largo del tiempo.

## 3.2 METODOLOGÍA UTILIZADA

### 3.2.1 Scrum

Scrum es un marco de trabajo ágil a través del cual las personas pueden abordar problemas complejos adaptativos a la vez que se entregan productos de forma eficiente y creativa con el máximo valor. Así, Scrum es una metodología que ayuda a los equipos a colaborar y realizar un trabajo de alto impacto. La metodología Scrum proporciona un plan de valores, roles y pautas para ayudar a tu equipo a concentrarse en la iteración y la mejora continua en proyectos complejos [9].

Dicho esto, se puede concluir que la metodología Srum es idónea para este proyecto en concreto, ya que estamos ante un tipo de proyecto donde los requisitos son cambiantes y es necesaria una adaptabilidad ante estos cambios para asegurar la entrega del proyecto en la fecha esperada con éxito.

### 3.2.2 Organización del equipo

En cuanto a la organización de equipo dentro del marco Scrum, este suele estar compuesto por grupos de entre 3 a 9 miembros del equipo de desarrollo, más el Scrum Máster y el Product Owner. Cada uno tiene una serie de responsabilidades y juegan papeles bien diferenciados.

- **Product Owner**: Es quien optimiza y maximiza el valor del producto. Entre sus funciones se incluyen la interacción con los stakeholders, la representación de los requerimientos del cliente y la definición de los valores del producto. En cada Sprint el Product Owner establece metas claras y trabaja estrechamente con el equipo de desarrollo para tomar decisiones que afecten al producto y transmitírselas al cliente.

- **Scrum Máster**: El Scrum Master tiene dos funciones principales. En primer lugar, es el encargado de asegurar la correcta ejecución de las mecánicas Scrum y, en segundo lugar, su labor incluye la eliminación de impedimentos que puedan surgir durante el proyecto, garantizando que se cumple de manera íntegra con la metodología Scrum.

- **Equipo de desarrollo**: Son los encargados de desarrollar el producto, de manera que se auto organizan y se auto gestionan para conseguir entregar los incrementos de software al final de cada ciclo del desarrollo.

Una vez introducidos de manera teórica como es la organización de un equipo, en el caso concreto de este proyecto el alumno será el Product Owner y a la vez será el único miembro del equipo de desarrollo. Por otro lado, tanto el tutor académico como los tutores de empresa serán los Scrum Master.

### 3.2.3 Eventos

Los eventos son bloques de tiempo de una duración determinada cuya finalidad es evitar reuniones innecesarias gracias a la regularidad y la consistencia.

Existes diferentes tipos de eventos dentro del marco Scrum. A continuación, se presentan los principales:

- **Sprint**: Es un periodo de tiempo fijo y corto durante el cual se desarrolla un incremento del producto. Por lo general, los Sprints tienen una duración de una a cuatro semanas, durante la cual el equipo de desarrollo trabaja para completar los elementos seleccionados del Product Backlog y entregar un incremento del producto.

- **Reunión de planificación**: Al comienzo de cada Sprint se realiza una reunión de planificación. En esta reunión, el Product Owner presenta los elementos de mayor prioridad del Product Backlog, y el equipo de desarrollo selecciona los elementos que se comprometerá a implementar durante dicho Sprint.

- **Daily Scrum**: Es una reunión diaria de 15 minutos que se realiza durante todos los días hábiles que dura el Sprint. En esta reunión, el equipo de desarrollo inspecciona el progreso realizado durante el Sprint y adapta el plan para alcanzar el objetivo acordado.

- **Revisión del Sprint**: Se revisa todo el trabajo realizado por el equipo de desarrollo para determinar qué elementos del Product Backlog se han completado.

- **Retrospectiva del Sprint**: Se realiza posteriormente a la revisión del sprint y tiene como objetivo obtener una reflexión acerca de los resultados del sprint para detectar dificultades y proponer mejoras.

Una vez introducidos todos los tipos de evento, en el caso concreto de este proyecto cada Sprint durará entre una y dos semanas en función de las necesidades y funcionalidades a implementar. En cuanto a las reuniones de planificación, estas serán realizadas los miércoles a las 15:00h por medio de videoconferencia, además al inicio de estas reuniones se realizará una revisión del Sprint anterior y una retrospectiva del Sprint. Por otro lado, no se realizarán Daily Scrums y en su lugar en caso de que en cualquier momento sea necesaria alguna consulta por parte del alumno a los tutores o viceversa se realizará por medio de correo electrónico.

### 3.2.4 Artefactos

Los artefactos son información que un equipo de scrum y las partes interesadas utilizan para detallar el producto en desarrollo, las acciones para producirlo y las tareas realizadas durante el proyecto. Estos artefactos ofrecen metadatos que dan una idea del rendimiento de un sprint. Son herramientas esenciales para todos los equipos de scrum, ya que posibilitan los atributos básicos de transparencia, inspección y adaptación [10].

Los principales artefactos del Scrum ágil son:

- **Product Backlog**: Se trata de una lista dinámica que prioriza todas las funcionalidades, características, mejoras y correcciones que se desean implementar en el producto.

- **Sprint Backlog**: Es una lista de elementos del Product Backlog seleccionados para el Sprint actual, junto con un plan para entregar el incremento del producto y alcanzar el objetivo del Sprint. Una vez establecido es inmutable durante en desarrollo del sprint

- **Incremento**: Es la suma de todos los elementos del Product Backlog completados durante un Sprint y el valor de los incrementos de todos los Sprints anteriores.

Tras haber explicado de manera teórica que son los artefactos y que tipos existen, cabe indicar que en este proyecto se ha utilizado la plataforma GitLab para registrar todos los Sprints e incrementos dentro del Product Backlog. Esto se ha debido a la facilidad de integrar código y planificación en una misma plataforma.

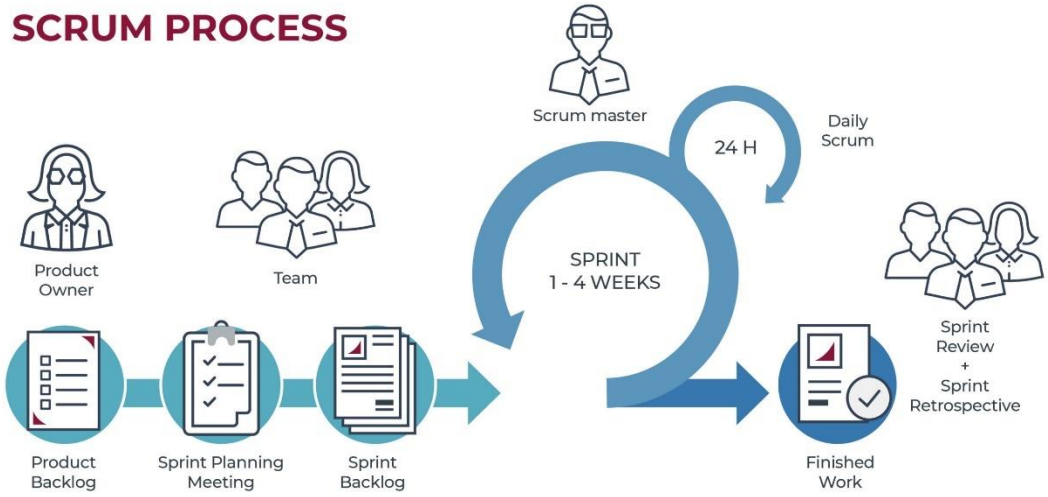


*Ilustración 6 Esquema metodología Scrum [11]*

## 3.3 PLANIFICACIÓN

### 3.3.1 Planificación de los Sprint

A continuación, se presenta la planificación inicial de los sprint, cuya duración estimada es de una semana. La fecha de inicio estimada es el 28 de febrero de 2024 y la fecha límite de entrega del proyecto es el 7 de julio de 2024(convocatoria extraordinaria). Se ha decidido estimar el proyecto para entregarlo en convocatoria extraordinaria debido a la disponibilidad del desarrollador Pablo Gutiérrez Martínez de manera que haya una mayor garantía de entregar un proyecto de calidad a tiempo. Durante este periodo se planea dedicar un total de 300 horas divididas en 18 sprint, es decir, un sprint por semana empleando una media aproximada de 16,6 horas a la semana en el desarrollo del proyecto.

En la siguiente tabla se presenta la planificación de manera global, indicando las fechas de inicio y fin de cada sprint, además de una descripción donde se indica el objetivo de cada uno.

| | Fecha inicio | Fecha Fin | Objetivo Principal |
|---|---|---|---|
| **Sprint 1** | **28/2/2024** | **6/3/2024** | Investigación de la tecnología blockchain |
| **Sprint 2** | **6/3/2024** | **13/3/2024** | Investigación de la tecnología blockchain |
| **Sprint 3** | **13/3/2024** | **20/3/2024** | Aprendizaje del lenguaje Solidity |
| **Sprint 4** | **20/3/2024** | **27/3/2024** | Aprendizaje del lenguaje Solidity |
| **Sprint 5** | **27/3/2024** | **3/4/2024** | Planteamiento de requisitos |
| **Sprint 6** | **3/4/2024** | **10/4/2024** | Diseño de la aplicación |
| **Sprint 7** | **10/4/2024** | **17/4/2024** | Diseño de la aplicación |
| **Sprint 8** | **17/4/2024** | **24/4/2024** | Creación de interfaces con Figma |
| **Sprint 9** | **24/4/2024** | **1/5/2024** | Implementación del Smart Contract |
| **Sprint 10** | **1/5/2024** | **8/5/2024** | Implementación de la conexión con la blockchain |
| **Sprint 11** | **8/5/2024** | **15/5/2024** | Implementación de inicio de sesión, incluyendo registro de usuarios y asignación de carteras |
| **Sprint 12** | **15/5/2024** | **22/5/2024** | Implementación de la creación, modificación y firma de contratos |
| **Sprint 13** | **22/5/2024** | **29/5/2024** | Implementación de la creación, modificación y firma de contratos |
| **Sprint 14** | **29/5/2024** | **5/6/2024** | Implementación de la creación, modificación y firma de contratos |
| **Sprint 15** | **5/6/2024** | **12/6/2024** | Implementación de la interfaz de usuario |
| **Sprint 16** | **12/6/2024** | **19/6/2024** | Implementación de la interfaz de usuario |
| **Sprint 17** | **19/6/2024** | **26/6/2024** | Retocar últimos detalles y pruebas |
| **Sprint 18** | **26/6/2024** | **3/7/2024** | Retocar últimos detalles y pruebas |

*Tabla 1 Planificación por Sprint*

### 3.3.2 Análisis de riesgos

En este apartado se analizarán los posibles riesgos que pueden ocurrir, la probabilidad de que estos ocurran, el impacto que pueden tener y las consecuencias de estos. Además, se tendrá en cuenta el nivel de cada riesgo y el plan tanto de prevención como de respuesta de cada uno.

Para facilitar el entendimiento, a continuación, se proporciona una leyenda para el cálculo de una matriz de riesgos donde se puede observar de manera gráfica cómo se clasifican los riesgos en función de su probabilidad e impacto en una escala del 1 al 5, de manera que en la intersección de ambos podemos determinar el nivel de importancia de cada riesgo.

**LEYENDA**

| | | | GRAVEDAD (IMPACTO) | | | | |
|---|---|---|---|---|---|---|---|
| | | | MUY BAJO **1** | BAJO **2** | MEDIO **3** | ALTO **4** | MUY ALTO **5** |
| **PROBABILIDAD** | MUY ALTA | **5** | 5 | 10 | 15 | 20 | 25 |
| | ALTA | **4** | 4 | 8 | 12 | 16 | 20 |
| | MEDIA | **3** | 3 | 6 | 9 | 12 | 15 |
| | BAJA | **2** | 2 | 4 | 6 | 8 | 12 |
| | MUY BAJA | **1** | 1 | 2 | 3 | 4 | 5 |

| | |
|---|---|
| | Riesgo muy grave. Requiere medidas preventivas urgentes. No se debe iniciar el proyecto sin la aplicación de medidas preventivas urgentes y sin acotar sólidamente el riesgo. |
| | Riesgo importante. Medidas preventivas obligatorias. Se deben controlar fuertemente llas variables de riesgo durante el proyecto. |
| | Riesgo apreciable. Estudiar económicamente ai es posible introducir medidas preventivas para reducir el nivel de riesgo. Si no fuera posible, mantener las variables controladas. |
| | Riesgo marginal.Se vigilará aunque no requiere medidas preventivas de partida. |

*Tabla 2 Leyenda cálculo matriz de riesgos [12]*

Seguidamente, se realizará una clasificación de los posibles riesgos y se analizará cada uno de ellos en detalle. Por otro lado, para realizar los cálculos necesarios nos guiaremos de la figura anterior.

A continuación, se muestra una matriz de riesgos donde obtenemos el nivel de cada riesgo en base a la probabilidad y el impacto de cada uno.

| Riesgo | Probabilidad | Impacto | Valor del Riesgo | Nivel de Riesgo |
|---|---|---|---|---|
| R1 - El desarrollador no dispone de suficiente tiempo o se pone enfermo | 2 | 5 | 10 | Importante |
| R2 – El desarrollador tiene dificultades para aprender las nuevas tecnologías | 3 | 4 | 12 | Importante |
| R3 - Se pierde parte del código o documentación del proyecto | 1 | 4 | 4 | Apreciable |
| R4 – No es posible realizar alguna de las reuniones planificadas | 2 | 1 | 2 | Marginal |
| R5 - Alguno de los tutores se ausenta debido a una baja | 2 | 3 | 6 | Apreciable |
| R6 – Los requisitos del proyecto son demasiado cambiantes | 3 | 4 | 12 | Importante |
| R7- Se realiza la planificación de manera incorrecta | 3 | 4 | 12 | Importante |
| R8- Aparecen fallos de codificación en la aplicación | 4 | 3 | 12 | Importante |
| R9 – No se han planteado todos los riesgos | 2 | 3 | 6 | Apreciable |

*Tabla 3 Matriz de riesgos*

Posteriormente se presentan en detalle cada uno de los riesgos, incluyendo una descripción, las posibles consecuencias y los planes de contingencia y mitigación

| | |
|---|---|
| *Identificador* | *R1* |
| *Nombre* | Desarrollador no disponible |
| *Descripción* | El desarrollador sufre una baja médica o se encuentra disponible |
| *Impacto* | Muy Alto |
| *Probabilidad* | Baja |
| *Nivel de riesgo* | Importante |
| *Consecuencia* | No es posible desarrollar la aplicación |
| *Plan de mitigación* | Planificar de manera previa que hacer en caso de que el desarrollador no esté disponible |
| *Plan de contingencia* | Modificar la planificación |

*Tabla 4 R1- Desarrollador no disponible*

| | |
|---|---|
| *Identificador* | *R2* |
| *Nombre* | Dificultades en la formación |
| *Descripción* | El desarrollador tiene dificultades a la hora de aprender las nuevas tecnologías |
| *Impacto* | Alto |
| *Probabilidad* | Media |
| *Nivel de riesgo* | Importante |
| *Consecuencia* | Se utiliza mayor tiempo del necesario en formación del desarrollador |
| *Plan de mitigación* | Pedir ayuda a los tutores |
| *Plan de contingencia* | Revisar los conocimientos adquiridos |

*Tabla 5 R2-Dificultades en la formación*

| | |
|---|---|
| *Identificador* | *R3* |
| *Nombre* | Pérdida de los entregables |
| *Descripción* | Se pierden los entregables, ya sea el código o la memoria |
| *Impacto* | Alto |
| *Probabilidad* | Muy baja |
| *Nivel de riesgo* | Importante |
| *Consecuencia* | No sería posible realizar la entrega del proyecto |
| *Plan de mitigación* | Hacer uso de un sistema de control de versiones como GitLab |
| *Plan de contingencia* | Recuperar copias de seguridad |

*Tabla 6 R3- Pérdida de los entregables*

| | |
|---|---|
| *Identificador* | *R4* |
| *Nombre* | Imposibilidad para realizar alguna reunión |
| *Descripción* | No es posible realizar alguna de las reuniones de seguimiento con alguno de los tutores |
| *Impacto* | Muy Bajo |
| *Probabilidad* | Baja |
| *Nivel de riesgo* | Marginal |
| *Consecuencia* | No sería posible revisar el sprint correspondiente y planificar el siguiente de manera correcta |
| *Plan de mitigación* | No aplica |
| *Plan de contingencia* | Realizar la reunión otro día |

*Tabla 7 R4-Imposibilidad para realizar alguna reunión*

| | |
|---|---|
| *Identificador* | *R5* |
| *Nombre* | No disponibilidad de alguno de los tutores |
| *Descripción* | Alguno de los tutores no está disponible por motivos médicos o cualquier otro motivo externo |
| *Impacto* | Medio |
| *Probabilidad* | Baja |
| *Nivel de riesgo* | Apreciable |
| *Consecuencia* | Posible confusión por parte del alumno para desarrollar el proyecto |
| *Plan de mitigación* | No aplica |
| *Plan de contingencia* | Tratar de resolver los problemas que surjan de manera individual o solicitar un nuevo tutor |

*Tabla 8 R5-No disponibilidad de alguno de los tutores*

| | |
|---|---|
| *Identificador* | *R6* |
| *Nombre* | Requisitos muy cambiantes |
| *Descripción* | Los requisitos del proyecto son muy cambiantes y ambiguos |
| *Impacto* | Alto |
| *Probabilidad* | Media |
| *Nivel de riesgo* | Importante |
| *Consecuencia* | Confusión por parte del desarrollador |
| *Plan de mitigación* | Definir correctamente cuales son los requisitos del proyecto |
| *Plan de contingencia* | Realizar modificaciones en los requisitos del proyecto |

*Tabla 9 R6-Requisitos muy cambiantes*

| | |
|---|---|
| *Identificador* | *R7* |
| *Nombre* | Planificación incorrecta |
| *Descripción* | La planificación se realiza de manera incorrecta e incoherente, teniendo fallos en la estimación de tiempo y recursos |
| *Impacto* | Alto |
| *Probabilidad* | Media |
| *Nivel de riesgo* | Importante |
| *Consecuencia* | No se cumplen las fechas de entrega propuestas |
| *Plan de mitigación* | Revisar detalladamente de la planificación por parte de los tutores |
| *Plan de contingencia* | Modificar el Product Backlog |

*Tabla 10 R7-Planificación incorrecta*

| | |
|---|---|
| *Identificador* | *R8* |
| *Nombre* | Fallos en la codificación |
| *Descripción* | La planificación se realiza de manera incorrecta e incoherente, teniendo fallos en la estimación de tiempo y recursos |
| *Impacto* | Medio |
| *Probabilidad* | Alta |
| *Nivel de riesgo* | Importante |
| *Consecuencia* | Código de baja calidad |
| *Plan de mitigación* | Seguir buenas prácticas de codificación |
| *Plan de contingencia* | Refactorizar el código |

*Tabla 11 R8-Fallos en la codificación*

| | |
|---|---|
| *Identificador* | *R8* |
| *Nombre* | Falta de riesgos |
| *Descripción* | No se han planteado todos los riesgos |
| *Impacto* | Medio |
| *Probabilidad* | Baja |
| *Nivel de riesgo* | Apreciable |
| *Consecuencia* | Aparición de nuevos riesgos inesperados |
| *Plan de mitigación* | Planificar todos los riesgos posibles |
| *Plan de contingencia* | Analizar un posible nuevo riesgo de manera detenida |

*Tabla 12 R9-Falta de riesgos*

## 3.4 ANÁLISIS DE COSTES

En este apartado se va a realizar un análisis de los costes estimados para el desarrollo del proyecto, incluyendo el costo humano y el costo de los recursos utilizados, tanto a nivel de hardware como de software.

Hay que tener en cuenta que para realizar los cálculos se han estimado un total de 300 horas para realizar este proyecto.

### 3.4.1 Costes de mano de obra

El sueldo medio de un ingeniero informático Junior en España es de 24.000 euros brutos anuales [13]. Suponiendo una media de 40 horas semanales de trabajo y teniendo en cuenta que un año

tiene 52 semanas, podemos estimar un total de 2.080 horas aproximadamente para dicho salario, de manera que obtenemos una media de unos 11,54 €/hora. Una vez sabiendo esto si lo multiplicamos por las 300 horas que ocupa este proyecto obtenemos un total de 3.462€ como coste de la mano de obra.

### 3.4.2 Costes de hardware

Para realizar este proyecto se ha utilizado un portátil y una pantalla auxiliar. El portátil tiene un costo de 900€ y la pantalla de 200€. A mayores se han utilizado periféricos como un teclado, un ratón y un cargador y su coste total ha sido de 50€.

Suponiendo que la vida útil de cada dispositivo es de entre 3 a 4 años aproximadamente y que un dispositivo es utilizado una media de 4 horas al día, para poder calcular el costo total del hardware aplicado al proyecto se ha tenido en cuenta esta suposición, de manera que podamos obtener el precio por hora de uso de cada dispositivo y así poder calcular el coste total.

A continuación, se muestra una tabla con la información relacionada con los costes del hardware.

| Dispositivo | Precio | Horas de uso totales | Precio/hora | Horas de uso en el proyecto | Coste |
|---|---|---|---|---|---|
| Ordenador portátil | 900 € | 5844 h (4 años, 4 horas/dia | 0,154 €/hora | 300 horas | 46,2 € |
| Pantalla auxiliar | 200 € | 4384 h (3 años, 4 horas/dia | 0,0456 €/hora | 300 horas | 13,68 € |
| Periféricos | 50 € | 4384 h (3 años, 4 horas/dia | 0,0114 €/hora | 300 horas | 3,42 € |

*Tabla 13 Costes de hardware*

Si realizamos la suma de todos los costes, obtenemos que el coste total es de **63,3€.**

### 3.4.3 Costes de software

En cuanto al software la mayoría de las herramientas utilizadas han sido gratuitas, por lo que en este caso solo debemos de prestar atención a aquellas herramientas que sean de pago. Por un lado, se ha utilizado la licencia "Microsoft 365 Empresa Premium" cuyo coste al mes es de 20,6€ y si tenemos en cuenta los 5 meses de desarrollo del proyecto obtenemos un total de 103€. Por otro lado, el portátil en el que se ha realizado este proyecto utiliza Windows 11, cuyo coste de la licencia "Windows 11 home" es de 145€. Por último, se ha utilizado la licencia de "Astah professional" cuyo coste es de 36 €/año.

En la siguiente tabla se muestra de manera resumida todo el software utilizado incluyendo sus costes:

| Software | Costo |
|---|---|
| Windows 11 home | 145 € |
| VS Code | 0 € |
| Gitlab | 0 € |
| Ganache | 0 € |
| Figma | 0 € |

| | |
|---|---|
| Astah | 36 € |
| Microsoft 365 | 103 € |
| **Total** | 284 € |

*Tabla 14 Costes de software*

### 3.4.4 Costes totales

Una vez analizados los costes del proyecto por separado, podemos obtener cual es el coste estimado para la realización de este proyecto. Para ello se ha tenido en cuenta la suma de todos ellos y se han mostrado los resultados en la siguiente tabla.

| Elemento | Costo |
|---|---|
| Software | 284 € |
| Hardware | 63,3 € |
| Mano de obra | 3.462 € |
| **Total** | 3.809,3 € |

*Tabla 15 Costes totales*

# 4 ANÁLISIS

A lo largo de este capítulo se va a realizar un análisis de los requisitos del proyecto, además de los usuarios objetivo a los que va dirigida la aplicación y un análisis de casos de usos.

## 4.1 USUARIOS OBJETIVOS

Con la realización de este proyecto se pretende abarcar a una serie de usuarios con diferentes objetivos y características.

El objetivo principal de esta aplicación es que los usuarios individuales puedan mantener el control sobre sus acuerdos y datos personales de forma segura y sin sacrificar su privacidad. Gracias a las características que brinda la blockchain, se espera que los usuarios puedan crear y firmar contratos con total transparencia y confianza.

Además, la aplicación podría ser beneficiosa para empresas y emprendedores que buscan agilizar y asegurar sus transacciones comerciales y acuerdos de comunicación ya que les permite automatizar el cumplimiento de acuerdos, reducir la posibilidad de disputas y minimizar la necesidad de intermediarios costosos. Por otro lado, la naturaleza descentralizada de la blockchain permite una seguridad mejorada contra fraudes y ataques cibernéticos.

Finalmente, organizaciones sin ánimo de lucro y entidades gubernamentales también se podrían ver beneficiadas ya que a menudo manejan fondos y contratos donde la transparencia es crucial, por lo que la aplicación puede ayudar a rastrear el flujo de fondos y asegurar que se cumplen los términos y condiciones de los contratos, lo que es esencial para mantener la confianza pública y cumplir con las regulaciones legales y éticas.

## 4.2 ANÁLISIS DE REQUISITOS

El análisis de requisitos en un proyecto software consiste en identificar, documentar y gestionar las necesidades del proyecto que se va a desarrollar. En este proceso se recoge información detallada sobre las funcionalidades, características y restricciones que se deben cumplir. De esta manera se establecen las bases para posteriormente realizar el diseño y la implementación.

### 4.2.1 Requisitos Funcionales

Los requisitos funcionales son especificaciones detalladas de las funcionalidades del sistema que definen las interacciones entre el sistema y los usuarios. Aquí se incluyen aspectos relacionados con la entrada/salida de datos, procesamiento de la información y comportamiento del sistema. A continuación, se muestra una tabla que describe los requisitos funcionales de este proyecto:

| ID | Requisito | Descripción |
|---|---|---|
| **RF1** | **Creación de contrato** | El sistema deberá permitir crear un contrato |
| **RF2** | **Modificación de contrato** | El sistema deberá permitir modificar un contrato |
| **RF3** | **Firmar contrato** | El sistema deberá permitir firmar un contrato |
| **RF4** | **Contratos creados** | El sistema deberá permitir la visualización de los contratos creados por un usuario |
| **RF5** | **Contratos firmados** | El sistema deberá permitir la visualización de los contratos firmados |
| **RF6** | **Contratos disponibles para firmar** | El sistema deberá permitir la visualización de los contratos que están disponibles para firmar |
| **RF7** | **Registro de usuario** | El sistema deberá permitir al usuario registrase en la aplicación |
| **RF8** | **Inicio de sesión** | El sistema deberá permitir a usuarios registrados iniciar sesión |
| **RF9** | **Vuelta atrás** | El sistema deberá permitir volver o la pantalla anterior en cualquier momento |
| **RF10** | **Actualizar listas** | El sistema deberá permitir actualizar las listas de contratos cada vez que el usuario decida |
| **RF11** | **Información sobre la aplicación** | El sistema deberá permitir la visualización de la información referida a la descripción de la aplicación |
| **RF12** | **Detalles de un contrato sin firmar** | El sistema deberá permitir la visualización de todos los detalles de un contrato sin firmar |
| **RF13** | **Detalles de un contrato firmado** | El sistema deberá permitir la visualización de todos los detalles de un contrato firmado |
| **RF14** | **Cambio entre ETH y euros** | El sistema deberá permitir la conversión entre ETH y euros teniendo en cuenta el valor de ETH a tiempo real |
| **RF15** | **Visualización de cartera** | El sistema deberá permitir a cada usuario visualizar la cartera que se le ha asignado |
| **RF16** | **Detalles del propietario** | El sistema deberá permitir la visualización de todos los detalles de la persona que ha creado el contrato |
| **RF17** | **Detalles del firmante** | El sistema deberá permitir la visualización de todos los detalles de la persona que ha firmado el contrato |

*Tabla 16 Requisitos funcionales*

### 4.2.2 Requisitos no funcionales

Los requisitos no funcionales son especificaciones que describen los criterios que pueden ser utilizador para el funcionamiento de un sistema, de manera que describen atributos de calidad y restricciones en el sistema. A continuación, se muestra una tabla que recoge los requisitos no funcionales de este proyecto:

| ID | Requisito | Descripción |
|---|---|---|
| **RNF1** | **Red Ethereum** | El sistema deberá de utilizar una red Ethereum |
| **RNF2** | **Aplicación móvil** | El sistema deberá ser desarrollado como una aplicación móvil para Android |
| **RNF3** | **Utilización de Truffle** | El sistema utilizará el ecosistema de Truffle para todo aquello relacionado con la tecnología blockchain |
| **RNF4** | **Seguridad** | El sistema deberá garantizar la seguridad a la hora de crear, modificar y firmar contratos |
| **RNF5** | **Interfaz sencilla e intuitiva** | El sistema deberá proporcionar al usuario una interfaz sencilla e intuitiva |
| **RNF6** | **Formato UTF-8** | El sistema deberá utilizar el formato UTF-8 en todos los textos |

*Tabla 17 Requisitos no funcionales*

### 4.2.3 Reglas de negocio

Las reglas de negocio son directrices específicas que determinan como se debe operar en el contexto comercial de la aplicación. A continuación, se muestra una tabla con las reglas de negocio de este proyecto.

| ID | Requisito | Descripción |
|---|---|---|
| **RDN1** | **Uso responsable** | Loa usuarios deberán de utilizar la aplicación de forma responsable y no realizar acciones fraudulentas |
| **RDN2** | **Transparencia** | Los usuarios deben ser transparentes a la hora de crear los contratos, mostrando toda la información necesaria |
| **RDN3** | **Cumplimiento legal** | Los usuarios deben de cumplir con toda la legalidad a la hora de crear y firmar contratos |

*Tabla 18 Reglas de negocio*

## 4.3 CASOS DE USO

En esta sección se van a presentar los casos de uso del sistema. El objetivo principal es brindar una visión global de las interacciones del usuario con el sistema de modo que se entienda el funcionamiento de la aplicación.

En cuanto al diagrama de casos de uso se presenta teniendo en cuenta un solo actor como representante de una dirección de cartera en la red de blockchain. De esta manera se simplifica la representación y es más fácilmente entendible. Así mismo se podría generalizar a un número de usuarios mayor que 1. A continuación se presenta el diagrama de casos de uso ya mencionado:

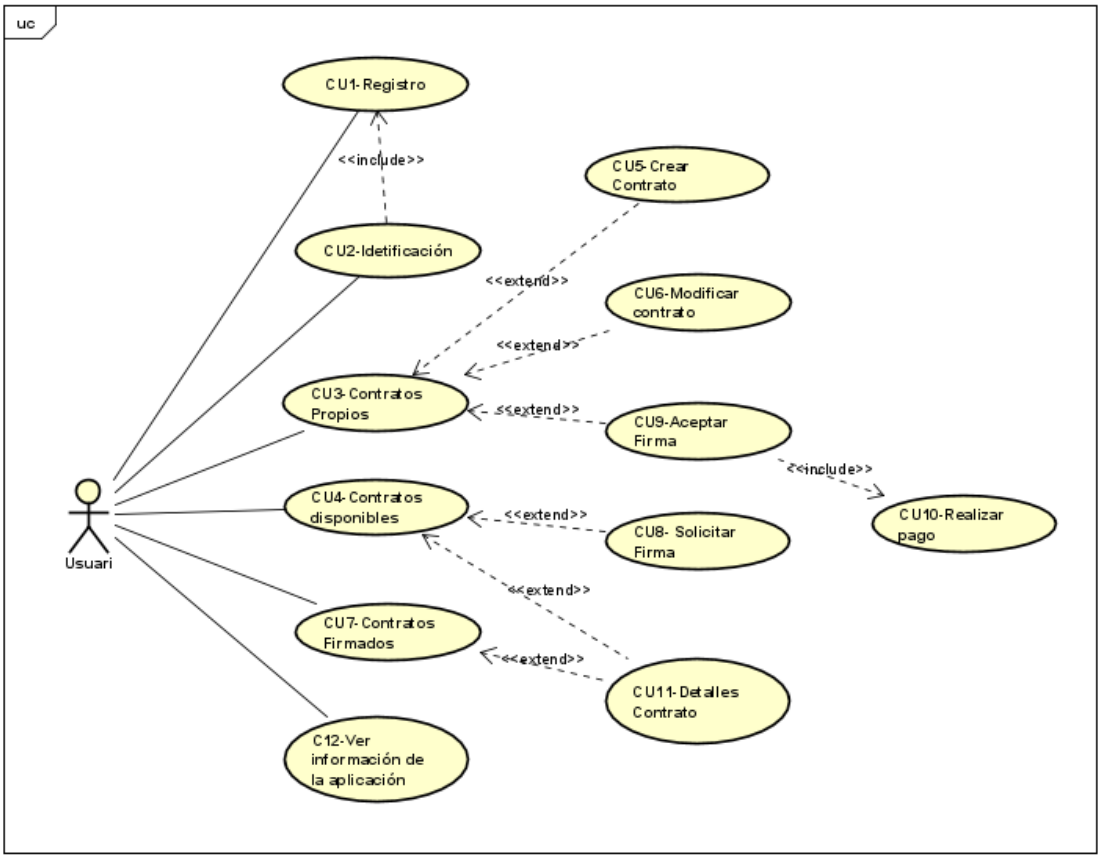


*Ilustración 7 Diagrama de casos de uso*

### 4.3.1 Descripción de los casos de uso

A continuación, se presenta una descripción de cada caso de uso indicando su flujo normal, así como su flujo alternativo o excepciones según corresponda. Para identificar cada caso de uso se ha utilizado la nomenclatura CU más un número de identificación.

| CU01 | Registro |
|---|---|
| **Descripción** | El usuario se registra en el sistema |
| **Actor** | Usuario |
| **Precondiciones** | Debe de haber una cartera disponible para asignar al usuario |
| **Postcondiciones** | El sistema muestra la cartera a la que está asociado el usuario |
| **Flujo normal** | 1. El usuario inicia la aplicación<br>2. El usuario pulsa el botón "Registrarse" en la pantalla de inicio<br>3. El usuario introduce su nombre de usuario, DNI y contraseña<br>4. El sistema asigna una cartera al usuario<br>5. El sistema redirige al usuario a la pantalla "Iniciar sesión" |
| **Flujo Alternativo** | 3.A El usuario pulsa el botón de volver y se ejecuta el CU6-Ver información de la aplicación |
| **Excepciones** | 3.B Si el usuario escribe un DNI incorrecto se le informará<br>3.C Si el usuario escribe un nombre se usuario ya existente se le informará |

*Tabla 19 CU1-Registro*

| CU02 | Identificación |
|---|---|
| **Descripción** | El usuario inicia sesión en el sistema |
| **Actor** | Usuario |
| **Precondiciones** | El usuario debe estar registrado |
| **Postcondiciones** | El sistema muestra la cartera a la que está asociado el usuario |
| **Flujo normal** | 1. El usuario inicia la aplicación<br>2. El usuario pulsa el botón "Iniciar Sesión" en la pantalla de inicio<br>3. El usuario introduce su nombre de usuario y contraseña<br>4. El sistema redirige al usuario a la pantalla principal |
| **Flujo Alternativo** | 3.A El usuario pulsa el botón de "volver" y se ejecuta el CU6-Ver información de la aplicación |
| **Excepciones** | 3.B Si el usuario escribe mal su contraseña se le informará |

*Tabla 20 CU2-Identificación*

| CU03 | Contratos propios |
|---|---|
| **Descripción** | El usuario consulta la lista de contratos que han sido creados por él |
| **Actor** | Usuario |
| **Precondiciones** | El usuario debe estar conectado a la red blockchain y se encuentra en la pantalla principal |
| **Postcondiciones** | Los contratos mostrados han sido creados por el usuario |
| **Flujo normal** | 1. El usuario pulsa el botón de “Mis Contratos”<br>2. El sistema muestra un listado con los contratos creados |
| **Flujo Alternativo** | 2.A El usuario pulsa el botón de volver y se ejecuta el CU6-Ver información de la aplicación<br>2.B El usuario no ha creado contratos aún y el sistema muestra un mensaje indicándolo |

*Tabla 21 CU3-Conrtratos propios*

| CU04 | Contratos disponibles |
|---|---|
| **Descripción** | El usuario consulta la lista de todos los contratos que están disponibles para firmar o que ya han sido firmados |
| **Actor** | Usuario |
| **Precondiciones** | El usuario debe estar conectado a la red blockchain y se encuentra en la pantalla principal |
| **Postcondiciones** | Los contratos consultados no son propiedad del usuario que está consultando |
| **Flujo normal** | 1. El usuario pulsa el botón de “Contratos disponibles”<br>2. El sistema muestra un listado de contratos disponibles para firmar o que ya hayan sido firmados |
| **Flujo Alternativo** | 2.A El usuario pulsa el botón de volver y se ejecuta el CU6-Ver información de la aplicación<br>2.B Aun no existen contratos disponibles y el sistema muestra un mensaje indicándolo |

*Tabla 22 CU4-Contratos disponibles*

| CU05 | Crear Contrato |
|---|---|
| **Descripción** | El usuario crea un contrato |
| **Actor** | Usuario |
| **Precondiciones** | El usuario debe estar conectado a la red blockchain y se encuentra en la pantalla principal |
| **Postcondiciones** | El contrato es añadido como propiedad del usuario que lo ha creado |
| **Flujo normal** | 1. El usuario pulsa el botón de "Añadir contrato"<br>2. El sistema redirecciona al usuario a la pantalla "Crear contrato"<br>3. El usuario rellena todos los campos necesarios<br>4. El usuario pulsa el botón "Crear contrato"<br>5. El sistema pide confirmación al usuario<br>6. El usuario confirma la acción<br>7. El sistema informa de que el contrato ha sido creado<br>8. El sistema redirige al usuario a la pantalla principal |
| **Flujo Alternativo** | 3.A El usuario pulsa el botón de volver y se ejecuta el CU6-Ver información de la aplicación<br>6.A El usuario cancela la acción y el caso de uso vuelve al paso 2 |
| **Excepciones** | 3.B Si el usuario no rellena todos los campos necesarios se le informará<br>3.C Si el usuario selecciona una fecha anterior a la actual se le informará<br>3.D Si el usuario introduce caracteres inválidos en el precio se le informará |

*Tabla 23 CU5-Crear contrato*

| CU06 | Modificar Contrato |
|---|---|
| **Descripción** | El usuario modifica un contrato |
| **Actor** | Usuario |
| **Precondiciones** | El contrato debe de haber sido creado por el usuario y no haber sido firmado aún |
| **Flujo normal** | 1. El usuario pulsa el botón de "Mis Contratos"<br>2. Se ejecuta de CU3-Contratos propios<br>3. El usuario pulsa sobre uno de los contratos de la lista<br>4. El sistema redirecciona al usuario a la pantalla "Modificar"<br>5. El usuario realiza los cambios que desee en el contrato<br>6. El usuario pulsa el botón "Guardar Cambios<br>7. El sistema pide confirmación al usuario<br>8. El usuario confirma la acción<br>9. El sistema informa de que el contrato ha sido modificado<br>10. El sistema redirecciona al usuario a la pantalla principal |
| **Flujo Alternativo** | 3.A El usuario pulsa el botón de volver y se ejecuta el CU6-Ver información de la aplicación<br>5.A El usuario pulsa el botón de volver y se ejecuta el CU3- Contratos propios<br>8.A El usuario cancela la acción y el caso de uso vuelve al paso 4 |
| **Excepciones** | 5.B Si el usuario no rellena todos los campos necesarios se le informará<br>5.C Si el usuario selecciona una fecha anterior a la actual se le informará<br>5.D Si el usuario introduce caracteres inválidos en el precio se le informará |

*Tabla 24 CU6-Modificar contrato*

| CU07 | Contratos firmados |
|---|---|
| **Descripción** | El usuario consulta la lista de todos los contratos que ha firmado |
| **Actor** | Usuario |
| **Precondiciones** | El usuario debe estar conectado a la red blockchain y se encuentra en la pantalla principal |
| **Postcondiciones** | Los contratos firmados no han sido creados por el usuario que está haciendo la consulta |
| **Flujo normal** | 3. El usuario pulsa el botón de "Firmados"<br>4. El sistema muestra un listado de contratos firmados por el usuario |
| **Flujo Alternativo** | 2.A El usuario pulsa el botón de volver y se ejecuta el CU6-Ver información de la aplicación<br>2.B Aun no existen contratos firmados y el sistema muestra un mensaje indicándolo |

*Tabla 25 CU7-Contratos firmados*

| CU08 | Solicitar firma |
|---|---|
| **Descripción** | El usuario solicita la firma de un contrato |
| **Actor** | Usuario |
| **Precondiciones** | El usuario debe estar conectado a la red blockchain y se encuentra en la pantalla principal |
| **Postcondiciones** | El contrato no ha sido creado por el solicitante |
| **Flujo normal** | 1. El usuario pulsa el botón de "Disponibles"<br>2. Se ejecuta el CU4-Contratos disponibles<br>3. El usuario pulsa sobre uno de los contratos de la lista<br>4. El sistema redirecciona al usuario a la pantalla "Firmar"<br>5. El usuario pulsa el botón "Solicitar firma"<br>6. El sistema pide confirmación al usuario<br>7. El usuario confirma la acción<br>8. El sistema informa al usuario de que ha solicitado correctamente la firma del contrato |
| **Flujo Alternativo** | 3.A El usuario pulsa el botón de volver y se ejecuta el CU6-Ver información de la aplicación<br>5.A El usuario pulsa el botón de volver y se ejecuta el CU4- Contratos disponibles<br>6.A El usuario cancela la acción y el caso de uso vuelve al paso 4 |
| **Excepciones** | 6.B El estado del contrato es distinto a "Activo", entonces el sistema informa al usuario de que no es posible solicitar la firma |

*Tabla 26 CU8-Solicitar firma*

| CU09 | Aceptar firma |
|---|---|
| **Descripción** | El usuario acepta la solicitud de una firma |
| **Actor** | Usuario |
| **Precondiciones** | El usuario debe estar conectado a la red blockchain, se encuentra en la pantalla principal y ha tenido un solicitante de firma |
| **Postcondiciones** | El contrato ha sido por el usuario que va a aceptar la firma |
| **Flujo normal** | 1. El usuario pulsa el botón de "Mis Contratos"<br>2. Se ejecuta el CU3-Contratos propios<br>3. El usuario pulsa sobre uno de los contratos de la lista<br>4. El sistema redirecciona al usuario a la pantalla "ContratoPropio"<br>5. El usuario pulsa el botón "Aceptar firma"<br>6. El sistema pide confirmación al usuario<br>7. El usuario confirma la acción<br>8. El sistema informa al usuario de que se ha aceptado la firma del contrato<br>9. Se ejecuta el CU10-Realizar Pago |
| **Flujo Alternativo** | 3.A El usuario pulsa el botón de volver y se ejecuta el CU6-Ver información de la aplicación<br>5.A El usuario pulsa el botón de volver y se ejecuta el CU4- Contratos disponibles<br>5.B El usuario pulsa el botón de "rechazar firma" y la firma se rechaza quedando el contrato libre para que otro usuario solicite su firma<br>6.A El usuario cancela la acción y el caso de uso vuelve al paso 4 |

*Tabla 27 CU9-Aceptar firma*

| CU10 | Realizar pago |
|---|---|
| **Descripción** | El usuario realiza el pago del contrato |
| **Actor** | Usuario |
| **Precondiciones** | El usuario debe estar conectado a la red blockchain y se encuentra en la pantalla "Firmar" |
| **Flujo normal** | 1. El sistema informa al usuario de que se va a realizar el pago<br>2. El usuario acepta el mensaje<br>3. El sistema realiza la transferencia |
| **Excepciones** | 3.A Si no se dispone de suficientes fondos para realizar la transacción el sistema informará de ello y el caso de uso queda sin efecto |

*Tabla 28 CU10-Realizar pago*

| CU11 | Detalles contrato |
|---|---|
| **Descripción** | El usuario consulta todos los detalles de un contrato |
| **Actor** | Usuario |
| **Precondiciones** | El usuario debe estar conectado a la red blockchain y se encuentra en la pantalla principal en el apartado “Disponibles” o “Firmados” |
| **Postcondiciones** | El contrato consultado no es propiedad del usuario que está consultando |
| **Flujo normal** | 1. El sistema muestra una lista con los contratos que se pueden consultar<br>2. El usuario selecciona uno de los contratos<br>3. El sistema muestra los detalles del contrato |
| **Flujo Alternativo** | 2.A El usuario pulsa el botón de volver y se ejecuta el CU6-Ver información de la aplicación<br>2.B Aun no existen contratos disponibles para consultar y el sistema muestra un mensaje indicándolo |

*Tabla 29 CU11-Detalles contrato*

| CU12 | Ver información de la aplicación |
|---|---|
| **Descripción** | El usuario consulta la información de la aplicación |
| **Actor** | Usuario |
| **Precondiciones** | El usuario se encuentra en cualquier pantalla que tenga el botón “Volver” |
| **Flujo normal** | 1. El usuario pulsa el botón de “Volver”<br>2. El sistema redirige al usuario a la pantalla inicial<br>3. El sistema muestra un texto con información de la aplicación |

*Tabla 30 CU12-Ver información de la aplicación*

## 4.4 MODELO DE DOMINIO

En el modelo de dominio tenemos 2 clases principales que son “Contrato” y “Usuario”, luego por otro lado tenemos una clase asociativa de “Evento” que relaciona a estas 2 entidades. De esta manera podemos representar como interactúan los usuarios con los contratos dentro de la aplicación.

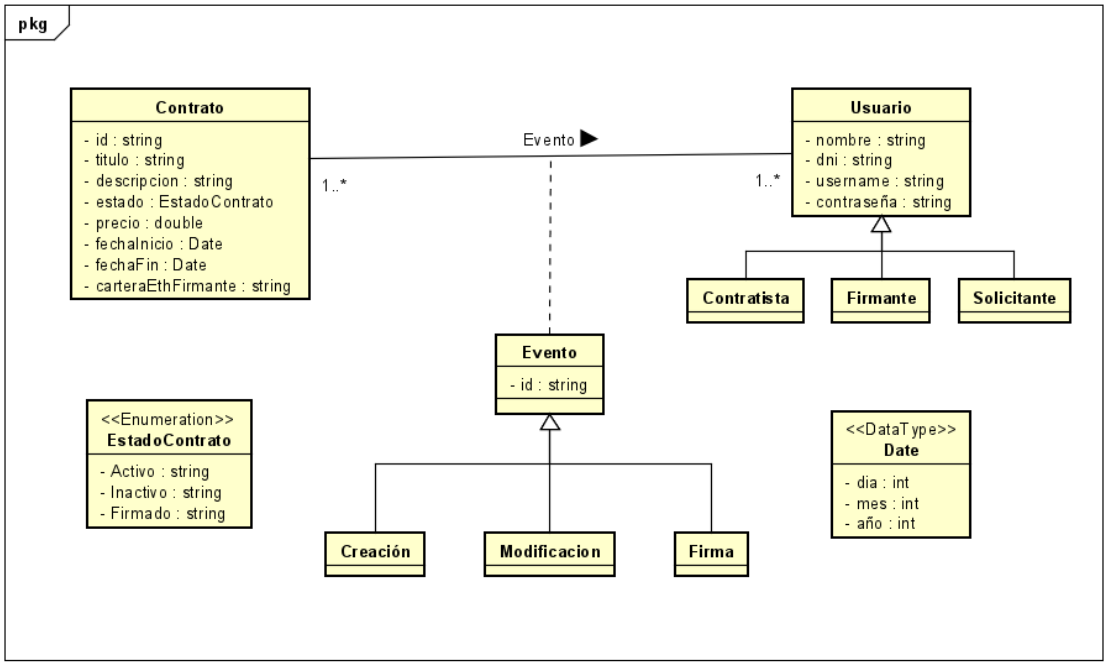


*Ilustración 8 Modelo de dominio*

A continuación, se entrará más en detalle acerca de las clases del dominio, además de sus atributos.

### 4.4.1 Contrato

Esta clase modela el contrato en sí que se va a estar manejando a lo largo de la aplicación, de modo que tiene los siguientes atributos:

- **Id**: Se trata del identificador único del contrato .
- **Título**: Este es el título que resume el objetivo del contrato.
- **Descripción**: Hace referencia a la descripción completa y detallada del contrato.
- **Estado**: Se trata del estado que puede tener el contrato, este puede ser activo, firmado o inactivo. Se considera inactivo si el contrato ha expirado y no ha sido firmado.
- **Precio**: Se trata del precio del contrato, el cual se puede medir en ETH o en euros.
- **Fecha Inicio**: Es la fecha de inicio del contrato.
- **Fecha Fin**: Es la fecha de finalización del contrato.

- **CarteraEthFirmante**: Se trata de la dirección de cartera del firmante del contrato. Las direcciones de los contratos se escriben comenzando con “0x” y seguido de una serie de caracteres alfanuméricos.

### 4.4.2 Usuario

Esta clase modela a los usuarios de la aplicación, que pueden ser contratistas o firmantes. En cuanto a los atributos nos encontramos con los siguientes:

- **Nombre:** Se trata del nombre completo del usuario.
- **DNI:** Se trata del DNI del usuario. Este debe ser único por cada usuario.
- **Username:** Se trata del nombre de usuario que será utilizado para iniciar sesión. Este nombre debe de ser único por cada usuario.
- **Contraseña:** Se trata de la contraseña que el usuario utilizará para iniciar sesión.

### 4.4.3 Contratista

Clase que modela a los usuarios que crean los contratos

### 4.4.4 Solicitante

Clase que modela a los usuarios que solicitan la firma del contrato. Este tipo de usuario es el previo a convertirse en firmantes, que es cuando la firma del contrato se hace oficial.

### 4.4.5 Firmante

Clase que modela a los usuarios que firman los contratos. Para que un usuario se considere firmante tiene que haber sido previamente un solicitante del contrato y que el contratista haya aceptado la firma del contrato.

### 4.4.6 Evento

Clase para modelar los eventos dentro del Smart Contract que tiene como atributo un **Id** único para poder identificar a cada evento.

### 4.4.7 Creación

Clase para modelar la creación de contratos.

### 4.4.8 Modificación

Clase para modelar la modificación de contratos.

### 4.4.9 Firma

Clase para modelar la firma de contratos.

# 5 DISEÑO

Una vez presentado el análisis de la aplicación el siguiente punto para tener en cuenta es el diseño de esta. En este capítulo se desglosarán las decisiones de diseño a nivel de arquitectura, además del diseño de las interfaces y la estructura interna de la misma.

## 5.1 PATRÓN MVVM

### 5.1.1 Fundamento teórico

El patrón MVVM (Model-View-ViewModel) se trata de una arquitectura de software que se utiliza para separar el desarrollo de la interfaz de usuario (UI) de la lógica de negocio, de manera que se facilita la creación de aplicaciones más mantenibles y reutilizables [14]. De acuerdo con este patrón nos encontramos ante 3 capas principales, que son las siguiente:

- **Modelo (Model):** En esta capa se presentan los datos y la lógica de negocio de la aplicación, por lo que maneja operaciones de red o bases de datos encapsulando todo que la aplicación necesita.

- **Vista (View):** Esta capa es la responsable de la presentación de la interfaz de usuario (UI) y de la interacción con el mismo. Aquí se actualizan los datos expuestos por el ViewModel y se actualiza la interfaz según sea necesario.

- **Modelo de Vista (View-Model):** Esta capa actúa como intermediario entre la vista y el modelo gestionando la lógica de presentación y transformación de los datos, de manera que estos lleguen de forma adecuada a la vista.

A continuación, se muestra una imagen que representa la interacción entre las diferentes capas de manera gráfica

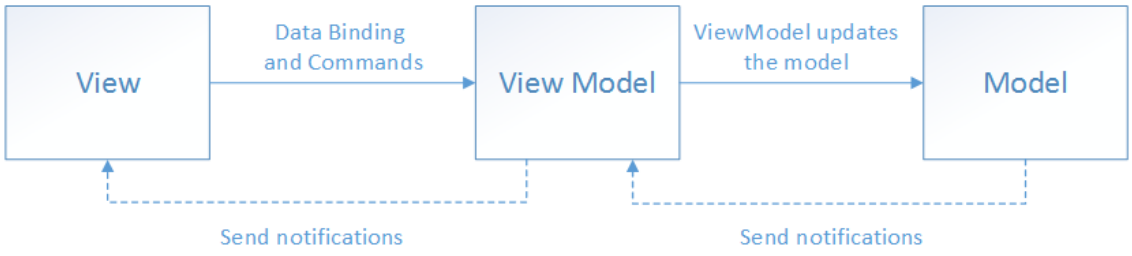


*Ilustración 9 Esquema del patrón MVVM [15]*

### 5.1.2 Adaptación del patrón MVVM

Una vez explicado en que consiste el patrón MVVM, vamos a pasar a explicar cómo se acopla este patrón al proyecto que se ha desarrollado. En este caso lo que se ha logrado es separar de manera clara la lógica relacionada con la implementación del Smart Contract de la interfaz del usuario (UI), de manera que se facilita la integración y comunicación entre ambas partes.

Por un lado, la **vista (View)** hace referencia a la biblioteca de React Native (Ver 6.2.1), donde se presentan todos los componentes visuales que están involucrados en la interfaz de usuario, como pueden ser botones, formularios, imágenes o textos.

Por otro lado, el **modelo (Model)** hace referencia al almacenamiento de los datos en la red blockchain a través del Smart Contract, que es donde se implementa toda la lógica de negocio.

Finalmente, la **vista del modelo (View-Model)** involucra la interacción entre el Smart Contract y la vista. Concretamente para este propósito se ha utilizado la tecnología Web3, necesaria para el acceso y la manipulación de los datos.

### 5.1.3 Ventajas de usar MVVM

El patrón MVVM (Model-View-ViewModel) ofrece varias ventajas significativas a la hora de desarrollar este proyecto. Entre las ventajas principales se encuentran las siguientes:

- **Separación de responsabilidades:** Esto facilita el mantenimiento y reduce la complejidad entre la lógica de negocio, la lógica de presentación y la interfaz de usuario. Algo que es crucial en una aplicación que combina la tecnología blockchain con tecnologías de desarrollo móvil.

- **Reutilización de código:** Esto es importante debido a que la lógica de negocio y la lógica de presentación no están acopladas a la vista, por lo que la posibilidad de reutilizar diferentes módulos y componentes permite compartir la misma lógica de negocio en toda la aplicación sin duplicar el código.

- **Escalabilidad y mantenibilidad:** Debido a que las aplicaciones móviles pueden crecer en complejidad, con una estructura MVVM cabe la posibilidad de ir agregando nuevas funcionalidades sin afectar significativamente a las partes ya existentes, algo que es clave para el desarrollo de una aplicación como la que se presenta.

- **Integración con Blockchain:** En este caso es de vital importancia la posibilidad que ofrece el patrón MVVM a la hora de integrarse con Blockchain debido a que facilita la gestión de las transacciones y eventos que son la base para el funcionamiento correcto de la aplicación.

## 5.2 DISEÑO MÓVIL

### 5.2.1 Diseño basado en componentes

Este proyecto consta de una estructura basada en componentes, de modo que cada componente corresponde con una pantalla dentro de la aplicación. De esta manera cada pantalla contiene todos los elementos necesarios para su funcionamiento, favoreciendo a los principios de simplicidad, separación, eficiencia y reutilización de código.

### 5.2.2 Componentes del proyecto

La aplicación costa de los siguientes componentes:

- **App:** Se trata del componente padre que alberga al resto de componentes. Este es el punto de entrada de la aplicación y maneja toda la navegación entre las diferentes pantallas.
- **Pantalla inicial:** Esta es la pantalla de bienvenida que muestra una breve descripción de la aplicación, además del logo de esta.
- **Crear Cuenta:** Esta pantalla consta de un formulario para que los usuarios puedan registrarse en la aplicación.
- **SignIn:** Esta pantalla es donde los usuarios introducen su nombre de usuario y contraseña para iniciar sesión en la aplicación.
- **Main:** Esta es la pantalla principal que muestra todas las listas de contratos. Incluyendo contratos propios, contratos disponibles y contratos firmados.
- **Nuevo Contrato:** Se trata de la pantalla que incluye el formulario para crear un nuevo contrato.
- **Contrato Propio:** Se trata de la pantalla que permite modificar un contrato o aceptar la firma de un contrato según corresponda.
- **Firmar:** En esta pantalla se ofrece la posibilidad de solicitar la firma de un contrato.
- **Firmado:** En esta pantalla se pueden ver los detalles de un contrato que ya ha sido firmado.

A continuación, se muestra el diagrama de componentes correspondiente.

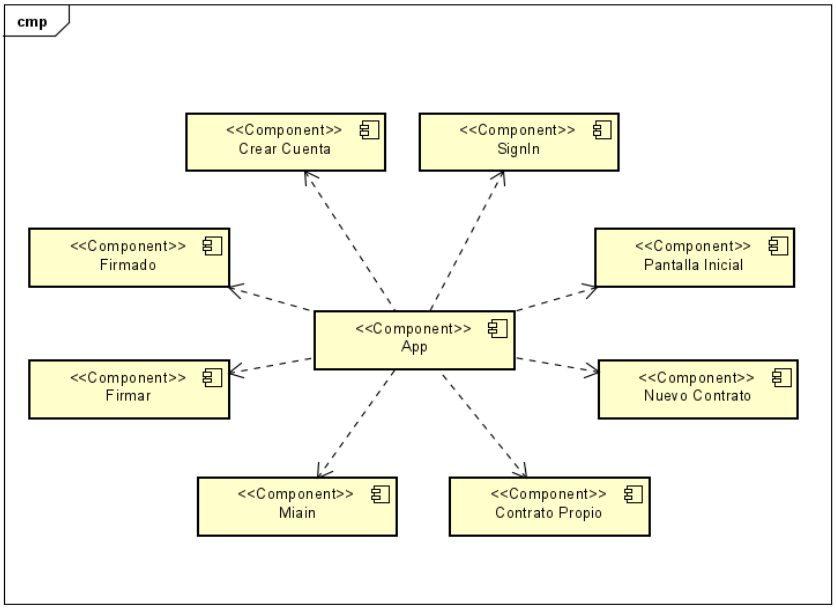


*Ilustración 10 Diagrama de componentes*

## 5.3 INTERFAZ DE USUARIO

Para el diseño de la interfaz de usuario se ha utilizado Figma, que se trata de una herramienta para el diseño de interfaces y prototipado desde la web, donde se pueden realizar, entre otras cosas diseños para las interfaces de una aplicación móvil.

A continuación, se muestra el diseño detallado de la pantalla inicial donde podemos ver el logo de la aplicación y un texto explicativo para que el usuario entienda a grandes rasgos en que consiste la aplicación. Por otro lado, tenemos un botón que nos lleva a la pantalla de registro y otro que nos lleva a la pantalla de inicio de sesión. En cuanto a estas pantallas podemos observar que para el registro es necesario indicar nombre de usuario, DNI y contraseña, mientras que para el inicio de sesión es necesario indicar el nombre de usuario y la contraseña.

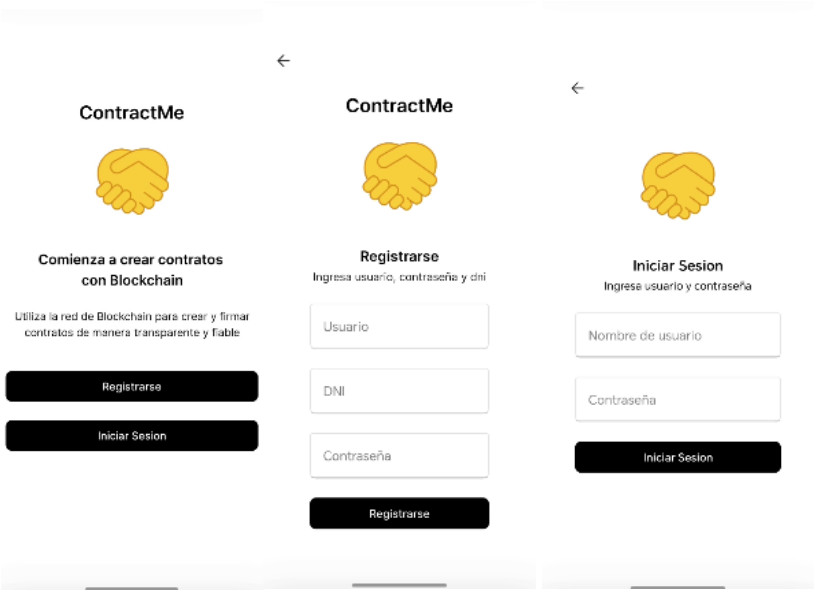


*Ilustración 11 Interfaces de pantalla inicial, de registro y de inicio de sesión*

Si ahora nos movemos a la pantalla principal, en la siguiente imagen podemos observar que disponemos de 3 pestañas.


*Ilustración 12 Interfaz de pantalla principal*

Una pestaña llamada “Mis contratos” donde aparecerán los contratos que ha creado un usuario, otra pestaña llamada “Disponibles”, donde aparecerán aquellos contratos creados por otros usuarios que están disponibles para firmar y finalmente una pestaña llamada “Mis firmados” donde aparecerán los contratos que han sido firmados por un usuario. En todas las pestañas el formato para mostrar los contratos es el mismo, de manera que se pueda observar el título del contrato, sus fechas de inicio y de fin y su estado.

Además, en la parte superior tenemos 3 botones. El botón situado más a la izquierda se utiliza para volver a la pantalla inicial mencionada anteriormente, mientras que de los otro 2 botones situados más a la derecha, el primero de ellos se utiliza para actualizar la lista de contratos cada vez que se ha producido un cambio y el segundo se utiliza para abrir una ventana de creación de un nuevo contrato.

Por otro lado, si pinchamos en cualquiera de los contratos nos llevará a una pantalla donde podremos modificar campos del contrato, firmar el contrato o ver detalles del contrato del modo que se ha explicado anteriormente (Ver 4.3).

Para el caso de la creación del contrato nos encontraremos con una pantalla como la siguiente

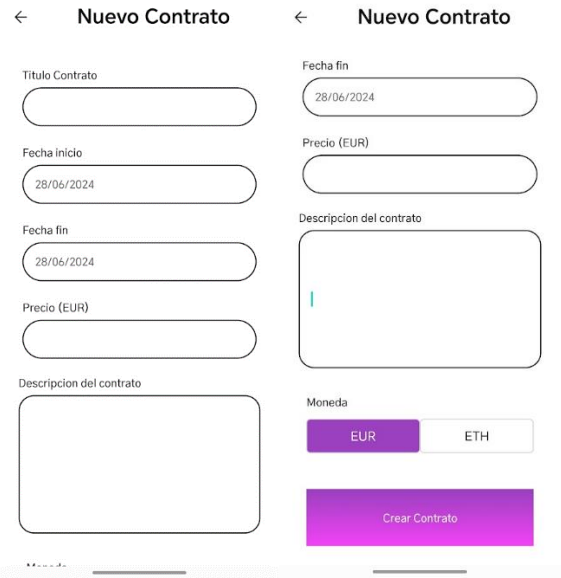


*Ilustración 13 Interfaz de creación de un contrato*

Se puede observar cómo tenemos todos los campos disponibles para la creación de un contrato a modo de formulario y además cabe destacar la posibilidad de indicar el precio en euros o en ETH pulsando el botón de moneda según se desee.

En cuanto a la modificación o firma de contrato nos encontramos con la siguiente interfaz con datos de ejemplo.


*Ilustración 14 Interfaz de modificación de un contrato*

Como vemos, en esta interfaz se mostrarán los datos del contrato y se podrán realizar las acciones mencionadas anteriormente de modificación y firma de contratos (Ver 4.3). La única diferencia entre las distintas interfaces será en botón de abajo (en este caso "Guardar cambios") y el título del texto de arriba (en este caso "Contrato Propio"), de manera que se adapte a cada situación, por lo que solo con mostrar un ejemplo es suficiente para hacerse a la idea del diseño de estas pantallas, ya que el resto tendrán un diseño análogo como se ha mencionado.

Cabe destacar que las 2 interfaces anteriores están diseñada pensado en una vista deslizante (ScrollView), de modo que para ver toda la información el usuario deberá deslizar hacia abajo o hacia arriba según sea necesario.

# 6 IMPLEMENTACIÓN

En este capítulo se va a profundizar en la implementación de la aplicación, incluyendo cuales han sido las tecnologías y herramientas de desarrollo utilizadas, además de una explicación detallada de la estructura del código.

## 6.1 HERRAMIENTAS DE DESARROLLO

Para el desarrollo de este proyecto se ha utilizado principalmente en editor de código Visual Studio Code, ya que es ampliamente utilizado y ofrece un gran set de herramientas que cubren las necesidades de desarrollo de este proyecto. Por otro lado, el equipo donde se ha desarrollado la aplicación tiene un sistema operativo Windows 11.

## 6.2 TECNOLOGÍAS UTILIZADAS

Una vez mencionadas las herramientas de desarrollo utilizadas, lo siguiente a tener en cuenta son las tecnologías en las que se basa la aplicación. Por un lado, tenemos todas aquellas relacionadas con el desarrollo blockchain ya mencionadas anteriormente (Ver 2.5) y por otro lado en este capítulo se abordarán el resto de las tecnologías.

### 6.2.1 React Native

React Native es un framework JavaScript para crear aplicaciones reales nativas en iOS y Android, basado en la librearía de React para la creación de componentes visuales, cambiando el propósito de estos para, en lugar de ser ejecutados en navegador, correr directamente sobre las plataformas móviles nativas [16]. Esto es un aspecto clave a tener en cuenta para el desarrollo de nuestra aplicación, ya que tenemos como requisito que se trate de una aplicación Android (Ver 4.2).

Al igual que React, se utiliza el mismo paradigma de construcción de bloques de UI. De este modo las aplicaciones hechas con React Native usan una mezcla de JavaScript y un lenguaje de marcado conocido como JSX [16].

En el caso concreto de esta aplicación toman importancia los Hooks, que son funciones especiales que permiten a los desarrolladores usar características de React Native como el estado y el ciclo de vida de los componentes funcionales [17].

Uno de los Hooks más utilizados en el desarrollo de esta aplicación es useState, que permite añadir estado a los componentes funcionales de manera que estos pueden mantener y modificar su estado interno. Otro Hook fundamental es useEffect, el cual permite ejecutar efectos secundarios en componentes funcionales, como por ejemplo llamadas a API o actualizaciones del DOM. Esta función se ejecuta tras cada renderizado y permite, entre otras cosas realizar tareas de limpieza para evitar fugas de memoria. El uso de ambos hooks es esencial para poder desarrollar una interfaz interactiva y fácilmente usable para el usuario.

### 6.2.2 Node

Node.js es un entorno en tiempo de ejecución multiplataforma, de código abierto, para la capa del servidor (pero no limitándose a ello) basado en el lenguaje de programación JavaScript, asíncrono, con E/S de datos en una arquitectura orientada a eventos y basado en el motor V8 de Google. Fue creado con el enfoque de ser útil en la creación de programas de red altamente escalables, como, por ejemplo, servidores web [16].

Entre las ventajas principales de Node.js destacan su asincronía y modelo no bloqueante, permitiendo un muy buen rendimiento en aplicaciones de alta concurrencia. Además, ofrece la posibilidad de desarrollar en JavaScript tanto en el cliente como en el servidor, lo cual simplifica el desarrollo de manera considerable. Todo esto hace que se adecúe correctamente para el desarrollo de una aplicación móvil como la que se desarrolla en este proyecto.

### 6.2.3 Npm

NPM es una herramienta fundamental en el desarrollo con JavaScript. Corresponde con las siglas de Node Package Manager y consiste en un administrador de paquetes que permite a los desarrolladores de JavaScript trabajar con lo que llamamos dependencias. Las dependencias son todas aquellas librerías que usamos al desarrollar un programa y que nos permiten evitar hacer todo el trabajo desde cero [18].

Para el desarrollo de esta aplicación NPM ha sido utilizado principalmente en la instalación de paquetes, además de la gestión de dependencias. Algunos módulos que se han instalado a través de NPM han sido web3 o truffle, que son claves para poder desarrollar contratos inteligentes.

### 6.2.4 Web3.js

Web3.js es una colección de bibliotecas que permite interactuar con una blockchain Ethereum local o remota. Esta solución ha sido desarrollada por la comunidad de Ethereum facilitando la conexión a nodos Ethereum a través de HTTP, IPC o WebSocket [19].

En esta aplicación se utiliza Web3.js para poder interactuar con el Smart Contract, de manera que desde el cliente se pueda acceder a los métodos y funciones necesarias para la correcta comunicación la red Ethereum.

Por otro lado, también se ha utilizado para la gestión de carteras y transacciones entre ellas, de manera que a la hora de firmar los contratos las transacciones entre usuarios sean válidas y transparentes.

### 6.2.5 Astah

Astah es una herramienta de modelado de software desarrollada por Change Vision, Inc, utilizada principalmente para la creación y visualización de diagramas UML (Undefined Modeling Language) y otros tipos de diagramas de diseño de sistemas. Esta herramienta proporciona una variedad de funciones y diagramas que incluyen [20].

En este proyecto, Astah se ha utilizado para realizar los diagramas que se muestran en el análisis y el diseño (Ver capítulos 4 y 5) de modo que se facilite la estructuración y planificación.

### 6.2.6 GitLab

Durante todo el desarrollo del proyecto se ha utilizado la plataforma Gitlab en un repositorio proporcionado por la empresa HP. Esta plataforma ofrece herramientas para todas las etapas del ciclo de vida del proyecto, incluyendo la planificación, el control de versiones y la monitorización [21].

Concretamente se ha hecho uso de la plataforma para todo aquello relacionado con la planificación de las tareas y sprints, además de la creación de ramas de desarrollo donde poder desarrollar las diferentes funcionalidades del proyecto. Todo esto ha hecho posible que la comunicación remota con los tutores de la empresa HP haya sido fluida y efectiva.

### 6.2.7 Expo y Expo Go

Expo es una plataforma y conjunto de herramientas que facilita el desarrollo de aplicaciones móviles usando React Native. Con esta herramienta es posible construir aplicaciones móviles nativas de manera simple y ágil gracias a que incluye un SDK con APIs y componentes nativos previamente construidos, una CLI para la gestión y creación de proyectos y una aplicación móvil llamada Expo Go, que ofrece la posibilidad de probar la aplicación que se está desarrollando en tiempo real sin necesidad de compilación previa [22].

Entre las ventajas que ofrece expo nos encontramos con el manejo automático de dependencias, simplificando la configuración y evitando problemas de compatibilidad, sobre todo relacionados con el módulo Web3.js mencionado anteriormente (Ver 6.2.4). Además, ofrece un servicio de construcción y publicación de aplicaciones generando archivos .apk en el caso de Android y .ipa en el caso de IOS.

## 6.3 IMPLEMENTACION EN LA RED BLOCKCHAIN

En este apartado se va a presentar todo aquello relacionado con la implementación realizada en la red Blockchain, incluyendo la implementación del Smart Contract y los componentes que este incluye.

### 6.3.1 Implementación del Smart Contract

Para poder implementar y desplegar en el Smart Contract se ha utilizado el ecosistema de Truffle presentado anteriormente (Ver 2.5.3) mientras que la red de Blockchain está albergada por Ganache, que como vimos anteriormente nos ofrece la posibilidad de generar una red de Ethereum privada con un amplio abanico de configuración y monitorización (Ver 2.5. 4). También hay que recordar que la implementación del Smart Contract está realizada utilizando el lenguaje Solidity (Ver 2.5.2).

#### *6.3.1.1 Estructuras y mapeos*

Si ahora entramos más en detalle acerca de la implementación del Smart Contract nos encontramos con las siguientes estructuras y mapeos:

```
contract MyContract {

    struct Contrato {
        uint id;
        string titulo;
        string descripcion;
        EstadoContrato estado;
        uint256 precio;
        string fechaInicio;
        string fechaFin;
        address solicitante;
        address firmante;
    }

    enum EstadoContrato {  Activo, Solicitado, Firmado, Expirado }
    Contrato[] public contratos;

    mapping (uint => address) public contractOwner;
    mapping (address => uint) private ownerCount;
    mapping(address => uint256[]) public contractsSignedByAccount;
```

*Ilustración 15 Estructuras y mapeos del Smart Contract*

Por un lado, tenemos la estructura principal que representa las características principales del contrato con el nombre de "Contrato", donde vemos los siguientes atributos:

- **Id**: Identificador de cada contrato.
- **Título**: Título correspondiente a cada contrato que resume el objetivo principal del mismo.
- **Descripción**: Utilizado para describir en detalle el contrato que se ha creado, incluyendo los objetivos a cubrir, los aspectos legales o ciertos matices a tener en cuenta a la hora de realizar la firma del contrato.
- **EstadoContrato:** Se utiliza para guardar el estado del contrato que puede ser "Activo", "Solicitado", "Firmado" o "Expirado.
- **Precio**: Cantidad a pagar por el contratista a la persona que firma el contrato.
- **FechaInicio**: Fecha en la cual se inicia el periodo de validez del contrato.
- **FechaFin**: Fecha en la cual finaliza el periodo de validez del contrato.
- **Solicitante**: Dirección de la cartera de usuario que solicita la firma del contrato.
- **Firmante**: Dirección de la cartera del usuario que finalmente realiza la firma del contrato.

Por otro lado, nos encontramos con los mapeos que nos permiten relacionar elementos entre sí. En concreto contamos con los siguientes:

- **Contrato [] public contratos:** Se trata de la estructura de memoria donde se van a almacenar todos los contratos de la aplicación.

- **mapping (uint => address) public ContractOwner :** Establece una relación entre la dirección de cartera del dueño del contrato y el contrato creado. De esta manera es posible identificar cada contrato sabiendo su creador.

- **mapping (address => uint) pivate ownerCount:** Establece una relación entre la dirección de cartera de un usuario y el número de contratos que ha creado.

- **mapping (address => uint256[]) pivate contractsSignedByAccount:** Se trata de un mapeo que relaciona la dirección de una cartera con la lista de contratos que han sido firmados por una esa misma cartera.

#### *6.3.1.2 Eventos*

Una vez se han presentado todas las estructuras y mapeos del Smart Contract, lo siguiente a tener en cuenta son los eventos, que son mecanismos que permiten almacenar y extraer la información de la cadena de bloques. Para este contrato el uso que tienen los eventos es el de transmitir, recibir y procesar toda la información enviada por los usuarios que crean, modifican y firman los contratos.

Concretamente en esta aplicación contamos con los siguientes eventos:

```
event ContratoCreado(uint indexed id, string titulo, string descripcion, address ownerAddress, uint256 precio, string fechaInicio, string fechaFin);
event ContratoModificado(uint indexed id, string nuevoTitulo, string nuevaDescripcion, uint256 nuevoPrecio, string nuevaFechaInicio, string nuevaFechaFin);
event ContratoFirmado(uint indexed id, address firmante);
event FirmaSolicitada(uint indexed id, address firmante);
event FirmaRechazada(uint indexed id);
```

*Ilustración 16 Eventos del Smart Contract*

Como se puede observar contamos con los eventos de "ContratoCreado", "ContratoModificado", "ContratoFirmado", "FirmaSolicitada" y "FirmaRechazada". Los 3 primeros hacen referencia a la creación, modificación y firma, mientras que los 2 últimos hacen referencia a la solicitud de las firmas cuyo funcionamiento explicamos anteriormente (Ver 4.3.1).

#### *6.3.1.3 Funciones y métodos*

Finalmente es necesario indagar en las funciones y métodos implementados dentro del Smart Contract, de modo que son estos los que invocan a los eventos y procesan toda la información. En concreto nos encontramos con los siguientes:

- **crearContrato(titulo, descripcion, precio, fechaInicio, fechaFin):** Este método se utiliza para crear un nuevo contrato y es el que invoca al evento "ContratoCreado". Para que el contrato pueda ser creado se comprueba que el precio de este sea superior a 0.

- **modificarContrato(id, nuevoTitulo, nuevaDescripcion, nuevoPrecio, nuevaFechaInicio, nuevaFechaFin):** Con este método es posible modificar un contrato que ya se había creado con anterioridad y que no haya sido firmado aún. Se puede modificar el precio, el título, la descripción y las fechas de inicio y fin según se desee. Además, este método invoca al evento "ContratoModificado".

- **firmarContrato(id, address firmante):** Este método es el encargado de "sellar" la firma de un contrato, asignando al firmante correspondiente y transfiriéndole al mismo la cantidad de ETH indicada en el contrato. Dentro del método se comprueba que el contratista dispone de la cantidad suficiente como para realizar la transacción y si es así se invoca al evento "ContratoFirmado".

- **solicitarFirma(id):** A partir del identificador del contrato, este método es el encargado de asignar temporalmente a un solicitante hasta que este se convierta temporalmente en firmante. Para ello se invoca al evento "FirmaSolicitada".

- **rechazarFirma(id):** A partir del identificador del contrato, este método es utilizado para eliminar al solicitante del contrato. Para ello se invoca al método "FirmaRechazada".

## 6.4 IMPLEMENTACIÓN DEL CLIENTE

En este apartado se va a desglosar la estructura general del cliente, el cual se ha desarrollado utilizando la biblioteca React Native (Ver 6.2.1). Se destacarán las partes más relevantes y se hará hincapié en la funcionalidad principal que ha sido desarrollada.

### 6.4.1 Estructura general

Para poder entender la implementación por parte del cliente primero es importante presentar la estructura general de carpetas que este contiene, de modo que lo vamos a dividir de la siguiente manera:

- **/App.js:** Este archivo se encuentra en el directorio raíz del cliente y es el principal punto de entrada a la aplicación, ya que desde aquí se gestionan todas las pantallas que conforman la aplicación.

- **/Screens:** En esta carpeta nos encontramos con todas las pantallas que contiene la aplicación, incluyendo aquí toda la lógica de componentes mencionada anteriormente (Ver 5.2.1).

- **/rowComponents:** Esta carpeta actúa como auxiliar para proporcionar las filas con la información de los contratos como componentes según sea solicitado por cada pantalla en cada momento.

- **/assets:** En esta carpera nos encontramos con los elementos gráficos, como imágenes o logos, además de los formatos de las fuentes utilizadas para los textos de la aplicación.

- **/contracts:** Aquí encontramos las compilaciones del Smart Contract necesarias para que la comunicación entre el cliente y la Blockchain a través de web3.js (Ver 6.2.4) sea posible. Cabe destacar que el tipo de archivos que encontramos en esta carpeta es JSON.

- **/global.js:** En este archivo se incluyen elementos globales en toda la aplicación como la dirección http del proveedor de la red Blockchain, en este caso Ganache (Ver 2.5.4).

Mas allá de la estructura mencionada anteriormente tenemos otros archivos en formato JSON menos relevantes como son package.json, utilizado por la aplicación para gestionar las dependencias o rn-cli.json, necesario para que Expo pueda funcionar correctamente. En general estos archivos son generados de manera automática a la hora de crear el proyecto y algunos de ellos se van modificando de manera automática a medida que el programador va añadiendo dependencias al proyecto.

### 6.4.2 Navegabilidad de la aplicación

```
<Stack.Navigator screenOptions={{ headerShown: false }}>
  <Stack.Screen
    name="PantallaInicial"
    component={PantallaInicial}
    options={{ headerShown: false }}
  />

  <Stack.Screen
    name="SignIn"
    component={SignIn}
    options={{ headerShown: false }}
  />

  <Stack.Screen
    name="CrearCuenta"
    component={CrearCuenta}
    options={{ headerShown: false }}
  />

  <Stack.Screen
    name="NuevoContrato"
    component={NuevoContrato}
    options={{ headerShown: false }}
  />
  <Stack.Screen
    name="Lista"
    key={"Lista_" + listaKey}
    component={Lista}
    options={{ headerShown: false }}
    listeners={{ focus: () => refreshLista() }}
  />
  <Stack.Screen
    name="Firmar"
    component={Firmar}
    options={{ headerShown: false }}
  />
  <Stack.Screen
    name="ContratoPropio"
    component={ContratoPropio}
    options={{ headerShown: false }}
  />
  <Stack.Screen
    name="Firmado"
    component={Firmado}
    options={{ headerShown: false }}
  />
</Stack.Navigator>
```

*Ilustración 17 Implementación de Stack.Nagivator y Stack.Screen*

Para gestionar la navegabilidad de toda la aplicación se ha hecho uso de Stack.Navigator, que gestiona una pila de pantallas (Stack.Screen) donde los usuarios pueden navegar hacia adelante y hacia atrás de manera que cada una de las pantallas (Screen) es un componente de React Native que define una vista o una página completa de la aplicación.

Para configurar Stack.Navigator, se definen todas las pantallas de la aplicación junto con sus configuraciones específicas, incluyendo entre otras cosas opciones de navegación o el nombre de cada componente. Además, a la hora de declarar las pantallas, se sigue una estructura jerárquica donde cada pantalla es navegable desde y hacia otras pantallas es función de esta jerarquía.

### 6.4.3 Conexión con Blockchain

```
useEffect(() => {
  const connectToBlockchain = async () => {
    try {
      const web3 = new Web3(
        new Web3.providers.HttpProvider(WEB3_PROVIDER_URL)
      );
      const contract = await getContract(web3);
      setMyContract(contract);
    } catch (error) {
      alert("No se ha podido conectar a la blockchain");
    }
  };

  connectToBlockchain();
}, []);
```

*Ilustración 18 Conexión con Blockchain*

Para realizar la conexión con blockchain se hace uso del hook de React Native mencionado anteriormente "useEffect" (Ver 6.2.1), dentro del cual se ejecuta una función llamada "connectToBlockchain", donde podemos observar que primeramente se instancia un objeto Web3 tomando como argumento un nuevo proveedor HTTP con la dirección definida en la variable global WEB3_PROVIDER_URL, la cual se encuentra en el fichero global.js (Ver 6.4.1).

Una vez se ha realizado la instancia del objeto Web3, lo siguiente que se hace es llamar al método getContract, donde se obtiene el contrato que se ha implementado con Solidity. El siguiente extracto de código muestra cómo se obtiene el contrato.

```
const getContract = async (web3) => {
  const networkID = await web3.eth.net.getId();
  const network = MyContract.networks[networkID];
  return new web3.eth.Contract(MyContract.abi, network && network.address);
};
```

*Ilustración 19 Obtención del Smart Contract*

Como se puede observar primero se obtiene el identificador de red y la configuración de la misma y posteriormente se devuelve la instancia del contrato en base a la interfaz de este en formato ABI (Aplication Binary Interface) y los atributos de red ya mencionados.

Gracias a esta instancia es posible realizar las operaciones implementadas en el a través del cliente en React Native y también se pueden gestionar las carteras que ofrece Ganache para poder realizar las transacciones (Ver 2.5.4).

### 6.4.4 Registro e inicio de sesión

Para realizar el registro de usuarios se hace uso del método “handleRegister” en la pantalla “CrearCuenta”. Dentro de este método primero se validan los campos de usuario, contraseña y DNI, de manera que el usuario no debe coincidir con ningún otro usuario registrado previamente, la contraseña debe de tener un mínimo de 4 caracteres y el DNI debe constar de 8 números y una letra al final.

Una vez validados todos estos campos lo que se hace es asignar la primera dirección de cartera disponible proporcionada por Ganache al usuario que se está registrando y se guarda esta información en el dispositivo a modo de “importación de cartera”. A continuación, se muestra el fragmento de código encargado de esto.

```
const usedAccounts = users.map((user) => user.account);
const availableAccount = accounts.find(
  (acc) => !usedAccounts.includes(acc)
);

if (!availableAccount) {
  window.alert("No hay cuentas disponibles");
  return;
}

users.push({ username, password, dni, account: availableAccount });
await AsyncStorage.setItem("users", JSON.stringify(users));

window.alert(
  "Usuario registrado con dirección de cartera:" + availableAccount
);
```

*Ilustración 20 Método "handleRegister"*

Como se puede observar en caso de que no queden cuentas disponibles se le notifica al usuario y en el caso de que se pueda asignar una dirección de cartera se le informará a cada usuario de cual es dicha dirección.

En cuanto al inicio de sesión, se hace uso del método “handleLogin” en la pantalla “SignIn”, donde se comprueba que el usuario existe y en caso de existir se comprueba que la contraseña sea la correcta. Posteriormente se navega a la pantalla principal “Main”. A continuación, se muestra el código encargado de esto.

```
const handleLogin = async () => {
  const storedUsers = await AsyncStorage.getItem("users");
  const users = storedUsers ? JSON.parse(storedUsers) : [];
  const user = users.find(
    (user) => user.username === username && user.password === password
  );

  if (!username || !password) {
    window.alert("Por favor, complete todos los campos");
    return;
  }

  if (!user) {
    window.alert("Usuario o contraseña incorrectos");
    return;
  }

  navigation.navigate("Lista", { account: user.account });
};
```

*Ilustración 21 Método "handleLogin"*

### 6.4.5 Creación y modificación de contratos

Para la creación de un contrato se hace uso del método “handleCreateContract” en la pantalla “NuevoContrato”, donde primero se comprueba que todos los campos del contrato son válidos y posteriormente se llama al método “crearContrato” implementado en el Smart Contract (Ver 6.3.1.3). La implementación para realizar la llamada a este método es la siguiente.

```
await MyContract.methods
  .crearContrato(
    formData.tituloContrato,
    formData.descripcionContrato,
    precioEnWei,
    formatDate(formData.fechaInicio),
    formatDate(formData.fechaFin)
  )
  .send({ from: account, gas: "1000000" });
```

*Ilustración 22 Llamada al método "crearContrato"*

Aquí se puede observar cómo se hace uso de la palabra clave “send”, que indica que estamos realizando una operación con coste. En este caso el coste de la operación se mide en “gas”, que es la unidad de medida para cuantificar el trabajo que tiene que hacer la red de Blockchain para realizar cada operación. Si nos fijamos, el “send” utiliza 2 argumentos, el primero de ellos hace referencia a la dirección de cartera del usuario que está realizando la creación del contrato y el segundo hace referencia al límite de gas permitido en la operación. Se ha decidido poner un número elevado de modo que no haya problema por falta de gas.

En cuanto a la modificación del contrato el procedimiento es muy similar, de modo que primero se valida que aquellos campos que se han modificado y se realiza la llamada al método del Smart Contract “modificarContrato”, situado en el “handleModificar” de la pantalla “ContratoPropio”. A continuación, se muestra el código que hace la llamada necesaria, que podemos observar que es prácticamente igual que en el caso anterior.

```
await contract.methods
  .modificarContrato(
    idContrato,
    formData.titulo,
    formData.descripcion,
    precioEnWei,
    formatDateString(formData.fechaInicio),
    formatDateString(formData.fechaFin)
  )
  .send({ from: account, gas: "1000000" });
```

*Ilustración 23 Llamada al método "modificarContrato"*

### 6.4.6 Firma de contratos

Para realizar la firma de contratos, como se ha explicado anteriormente primero es necesario que un usuario solicite la firma y posteriormente que el usuario que ha creado el contrato acepte esta solicitud (Ver 4.3). De modo que para implementar esta funcionalidad se ha hecho uso de los métodos "handleFirmar", para solicitar la firma, "handleAceptarFirma", para aceptar la firma que se ha solicitado y "handleRechazarFirma", para rechazar la firma solicitada.

Primero vamos a analizar el método "handleFirmar", situado en la pantalla "Firmar". Este método realiza la comprobación del estado y en caso de que sea un estado válido para realizar la firma se procede con ello. Como ya se explicó anteriormente el único estado válido para solicitar una firma de contrato es "Activo" (Ver 4.3). Por otro lado, dentro de este método encontramos la llamada a "solicitarContrato", que de la misma forma que en el caso de creación de contrato utiliza "gas" debido a que es una operación que tiene un coste.

```
await contract.methods.solicitarFirma(idContrato).send({
  from: userAccount,
  gas: "1000000",
});
```

*Ilustración 24 Llamada al método "SolicitarFirma"*

Posteriormente una vez solicitada la firma tenemos que movernos a la pantalla "ContratoPropio" donde nos encontramos con "handleAceptarFirma" y "handleRechazarFirma". En el caso de "handleAceptarFirma" nos encontramos con la siguiente llamada al método "firmarContrato" del Smart Contract.

```
await contract.methods.firmarContrato(idContrato, firmante).send({
  from: account,
  value: precioEnWei,
  gas: "1000000",
});
```

*Ilustración 25 Llamada al método "firmarContrato"*

En este caso nuevamente estamos ante una operación que tiene un coste en "gas", pero a la vez se trata de una transacción de una cuenta a otra, como vimos en la implementación del contrato (Ver 6.3.1.3), por lo que a mayores hay que indicar la dirección de la cuenta de origen desde la que se realiza la transacción. En este caso se indica en el atributo "from".

En el caso de que el usuario creador del contrato decida rechazar la firma, dentro de “handleRechazarFirma” tenemos la siguiente llamada al método “rechazarFirma” del Smart Contract.

```
await contract.methods.rechazarFirma(idContrato).send({
  from: account,
  gas: "1000000",
});
```

*Ilustración 26 Llamada al método "rechazarFirma"*

# 7 PRUEBAS

En este capítulo se presentan las pruebas realizadas para verificar el correcto funcionamiento de la aplicación, de modo que se compruebe el funcionamiento de todos los elementos, tanto en aquellos casos de éxito, como en aquellos casos de fallo. A continuación, se muestran las pruebas realizadas junto con sus respectivos resultados.

## 7.1 PRUEBAS SOBRE EL INICIO DE SESIÓN Y REGISTRO DE USUARIOS

| | |
|---|---|
| **PIR1** | Registro de un nuevo usuario correctamente |
| **Descripción** | El usuario se registra de manera satisfactoria introduciendo un nombre de usuario, DNI y contraseña |
| **Resultado esperado** | Se notifica al usuario de que se ha registrado correctamente y se le informa de la dirección de cartera que se le ha asignado |
| **Resultado Obtenido** | Resultado esperado |

*Tabla 31 PIR1- Registro de un nuevo usuario correctamente*

| | |
|---|---|
| **PIR2** | Inicio de sesión exitoso |
| **Descripción** | El usuario inicia sesión de forma exitosa introduciendo nombre de usuario y contraseña |
| **Resultado esperado** | Se le informa al usuario que ha iniciado sesión correctamente y se carga la página principal |
| **Resultado Obtenido** | Resultado esperado |

*Tabla 32 PIR2 - Inicio de sesión exitoso*

| | |
|---|---|
| **PIR3** | DNI Inválido |
| **Descripción** | Se introduce un DNI inválido a la hora de registrarse |
| **Resultado esperado** | Se le informa al usuario de que el DNI introducido no es válido |
| **Resultado Obtenido** | Resultado esperado |

*Tabla 33 PIR3 - DNI Inválido*

| | |
|---|---|
| **PIR4** | Usuario ya existente |
| **Descripción** | Se intenta hacer el registro de un usuario ya existente |
| **Resultado esperado** | Se informa de que el nombre de usuario ya existe |
| **Resultado Obtenido** | Resultado esperado |

*Tabla 34 PIR4 - Usuario ya existente*

| | |
|---|---|
| **PIR5** | Contraseña incorrecta |
| **Descripción** | El usuario introduce una contraseña incorrecta al iniciar sesión |
| **Resultado esperado** | Se lanza un mensaje informativo indicando que ha habido un error en el inicio de sesión |
| **Resultado Obtenido** | Resultado esperado |

*Tabla 35 PIR5 - Contraseña incorrecta*

| | |
|---|---|
| **PIR6** | Algún campo vacío |
| **Descripción** | Alguno de los campos en el registro está vacío |
| **Resultado esperado** | Se le informa al usuario de que debe de rellenar todos los campos |
| **Resultado Obtenido** | Resultado esperado |

*Tabla 36 PIR6 - Algún campo vacío*

| | |
|---|---|
| **PIR7** | Usuario inexistente |
| **Descripción** | El usuario introducido en el inicio de sesión aún no se ha registrado |
| **Resultado esperado** | Se informa al usuario de que ha habido un error en el inicio de sesión |
| **Resultado Obtenido** | Resultado esperado |

*Tabla 37 PIR7 - Usuario inexistente*

## 7.2 PRUEBAS SOBRE LA CREACIÓN Y MODIFICACIÓN DE CONTRATOS

| | |
|---|---|
| **PCM1** | Crear nuevo contrato correctamente |
| **Descripción** | El usuario crea correctamente un nuevo contrato cumpliendo con todos los requisitos |
| **Resultado esperado** | Se informa al usuario de que el contrato se ha creado correctamente y se crea el contrato en la red de Blockchain |
| **Resultado Obtenido** | Resultado esperado |

*Tabla 38 PCM1 - Crear nuevo contrato correctamente*

| | |
|---|---|
| **PCM2** | Modificar contrato correctamente |
| **Descripción** | El usuario modifica alguno de los campos del contrato de manera satisfactoria |
| **Resultado esperado** | Se informa al usuario de que el contrato se ha modificado correctamente y se crea el contrato en la red de Blockchain |
| **Resultado Obtenido** | Resultado esperado |

*Tabla 39 PCM2 - Modificar contrato correctamente*

| | |
|---|---|
| **PCM3** | Algún campo vacío en la creación de un contrato |
| **Descripción** | A la hora de crear un contrato alguno de los campos está vacío |
| **Resultado esperado** | Se lanza un mensaje informativo indicando al usuario que rellene todos los campos |
| **Resultado Obtenido** | Resultado esperado |

*Tabla 40 PCM3 - Algún campo vacío en la creación de un contrato*

| | |
|---|---|
| **PCM4** | Fecha de finalización anterior a la fecha de inicio |
| **Descripción** | A la hora de crear o modificar un contrato se introduce una fecha de finalización anterior a la fecha actual |
| **Resultado esperado** | Se lanza un mensaje informativo indicando al usuario que las fechas introducidas son incorrectas |
| **Resultado Obtenido** | Resultado esperado |

*Tabla 41 PCM4 - Fecha de finalización anterior a la fecha de inicio*

| | |
|---|---|
| **PCM5** | Error en el precio introducido |
| **Descripción** | El precio introducido contiene caracteres inválidos como símbolos o caracteres especiales |
| **Resultado esperado** | Se lanza un mensaje informativo indicando al usuario que el precio introducido contiene caracteres inválidos |
| **Resultado Obtenido** | Resultado esperado |

*Tabla 42 PCM5 - Error en el precio introducido*

| | |
|---|---|
| **PCM6** | Precio introducido menor o igual que 0 |
| **Descripción** | El precio introducido es menor o igual que 0 |
| **Resultado esperado** | Se lanza un mensaje informativo indicando al usuario que el precio no puede ser menor o igual que 0 |
| **Resultado Obtenido** | Resultado esperado |

*Tabla 43 PCM6 - Precio introducido menor o igual que 0*

| | |
|---|---|
| **PCM7** | Modificación de contrato por propietario incorrecto |
| **Descripción** | Se trata de modificar el contrato por un propietario diferente a quien ha creado el contrato |
| **Resultado esperado** | El Smart Contract bloquea esta acción y se lanza una excepción |
| **Resultado Obtenido** | Resultado esperado |

*Tabla 44 PCM7 - Modificación de contrato por propietario incorrecto*

## 7.3 PRUEBAS SOBRE LA FIRMA DE CONTRATOS

| | |
|---|---|
| **PF1** | Solicitud de firma correcta |
| **Descripción** | El usuario solicita la firma de un contrato |
| **Resultado esperado** | Se informa al usuario de que la solicitud de firma ha sido realizada correctamente |
| **Resultado Obtenido** | Resultado esperado |

*Tabla 45 PF1 - Solicitud de firma correcta*

| | |
|---|---|
| **PF2** | Solicitud inválida por estado “Expirado” |
| **Descripción** | Se realiza una petición de firma a un contrato que está expirado |
| **Resultado esperado** | Se lanza un mensaje informativo indicando al usuario que no es posible realizar la solicitud de firma |
| **Resultado Obtenido** | Resultado esperado |

*Tabla 46 PF2 - Solicitud inválida por estado “Expirado”*

| | |
|---|---|
| **PF3** | Solicitud inválida por estado “Solicitado” |
| **Descripción** | Se realiza una solicitud de firma a un contrato que ya ha sido solicitado |
| **Resultado esperado** | Se lanza un mensaje informativo indicando al usuario que no es posible realizar la solicitud de firma |
| **Resultado Obtenido** | Resultado esperado |

*Tabla 47 PF3 - Solicitud inválida por estado “Solicitado”*

| | |
|---|---|
| **PF4** | Aceptar solicitud correctamente |
| **Descripción** | Un usuario acepta la solicitud de firma recibida |
| **Resultado esperado** | Se informa al usuario de que la solicitud de firma ha sido aceptada y se realiza la transacción correspondiente |
| **Resultado Obtenido** | Resultado esperado |

*Tabla 48 PF4 - Aceptar solicitud correctamente*

| | |
|---|---|
| **PF5** | Rechazar solicitud correctamente |
| **Descripción** | Un usuario rechaza la solicitud de firma recibida |
| **Resultado esperado** | Se informa al usuario de que la solicitud de firma ha sido rechazada y el contrato queda libre para poder obtener una nueva solicitud |
| **Resultado Obtenido** | Resultado esperado |

*Tabla 49 PF5 -Rechazar solicitud correctamente*

| | |
|---|---|
| **PF6** | Fondos insuficientes |
| **Descripción** | El usuario no dispone de fondos suficientes para pagar a la persona contratada |
| **Resultado esperado** | Se lanza una excepción y se le informa al usuario de que no es posible finalizar la firma del contrato |
| **Resultado Obtenido** | Resultado esperado |

*Tabla 50 PF6 - Fondos insuficientes*

# 8 CONCLUSIONES Y TRABAJO FUTURO

## 8.1 CONCLUSIONES

En cuanto al trabajo desarrollado, se han cumplido todos los objetivos planteados en los apartados de objetivos académicos y personales (Ver 1.3 y 1.4) menos el de desplegar la aplicación en una red de pruebas de Ethereum, que será un objetivo a tener en cuenta en líneas de trabajo futuras. De este modo, se ha conseguido desarrollar una aplicación móvil en Android orientada a la creación y firma de contratos utilizando la tecnología Blockchain.

Por otro lado, durante el desarrollo de este proyecto se han llevado a cabo metodologías ágiles de manera correcta, siendo posible experimentar su utilidad en un proyecto complejo como es este caso.

Además, a la hora de realizar este proyecto se han podido poner en práctica los conocimientos adquiridos durante la carrera, especialmente aquellos relacionados con las asignaturas de Diseño de Software, Sistemas Móviles, Planificación y Gestión de Proyectos y Análisis y Diseño de Bases de Datos. En cuanto a esta última asignatura ha sido posible profundizar en el entendimiento de aquellos conceptos relacionados con la tecnología Blockchain.

Gracias a este proyecto espero que se haya podido reflejar la utilizad y beneficios que ofrece esta tecnología tan recientemente implantada. En base a esto, queda la puerta abierta a nuevas investigaciones y desarrollos punteros que puedan poner en valor esta tecnología.

## 8.2 TRABAJO FUTURO

A pesar de que el desarrollo del proyecto ha sido realizado de manera satisfactoria en base a la funcionalidad requerida, debido a la limitación de tiempo hay ciertas funcionalidades extra que no ha sido posible implementar, además de objetivos más ambiciosos a los planteados que haría de la aplicación mucho más completa. Algunas de estas posibles funcionalidades y objetivos son los siguientes:

- Publicar la aplicación en la tienda de aplicaciones de Android (Play Store), haciendo uso de una red de Ethereum pública, ya que por el momento se utiliza una red privada soportada por Ganache y la aplicación hay que desplegarla en local para poder utilizarla.

- Permitir al usuario subir documentos legales asociados al contrato.

- Implementar el pago de manera automática cuando el contrato haya finalizado en tiempo real.

- Añadir la funcionalidad de firma de contrato haciendo uso de un código QR con el objetivo de agilizar la firma de contratos.

- Implementar una solución basada en GPS para hacer uso de la localización a la hora de firmar contratos.

# 9 BIBLIOGRAFÍA


[1] Z. Zheng, S. Xie, H. Dai, X. Chen y H. Wang, «An Overview of Blockchain Technology:Architecture, Consensus, and Future Trends,» 2017. [En línea]. Available: https://www.researchgate.net/publication/318131748_An_Overview_of_Blockchain_Technology_Architecture_Consensus_and_Future_Trends. [Último acceso: 7 Abril 2024].

[2] «Binance Academy,» 2024. [En línea]. Available: https://academy.binance.com/es/articles/what-is-blockchain-and-how-does-it-work. [Último acceso: 17 Mayo 2024].

[3] «IBM,» 2024. [En línea]. Available: https://www.ibm.com/es-es/topics/blockchain. [Último acceso: 20 Abril 2024].

[4] G. KAUR, «Cointelegraph,» 30 Octubre 2023. [En línea]. Available: https://es.cointelegraph.com/learn/a-beginners-guide-to-understanding-the-layers-of-blockchain-technology. [Último acceso: 5 Abril 2024].

[5] «Ethereum.org,» 2024. [En línea]. Available: https://ethereum.org/es. [Último acceso: 11 Abril 2024].

[6] «Wikipedia,» 2024. [En línea]. Available: https://es.wikipedia.org. [Último acceso: 3 Junio 2024].

[7] «Solidity Docs,» 2024. [En línea]. Available: https://solidity-es.readthedocs.io/es/latest/. [Último acceso: 31 Mayo 2024].

[8] «Truffle Suite,» 2022. [En línea]. Available: https://archive.trufflesuite.com/docs/truffle/. [Último acceso: 31 Mayo 2024].

[9] «Asana,» 2024. [En línea]. Available: https://asana.com/es/resources/what-is-scrum. [Último acceso: 28 Mayo 2024].

[10] «Altassian,» 2024. [En línea]. Available: https://www.atlassian.com/es/agile/scrum. [Último acceso: 24 Mayo 2024].

[11] «Escueladenegocios y Direccion,» 2024. [En línea]. Available: https://www.escueladenegociosydireccion.com/revista/business/scrum-framework-agiliza-trabajo-equipo/. [Último acceso: 3 Junio 2024].

[12] «Coordinacae,» 2024. [En línea]. Available: https://www.coordinacae.com/blog/matriz-de-riesgos/. [Último acceso: 13 6 2024].

[13] «Glassdoor,» 2024. [En línea]. Available: https://www.glassdoor.es/Sueldos/ingeniero-informatico-sueldo-SRCH_KO0,21.htm. [Último acceso: 17 Junio 2024].

[14] A. Vijaywargi y U. K. Boddapati, «Architectural Patterns in Android Development: Comparing MVP, MVVM, and MVI,» 2024. [En línea]. Available: https://www.ijraset.com/best-journal/architectural-patterns-in-android-development-comparing-mvp-mvvm-and-mvi. [Último acceso: 25 Junio 2024].

[15] «Microsoft Learn,» [En línea]. Available: https://learn.microsoft.com/es-es/dotnet/architecture/maui/mvvm. [Último acceso: 26 Junio 2024].

[16] B. Eisenman, «What is React Native?,» de *Learning React Native* , O´Reilly, 2016.

[17] «React,» 2024. [En línea]. Available: https://es.legacy.reactjs.org/. [Último acceso: 24 Junio 2024].

[18] «Arsys,» 2024. [En línea]. Available: https://www.arsys.es/blog/que-es-npm-javascript-para-principiantes. [Último acceso: 24 Junio 2024].

[19] «Web3.js,» 2024. [En línea]. Available: https://web3js.readthedocs.io/en/v1.10.0/#. [Último acceso: 27 Junio 2024].

[20] Astah, 2024. [En línea]. Available: https://astah.net/pricing/academic/. [Último acceso: 17 Junio 2024].

[21] «Gitlab,» 2024. [En línea]. Available: https://about.gitlab.com/solutions/devops-platform/. [Último acceso: 28 Junio 2024].

[22] «Expo,» 2024. [En línea]. Available: https://docs.expo.dev/. [Último acceso: 29 Junio 2024].

[23] «B2BInPlay,» Junio 2023. [En línea]. Available: https://b2binpay.com/es/what-are-the-different-layers-of-blockchain-technology/. [Último acceso: 16 Abril 2024].

[24] S. Nakamoto, «Bitcoin: Un Sistema de Efectivo Electrónico Usuario-a-Usuario,» 2020. [En línea]. Available: https://bitcoin.org/files/bitcoin-paper/bitcoin_es_latam.pdf. [Último acceso: 5 Abril 2024].

[25] «NodeJs,» 2024. [En línea]. Available: https://nodejs.org/en/. [Último acceso: 24 Junio 20204].

# 10 ANEXO

## 10.1 GUÍA DE INSTALACIÓN Y USO

A continuación, se presenta una guía de instalación y de uso de la aplicación con el objetivo de clarificar cueles son los pasos a seguir para que la aplicación funcione correctamente. Es importante aclarar que esta guía está orientada a la instalación en un equipo con sistema operativo Windows.

Por otro lado, es importante destacar que es necesario contar con un dispositivo móvil Android, o en su defecto utilizar un emulador, aunque esta segunda opción es poco recomendable ya que consume demasiados recursos y es posible que el funcionamiento no sea el adecuado.

El proyecto se puede descargar mediante git desde el siguiente enlace https://gitlab.com/HP-SCDS/Observatorio/2023-2024/contractme/uva-contractme.git o directamente hacer la descarga en zip y descomprimir el proyecto en el directorio que se desee. Cabe destacar que para realizar la descarga es necesario contar con una cuenta de Gitlab que disponga de los permisos necesarios concedidos por la empresa HP.

### *10.1.1 INSTALACIÓN DE NODE, NPM Y TRUFFLE*

Para el correcto funcionamiento de la aplicación primero es necesario que el equipo cuente con Node y npm, de modo que luego sea posible instalar Truffle. Los pasos a seguir son los siguientes:

1. Descargar Node desde este enlace https://nodejs.org/en

2. Para comprobar que Node y npm se han instalado correctamente ejecutar los siguientes comandos, de manera que debemos obtener como salida la versión tanto de Node como de npm.

```
node -v
npm -v
```

3. Instalar truffle con el siguiente comando

```
npm install -g truffle
```

4. De mismo modo que antes para comprobar que Truffle está instalado correctamente ejecutar el siguiente comando

```
npm install -g truffle
```

### *10.1.2 INSTALACIÓN Y CONFIGURACIÓN DE GANACHE*

Lo siguiente es configurar la red de blockchain local, para ello necesitamos la herramienta Ganache, por lo que habrá que seguir estos pasos :

1. Instalar ganache desde el siguiente enlace https://archive.trufflesuite.com/ganache/

2. Ejecutar la aplicación y pulsar en el botón “NEW WORKSPACE”

3. Ir a la pestaña “Server” y seleccionar la opción de WI-FI, en el menú desplegable “HOSTNAME”. Opcionalmente en la pestaña “Workspace” es posible personalizar el nombre de la red si se desea, en caso contrario se generará un nombre aleatorio.

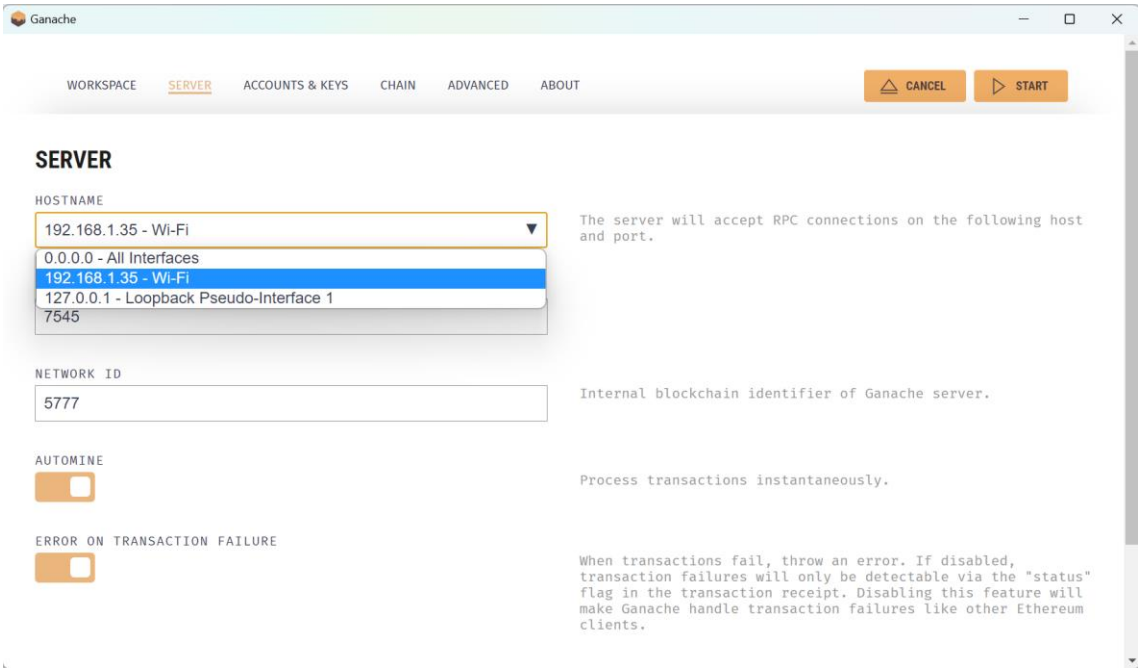


*Ilustración 27 Configuración de Ganache*

4. Pulsar sobre el botón “Start”, de manera que ya se habrá generado una nueva red local de Ethereum que consta de 10 carteras de prueba con un saldo de 100 ETH cada una

5. Finalmente ir al proyecto y en el archivo truffle-config.js cambiar la dirección del puerto a la seleccionada anteriormente. En la imagen del ejemplo sería la dirección “192.168.1.35”, el resto de los campos dejarlos como están por defecto

```
development: {
  host: "192.168.1.35", // Localhost (default: none)
  port: 7545, // Standard Ethereum port (default: none)
  network_id: "*", // Any network (default: none)
},
```

*Ilustración 28 Configuración en truffle-config.js*

### *10.1.3 COMPILACIÓN Y DESPLIEGUE DEL SMART CONTRACT*

Para realizar la compilación del Smart Contract se puede hacer uso de un editor de código como VSCode o directamente usar la terminal de comandos. Los pasos a seguir son los siguientes:

1. Navegar hasta el directorio raíz del proyecto

2. Ejecutar el siguiente comando

```
truffle migrate
```

3. Una vez ejecutado el comando, hay que comprobar que se ha desplegado correctamente en la red de Blockchain albergada por Ganache. Para ello iremos a la pestaña de "Transactions" y comprobaremos que se ha realizado una transacción similar a esta.

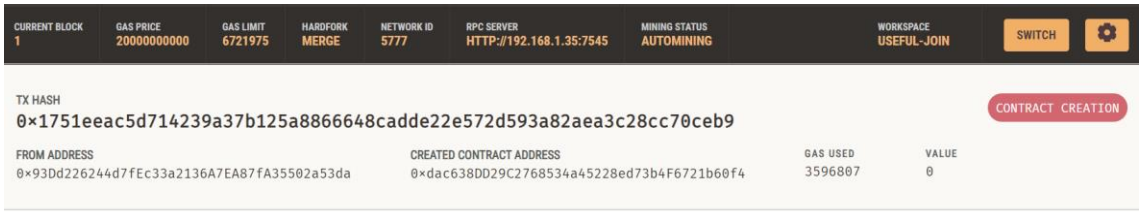


*Ilustración 29 Transacción en Ganache*

4. Una vez desplegado el contrato observaremos como se ha generado un archivo .json en la ruta /**build/contracts**. Este archivo es necesario copiarlo a la ruta /**client/contracts.** En caso de que ya exista un archivo con el mismo nombre, simplemente sobreescribirlo.

### *10.1.4 EJECUCIÓN DE LA APLICACIÓN*

Tras haber realizado toda la configuración y despliegue de la red Blockchain y el respectivo Smart Contract, lo que falta por hacer es por un lado ejecutar el proyecto desde el ordenador y por otro lado inicializar la aplicación en un móvil Android. Para ello hay que seguir los siguientes pasos:

1. Situarse en la carpeta /**client**

2. Ejecutar el siguiente comando para instalar todas las dependencias necesarias

```
npm install
```

3. Una vez instaladas todas las dependencias ejecutar el siguiente comando. En este momento expo arrancará la aplicación y quedará a la escucha de ser ejecutada en un dispositivo móvil

```
npx expo start
```


*Ilustración 30 Código QR generado por Expo*

4. Finalmente, desde un dispositivo Android descargar la aplicación Expo Go en su versión de SDK 50. La descarga se puede realizar desde el siguiente enlace https://expo.dev/go?sdkVersion=50&platform=android&device=true donde se puede obtener el archivo .apk correspondiente. Una vez instalada la aplicación escanear el código QR que ha generado Expo y la aplicación estará lista para ser utilizada desde el dispositivo móvil.