



UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática



TFG del Grado en Ingeniería
Informática

título del TFG
Documentación Técnica



Presentado por nombre alumno
en Universidad de Burgos — 17 de junio
de 2024

Tutor: nombre tutor

Índice general

Índice general	i
Índice de figuras	iii
Índice de tablas	iv
Apéndice A Plan de Proyecto Software	1
A.1. Introducción	1
A.2. Planificación temporal	1
A.3. Estudio de viabilidad	9
Apéndice B Especificación de Requisitos	13
B.1. Introducción	13
B.2. Objetivos generales	13
B.3. Catálogo de requisitos	14
B.4. Especificación de requisitos	17
Apéndice C Especificación de diseño	51
C.1. Introducción	51
C.2. Diseño de datos	51
C.3. Diseño procedimental	55
C.4. Diseño arquitectónico	58
C.5. Diseño de interfaces	58
Apéndice D Documentación técnica de programación	65
D.1. Introducción	65
D.2. Estructura de directorios	65

D.3. Manual del programador	68
D.4. Compilación, instalación y ejecución del proyecto	69
D.5. Pruebas del sistema	76
Apéndice E Documentación de usuario	79
E.1. Introducción	79
E.2. Requisitos de usuarios	79
E.3. Instalación	79
E.4. Manual del usuario	79
Apéndice F Anexo de sostenibilización curricular	81
F.1. Introducción	81
F.2. Conclusión	82

Índice de figuras

B.1. Diagrama de casos de uso para el trabajador	19
B.2. Diagrama de casos de uso para el empleador	20
C.1. Regla definida en FireBase para el control de acceso a los datos.	53
C.2. Diagrama entidad-relación	55
C.3. Diagrama de secuencia para crear un contrato	56
C.4. Diagrama de secuencia para firmar un contrato	57
C.5. Prototipo pantalla inicio de sesión	59
C.6. Prototipo pantalla principal	60
C.7. Prototipo pantalla crear contrato	61
C.8. Prototipo pantalla ver contratos del usuario	62
C.9. Prototipo pantalla información y utilidades	63
C.10. Prototipo pantalla estado cuenta del usuario	64
D.1. Directorios del proyecto	67
D.2. Descarga de Node.js	70
D.3. Instalar Node.js y npm	70
D.4. Configuración red Ganache en truffle-config.js	71
D.5. Configuración del compilador en truffle-config.js	72
D.6. Levantar red local usando Ganache	74
D.7. Compilar SmartContract usando Truffle	75
D.8. Despliegue del contrato en la red de Ganache usando Truffle	75
D.9. Configuración de conexión con el contrato	76
D.10. Pruebas unitarias sobre el SmartContract usando Truffle	77

Índice de tablas

A.1. Coste total del personal	9
A.2. Coste total del Software	10
A.3. Dependencias del proyecto y licencias bajo las cuales están regis- tradas	12
B.1. CU-1 Iniciar sesión.	21
B.2. CU-2 Gestionar perfil.	22
B.3. CU-2.1 Visualizar perfil.	22
B.4. CU-2.2 Modificar perfil.	23
B.5. CU-3 Cerrar sesión.	24
B.6. CU-4 Gestionar billetera.	25
B.7. CU-4.1 Visualizar cuenta.	26
B.8. CU-4.2 Visualizar últimos movimientos.	27
B.9. CU-5 Crear contrato.	28
B.10.CU-5.1 Crear contrato genérico.	29
B.11.CU-5.2 Crear contrato específico.	30
B.12.CU-6 Firmar contrato.	31
B.13.CU-6.1 Firmar con biométrica.	32
B.14.CU-6.2 Firmar con código QR.	33
B.15.CU-7 Buscar contratos.	34
B.16.CU-8 Visualizar contratos.	35
B.17.CU-8.1 Visualizar contratos como empleador.	36
B.18.CU-8.2 Visualizar contratos como trabajador.	37
B.19.CU-9 Administrar contratos.	38
B.20.CU-9.1 Cancelar contrato.	39
B.21.CU-9.2 Finalizar contrato.	40
B.22.CU-9.3 Liberar contrato.	41
B.23.CU-9.4 Añadir manager.	42

B.24.CU-9.5 Eliminar manager.	43
B.25.CU-10 Modificar contratos.	44
B.26.CU-10.1 Proponer modificación.	45
B.27.CU-10.2 Aceptar modificación.	46
B.28.CU-10.3 Denegar modificación.	47
B.29.CU-11 Visualizar alertas.	48
B.30.CU-12 Utilizar herramientas.	48
B.31.CU-12.1 Consultar información Ethereum.	49
B.32.CU-12.2 Usar calculadora de divisas.	50

Apéndice A

Plan de Proyecto Software

A.1. Introducción

En este apartado se recoge el ciclo de vida del proyecto, detallando los aspectos más relevantes del mismo y como se han resuelto los diferentes problemas a lo largo de su desarrollo. Se presentarán secciones que muestran de manera cronológica la justificación de las decisiones tomadas.

Para llevar a cabo este seguimiento y planificación del proyecto se ha utilizado una metodología Scrum, que ha permitido un desarrollo ágil dividido en sprints de dos semanas cada uno. Al comienzo de cada sprint se establecen las tareas y objetivos a realizar durante ese periodo. Al final de cada sprint se realizan reuniones con los tutores para valorar los resultados obtenidos y definir nuevas tareas para el siguiente sprint. Para organizar las diferentes tareas se ha usado Gitlab que permite visualizar los diferentes estados de desarrollo de las tareas.

A.2. Planificación temporal

La propuesta del proyecto consistía en crear una aplicación android basada en blockchain que simplifica la contratación y la verificación a través de contratos inteligentes. Los requisitos principales del proyecto se pueden dividir en los siguiente puntos:

- **Tecnología Blockchain:** Utilizar la tecnología blockchain para desplegar contratos inteligentes que gestionen automáticamente los contratos laborales, desde su creación hasta su ejecución.

- **Contratos Inteligentes:** Implementar contratos inteligentes en Python, Vyper y Solidity.
- **Localización GPS:** Integrar tecnología GPS para permitir a los empleadores imponer zonas de trabajo específicas.
- **Identificación segura mediante dispositivo móvil:** Implementar autenticación biométrica y el escaneo de códigos QR.
- **Integración con Pagos:** Incorporar procesamiento de pagos dentro de la aplicación para facilitar transacciones rápidas y seguras
- **Desarrollo de Aplicación Móvil:** Diseñar una interfaz de usuario amigable para dispositivos móviles que facilite la creación de contratos, seguimiento y pago.

Una vez expuestos los requerimientos principales del proyecto, la etapa inicial del proyecto se basó en una exhaustiva investigación que permitió obtener un conocimiento detallado sobre las tecnologías y herramientas necesarias. Dado que inicialmente no contaba con conocimientos previos en desarrollo de aplicaciones móviles y tecnología blockchain, esta investigación fue crucial para identificar las mejores prácticas y soluciones en estos campos. Esta etapa de investigación y adaptación a las nuevas tecnologías tuvo una duración de aproximadamente 3 semanas, durante el cual se hizo especial énfasis en entender a fondo el funcionamiento de la blockchain y los contratos inteligentes. Este aprendizaje teórico se reforzó de manera práctica con diversos proyectos usando Truffle, completando videoTutoriales y realizando el curso interactivo "CryptoZombies", además de consultar numerosos artículos especializados. Estas actividades facilitaron la asimilación del nuevo lenguaje de programación y familiarización con el entorno blockchain.

Sprint 0

Este sprint se desarrolló entre los días 3 y 17 de Noviembre de 2023. Se realizaron las siguientes tareas y objetivos:

1. **Configuración repositorio:** Se configuró el repositorio en GitLab y se establecieron ramas principales basadas en el flujo de trabajo de GitFlow, incluyendo 'main' para la producción y 'develop' para el desarrollo

2. **Configuración entorno para la redacción de la memoria:** Se descargaron las plantillas y se configuró el software de redacción "Texmaker".
3. **Investigación de tecnologías para realizar una app móvil:** Se realizó un análisis comparativo de las plataformas de desarrollo móvil más populares. Se evaluaron criterios como el rendimiento, la facilidad de uso y la comunidad de desarrolladores.
4. **Diseñar la arquitectura del proyecto:** Se definieron los componentes principales del sistema y su interacción. Para facilitar la comprensión de la estructura y el flujo de datos se desarrollaron diagramas.

Sprint 1

Este sprint se desarrolló entre los días 17 de Noviembre y 1 de Diciembre de 2023. Se realizaron las siguientes tareas y objetivos:

1. **Investigación Smart Contracts:** Se evaluó la elección entre usar tokens fungibles o no fungibles para el desarrollo de los contratos, optándose finalmente por el estándar ERC-721, característico de los NFTs, debido a su adaptabilidad al proyecto. A partir de esta decisión fue necesario evaluar la versión adecuada para el compilador de solidity que asegure la compatibilidad con ciertas bibliotecas necesarias.
2. **Creación de un prototipo visual:** Se diseñó un prototipo visual inicial para la aplicación móvil utilizando herramientas como Figma y Canvas. El prototipo se centró en crear una interfaz intuitiva y atractiva que refleje las funcionalidades claves del proyecto.

Sprint 2

Este sprint se desarrolló entre los días 1 de Diciembre y 21 de Diciembre de 2023. Se realizaron las siguientes tareas y objetivos:

1. **Primera implementación del contrato inteligente:** Siguiendo el estándar ERC-721 se planteó una primera solución que incluyese las funcionalidades básicas de creación y transferencia de NFTs de manera segura y eficiente.
2. **Configuración entorno de desarrollo aplicación móvil:** Se configuró el entorno de desarrollo para asegurar la correcta conectividad

entre la aplicación móvil y el backend, representado por el blockchain y el contrato inteligente.

Sprint 3

Este sprint se desarrolló entre los días 21 de Diciembre de 2023 y 12 de Enero de 2024. Se realizaron las siguientes tareas y objetivos:

1. **Continuación con la implementación del contrato inteligente:** Se continuó con el desarrollo del contrato inteligente implementando funciones que manejasen la lógica de los pagos dentro de la app. También se programación la lógica para la correcta destrucción del contrato una vez ha transcurrido su ciclo de vida.
2. **Creación de pruebas unitarias para comprobar el correcto funcionamiento del contrato inteligente:** Para garantizar el correcto funcionamiento del contrato inteligente se realizaron test en Java utilizando las herramientas Ganache y Truffle. Ganache sirvió para simular un entorno blockchain local y Truffle sirvió para compilar, migrar y probar el contrato inteligente con los test creados.
3. **Implementar navegación en la aplicación móvil:** Se programaron las pantallas principales de la aplicación. Estas pantallas se integraron mediante un menú interactivo que facilita la navegación dentro de la app mejorando la experiencia del usuario.
4. **Tareas de ordenación de repositorio:** Se realizaron tareas de ordenación y limpieza en el repositorio del código. Incluyó la reorganización de directorios, la eliminación de archivos obsoletos y la estandarización de nombres de archivos y carpetas para mejorar la accesibilidad y el mantenimiento.
5. **Modificación del ReadMe:** Se actualizó el archivo ReadMe para reflejar los cambios recientes en el proyecto y proporcionar una guía más clara y detallada para los usuarios que quieran replicar el proyecto.

Debido a las fechas de vacaciones de Navidad, a los compromisos académicos y la gran demanda de los issues, no se logró completar lo previsto para este sprint, por lo que se decidió en la reunión del 12 de Enero ampliar 2 semanas más este sprint.

Sprint 4

Este sprint se desarrolló entre los días 26 de Enero y 9 de Febrero de 2024. Se realizaron las siguientes tareas y objetivos:

1. **Integración de Metamask:** Se integró la billetera Metamask para facilitar las transacciones en la aplicación. Esto incluyó configurar la autenticación y las firmas de transacciones a través de la popular billetera de criptomonedas, permitiendo a los usuarios interactuar de manera segura a través de la extensión de navegador de Metamask.
2. **Documentación de tecnologías utilizadas:** Tras el progresivo avance del proyecto y el amplio repertorio de frameworks y librerías utilizadas, se plasmaron y explicaron todas ellas en su correspondiente apartado en la memoria.

Sprint 5

Este sprint se desarrolló entre los días 9 de Febrero y 23 de Febrero de 2024. Se realizaron las siguientes tareas y objetivos:

1. **Creación de la guía de instalación:** Se elaboró una guía de instalación detallada para facilitar la configuración inicial del sistema por parte de los usuarios. Recogiendo las instrucciones paso a paso para la instalación del software, configuración del entorno y la conexión con las dependencias necesarias.
2. **Documentación del contexto teórico:** Se empezó a recoger en la memoria todos los fundamentos teóricos que fundamentan este proyecto, proporcionando una base sólida que ayude a entender el marco en el que se desarrolla la aplicación.
3. **Implementación de pantallas informativas:** Se implementaron dos pantallas clave en la aplicación móvil: la pantalla "Wallet", que permite al usuario la conexión con la billetera y muestra los datos de la cuenta seleccionada en la billetera Metamask, y la pantalla "Info", que incluye información actualizada sobre el precio del Ethereum, junto con un gráfico interactivo y una calculadora para conversiones.
4. **Implementar funcionalidad modificar contrato:** Se extendieron las funciones del contrato inteligente para permitir modificar un contrato existente. Esto incluyó la adición de métodos para actualizar

parámetros del contrato y funciones que permiten a los usuarios adaptar el contrato a necesidades cambiantes, asegurando así una mayor flexibilidad.

5. **Creación base de datos para el almacenamiento de las cuentas de los usuario:** Se implementó una base de datos utilizando Firebase de Google, aprovechando la amplia gama de funcionalidades que incluye. De esta forma se permitía a los usuarios iniciar sesión con su cuenta google, Facebook y Github entre muchas otras, facilitando un proceso de autenticación rápido y seguro, además de almacenar de manera eficiente los datos de las cuentas de los usuarios.

Sprint 6

Este sprint se desarrolló entre los días 23 de Febrero y 8 de Marzo de 2024. Se realizaron las siguientes tareas y objetivos:

1. **Modularizar código:** Debido a que los smart contracts en la Ethereum Virtual Machine (EVM) están sujetos a una restricción de tamaño para prevenir ataques de denegación de servicio, el contrato que me encontraba desarrollando se encontraba en el límite de tamaño permitido, impidiendo futuras ampliaciones. Para resolver esto, se optó por modularizar el contrato inteligente en diferentes componentes. Esto no solo evitó la necesidad de desplegar múltiples contratos, lo cual hubiera sido más costoso en términos de gas, sino que también ayudó en la organización del código.
2. **Validación de datos de entrada en Smart Contract:** Aunque inicialmente el contrato inteligente ya incluía algunas comprobaciones básicas en el código Solidity y la aplicación también verificaba las entradas, se realizaron mejoras significativas en estas validaciones para reforzar la seguridad.
3. **Ampliación funcionalidad SmartContract:** Se añadió código adicional al contrato inteligente para permitir que usuarios adicionales, aparte del dueño original, pudieran tener permisos de gestión del contrato. Esto fue diseñado para facilitar la colaboración permitiendo que usuarios autorizados puedan realizar cambios sin depender el propietario inicial.
4. **Modificación lógica contrato:** Se modificó la lógica del contrato para permitir la creación de contratos sin asignar inicialmente un

trabajador. Esta funcionalidad permitió mostrar contratos disponibles en una especie de "tablón de ofertas de trabajo", donde las personas en búsqueda de empleo pueden encontrar y seleccionar contratos que coincidan con sus habilidades y expectativas.

Los siguientes objetivos que se exponen a continuación, no han podido ser completados y han sido pospuestos hasta el sprint 10. Hasta ahora, la aplicación se ha estado probando de manera local en una pestaña del navegador por razones de eficiencia y comodidad. Sin embargo, al intentar realizar las pruebas con un teléfono móvil para funcionalidades que requieren ejecución nativa, como el escaneo de códigos QR o el acceso a la ubicación, se han encontrado errores debido a incompatibilidades con las bibliotecas y el sistema operativo Android. Por tanto, estos problemas se han tenido que posponer hasta que se solucione el problema de compatibilidad de la app con la biblioteca web3.

1. **Implementación de códigos QR:** La integración de códigos QR, diseñada para facilitar transacciones rápidas y seguras mediante escaneo, se ha pospuesto debido a problemas técnicos encontrados durante las pruebas en dispositivos móviles.
2. **Inclusión ubicación a los parámetros del contrato:** Del mismo modo que con los códigos QR, por la imposibilidad de probar los servicios de ubicación en un dispositivo móvil, se ha tenido que posponer.
3. **Creación de un filtro para buscar contratos:** Este issue ha sido pospuesto ya que depende directamente de la incorporación de la ubicación como parámetro para realizar el filtro.
4. **Creación de eventos para registrar alertas en los contratos:** Debido a la importancia de solucionar el problema de despliegue en teléfonos móvil, este issue ha pasado a segundo plano.

Sprint 7

Este sprint se desarrolló entre los días 8 de Marzo y 22 de Marzo de 2024. Se realizaron las siguientes tareas y objetivos:

1. **Resolver error compatibilidad app android y biblioteca web3:** Este issue tuvo prioridad máxima y se le dedicó únicamente para

el. Al ejecutar la aplicación en un dispositivo móvil aparecía un problema de compatibilidad significativo entre la biblioteca web3 y el motor JavaScript Hermes utilizado en React Native. Este problema incapacitaba cualquier avance para el que se requiriese las funcionales nativas del dispositivo móvil.

2. **adaptar aplicación web a aplicación móvil:** La aplicación en su primera etapa de desarrollo fue hecha desde un entorno web. Por lo que una vez solucionado el problema anterior que impedía desplegar el proyecto en un dispositivo móvil, fue necesario enfrentarse a diversos desafíos adicionales al adaptar características específicas de la web para funcionar en un entorno móvil.

Después de varias horas de esfuerzo, se logró identificar y corregir el error más importante hasta el momento. Además de cumplir con los requisitos establecidos para este sprint, también se consiguió implementar la funcionalidad para mostrar códigos QR con los detalles del contrato, el cual era uno de los objetivos que habían quedado pendientes del sprint anterior.

Sprint 8

Este sprint se desarrolló entre los días 22 de Marzo y 5 de Abril de 2024. Se realizaron las siguientes tareas y objetivos:

Este sprint se centró principalmente en resolver desafíos relacionados con la migración de la aplicación web a móvil, especialmente en lo que respecta a la conectividad con la billetera externa Metamask.

1. **Importar cuentas de Ganache a la app móvil de Metamask:** Esto representaba inicialmente una fase intermedia entre las pruebas locales y el despliegue en una red real. Por errores de conexión se llegó a la conclusión de que no era posible, por lo que se decidió omitir este paso ya que su eliminación no afecta el progreso hacia el entorno de producción.
2. **Gestionar conexión de la billetera desde la app:** Directamente conectar con Metamask resultó inviable debido a restricciones técnicas, por lo que se evaluó para adaptar el sistema para utilizar Wallet Connect junto con las librerías wagmi y viem, permitiendo la conexión con más de 300 billeteras diferentes

A.3. Estudio de viabilidad

El estudio de viabilidad es un análisis crítico realizado para evaluar la posibilidad y conveniencia de llevar a cabo un proyecto específico. Para determinar si este proyecto es viable, se van a desarrollar la viabilidad económica y la viabilidad legal.

Viabilidad económica

La viabilidad económica de un proyecto evalúa si los beneficios económicos previstos justifican los costos involucrados. Se considerarán los costes de recursos humanos, el material empleado y el Software usado

Coste de personal

Para realizar una estimación de los costos del proyecto, vamos a acotar el calculo a una duración de 9 meses, que ha sido el tiempo empleado en realizar el proyecto. Por otro lado, como solo ha habido una persona desarrollando (el alumno), se va a considerar un único trabajador para los cálculos Partiendo de un salario base promedio de un programador junior en España de 21000 brutos anuales, podemos estimar que el coste total personal sería:

Concepto	Coste (€)
Salario bruto (anual)	21000
Retención del IRPF (12 %)	2520
Seguridad Social (28,3 %)	5943
Salario neto (anual)	12537
Salario neto (mensual)	1044.75
Coste total en los 9 meses	15750

Tabla A.1: Coste total del personal

Coste hardware

El coste del hardware para la realización del proyecto únicamente ha sido necesario contar con un ordenador para el desarrollador. El ordenador utilizado para el desarrollo tiene una antigüedad de 7 años, por lo que ya ha sido amortizado y se desprecia el costo del ordenador.

Coste SoftWare

Las herramientas de software empleadas en este proyecto son principalmente gratuitas; sin embargo, los servicios de Firebase e Infura presentan ciertas restricciones en sus versiones sin costo que podrían limitar la operatividad al escalar la aplicación. FiraBase en su versión gratuita ofrece 1 GiB de almacenamiento, que bien es un limitante, y soporta hasta 50000 usuarios activos mensuales. Por otro lado, Infura proporciona 100000 peticiones diarias gratuitamente.

Considerando un escenario de total éxito en el lanzamiento de la aplicación, donde se estimen unos 200000 usuarios activos mensuales, las necesidades podrían requerir una gran ampliación de almacenamiento, lo cual vendría acompañado de un gran volumen de operaciones sobre la base de datos. Para Infura se contemplaría la necesidad de escalar al plan más avanzado, que permite hasta 5 millones de peticiones diarias. Los costos, inicialmente en dólares han sido convertidos a euros a 0.92€

Concepto	Coste (€)
Infura	917.90
190.500.000 operaciones lectura	113
999 GiB Almacenamiento	165.22
53.100.000 operaciones escritura	86.74
200.000 usuarios activos	633.35
Coste total mensual	1916.21

Tabla A.2: Coste total del Software

Beneficios

El proyecto no se ha realizado con el objetivo de obtener un beneficio económico. Este producto ha sido desarrollado con el fin de implementar una herramienta para abordar los desafíos del mercado laboral, especialmente en sectores afectados por la informalidad como la agricultura y los servicios domésticos. Se considera por tanto que este proyecto tiene un carácter social, destinado a mejorar las condiciones laborales y a promover la equidad en el trabajo.

Se considera que la financiación del proyecto debería provenir principalmente de inversores interesados en la optimización y reducción de costos del proceso de contratación. Adicionalmente, el gobierno español representa

un potencial inversor clave, ya que podría promover activamente el uso de esta aplicación para fomentar prácticas laborales adecuadas y combatir la economía sumergida.

Viabilidad legal

Esta sección se centra en analizar los aspectos legales asociados con el desarrollo y distribución de una solución basada en blockchain. Los puntos claves que incluyen el análisis de las licencias de software y el cumplimiento de las leyes vigentes.

Licencias de software

En el proyecto se emplea diversas bibliotecas y frameworks que están regidos bajo licencias de software específicas, que definen los términos y condiciones para su uso, modificación y distribución. A continuación, se va a listar las diferentes licencias de cada dependencia de mi proyecto, en orden de menos restrictiva a mas restrictiva:

- **Licencia MIT:** No impone restricciones significativas, permitiendo uso comercial, modificación, distribución y uso privado.
- **Licencia Apache 2.0:** Similar a la licencia MIT, pero también proporciona una protección explícita contra patentes.
- **Términos de servicio de Google:** Específicos para los productos de Google, no son licencias de software per se, pero regulan cómo se pueden usar los servicios.
- **GNU GPL:** Permite uso comercial, modificación, distribución y uso privado, pero cualquier versión modificada debe también ser libre.
- **GNU GPL-3.0:** Similar a la GPL pero con términos adicionales para cerrar algunas brechas legales en ciertos escenarios.

Dependencia	Licencia
React Native	Licencia MIT
Expo	Licencia MIT
Firebase	términos de servicio de Google
Solidity	Licencia GPL-3.0
Remix	Licencia GPL-3.0
Truffle	Licencia MIT
Ganache	Licencia MIT
MetaMask	Licencia GPL-3.0
WalletConnect	Licencia MIT
Node.js	Licencia MIT
OpenZeppelin	Licencia Apache 2.0
web3	Licencia MIT
ethers	Licencia MIT

Tabla A.3: Dependencias del proyecto y licencias bajo las cuales están registradas

Regulación legal

Debido a que este proyecto implicará la gestión de datos personales, es fundamental cumplir con el Reglamento General de Protección de Datos (RGPD) de la UE y la legislación española correspondiente, como la Ley Orgánica de Protección de Datos Personales y garantía de los derechos digitales (LOPDGDD). Esto implica garantizar la protección adecuada de los datos personales y obtener consentimientos claros por parte de los usuarios para usar sus datos.

Actualmente no hay ninguna legislación vigente en España en cuanto a los contratos inteligentes. Sin embargo, estos debería de cumplir con el código civil en términos de formación de contratos, capacidad para contratar y consentimiento. Por su puesto, la aplicación deberá cumplir con la legislación laboral española, asegurando que los contratos creados cumplan con las normativas de trabajo, incluyendo las mínimas garantías y derechos para los trabajadores.

Apéndice B

Especificación de Requisitos

B.1. Introducción

En esta sección se hace referencia a los requerimientos que debe cumplir el software para satisfacer las necesidades del cliente e identificar los componentes necesarios para entregar un producto adecuado.

B.2. Objetivos generales

A lo largo de la memoria ya se han definido distintos objetivos del proyecto, pero estos pueden ser resumidos en los siguientes puntos:

- **Tecnología Blockchain:** Uso de la tecnología blockchain como base para la creación de un sistema de contratación descentralizado y seguro. Se busca garantizar la transparencia e inmutabilidad de los datos, permitiendo que todas las transacciones queden registradas de forma segura.
- **Contratos Inteligentes:** Desarrollo de contratos inteligentes que aseguran el cumplimiento de acuerdos laborales. Estos contratos deben de ser capaz de autoejecutarse en función de las condiciones preestablecidas por las partes. El principal objetivo es reducir la necesidad de intermediarios.
- **Identificación segura mediante dispositivo móvil:** Implementación de métodos de autenticación y verificación seguros utilizando

dispositivos móviles, como biometría o códigos QR. Se pretende asegurar que solo los usuarios autorizados puedan acceder a un contrato y realizar operaciones dentro del sistema.

- **Desarrollo de Aplicación Móvil:** Creación de una aplicación móvil android que sirva de interfaz para interactuar con los contratos inteligentes y la blockchain, escondiendo toda la complejidad al usuario. La aplicación debe de estar diseñada para ser intuitiva, con el objetivo de que sea fácilmente utilizable para personas con cualquier nivel de habilidades tecnológicas.

B.3. Catálogo de requisitos

En este apartado se van a listar los requisitos específicos, agrupados en requisitos funcionales y no funcionales.

Requisitos funcionales

Este tipo de requerimientos están orientados a detallar las funciones que debe realizar la aplicación para cumplir con las expectativas del usuario. Por lo tanto se expondrá en como debe de ser el comportamiento del sistema en cada caso determinado.

Usuario: El usuario dependiendo del contexto puede adoptar tanto los roles de empleador como de trabajador. Ofreciendo una gran flexibilidad permitiendo que los usuarios puedan gestionar sus contratos desde la perspectiva de quien ofrece trabajo o desde quien lo ejecuta. Por lo que se pueden diferenciar estos dos roles en la aplicación, usuarios empleadores y usuarios trabajadores.

- **RF.1 Usuarios:** Se ha de incluir una gestión básica de los usuarios, garantizando una experiencia de usuario segura y personalizada.
 - **RF.1.1 Inicio de sesión:** Un usuario podrá acceder a la aplicación con su correo y contraseña.
 - **RF.1.2 Visualización del perfil:** Cualquier usuario podrá consultar los datos de su perfil.
 - **RF.1.3 Modificación del perfil:** Cualquier usuario podrá modificar los datos de su perfil

- **RF.1.4 Cierre de sesión:** Un usuario podrá cerrar sesión de forma segura, borrando las credenciales guardadas de su dispositivo.
- **RF.2 Billetera:** Se ha de proveer una herramienta que permita al usuario manejar y visualizar los aspectos financieros asociados con su cuenta en la aplicación.
 - **RF.2.1 Visualización estado cuenta:** El usuario debe poder consultar el saldo actual de su billetera.
 - **RF.2.2 Visualización últimos movimientos** El usuario debe poder revisar sus últimos movimientos, incluyendo ingresos y regresos.
- **RF.3 Creación de contratos:** Un usuario podrá crear y configurar un contrato, permitiendo formular acuerdos claros y vinculantes.
 - **RF.3.1 Creación de contratos laborales genéricos:** Posibilidad de crear contratos que sin especificar previamente la identidad del trabajador.
 - **RF.3.2 Creación de contratos laborales específicos:** Posibilidad de crear un contrato dirigido a una persona específica.
- **RF.4 Firma de contratos:** Se debe asegurar que la firma de contratos sea segura y verificable, utilizando tecnología avanzada para la autenticación.
 - **RF.4.1 Firma con autenticación biométrica:** Requerir firma de contratos utilizando autenticación biométrica, como reconocimiento facial o huella digital
 - **RF.4.2 Firma empleando código QR:** Facilitar el uso de un código QR para verificar y firmar un contrato de manera segura.
- **RF.5 Búsqueda de contratos:** Se ha de proveer de una herramienta que facilite el descubrimiento de ofertas de trabajo. Permitiendo al usuario encontrar rápidamente las ofertas que mejor se adapten a sus necesidades y habilidades.
- **RF.6 Visualización de contratos:** El usuario ha de poder consultar los contratos en los que esté involucrado, así como poder acceder a aquellos contratos que hayan expirado.

- **RF.6.1 Acceso a los contratos como empleador:** Permite al usuario revisar y gestionar todos los contratos en los que posee el rol de empleador.
- **RF.6.2 Acceso a los contratos como trabajador:** Permite al usuario revisar y gestionar todos los contratos en los que posee el rol de trabajador.
- **RF.7 Datos contratos:** El usuario ha de poder revisar detalladamente los contratos en los que esté involucrado, ofreciendo un acceso completo a toda la información vinculante.
- **RF.8 Administración de contratos:** El usuario ha de poder gestionar los activamente sus contratos, proporcionando herramientas para cancelar, finalizar y liberar el pago de los acuerdos según sea necesario.
 - **RF.8.1 Cancelación contrato:** Permite al usuario cancelar un contrato antes de que se firme.
 - **RF.8.2 Finalización contrato:** Posibilita la finalización de un contrato de manera formal una vez que se hayan cumplido los términos acordados.
 - **RF.8.3 Liberación del pago:** Facilita la liberación y el traspaso de los fondos acordados una vez cumplidos los términos del contrato.
 - **RF.8.4 Añadir nuevo manager:** Permite agregar un nuevo manager para la gestión del contrato.
 - **RF.8.5 Eliminar manager existente:** Posibilita la eliminación de un manager actualmente asignado al contrato.
- **RF.9 Modificación de contratos:** Se debe proporcionar una herramienta para que el usuario pueda registrar modificaciones en los contratos, asegurando que todos los cambios sean consensuados y legalmente vinculantes.
 - **RF.9.1 Propuesta modificación:** Permite al usuario proponer modificaciones al contrato existente.
 - **RF.9.2 Aceptar modificación:** Opción para que la otra parte acepte las modificaciones propuestas.
 - **RF.9.3 Denegar modificación:** Opción para que la otra parte rechace las modificaciones propuestas.

- **RF.10 Alertas de usuario:** Se debe garantizar que el usuario esté siempre informado sobre los eventos importantes relacionados con sus contratos, se debe de crear un sistema de alertas que notifiquen al usuario de cualquier cambio en sus contratos.
- **RF.11 Proporcionar herramientas:** Se ofrecer herramientas adicionales que apoyen al usuario con la gestión y análisis de sus contratos.
 - **RF.11.1 Precio Ethereum:** Se debe mostrar el precio actual del Ethereum en euros.
 - **RF.11.2 Gráfico histórico:** Se han de mostrar gráficos con el precio histórico del ethereum a lo largo del tiempo.
 - **RF.11.3 Calculadora divisas:** Se debe proveer una calculara que permita hacer conversiones ethereum-euro y viceversa.

Requisitos no funcionales

Estos requisitos son utilizados para especificar criterios que puedan ser usados ara juzgar la operación del sistema, más que detallar un comportamiento específico.

- **RNF.1 Seguridad:**Garantizar la integridad y confidencialidad de la información del usuario mediante tecnologías avanzadas.
- **RNF.2 Usabilidad:** El sistema debe ser fácil de usar y navegar, minimizando la necesidad de capacitación extensiva.
- **RNF.3 Mantenibilidad:** Facilitar la actualización y el mantenimien- to del sistema sin afectar su funcionamiento continuo.
- **RNF.4 Rendimiento:** El sistema debe mantener un alto rendimiento bajo variadas condiciones de carga.

B.4. Especificación de requisitos

Se detalla a continuación la especificación de requisitos de la aplicación propuesta basada en el catálogo de requisitos.

Diagrama de casos de uso

En esta sección se mostrarán los diagramas de casos de uso. En la aplicación hay dos actores: el empleador con capacidad de crear contratos y administrarlos, y el trabajador, con la capacidad de firmar contratos. Por motivos de legibilidad, se ha decidido dividir el diagrama general en dos, creando un diagrama para cada actor.



Figura B.1: Diagrama de casos de uso para el trabajador

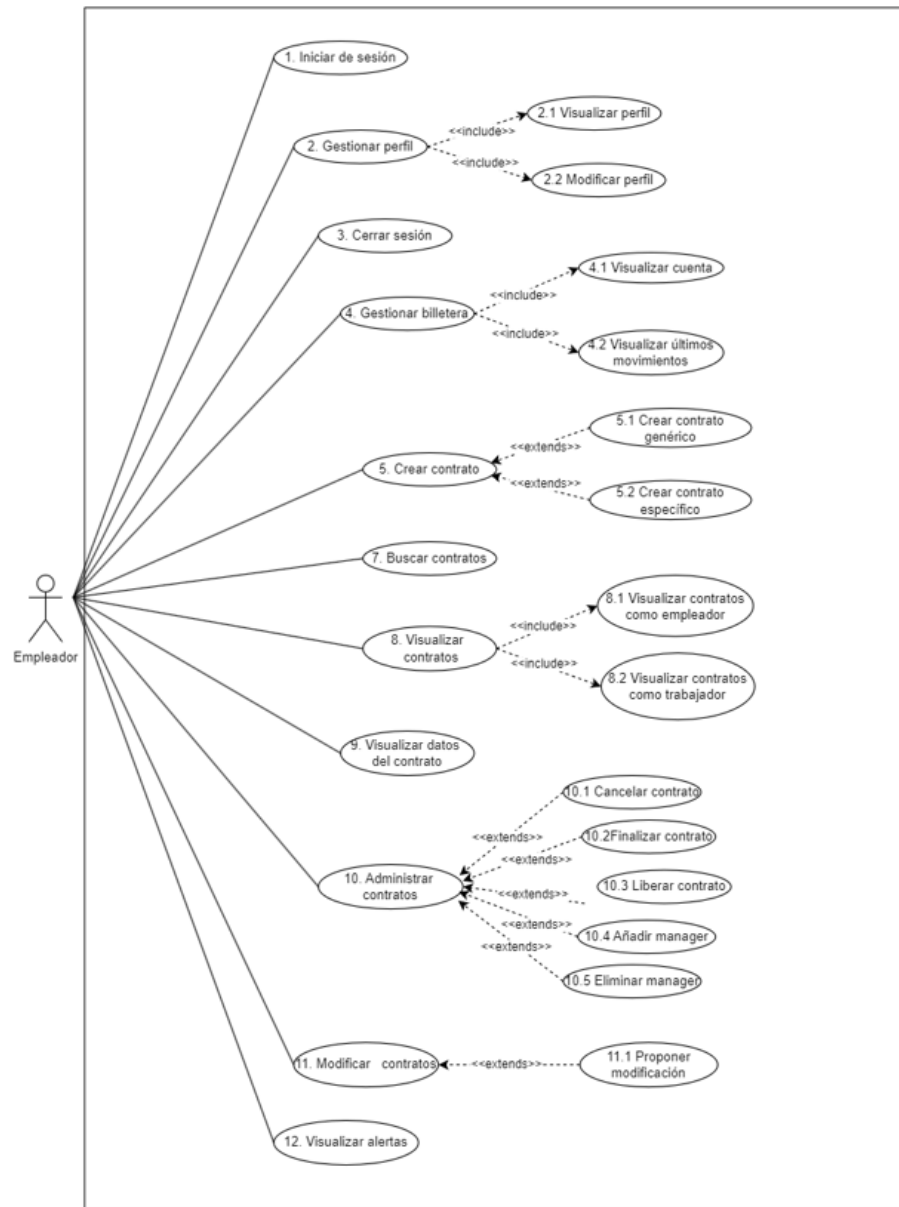


Figura B.2: Diagrama de casos de uso para el empleador

CU-1	Iniciar sesión
Versión	1.0
Autor	David Martínez Bahillo
Requisitos asociados	RF-1.1
Descripción	Permite al usuario acceder a su cuenta proporcionando sus credenciales de autenticación.
Precondición	<ol style="list-style-type: none">1. El usuario debe estar registrado.2. El usuario no debe tener una sesión activa.
Acciones	<ol style="list-style-type: none">1. El usuario introduce su nombre de usuario y contraseña.2. El usuario selecciona la opción "Iniciar sesión".3. El sistema verifica la información y concede el acceso si los datos son correctos.
Postcondición	El usuario accede a su cuenta y puede utilizar las funcionalidades del sistema.
Excepciones	Excepciones administradas por FireBase
Importancia	Alta

Tabla B.1: CU-1 Iniciar sesión.

CU-2	Gestionar perfil
Versión	1.0
Autor	David Martínez Bahillo
Requisitos asociados	RF-1.2, RF-1.3
Descripción	Permite al usuario visualizar y modificar su perfil.
Precondición	<ol style="list-style-type: none"> 1. El usuario debe estar registrado y logueado. 2. El sistema debe estar operativo.
Acciones	El usuario selecciona «Perfil».
Postcondición	Se volverá a la pagina inicio tras completar la acción necesaria.
Excepciones	Ninguna
Importancia	Baja

Tabla B.2: CU-2 Gestionar perfil.

CU-2.1	Visualizar perfil
Versión	1.0
Autor	David Martínez Bahillo
Requisitos asociados	RF-1.2
Descripción	Permite al usuario consultar los datos de su perfil en la plataforma.
Precondición	<ol style="list-style-type: none"> 1. El usuario debe estar registrado y logueado. 2. El sistema debe estar operativo.
Acciones	El usuario selecciona «Perfil».
Postcondición	Se volverá a la página de inicio tras consultar el perfil.
Excepciones	Ninguna
Importancia	Baja

Tabla B.3: CU-2.1 Visualizar perfil.

CU-2.2	Modificar perfil
Versión	1.0
Autor	David Martínez Bahillo
Requisitos asociados	RF-1.3
Descripción	Permite al usuario modificar los datos de su perfil.
Precondición	Usuario autenticado y visualizando su perfil.
Acciones	<ol style="list-style-type: none">1. El usuario selecciona la opción de «modificar perfil».2. El usuario edita la información deseada y confirma los cambios.3. El sistema valida y guarda los cambios.
Postcondición	Los datos del perfil del usuario son actualizados.
Excepciones	Excepciones administradas por FireBase
Importancia	Baja

Tabla B.4: CU-2.2 Modificar perfil.

CU-3	Cerrar sesión
Versión	1.0
Autor	David Martínez Bahillo
Requisitos asociados	RF-1.4
Descripción	Permite al usuario cerrar su sesión activa en la plataforma.
Precondición	<ol style="list-style-type: none"> 1. El usuario debe estar registrado y logueado.
Acciones	<ol style="list-style-type: none"> 1. El usuario selecciona la opción «Cerrar sesión». 2. El sistema termina la sesión y redirige al usuario a la pantalla de inicio.
Postcondición	El usuario deja de estar logueado en el sistema.
Excepciones	Ninguna
Importancia	Media

Tabla B.5: CU-3 Cerrar sesión.

CU-4	Gestionar billetera
Versión	1.0
Autor	David Martínez Bahillo
Requisitos asociados	RF-2.1, RF-2.2
Descripción	Permite al usuario visualizar su cuenta y movimientos recientes.
Precondición	<ol style="list-style-type: none">1. El usuario debe estar registrado y logueado.
Acciones	<ol style="list-style-type: none">1. El usuario selecciona la opción «Cartera».2. El sistema muestra las opciones de visualización de cuenta y movimientos.
Postcondición	El usuario visualiza la información de su billetera.
Excepciones	Ninguna
Importancia	Media

Tabla B.6: CU-4 Gestionar billetera.

CU-4.1	Visualizar cuenta
Versión	1.0
Autor	David Martínez Bahillo
Requisitos asociados	RF-2.1
Descripción	Permite al usuario ver el la dirección y activos de su cuenta en la plataforma.
Precondición	<ol style="list-style-type: none">1. El usuario debe estar registrado y logueado.2. El usuario ha seleccionado «Cartera».
Acciones	<ol style="list-style-type: none">1. El sistema muestra los detalles de la cuenta.
Postcondición	El usuario conoce el estado de su cuenta.
Excepciones	Ninguna
Importancia	Media

Tabla B.7: CU-4.1 Visualizar cuenta.

CU-4.2	Visualizar últimos movimientos
Versión	1.0
Autor	David Martínez Bahillo
Requisitos asociados	RF-2.2
Descripción	Permite al usuario ver los últimos movimientos financieros en su cuenta.
Precondición	<ol style="list-style-type: none">1. El usuario debe estar registrado y logueado.2. El usuario ha seleccionado «Cartera».
Acciones	<ol style="list-style-type: none">1. El sistema muestra la lista de movimientos recientes.
Postcondición	El usuario conoce sus últimos movimientos financieros.
Excepciones	No hay últimos movimientos
Importancia	Media

Tabla B.8: CU-4.2 Visualizar últimos movimientos.

CU-5	Crear contrato
Versión	1.0
Autor	David Martínez Bahillo
Requisitos asociados	RF-3.1, RF-3.2
Descripción	Permite al usuario iniciar el proceso de creación de un contrato, seleccionando entre un contrato genérico o específico.
Precondición	<ol style="list-style-type: none"> 1. El usuario debe estar registrado y logueado. 2. El usuario debe tener activos suficientes para crear el contrato.
Acciones	<ol style="list-style-type: none"> 1. El usuario selecciona la opción «Crear contrato». 2. El usuario completa los campos necesarios y confirma la creación del contrato.
Postcondición	El usuario inicia el proceso de creación de un contrato.
Excepciones	Ninguna
Importancia	Alta

Tabla B.9: CU-5 Crear contrato.

CU-5.1	Crear contrato genérico
Versión	1.0
Autor	David Martínez Bahillo
Requisitos asociados	RF-3.1
Descripción	Permite al usuario crear un contrato genérico con plantillas prediseñadas.
Precondición	<ol style="list-style-type: none"> 1. El usuario debe estar registrado y logueado. 2. El usuario debe tener activos suficientes para crear el contrato. 3. El usuario no especifica la cuenta del trabajador.
Acciones	<ol style="list-style-type: none"> 1. El usuario selecciona la opción «Crear contrato». 2. El usuario completa los campos necesarios excepto el campo de "trabajador". 3. El usuario confirma la creación del contrato.
Postcondición	Se crea un nuevo contrato genérico según las especificaciones del usuario.
Excepciones	Errores en la entrada de datos.
Importancia	Alta

Tabla B.10: CU-5.1 Crear contrato genérico.

CU-5.2	Crear contrato específico
Versión	1.0
Autor	David Martínez Bahillo
Requisitos asociados	RF-3.2
Descripción	Permite al usuario crear un contrato personalizado, especificando todos los detalles necesarios.
Precondición	<ol style="list-style-type: none"> 1. El usuario debe estar registrado y logueado. 2. El usuario debe tener activos suficientes para crear el contrato. 3. El usuario especifica la cuenta del trabajador.
Acciones	<ol style="list-style-type: none"> 1. El usuario selecciona la opción «Crear contrato». 2. El usuario completa los campos necesarios incluyendo el campo de "trabajador". 3. El usuario confirma la creación del contrato.
Postcondición	Se crea un nuevo contrato específico conforme a los requerimientos del usuario.
Excepciones	Fallos de validación de datos.
Importancia	Alta

Tabla B.11: CU-5.2 Crear contrato específico.

CU-6	Firmar contrato
Versión	1.0
Autor	David Martínez Bahillo
Requisitos asociados	RF-4.1, RF-4.2
Descripción	Permite al usuario firmar un contrato utilizando métodos biométricos o un código QR.
Precondición	<ol style="list-style-type: none"> 1. El usuario debe estar registrado y logueado. 2. El usuario ha seleccionado un contrato para firmar.
Acciones	<ol style="list-style-type: none"> 1. El usuario elige el método de firma: biométrico o código QR. 2. El sistema procesa la firma según el método seleccionado.
Postcondición	El contrato queda firmado por el usuario.
Excepciones	Fallos en la autenticación de la firma.
Importancia	Alta

Tabla B.12: CU-6 Firmar contrato.

CU-6.1	Firmar con biométrica
Versión	1.0
Autor	David Martínez Bahillo
Requisitos asociados	RF-4.1
Descripción	Permite al usuario firmar un contrato utilizando su identificación biométrica.
Precondición	<ol style="list-style-type: none"> 1. El usuario debe estar registrado y logueado. 2. El contrato debe estar listo para ser firmado.
Acciones	<ol style="list-style-type: none"> 1. El usuario selecciona la opción de firma biométrica. 2. El sistema verifica la identidad del usuario mediante el método biométrico elegido.
Postcondición	El contrato queda firmado mediante un método biométrico.
Excepciones	Error en la lectura o verificación biométrica.
Importancia	Alta

Tabla B.13: CU-6.1 Firmar con biométrica.

CU-6.2	Firmar con código QR
Versión	1.0
Autor	David Martínez Bahillo
Requisitos asociados	RF-4.2
Descripción	Permite al usuario firmar un contrato mediante la generación y escaneo de un código QR.
Precondición	<ol style="list-style-type: none"> 1. El usuario debe estar registrado y logueado. 2. El contrato debe estar listo para ser firmado.
Acciones	<ol style="list-style-type: none"> 1. El usuario selecciona la opción de firma con código QR. 2. El sistema genera un código QR que el usuario debe escanear para confirmar la firma.
Postcondición	El contrato queda firmado mediante la validación del código QR.
Excepciones	Fallos en la generación o escaneo del código QR.
Importancia	Alta

Tabla B.14: CU-6.2 Firmar con código QR.

CU-7	Buscar contratos
Versión	1.0
Autor	David Martínez Bahillo
Requisitos asociados	RF-5.1
Descripción	Permite al usuario buscar contratos que se ofertan en la aplicación.
Precondición	El usuario debe estar registrado y logueado.
Acciones	<ol style="list-style-type: none">1. El usuario accede a «Buscar contratos».2. El sistema muestra los contratos que coinciden con los criterios especificados.
Postcondición	El usuario visualiza la lista de contratos disponibles.
Excepciones	Ninguna
Importancia	Media

Tabla B.15: CU-7 Buscar contratos.

CU-8	Visualizar contratos
Versión	1.0
Autor	David Martínez Bahillo
Requisitos asociados	RF-6
Descripción	Permite al usuario visualizar los contratos en los que el usuario esté involucrado.
Precondición	<ol style="list-style-type: none">1. El usuario debe estar registrado y logueado.2. El usuario a accedido a la pantalla «Mis Contratos».
Acciones	<ol style="list-style-type: none">1. El usuario accede a la pantalla «Mis contratos».2. El sistema muestra todos los contratos en los que el usuario esté involucrado.
Postcondición	El usuario visualiza los contratos en los que esté involucrado.
Excepciones	Ninguna
Importancia	Alta

Tabla B.16: CU-8 Visualizar contratos.

CU-8.1	Visualizar contratos como empleador
Versión	1.0
Autor	David Martínez Bahillo
Requisitos asociados	RF-6.1
Descripción	Permite al usuario visualizar todos los contratos como empleador en los que se encuentre involucrado.
Precondición	<ol style="list-style-type: none"> 1. El usuario debe estar registrado y logueado. 2. El usuario a accedido a la pantalla «Mis Contratos». 3. El usuario a accedido a la sección «Mostrar contratos como empleador».
Acciones	<ol style="list-style-type: none"> 1. El usuario selecciona la opción de visualizar contratos como empleador. 2. El sistema muestra la lista de contratos activos.
Postcondición	El usuario visualiza todos los contratos como empleador en la plataforma.
Excepciones	Ninguna
Importancia	Alta

Tabla B.17: CU-8.1 Visualizar contratos como empleador.

CU-8.1	Visualizar contratos como trabajador
Versión	1.0
Autor	David Martínez Bahillo
Requisitos asociados	RF-6.2
Descripción	Permite al usuario visualizar todos los contratos como empleador en los que se encuentre involucrado.
Precondición	<ol style="list-style-type: none"> 1. El usuario debe estar registrado y logueado. 2. El usuario a accedido a la pantalla «Mis Contratos». 3. El usuario a accedido a la sección «Mostrar contratos como trabajador».
Acciones	<ol style="list-style-type: none"> 1. El usuario selecciona la opción de visualizar contratos como trabajador. 2. El sistema muestra la lista de contratos activos.
Postcondición	El usuario visualiza todos los contratos como trabajador en la plataforma.
Excepciones	Ninguna
Importancia	Alta

Tabla B.18: CU-8.2 Visualizar contratos como trabajador.

CU-9	Administrar contratos
Versión	1.0
Autor	David Martínez Bahillo
Requisitos asociados	RF-8
Descripción	Permite al usuario gestionar activamente sus contratos, incluyendo cancelación, finalización, liberación de pagos y gestión de managers.
Precondición	El usuario debe estar registrado y logueado.
Acciones	El usuario se encuentra en la pantalla de detalles del contrato.
Postcondición	El usuario gestiona sus contratos de acuerdo a sus necesidades.
Excepciones	Ninguna
Importancia	Alta

Tabla B.19: CU-9 Administrar contratos.

CU-9.1	Cancelar contrato
Versión	1.0
Autor	David Martínez Bahillo
Requisitos asociados	RF-8.1
Descripción	Permite al usuario cancelar un contrato antes de que se firme.
Precondición	<ol style="list-style-type: none"> 1. El usuario debe estar registrado y logueado. 2. El contrato debe estar en estado de pendiente de firma.
Acciones	<ol style="list-style-type: none"> 1. El usuario selecciona la opción «Cancelar contrato». 2. El sistema confirma la cancelación del contrato.
Postcondición	El contrato queda cancelado y no es vinculante para ninguna de las partes.
Excepciones	Ninguna
Importancia	Media

Tabla B.20: CU-9.1 Cancelar contrato.

CU-9.2	Finalizar contrato
Versión	1.0
Autor	David Martínez Bahillo
Requisitos asociados	RF-8.2
Descripción	Permite al usuario finalizar un contrato de manera formal una vez que se hayan cumplido los términos acordados.
Precondición	<ol style="list-style-type: none"> 1. El usuario debe estar registrado y logueado. 2. Los términos del contrato deben estar cumplidos.
Acciones	<ol style="list-style-type: none"> 1. El usuario selecciona la opción «Finalizar contrato». 2. El sistema confirma la finalización del contrato.
Postcondición	El contrato queda formalmente finalizado.
Excepciones	Ninguna
Importancia	Alta

Tabla B.21: CU-9.2 Finalizar contrato.

CU-9.3	Liberar contrato
Versión	1.0
Autor	David Martínez Bahillo
Requisitos asociados	RF-8.3
Descripción	Permite al empleador liberar y transferir los fondos acordados una vez cumplidos los términos del contrato.
Precondición	<ol style="list-style-type: none"> 1. El usuario debe estar registrado y logueado. 2. El contrato debe estar finalizado.
Acciones	<ol style="list-style-type: none"> 1. El usuario selecciona la opción «Liberar salario». 2. El sistema procesa la liberación y transferencia de los fondos al trabajador.
Postcondición	Los fondos son transferidos según los términos del contrato.
Excepciones	Fallos en la transferencia de fondos.
Importancia	Alta

Tabla B.22: CU-9.3 Liberar contrato.

CU-9.4	Añadir manager
Versión	1.0
Autor	David Martínez Bahillo
Requisitos asociados	RF-8.4
Descripción	Permite al usuario agregar un nuevo manager para la gestión del contrato.
Precondición	<ol style="list-style-type: none"> 1. El usuario debe estar registrado y logueado. 2. El contrato debe estar activo. 3. El manager no puede ser el propio trabajador.
Acciones	<ol style="list-style-type: none"> 1. El usuario selecciona la opción «Añadir manager». 2. El usuario ingresa los datos del nuevo manager. 3. El sistema confirma la adición del nuevo manager.
Postcondición	El nuevo manager es añadido al contrato.
Excepciones	Fallos en la validación de los datos del nuevo manager.
Importancia	Media

Tabla B.23: CU-9.4 Añadir manager.

CU-9.5	Eliminar manager
Versión	1.0
Autor	David Martínez Bahillo
Requisitos asociados	RF-8.5
Descripción	Permite al usuario eliminar un manager actualmente asignado al contrato.
Precondición	<ol style="list-style-type: none">1. El usuario debe estar registrado y logueado.2. El contrato debe estar activo.3. El manager debe de estar asignado previamente a dicho contrato.
Acciones	<ol style="list-style-type: none">1. El usuario selecciona elimina al manager.2. El sistema confirma la eliminación del manager.
Postcondición	El manager es eliminado del contrato.
Excepciones	Ninguna
Importancia	Media

Tabla B.24: CU-9.5 Eliminar manager.

CU-10	Modificar contratos
Versión	1.0
Autor	David Martínez Bahillo
Requisitos asociados	RF-9
Descripción	Permite al usuario proponer, aceptar o denegar modificaciones a los contratos existentes.
Precondición	<ol style="list-style-type: none"> 1. El usuario debe estar registrado y logueado. 2. El contrato debe estar activo.
Acciones	El usuario selecciona la opción «Modificar contratos».
Postcondición	El usuario gestiona las modificaciones de sus contratos.
Excepciones	Ninguna
Importancia	Alta

Tabla B.25: CU-10 Modificar contratos.

CU-10.1	Proponer modificación
Versión	1.0
Autor	David Martínez Bahillo
Requisitos asociados	RF-9.1
Descripción	Permite al usuario proponer modificaciones a los contratos existentes.
Precondición	<ol style="list-style-type: none">1. El usuario debe estar registrado y logueado.2. El contrato debe estar activo.
Acciones	<ol style="list-style-type: none">1. El usuario selecciona la opción «Modificar contrato».2. El usuario introduce los cambios propuestos.3. El sistema guarda y notifica la propuesta de modificación a la otra parte.
Postcondición	La propuesta de modificación queda registrada y pendiente de aceptación o rechazo.
Excepciones	Fallos en la validación de los datos introducidos.
Importancia	Alta

Tabla B.26: CU-10.1 Proponer modificación.

CU-10.2	Aceptar modificación
Versión	1.0
Autor	David Martínez Bahillo
Requisitos asociados	RF-9.2
Descripción	Permite al usuario aceptar las modificaciones propuestas a un contrato.
Precondición	<ol style="list-style-type: none"> 1. El usuario debe estar registrado y logueado. 2. El contrato debe estar activo. 3. Existe una propuesta de modificación pendiente.
Acciones	<ol style="list-style-type: none"> 1. El usuario selecciona la opción «Aceptar modificación». 2. El sistema actualiza el contrato con las modificaciones aceptadas.
Postcondición	El contrato se actualiza con las modificaciones aceptadas.
Excepciones	Fallos en la actualización del contrato.
Importancia	Alta

Tabla B.27: CU-10.2 Aceptar modificación.

CU-10.3	Denegar modificación
Versión	1.0
Autor	David Martínez Bahillo
Requisitos asociados	RF-9.3
Descripción	Permite al usuario denegar las modificaciones propuestas a un contrato.
Precondición	<ol style="list-style-type: none"> 1. El usuario debe estar registrado y logueado. 2. El contrato debe estar activo. 3. Existe una propuesta de modificación pendiente.
Acciones	<ol style="list-style-type: none"> 1. El usuario selecciona la opción «Denegar modificación». 2. El sistema registra la denegación y notifica a la otra parte.
Postcondición	La propuesta de modificación queda denegada y el contrato permanece sin cambios.
Excepciones	Fallos en la comunicación de la denegación.
Importancia	Alta

Tabla B.28: CU-10.3 Denegar modificación.

CU-11	Visualizar alertas
Versión	1.0
Autor	David Martínez Bahillo
Requisitos asociados	RF-10
Descripción	Permite al usuario visualizar las alertas y notificaciones relacionadas con sus contratos y actividades en la plataforma.
Precondición	El usuario debe estar registrado y logueado.
Acciones	<ol style="list-style-type: none"> 1. El usuario accede a la sección de «Alertas». 2. El sistema muestra las alertas y notificaciones relevantes.
Postcondición	El usuario visualiza las alertas y notificaciones.
Excepciones	Ninguna
Importancia	Media

Tabla B.29: CU-11 Visualizar alertas.

CU-12	Utilizar herramientas
Versión	1.0
Autor	David Martínez Bahillo
Requisitos asociados	RF-11
Descripción	Permite al usuario acceder y utilizar diversas herramientas adicionales proporcionadas por la plataforma.
Precondición	El usuario debe estar registrado y logueado.
Acciones	El usuario selecciona la opción «Estadísticas».
Postcondición	El usuario accede a las herramientas disponibles en la plataforma.
Excepciones	Ninguna
Importancia	Baja

Tabla B.30: CU-12 Utilizar herramientas.

CU-12.1	Consultar información Ethereum
Versión	1.0
Autor	David Martínez Bahillo
Requisitos asociados	RF-11.1
Descripción	Permite al usuario consultar la información actual y el histórico del precio de Ethereum.
Precondición	<ol style="list-style-type: none">1. El usuario debe estar registrado y logueado.2. El usuario ha seleccionado «Estadísticas».
Acciones	<ol style="list-style-type: none">1. El sistema muestra el precio actual y el histórico del Ethereum.
Postcondición	El usuario visualiza la información actual y el histórico del precio de Ethereum.
Excepciones	Fallos en la actualización de datos.
Importancia	Baja

Tabla B.31: CU-12.1 Consultar información Ethereum.

CU-12.2	Usar calculadora de divisas
Versión	1.0
Autor	David Martínez Bahillo
Requisitos asociados	RF-11.2
Descripción	Permite al usuario utilizar una calculadora para convertir entre Ethereum y euros, y viceversa.
Precondición	<ol style="list-style-type: none"> 1. El usuario debe estar registrado y logueado. 2. El usuario ha seleccionado «Estadísticas».
Acciones	<ol style="list-style-type: none"> 1. El usuario introduce la cantidad a convertir y selecciona la divisa de destino. 2. El sistema muestra el resultado de la conversión.
Postcondición	El usuario obtiene la conversión de la cantidad introducida entre Ethereum y euros.
Excepciones	Fallos en la actualización de los tipos de cambio.
Importancia	Baja

Tabla B.32: CU-12.2 Usar calculadora de divisas.

Apéndice C

Especificación de diseño

C.1. Introducción

En este apéndice se explicarán los distintos diseños de la aplicación, desde la organización de los datos hasta la arquitectura del software.

C.2. Diseño de datos

A continuación se describirá cómo se ha desarrollado el diseño de datos de la aplicación. Para implementar la persistencia, a parte del uso de la blockchain, se ha decidido utilizar una base de datos de Firebase, para almacenar datos sobre los usuarios y gestionar su autenticación.

Base de datos

El uso de Firestore en la aplicación se dirige a los datos que no requieren las propiedades de inmutabilidad de la blockchain. Este enfoque estratégico permite desacoplar la gestión de información que es dinámica y mutable, como los datos de usuario y las sesiones de autenticación, de aquella que requiere una integridad absoluta, como los contratos laborales y sus transacciones asociadas. Al delegar la gestión de datos no críticos a Firestore, se minimiza la carga sobre la blockchain, lo que resulta en una reducción significativa en el consumo de gas durante las transacciones. Esta optimización no solo mejora la eficiencia de los contratos inteligentes sino que también reduce los costos operativos para los usuarios, haciendo que la aplicación sea más accesible y económicamente viable. La elección de Firebase como plataforma de gestión de bases de datos NoSQL debe

a su escalabilidad automática y rendimiento en tiempo real. Este diseño híbrido de manejo de datos aprovecha las fortalezas de ambas tecnologías: la flexibilidad y la escalabilidad de Firestore para datos operativos y la seguridad y la inmutabilidad de la blockchain para transacciones críticas. El diagrama Entidad-Relación se puede observar en [C.2](#).

La colección **users** incluye los siguientes atributos:

- **nombre** (**string**): Nombre de pila del usuario.
- **apellido1** (**string**): Primer apellido del usuario.
- **apellido2** (**string**): Segundo apellido del usuario.
- **pais** (**string**): Pais de residencia del usuario
- **paisCode** (**string**): Código del país de residencia en formato iso.
- **ciudad** (**string**): Ciudad de residencia del usuario.
- **direccion** (**string**): Dirección física donde vive el usuario.
- **dni** (**string**): Documento Nacional de Identidad del usuario.
- **fecha** (**timeStamp**): Fecha de nacimiento del usuario. Formato: día de mes de año, hora:minuto:segundo AM/PM UTC+offset.
- **telefono** (**number**): Número de teléfono del usuario.
- **ganache** (**string**): Dirección de la billetera del usuario en la red blockchain.

Como clave primaria del documento se usa el ID del usuario, generado automáticamente con la creación del mismo, que proporciona un identificador único para cada usuario.

Por otro lado, dentro de la colección **users** se ha definido una regla que asegura que los usuarios solo puedan acceder y modificar sus propios datos, protegiendo la privacidad y manteniendo la integridad de los datos personales.

```
1 rules_version = '2';
2 service cloud.firestore {
3   match /databases/{database}/documents {
4     match /users/{userId} {
5       allow read, write: if request.auth != null && request.auth.uid == userId;
6     }
7   }
8 }
```

Figura C.1: Regla definida en FireBase para el control de acceso a los datos.

Blockchain

Dentro de los contratos inteligentes los datos se estructuran utilizando estructuras y mapeos que facilitan la gestión y el acceso eficiente de la información.

Los atributos del contrato recogidos en **ContractDetails** incluyen:

- **salary** (uint256): Salario acordado para el contrato, manejado en wei.
- **startDate** (uint256): Fecha de inicio del contrato representada en segundos.
- **duration** (uint256): Duración total del contrato en segundos.
- **title** (string): Título del contrato.
- **description** (string): Descripción del contrato.
- **isSigned** (bool): Estado que indica si el contrato ha sido firmado.
- **worker** (address): Dirección Ethereum del trabajador asignado al contrato.
- **isFinished** (bool): Estado que indica si el contrato ha concluido.
- **isReleased** (bool): Estado que señala si el pago ha sido liberado.
- **isPaused** (bool): Indica si el contrato está actualmente pausado.
- **pauseTime** (uint256): Tiempo en segundos que indica cuando el contrato fue pausado.

- **pauseDuration** (uint256): Acumulación de tiempo de pausas durante la ejecución del contrato en segundos.

La estructura **ChangeProposal** se utiliza para gestionar propuestas de cambio en el contrato y contiene los siguientes campos:

- **newTitle** (string): Nuevo título propuesto para el contrato.
- **newSalary** (uint256): Nuevo salario propuesto, expresado en wei.
- **newDuration** (uint256): Nueva duración propuesta para el contrato, expresada en segundos.
- **newDescription** (string): Nueva descripción detallada propuesta para el contrato.
- **isPaused** (bool): Indica si la propuesta incluye una pausa del contrato.
- **isPending** (bool): Estado que indica si la propuesta está pendiente de aprobación.

En los contratos inteligentes, se usa mapeos y arrays para organizar y relacionar de manera eficiente a los usuarios con los contratos.

- **mapping(uint256 => ContractDetails)**: almacena los datos de cada contrato mediante el **tokenId**, que es un identificador único.
- **mapping(uint256 => ChangeProposal)**: almacena las propuestas de cambios en contratos mediante el **tokenId**, que es un identificador único.
- **mapping(address => uint256[]) public contractsOwner**: relaciona cada empleador con una lista de **tokenIds** que representa los contratos que ha emitido.
- **mapping(address => uint256[]) public activeContractsOfWorker**: Mantiene un registro de los contratos activos asignados a cada trabajador.
- **mapping(address => uint256[]) public unsignedContractsOfWorker**: Lista los contratos que han sido ofrecidos a un trabajador pero que aún no han sido firmados.

- **mapping(uint256 => address[]) public tokenManagersList:** Almacena una lista de direcciones que tienen permisos de gestión para un contrato.
- **mapping(uint256 => mapping(address => bool)) public tokenManagers:** Determina qué direcciones tienen permisos de gestión sobre un contrato específico, facilitando la administración.

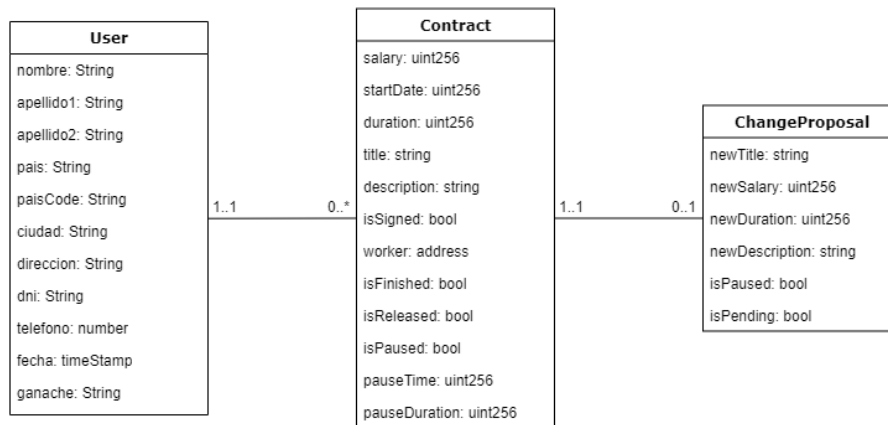


Figura C.2: Diagrama entidad-relación

C.3. Diseño procedimental

En esta sección se mostrarán los diagramas de secuencia de las dos tareas principales de la aplicación; crear un contrato (ver figura C.3 y firmar un contrato (ver figura C.4.

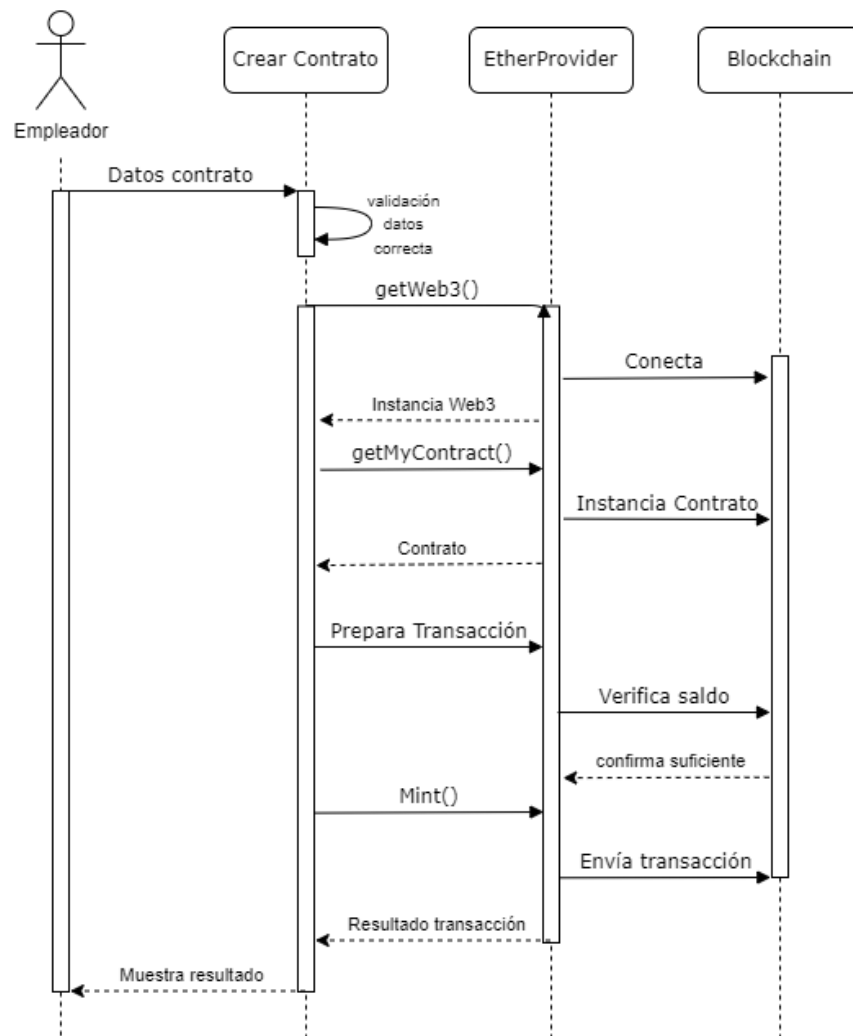


Figura C.3: Diagrama de secuencia para crear un contrato

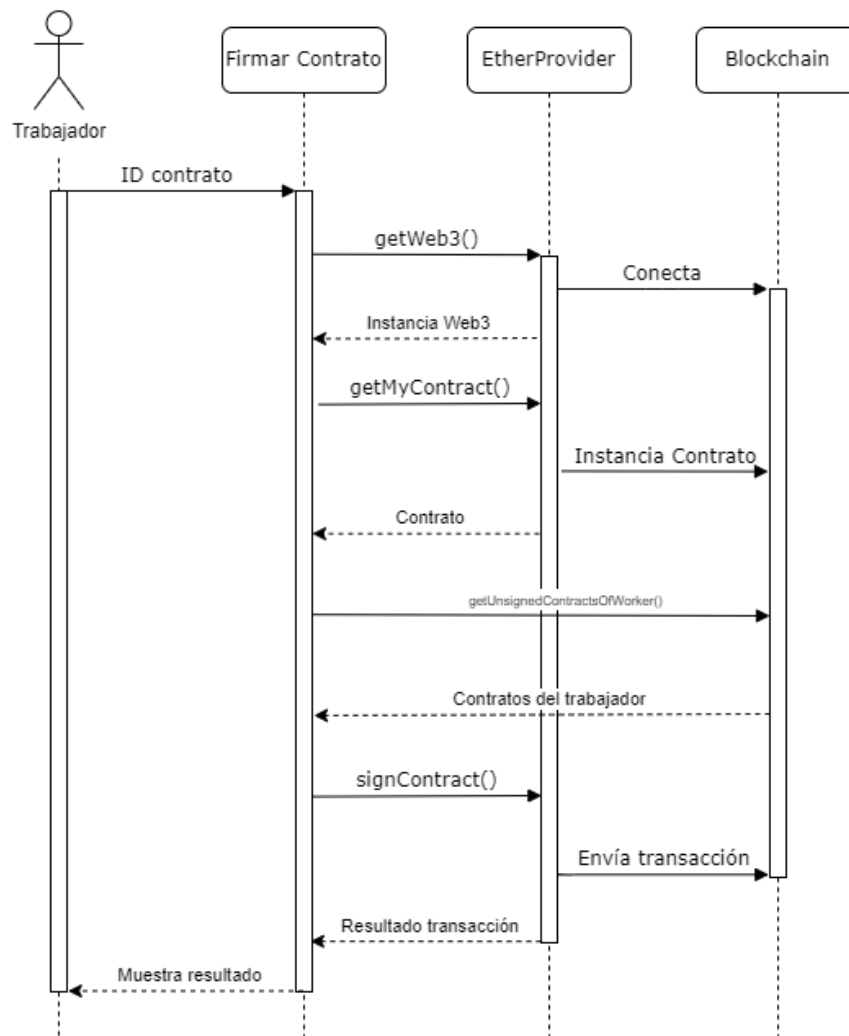


Figura C.4: Diagrama de secuencia para firmar un contrato

C.4. Diseño arquitectónico

C.5. Diseño de interfaces

Antes del desarrollo de la interfaz de la aplicación, se experimentó y depuró un prototipo de la misma. Las interfaces fueron diseñadas teniendo en cuenta la usabilidad de la aplicación, priorizando un diseño simple que permitiera que la aplicación fuera intuitiva, con una curva de aprendizaje suave. También se ha considerado que la aplicación sea adaptable a una amplia variedad de pantallas, ya que puede ser utilizada tanto en dispositivos móviles como en tabletas.

El diseño de los prototipos para la aplicación incluye varias pantallas clave: pantalla de inicio de sesión, permitiendo la autenticación de los usuarios (ver imagen C.5); la pantalla principal, que actúa como punto de entrada intuitivo (ver imagen C.5); la pantalla de creación de contratos, donde los usuarios pueden interactuar de manera clara y directa para establecer acuerdos (ver imagen C.5); la pantalla de visualización de contratos, diseñada para proporcionar una experiencia fluida al revisar contratos existentes (ver imagen C.5); la pantalla de información y utilidades, que ofrece recursos y herramientas fácilmente accesibles (ver imagen C.5); y la pantalla de estado de cuenta, que muestra de manera clara y concisa la información financiera relevante del usuario (ver imagen C.5).



ContracMe

Inicia sesión en tu cuenta

CORREO

CLAVE

¿Has olvidado tu contraseña?

Acceder

No tengo cuenta

Figura C.5: Prototipo pantalla inicio de sesión.

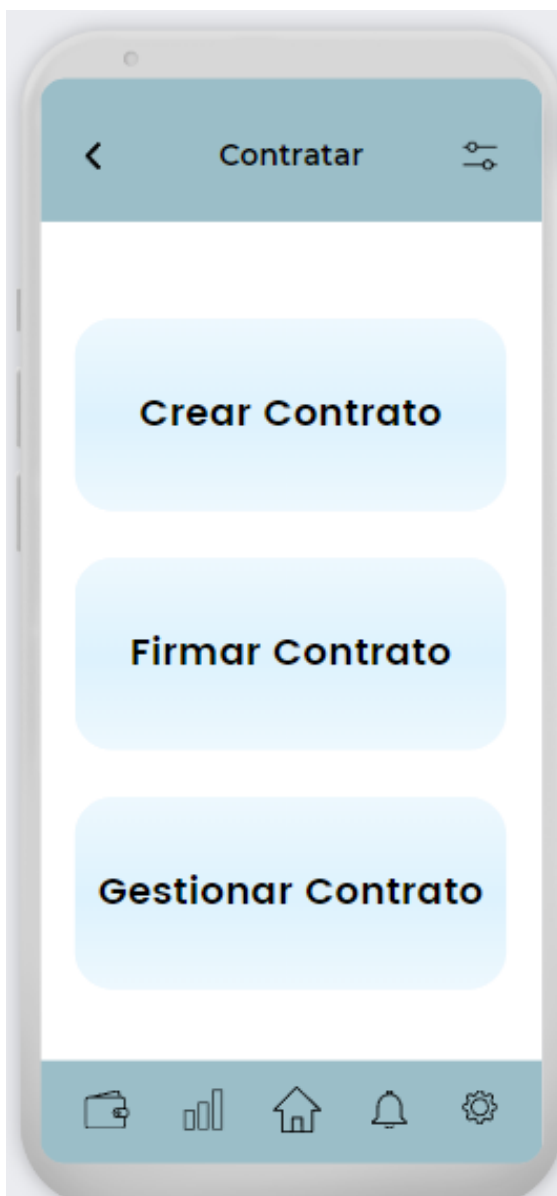


Figura C.6: Prototipo pantalla principal.

The image shows a mobile application prototype for creating a contract. The screen has a light blue header with a back arrow on the left, the title "Crear un contrato" in the center, and a settings icon on the right. Below the header, there are four input fields: "TITULO" (a single-line text box), "DESCRIPCIÓN" (a multi-line text box), "SALARIO" (a single-line text box), and "DURACIÓN" (a date picker with the text "seleccionar" and a calendar icon). At the bottom of the form is a dark blue button labeled "Crear Contrato". The entire screen is framed by a grey border representing a smartphone. At the very bottom, there is a light blue navigation bar with five icons: a folder, a bar chart, a house, a bell, and a gear.

Figura C.7: Prototipo pantalla crear contrato.

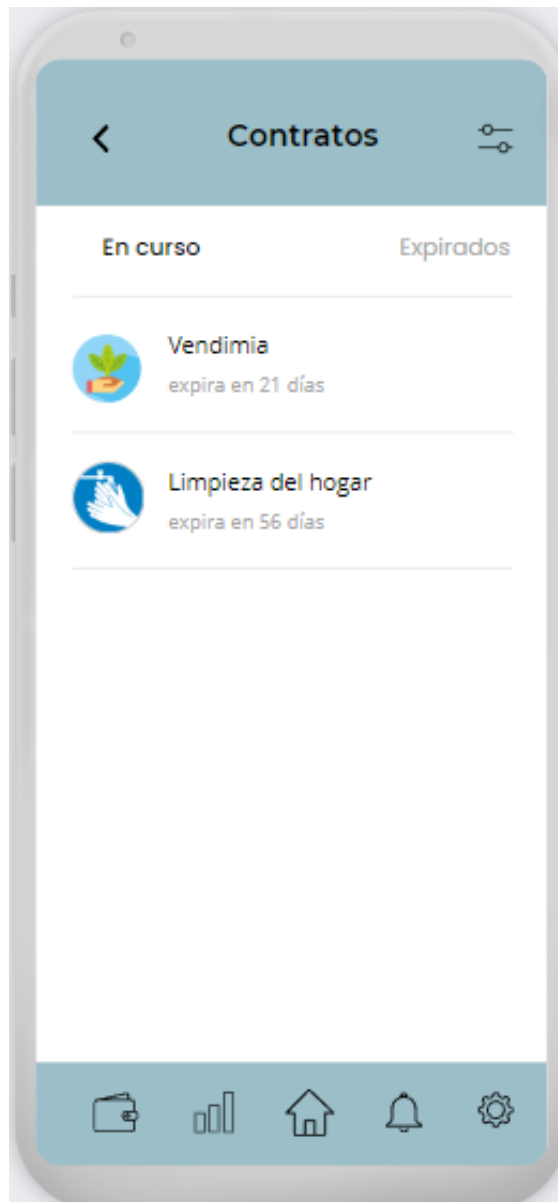


Figura C.8: Prototipo pantalla ver contratos del usuario.



Figura C.9: Prototipo pantalla información y utilidades.

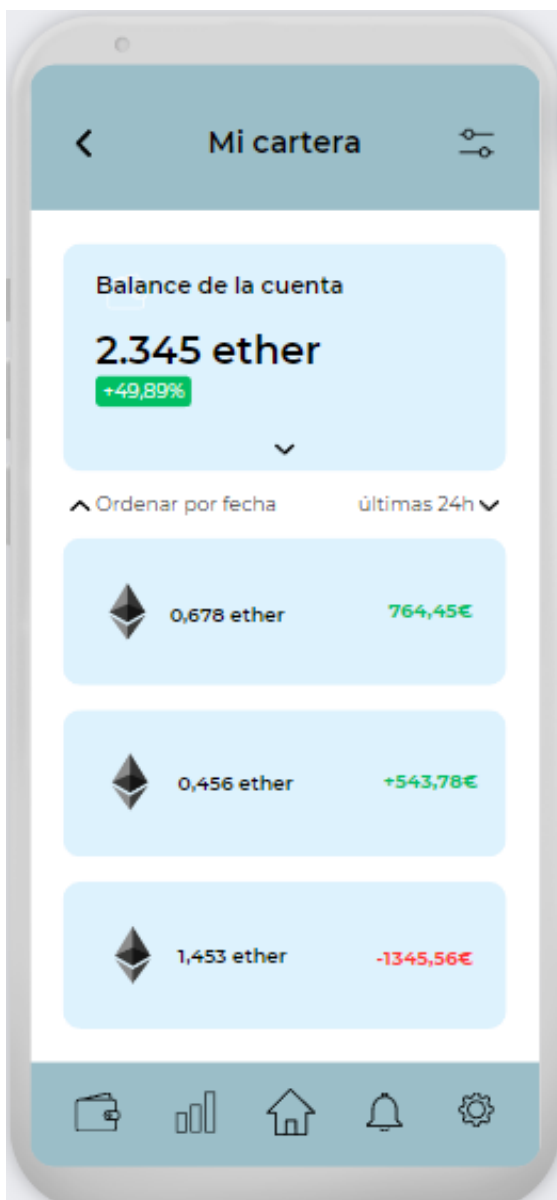


Figura C.10: Prototipo pantalla estado cuenta del usuario.

Apéndice D

Documentación técnica de programación

D.1. Introducción

En esta sección se describe la estructura del proyecto, el proceso de instalación y las herramientas necesarias para desarrollar el trabajo. También se explica cómo realizar la instalación de dependencias, la compilación, la ejecución del proyecto y el despliegue en Expo.

D.2. Estructura de directorios

En el primer nivel, se encuentran dos directorios, **Documentacion** y **Code**. El primero de ellos recoge los archivos de documentación sobre el proyecto, como la memoria y sus anexos, por otra parte, el segundo recoge la lógica de la aplicación.

Dentro de Code, el directorio **AppContractMe** contiene todo el código fuente necesario para la interfaz de usuario de la aplicación. Se divide en varios subdirectorios que organizan los recursos y scripts de manera lógica, siendo **scr** el más importante, incluyendo:

- **AppLogin:** Contiene los componentes para la autenticación y gestión de sesiones de usuario.
- **ContractConexion:** Incluye las funciones que facilitan la conexión y la interacción con los contratos inteligentes.

- **Screens:** Agrupa las diferentes pantallas de la aplicación, permitiendo la navegación al usuario.
- **components:** Reúne elementos reutilizables que se emplean en diversas partes de la aplicación, como botones y ciertas funcionalidades.

Por otro lado, también existe el directorio **SmartContract** que recoge toda la lógica del contrato inteligente e incluye:

- **build:** Contiene los archivos compilados de los contratos, que son necesarios para su despliegue y ejecución en la blockchain.
- **contracts:** Alberga los scripts de los contratos inteligentes.
- **migrations:** Gestiona el scripts que ayuda en la migración y despliegue de los contratos en la blockchain.
- **node_modules:** Directorio que incluye las dependencias de Node.js utilizadas en el proyecto.
- **test:** Contiene los tests escritos en java para asegurar el correcto funcionamiento de los contratos inteligentes.

Finalmente hay otro directorio llamada **ganache_db** el cual se utiliza para almacenar la configuración y los datos de la base de datos de Ganache.

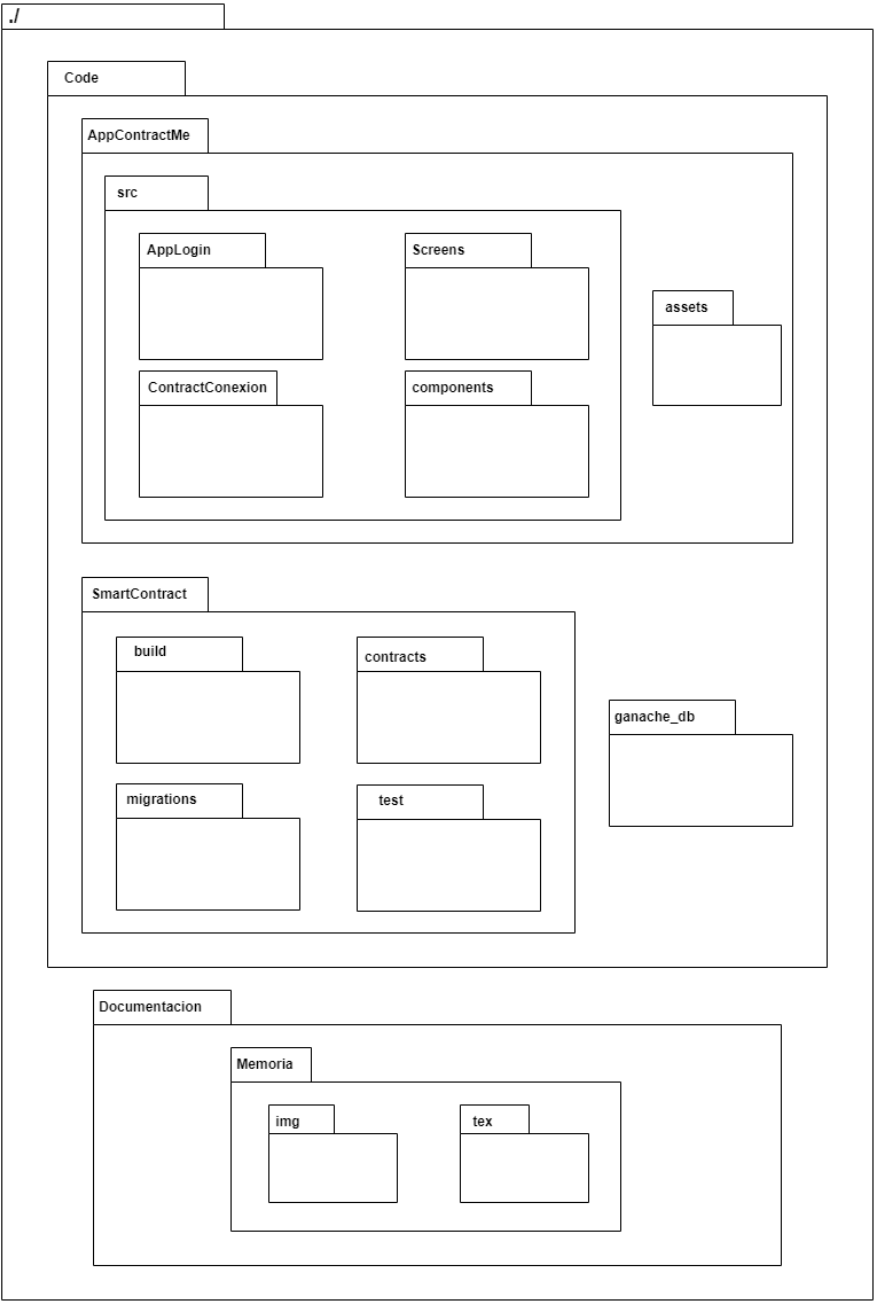


Figura D.1: Directorios del proyecto

D.3. Manual del programador

Este apartado servirá de ayuda y referencia a futuros desarrolladores que quieran replicar el proyecto. Por ello se detallará los requisitos necesarios y el proceso de instalación para el desarrollo de la aplicación.

Entorno desarrollo

Antes de explicar la instalación de los programas y herramientas para el desarrollo de la aplicación es necesario detallar unas especificaciones mínimas en cuanto al equipo de desarrollo para poder trabajar con el proyecto.

1. Procesador (CPU):

- **Mínimo:** Procesador con arquitectura x86_64, 2ª generación de Intel o procesador AMD que soporte "Hypervisor Framework", Permitiendo una compilación rápida de código y emular un dispositivo móvil para pruebas.
- **Recomendado:** Intel Core i7 o AMD ryzen 7, con 6-8 núcleos.

2. Memoria RAM:

- **Mínimo:** 8 GB de RAM como mínimo, para manejar diversas tareas y la ejecución de simuladores.
- **Recomendado:** 16 GB de RAM o más, pudiendo mantener múltiples dispositivos emulados simultáneamente.

3. Sistema operativo:

- **Mínimo:** Windows 10 64-bit.
- **Recomendado:** Windows 11.

Por otro lado, aunque no es un requisito obligatorio, se recomienda contar con un dispositivo Android físico para ejecutar la aplicación y hacer uso de funcionalidades nativas como el reconocimiento de huella, siendo necesario las siguientes especificaciones.

1. Versión Android:

- **Mínimo:** Android 6.0 (Marshmallow)
- **Recomendado:** Android 9.0 (Pie) o superior.

2. Procesador (CPU):

- **Mínimo:** Quad-core 1.2 GHz.
- **Recomendado:** Octa-core 2.0 GHz o superior.

3. RAM:

- **Mínimo:** 2 GB de RAM.
- **Recomendado:** 4-8 GB de RAM.

4. Compatibilidad con características nativas:

- **Huella digital:** El dispositivo debe contar con un sensor de huellas dactilares compatible con las API de Android para autenticación biométrica.
- **Reconocimiento facial:** cámara interior con capacidad de reconocimiento facial.
- **Cámara trasera:** Cámara de al menos 8 MP para la lectura de códigos QR.

D.4. Compilación, instalación y ejecución del proyecto

Instalación

Seguidamente se detallará como configurar un entorno de desarrollo para poder trabajar en la aplicación `ContractMe`.

1. **Node.js:** Node.js es una plataforma de ejecución para JavaScript del lado del servidor y es esencial para la gestión de paquetes y la ejecución de varias herramientas de desarrollo. Para instalar Node.js es necesario dirigirse a la [pagina oficial de Node.js](#) y seleccionar el instalador para Windows. Ver imagen [D.2](#).

Al ejecutar el instalador que se ha descargado, es necesario seleccionar la opción que permite instalar npm y añadir Node.js a tu path. Ver imagen [D.3](#). En concreto la versión utilizada de Node.js para el desarrollo del proyecto ha sido la v18.18.2 y la versión utilizada de npm ha sido la 10.2.2.

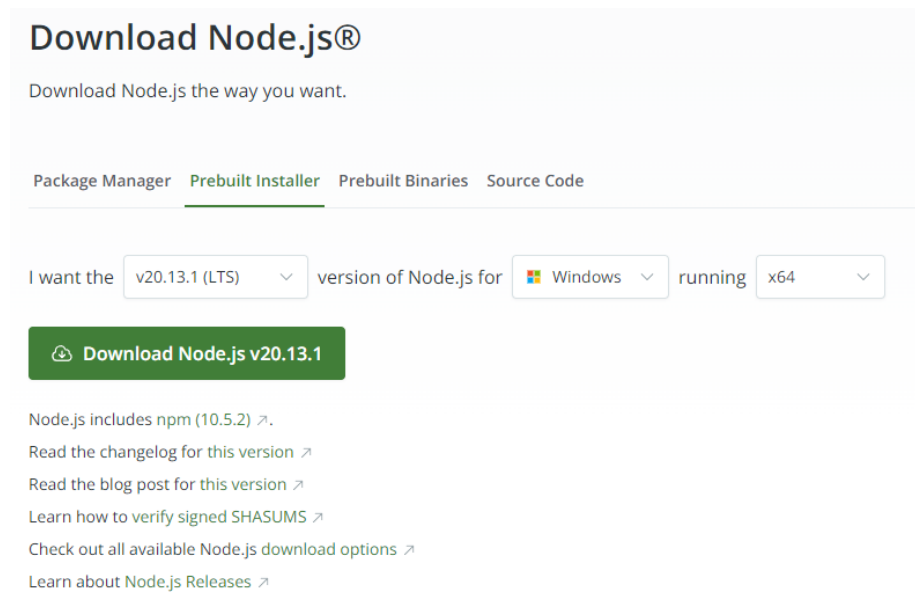


Figura D.2: Descarga de Node.js

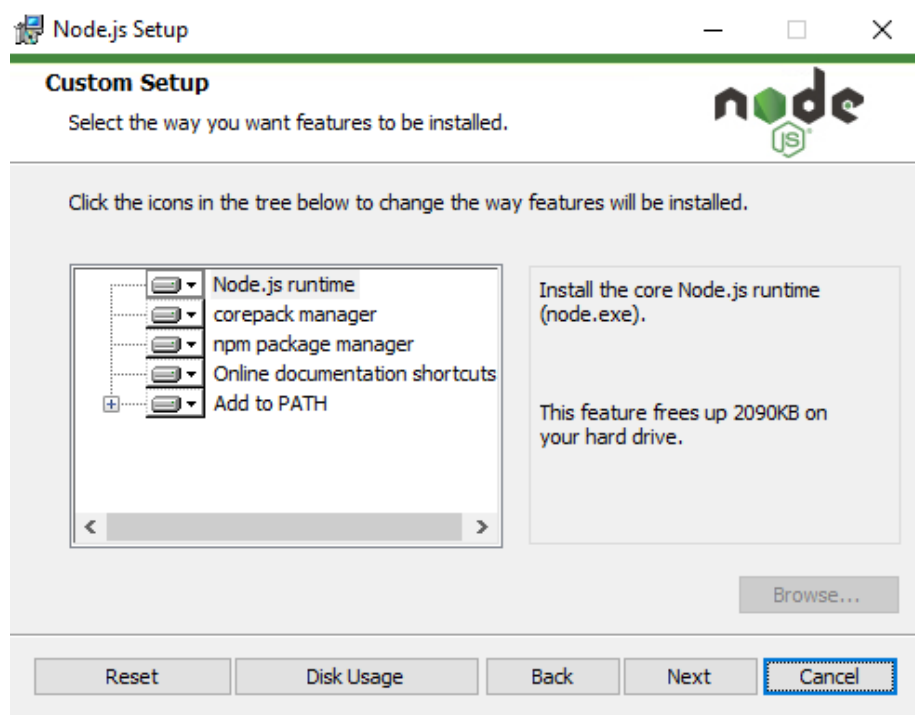
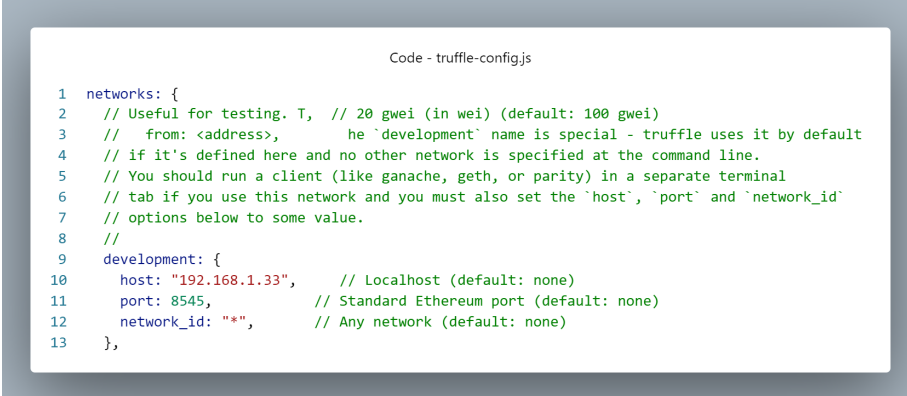


Figura D.3: Instalar Node.js y npm

2. **Truffle Suite:** Teniendo Node.js y npm instalados, procederemos a instalar Truffle globalmente, para ello desde la terminal ejecutaremos el siguiente comando: `npm install -g truffle` lo cual nos permitirá usar el comando `truffle` desde cualquier lugar en la terminal.

Una vez instalado Truffle, el siguiente paso es configurar el proyecto, para ello dentro del directorio *SmartContract*, se encuentra la estructura de carpetas generadas por Truffle, incluyendo *contracts* para almacenar la lógica de los contratos, *migrations* para almacenar el script de migración y *test* para las pruebas.

Truffle también crea un archivo de configuración llamado *truffle-config.js*, el cual se usa para configurar las redes sobre las cuales se desplegarán los contratos inteligentes. En este caso se usa la red de Ganache, que normalmente se ejecuta en *localhost* en el ordenador. Debido a que la aplicación se ejecutará en un dispositivo móvil independiente del ordenador, la configuración *localhost* no será suficiente. En su lugar, se usará la dirección IP del ordenador la cual actuará como servidor para Ganache. Esto permitirá que otros dispositivos de nuestra misma red local se conecten a la blockchain simulada por Ganache. Ganache por defecto usa el puerto 8545, aunque este podría ser reemplazado por cualquier puerto que se encontrase vacío. Siguiendo los pasos previamente mencionados, el archivo *truffle-config.js* debería verse como en la imagen [D.4](#).

The image shows a code editor window titled "Code - truffle-config.js". It contains a JavaScript configuration object for Truffle. The "networks" object has a "development" network configured with a specific IP address, port 8545, and network ID "*".

```
1  networks: {
2    // Useful for testing. T, // 20 gwei (in wei) (default: 100 gwei)
3    //   from: <address>,   he 'development' name is special - truffle uses it by default
4    // if it's defined here and no other network is specified at the command line.
5    // You should run a client (like ganache, geth, or parity) in a separate terminal
6    // tab if you use this network and you must also set the 'host', 'port' and 'network_id'
7    // options below to some value.
8    //
9    development: {
10     host: "192.168.1.33",    // Localhost (default: none)
11     port: 8545,             // Standard Ethereum port (default: none)
12     network_id: "*",       // Any network (default: none)
13   },
```

Figura D.4: Configuración red Ganache en truffle-config.js

1

Dentro del archivo *truffle-config.js* también será necesario realizar la configuración de compiladores para especificar como los contratos

inteligentes deben de ser compilados, optimizados y ejecutados en la máquina virtual de Ethereum. Se deberá especificar la versión **0.8.20**, la cual usará Truffle para compilar los contratos, esta versión coincide con la versión que se ha usado de Solidity para desarrollar los contratos inteligentes. Dentro de Settings se ha de especificar algunas opciones que afectan a la compilación de los contratos. La opción *optimizer* debe marcarse como *true*, ayudándonos a reducir el código de byte de los contratos y hacerlos más eficientes en términos de gas. Por otro lado, las runs se deben de establecer en 200, un número más alto puede resultar en un código más optimizado en términos de gas, pero con un proceso de compilación más lento. ver imagen [D.5](#).

The image shows a code editor window titled "Code - truffle-config.js". It contains a JavaScript configuration object for Truffle. The configuration includes a "compilers" section with a "solc" (Solidity Compiler) object. Inside "solc", there is a "version" set to "0.8.20" with a comment to fetch the exact version from solc-bin, a "docker" flag set to true with a comment to use the locally installed version, and a "settings" object. The "settings" object has an "optimizer" object with "enabled" set to true and "runs" set to 200. There is also an "evmVersion" set to "byzantium".

```
1 // Configure your compilers
2 compilers: {
3   solc: {
4     version: "0.8.20",    // Fetch exact version from solc-bin (default: truffle's version)
5     // docker: true,      // Use "0.5.1" you've installed locally with docker (default: false)
6     settings: {          // See the solidity docs for advice about optimization and evmVersion
7       optimizer: {
8         enabled: true,
9         runs: 200
10      },
11      // evmVersion: "byzantium"
12    }
13  }
14 },
```

Figura D.5: Configuración del compilador en truffle-config.js

1

3. **Ganache:** Como hemos comentado, en el proyecto se usa la red de *Ganache*. Por simplicidad hemos usado la versión de consola, por tanto para instalarla de forma global en nuestro ordenador será necesario ejecutar el siguiente comando: `npm install -g ganache-cli`.
4. **Expo CLI:** Del mismo modo que hemos realizado con *Ganache*, previamente es necesario descargar *Expo CLI*, el cual es necesario para iniciar, desarrollar y probar el proyecto. Utilizando el comando `npm install -g expo-cli` lo descargamos globalmente.
5. **Expo Go:** Para visualizar la aplicación se usará Expo Go permitiéndonos ejecutar la aplicación directamente en nuestro dispositivo móvil. Para ello, desde un dispositivo móvil Android utilizando *Google Play Store*, se buscará en la barra de búsqueda ".Expo Go". Seguidamente se procederá a descargar dicha aplicación.

6. **Descarga del repositorio:** Para adquirir el código fuente del proyecto “**ContracMe**”, se puede descargar de utilizando la interfaz gráfica de GitLab descargando el fichero ZIP o usando `git clone` desde la terminal.

Ejecución del proyecto

En este apartado se detallará como ejecutar el proyecto. Es muy importante que durante la ejecución tanto el ordenador como el dispositivo móvil se encuentren en la misma red.

1. **Levantar Ganache:** Antes de proceder a ejecutar la aplicación es necesario que la red Ganache se encuentre previamente en funcionamiento. Como hemos comentado anteriormente, la red debe de ser levantada sobre la dirección IP local de nuestro ordenador. Por ejemplo, suponiendo que la IP local de mi ordenador es 192.168.1.33, deberemos de ejecutar desde cualquier parte de la terminal, el siguiente comando:

```
ganache-cli --host 192.168.1.33 -d --db ganache_db
```

`-d` es una abreviatura de `deterministic`, lo que hace que Ganache genere las mismas cuentas y claves privadas cada vez que se ejecuta. `-db ganache_db` es una opción que permite que Ganache guarde el estado de la red en un directorio específico, en este caso el directorio será llamado `ganache_db` y se creará en la rama actual donde se encuentre el usuario en la terminal. Esto permite que la blockchain simulada mantenga su estado entre reinicios de *Ganache*.

```

E:\ubu-contractme\Code>ganache-cli --host 192.168.1.33 -d --db ganache_db
ganache v7.9.1 (@ganache/cli: 0.10.1, @ganache/core: 0.10.1)
Starting RPC server

Available Accounts
=====
(0) 0x90F8bf6A479f320ead074411a4B0e7944Ea8c9C1 (1000 ETH)
(1) 0xFFcf8FDEE72ac11b5c542428B35EEF5769C409f0 (1000 ETH)
(2) 0x22d491Bde2303f2f43325b2108D26f1eAbA1e32b (1000 ETH)
(3) 0xE11BA2b4D45Eaed5996Cd0823791E0C93114882d (1000 ETH)
(4) 0xd03ea8624C8C5987235048901fB614fDcA89b117 (1000 ETH)
(5) 0x95cED938F7991cd0dFcb48F0a06a40FA1aF46EBC (1000 ETH)
(6) 0x3E5e9111Ae8eB78Fe1CC3bb8915d5D461F3Ef9A9 (1000 ETH)
(7) 0x28a8746e75304c0780E011BEd21C72cD78cd535E (1000 ETH)
(8) 0xAca94ef8bD5ffEE41947b4585a84BdA5a3d3DA6E (1000 ETH)
(9) 0x1df62f291b2E969fB0849d99D9Ce41e2F137006e (1000 ETH)

Private Keys
=====
(0) 0x4f3edf983ac636a65a842ce7c78d9aa706d3b113bce9c46f30d7d21715b23b1d
(1) 0x6cbed15c793ce57650b9877cf6fa156fbef513c4e6134f022a85b1ffdd59b2a1
(2) 0x6370fd033278c143179d81c5526140625662b8daa446c22ee2d73db3707e620c
(3) 0x646f1ce2fdad0e6deeb5c7e8e5543bdde65e86029e2fd9fc169899c440a7913
(4) 0xad53f9a7e588d003326d1cbf9e4a43c061aadd9bc938c843a79e7b4fd2ad743
(5) 0x395df67f0c2d2d9fe1ad08d1bc8b6627011959b79c53d7dd6a3536a33ab8a4fd
(6) 0xe485d098507f54e7733a205420dfddbe58db035fa577fc294ebd14db90767a52
(7) 0xa453611d9419d0e56f499079478fd72c37b251a94bfde4d19872c44cf65386e3
(8) 0x829e924fd021ba3dbbc4225edfece9aca04b929d6e75613329ca6f1d31c0bb4
(9) 0xb0057716d5917badaf911b193b12b910811c1497b5bada8d7711f758981c3773

HD Wallet
=====
Mnemonic:      myth like bonus scare over problem client lizard pioneer submit female collect
Base HD Path:  m/44'/60'/0'/0/{account_index}

Default Gas Price
=====
2000000000

BlockGas Limit
=====
30000000

Call Gas Limit
=====
50000000

Chain
=====
Hardfork: shanghai
Id:      1337

```

Figura D.6: Levantar red local usando Ganache

2. **Compile y migrar contrato:** el contrato debe de ser compilado, probado y migrado antes de poder utilizarlo, para ello se utiliza Truffle.

- **Compile:** Es necesario convertir el código Solidity de los contratos en bytecode que pueda ser ejecutado por la Máquina Virtual de Ethereum (EVM). Para ello desde el directorio *SmartContract* donde se almacena toda la lógica de los contratos inteligentes, se debe ejecutar el comando `truffle compile`.

```

E:\ubu-contractme\Code\SmartContract>truffle compile

Compiling your contracts...
=====
> Compiling .\..\..\AppContract\ubu-contractme\Code\SmartContract\contracts\MyContract.sol
> Compiling .\contracts\MyContract.sol
> Compiling .\contracts\MyContract.sol
> Compiling .\contracts\MyContractAux.sol
> Compiling .\contracts\MyContractChanges.sol
> Compiling .\contracts\MyContractManagement.sol
> Compiling .\contracts\MyContractManagement.sol
> Artifacts written to E:\ubu-contractme\Code\SmartContract\build\contracts
> Compiled successfully using:
   - solc: 0.8.20+commit.alb79de6.Emscripten.clang

```

Figura D.7: Compilar SmartContract usando Truffle

- **Migrar:** Finalmente, el último paso es desplegar el contrato en la blockchain. Esto se realiza desde el directorio *SmartContract* usando el comando `truffle migrate`. Este comando desplegará los contratos inteligentes en la red configurada, en este caso Ganache y generará los archivos necesarios en el directorio *build/contracts*. Tras una correcta migración, se mostrará la dirección del contrato, la cuenta del creador, el costo total del despliegue, etcétera.

```

Starting migrations...
=====
> Network name:    'development'
> Network id:      1717691276361
> Block gas limit: 30000000 (0x1c9c380)

1_initial_migration.js
=====

Deploying 'MyContractAux'
=====
> transaction hash: 0x075c717ee3a4d1128190b3103c9434e673fdb87c8dc130e677ca20e6811af48d
> Blocks: 0        Seconds: 0
> contract address: 0xA586074FA4Fe3E546A132a16238abe37951D41fE
> block number:     48
> block timestamp:  1717691632
> account:          0x90F8bf6A479f320ead074411a4B0e7944Ea8c9C1
> balance:          987.687694296519782394
> gas used:         3338040 (0x32ef38)
> gas price:        2.501942642 gwei
> value sent:       0 ETH
> total cost:       0.00835158461670168 ETH

> Saving artifacts
=====
> Total cost:       0.00835158461670168 ETH

Summary
=====
> Total deployments: 1
> Final cost:       0.00835158461670168 ETH

```

Figura D.8: Despliegue del contrato en la red de Ganache usando Truffle

3. **Configuración conexión contrato:** Una vez hayamos desplegado un nuevo contrato en la blockchain, debemos actualizar la información de la aplicación para que esta pueda interactuar con él. Para ello, necesitamos dos cosas: la dirección del contrato, que se nos ha proporcionado en la terminal al migrarlo, y el archivo JSON del contrato que se encuentra en `SmartContract/build`.

Con el archivo JSON del contrato, es necesario copiarlo y pegarlo en la carpeta `AppContract/src/ContractConexion`. Por otro lado con la dirección del contrato, también es necesario copiarla, y actualizar la referencia `contractAddress` que se encuentra en el archivo `EtherProvider.js`.



```
Code - EtherProvider.js
1 export async function getMyContract() {
2   const web3 = await getWeb3();
3   const MyContract1 = require("../MyContractAux.json");
4   const contractAddress = "0xA586074FA4Fe3E546A132a16238abe37951D41fE";
5   const MyContract = new web3.eth.Contract(MyContract1.abi, contractAddress);
6   return MyContract;
7 }
```

Figura D.9: Configuración de conexión con el contrato

4. **Iniciar aplicación:** Desde el directorio raíz de la aplicación, `/Code/AppContract`, en la terminal ejecutamos el comando `npx expo start` para iniciar el servidor. Esto mostrará un código QR en la terminal. Será necesario comprobar que la opción Expo Go se encuentre activa. Seguidamente desde la aplicación Expo Go en el dispositivo móvil, se procederá a escanear el código QR que aparece en la terminal. La aplicación Expo Go cargará y mostrará la aplicación en el dispositivo móvil.

D.5. Pruebas del sistema

- **Probar:** Como paso opcional, pero altamente recomendado, para comprobar el correcto funcionamiento del contrato, se puede ejecutar una batería de tests. Estos tests están diseñados para asegurar que todas las funciones del contrato inteligente operen conforme a las especificaciones establecidas y para identificar cualquier posible falla antes de la implementación en un entorno de producción.

- **Proceso de ejecución de pruebas:** Para llevar a cabo estas pruebas unitarias, es necesario utilizar el comando `truffle test` en la terminal. Este comando debe ejecutarse desde el directorio *SmartContract*, donde se encuentran almacenados todos los scripts de prueba necesarios para la evaluación del contrato.
- **Descripción de los tests:** Los tests verifican cada función y transacción que el contrato inteligente puede realizar, incluyendo la creación, transferencia y administración de los contratos. Cada test está diseñado para garantizar que el contrato responda adecuadamente en cualquier situación.
- **Interpretación de resultados:** Al finalizar la ejecución de los tests, `truffle` proporcionará un reporte detallado que incluirá el resultado de cada prueba individual. Un test exitoso indica que la función correspondiente del contrato cumple con los requisitos y funciona como se espera. Para ver un ejemplo de cómo se presentan estos resultados, ver la imagen [D.10](#).

```
Contract: MyContract
  ✓Acuñar un NFT cuando se envía el suficiente dinero (218ms)
  ✓No se puede acuñar un NFT si no se envía suficiente dinero (293ms)

Contract: MyContract
  ✓Verifica que solo un usuario distinto al que acuñó el NFT pueda firmarlo (296ms)
  ✓Verifica que solo el usuario especificado en el contrato pueda firmarlo (198ms)
  ✓Verifica que el un contrato que haya sido firmado no pueda ser firmado de nuevo (324ms)
  ✓Verifica que un contrato no pueda ser firmado tras su vencimiento (280ms)

Contract: MyContract
  ✓Verifica que solo el propietario pueda liberar el salario (243ms)
  ✓Verifica que el salario no pueda ser liberado antes de que el contrato expire (240ms)
  ✓Comprueba que el salario se transfiera al trabajador cuando se libere el pago (293ms)
  ✓Comprueba que permita finalizar un contrato cuando no ha expirado pero el dueño lo haya declarado como finalizado (323ms)

Contract: MyContract
  ✓Comprueba que un contrato se pause de forma correcta (272ms)
  ✓Comprueba que el contrato se despause de forma correcta (320ms)
  ✓Comprueba que un dueño se pueda cancelar y este deje de tener valor. (468ms)
  ✓Comprueba que el rol de manager pueda interactuar con el contrato (346ms)
  ✓Comprueba que un manager pueda modificar un contrato (385ms)
  ✓Comprueba los contratos que ha emitido un owner (522ms)
  ✓Comprueba que se pueda eliminar a un manager (246ms)

17 passing (6s)
```

Figura D.10: Pruebas unitarias sobre el SmartContract usando Truffle

Apéndice E

Documentación de usuario

- E.1. Introducción
- E.2. Requisitos de usuarios
- E.3. Instalación
- E.4. Manual del usuario

Apéndice F

Anexo de sostenibilización curricular

F.1. Introducción

La transformación digital está remodelando diversos sectores económicos. Sin embargo, algunos sectores como la agricultura, los servicios domésticos y el trabajo freelance enfrentan desafíos significativos en el proceso de contratación y verificación de identidad. En esta sección, se propone una solución alineada con las directrices de sostenibilidad en el curriculum universitario aprobadas por la CRUE.

Sostenibilidad social y económico

la sostenibilidad social implica la creación de un entorno laboral justo y equitativo. La utilización de blockchain y contratos inteligentes asegura la transparencia y seguridad en los contratos laborales, garantizando que los términos acordados se cumplan automáticamente y sin intervención manual. Este enfoque combate con la informalidad laboral y protege al trabajador, asegurando que se cumplan unas condiciones laborales dignas, en línea con el principio ético de las directrices de sostenibilidad.

Desde una perspectiva económica, fomentar técnicas legales y justas que aseguren la reducción de la economía sumergida, fortalece la economía formal y aumenta la recaudación fiscal. Al implementar contratos inteligentes, la aplicación provee un registro inmutable de todas las transacciones, facilitando la fiscalización y el cumplimiento de las normativas. Esto incrementa la confianza en el sistema laboral y fomenta un desarrollo económico sostenible.

Formación para la sostenibilidad

En el proyecto también se reconoce y se valora la importancia de la educación en sostenibilidad. La formación de los usuarios en el uso de la blockchain y contratos inteligentes refuerza la capacidad de tomar decisiones informadas y responsable, fomentando la cultura de transparencia y equidad en el mercado laboral perseguida con el desarrollo de dicho proyecto. Esta formación es clave para desarrollar competencias transversales en sostenibilidad.

Accesibilidad

La accesibilidad es crucial en el desarrollo de nuestro proyecto, especialmente enfocado para los trabajadores de sectores marginados, permitiendo el acceso a unas condiciones laborales justas y transparentes. La aplicación está diseñada para ser simple y accesible desde dispositivos móviles, capaz de ejecutarse en incluso en dispositivos con muy pocos recursos. La integración de tecnologías de autenticación biométrica proporciona una capa adicional de seguridad, garantizando la privacidad y protección de la identidad de los trabajadores.

Impacto Medioambiental

Aunque el foco principal del proyecto se encuentra en la sostenibilidad social y económica, también se considera el impacto medioambiental de la solución. La digitalización de procesos laborales reduce la dependencia del papel y disminuye la necesidad de desplazamientos físicos, contribuyendo a la reducción de la huella de carbono. La eficiencia y automatización proporcionadas por las tecnologías empleadas optimizan el uso de recursos, minimizando el desperdicio y promoviendo prácticas más sostenibles.

F.2. Conclusión

La integración de sostenibilidad en el proyecto presentado aborda de manera integral los desafíos de la economía sumergida y promueve prácticas laborales justas y transparentes. Al adoptar tecnologías disruptivas como blockchain y contratos inteligentes, garantizamos la formalización del empleo, la protección de los derechos de los trabajadores y el desarrollo económico sostenible. Este enfoque holístico y transversal busca que no solo mejore la eficiencia operativa sino que también tenga un impacto positivo duradero en la sociedad. Al integrar principios de sostenibilidad en el diseño, promovemos

un entorno laboral más justo, seguro y equitativo, alineado con los objetivos de desarrollo sostenible.