



Escuela
Politécnica
Superior

Blockchain y Web 3.0: Creación de una Plataforma Electoral



Grado en Ingeniería en Sonido e
Imagen en Telecomunicación

Trabajo Fin de Grado

Autor:

Carlos A. Peña López

Tutores:

Higinio Mora Mora

Francisco Antonio Pujol López

Julio 2024

Blockchain y Web 3.0: Creación de una Plataforma Electoral

Una Aproximación Innovadora para la Seguridad y Transparencia en las Elecciones Electrónicas

Autor

Carlos A. Peña López

Tutores

Higinio Mora Mora

Tecnología Informática y Computación

Francisco Antonio Pujol López

Tecnología Informática y Computación



Grado en Ingeniería en Sonido e Imagen en Telecomunicación



Escuela
Politécnica
Superior



Universitat d'Alacant
Universidad de Alicante

ALICANTE, Julio 2024

Resumen

Este Trabajo de Fin de Grado (TFG) se centra en la modernización de los sistemas de votación mediante la aplicación de tecnologías emergentes como Blockchain, Web 3.0 (Web3) y Smart Contracts (SC). En la actualidad, la necesidad de actualizar los procesos electorales tradicionales se ha vuelto más evidente debido a los crecientes desafíos relacionados con la seguridad, la transparencia y la accesibilidad. A través de este proyecto, se exploran y analizan detalladamente estas tecnologías para demostrar cómo pueden integrarse en sistemas de votación electrónica y, de esta manera, mejorar significativamente estos aspectos críticos.

El estudio se divide en tres áreas principales: Bockchain, Web3 y Smart Contracts. Block-chain se analiza por su capacidad para ofrecer un registro inmutable y descentralizado, lo cual es crucial para asegurar la integridad de los votos. Web3 se examina como la próxima generación de internet que promueve la descentralización y el control de los datos por parte de los usuarios, ofreciendo una infraestructura más segura y eficiente para aplicaciones descentralizadas. Los smart contracts se estudian por su capacidad para automatizar y garantizar la ejecución de procesos de manera transparente y sin intermediarios.

Como resultado práctico del proyecto, se desarrollará una aplicación web de votación electrónica que utiliza estas tecnologías. Esta aplicación permitirá a los usuarios emitir sus votos de forma segura y verificar los resultados de manera transparente, incrementando así la confianza pública en el proceso electoral. Además, se identificarán y propondrán soluciones a los principales desafíos en la implementación de estos sistemas tecnológicos, asegurando que se superen las limitaciones de los métodos de votación tradicionales.

Este TFG no solo contribuirá al conocimiento teórico y práctico en el campo de las tecnologías emergentes y su aplicación en sistemas críticos, sino que también demostrará cómo las habilidades adquiridas en la formación en ingeniería pueden aplicarse para innovar y desarrollar soluciones tecnológicas avanzadas con un impacto positivo en la sociedad.

Agradecimientos

Quisiera expresar mi más sincero y profundo agradecimiento a todas las personas que han sido parte fundamental en mi vida a lo largo de mi carrera universitaria. Sin su apoyo incondicional, paciencia y amor, este logro no habría sido posible.

A mi familia que han sido mi pilar y mi mayor fuente de inspiración. Gracias por vuestro amor incondicional, por estar siempre a mi lado y por creer en mí incluso cuando yo mismo dudaba. A mis padres, gracias por los sacrificios que habéis hecho para proporcionarme las oportunidades que he tenido. Vuestros consejos y enseñanzas han sido la base sobre la cual he construido mi camino.

A mis amigos, quienes han compartido conmigo esta travesía, desde las largas noches de estudio hasta los momentos de celebración. Gracias por las risas, las conversaciones profundas y el apoyo emocional. Vuestra amistad ha sido un refugio y una fuente de fortaleza durante los momentos difíciles. Gracias por estar a mi lado, por vuestra comprensión y por hacer que este viaje fuera mucho más llevadero y memorable.

Quiero agradecer también a mis profesores y mentores, quienes me han guiado y motivado a lo largo de mi formación académica. Vuestro conocimiento y dedicación han sido fundamentales para mi crecimiento profesional y personal.

Finalmente, quiero agradecer a todos aquellos que, de una manera u otra, han contribuido a mi desarrollo y éxito. A mis compañeros de clase, gracias por el trabajo en equipo y la camaradería. A todas las personas que me han brindado su apoyo y consejos durante estos años, gracias por vuestra amabilidad y generosidad.

Este logro no es solo mío, sino de todos vosotros que habéis estado a mi lado y habéis contribuido a mi crecimiento. De todo corazón, **gracias**.

*A mi madre y a mi familia sin los cuales
no habría podido lograr nada.*

El avance de la tecnología se basa en hacer que encaje de forma que ni siquiera se note, de manera que forme parte de la vida cotidiana.

Bill Gates.

Índice general

1	Introducción.	1
1.1	Motivación y alcance.	2
2	Objetivos	3
3	Marco Teórico	5
3.1	Blockchain	5
3.1.1	Introducción a la Blockchain	5
3.1.1.1	Elementos básicos de la Blockchain	6
3.1.2	Sistemas descentralizados	8
3.1.2.1	Ventajas y desventajas de los sistemas descentralizados	9
3.1.2.2	Sistemas Distribuidos <i>Peer-to-Peer</i> (P2P)	12
3.1.2.3	El Propósito de la Blockchain	13
3.1.3	Planificación de la Blockchain	15
3.1.3.1	Documentación de la Propiedad	16
3.1.4	Fundamentos Criptográficos de la Blockchain	17
3.1.4.1	Fundamentos de las funciones Hash	18
3.1.4.2	Puzzles Hash	23
3.1.4.3	Criptografía Asimétrica.	25
3.1.5	Estructura de Datos de la Blockchain	27
3.1.6	Aplicaciones de la Blockchain	29
3.2	Smart Contracts	31
3.2.1	Herramientas Para el Desarrollo Blockchain	32
3.2.1.1	Solidity	32
3.2.1.2	Ganache	35
3.2.1.3	Truffle	36
3.3	Web 3.0	37
3.3.1	Principios Fundamentales de Web3	38
3.3.2	Características Clave de Web3	38
3.3.2.1	Carteras Digitales	38
3.3.2.2	Identidad Descentralizada	40
3.3.2.3	Seguridad de los Smart Contracts	41
3.3.2.4	Desarrollo y Herramientas para Smart Contracts	41
3.3.2.5	Mecanismos de Ejecución y Verificación	41
3.3.2.6	Lenguajes de Programación para Smart Contracts	42
3.3.2.7	Beneficios y Limitaciones de los Smart Contracts	42

4 Estado del Arte	43
4.1 Blockchain	43
4.1.1 Historia de la Blockchain	45
4.1.2 Clasificación de Redes Blockchain	46
4.1.3 Direcciones Futuras	48
4.1.3.1 Desafíos	50
4.1.4 Conclusiones	51
4.2 Smart Contracts	51
4.2.1 Historia de los Smart Contracts	51
4.2.2 Marco de Investigación	53
4.2.3 Direcciones Futuras	54
4.3 Web 3.0	56
4.3.1 Historia de la Web 3.0	56
4.3.1.1 Nacimiento de Internet	57
4.3.1.2 Web 1.0	58
4.3.1.3 Web 2.0	58
4.3.1.4 Web 3.0	59
5 Desarrollo Web3 de la Plataforma Electoral	61
5.1 Estudio Comparativo	61
5.2 Herramientas Para el Desarrollo de la Dapp	63
5.3 Desarrollo Web3	66
5.4 Ganache y Truffle Suite	66
5.5 Metamask	76
5.6 Plataforma Electoral	78
5.6.1 Smart Contrates	93
6 Propuestas de Mejora	99
7 Conclusiones	101
Bibliografía	103

Índice de figuras

3.1 Como funciona la Blockchain (PwC, 2023).	7
3.2 Arquitectura de sistema distribuido vs centralizado	9
3.3 Ejemplo de red P2P (Napster) (RedesZone, 2024).	12
3.4 Ilustración esquemática de los conceptos criptográficos básicos y su terminología.	18
3.5 Cálculo de los valores hash de un texto.	20
3.6 Ilustración esquemática del hash de diferentes datos de forma independiente.	21
3.7 Cálculo de valores hash repetidamente.	21
3.8 Combinación de datos y cálculo del valor hash.	22
3.9 Cálculo de los valores hash de un texto de manera secuencial.	22
3.10 Cálculo de los valores hash de un texto jerárquicamente.	23
3.11 Ejemplo de puzzle hash.	24
3.12 Funcionamiento de la criptografía asimétrica.	26
3.13 Ejemplo de árbol de Merkle (Wikipedia, 2022).	28
4.1 Cronología de la Web3.	43
4.2 Capitalización de mercado de las principales criptomonedas según la web <i>Statista</i> (Roa, 2021).	44
4.3 Cronología de la Blockchain.	45
4.4 Mapa mental de los distintos tipos de aplicaciones de blockchain (Casino y cols., 2019).	49
4.5 Cronología de los Smart Contracts.	53
4.6 Mapa de Arpanet 1972 (Wikipedia contributors, 2023).	57
4.7 Cronología de la Web3.	60
5.1 Ideas para la aplicación encontradas en la web.	62
5.2 Página principal de Ganache.	67
5.3 Página de creación del Workspace.	68
5.4 Página principal del Workspace con las diferentes cuentas creadas.	68
5.5 Configuración de cuentas	69
5.6 Configuración de la cadena	69
5.7 Configuración del servidor	69
5.8 Otras configuraciones	69
5.9 Vista de las diferentes ventanas en Ganache.	70
5.10 Contratos compilados.	70
5.11 Información detalla del despliegue de los contratos.	71
5.12 Registro de cuentas donde aparece las transacciones (2) de los contratos desplegados.	73
5.13 Registro de las transacciones al desplegar los contratos inteligentes.	74
5.14 Información detalla del contrato Migraciones.	75

5.15 Información detalla del contrato Elecciones.	75
5.16 Vista general de la cadena de bloques.	75
5.17 Información detallada del Bloque 2.	75
5.18 Agregando una red manualmente en Metamask.	76
5.19 Registro de cuentas de Metamask.	77
5.20 Pestaña con la clave privada de cada cuenta en Ganache.	78
5.21 Cuenta de Administración.	78
5.22 Cuenta de Usuario.	78
5.23 Inicio de sesión en Metamask al inicializar la Dapp.	79
5.24 Inicio de sesión la Dapp como administrador.	84
5.25 Inicio de sesión la Dapp como usuario.	85
5.26 Prompt de firma del contrato en Metamask.	86
5.27 Información detallada de la interacción con el contrato inteligente en Ganache.	87
5.28 Panel de Administración.	88
5.29 Añadir nuevo candidato.	88
5.30 Vista del Panel de Administración con candidatos añadidos.	89
5.31 Vista del Panel de Usuarios con candidatos añadidos.	89
5.32 Firma del contrato para votar.	90
5.33 Firma del contrato para votar.	91
5.34 Vista del la ventana "Eventos".	91
5.35 Vista detallada del evento.	91
5.36 Vista del la ventana "Eventos".	92
5.37 Vista detallada del evento.	92
5.38 Pestaña de resultados al cerrar las elecciones.	92
5.39 Pestaña para votar al cerrar las elecciones.	92

Índice de tablas

4.1 Clasificación y características principales de las redes blockchain.	47
4.2 Características/requisitos que permiten/requieren cada familia de aplicaciones blockchain. "Sí" indica que este requisito es obligatorio, mientras que "Depende" indica que depende del caso (Casino y cols., 2019).	47
4.3 Inversión en tecnología Blockchain en diferentes regiones en 2017(Namasudra y cols., 2021).	48
5.1 Resumen de herramientas utilizadas en la aplicación.	65

Índice de Códigos

3.1 Ejemplo de sintaxis en Solidity	33
3.2 Ejemplo de datos en Solidity	33
3.3 Ejemplo de funciones y modificadores en Solidity	33
3.4 Ejemplo de funciones y modificadores en Solidity	34
3.5 Ejemplo de configuración en Ganache	35
5.1 Configuración de Ganache	67
5.2 Código para el despliegue del contrato "Migraciones.sol"	71
5.3 Código para el despliegue del contrato "Elecciones.sol"	71
5.4 Inicialización de la conexión a MetaMask	79
5.5 Función render	81
5.6 Función para agregar usuarios	85
5.7 Función para agregar usuarios en el Smart Contract	86
5.8 Función para agregar candidatos en app.js	89
5.9 Función para agregar candidatos en el Smart Contract	89
5.10 Función para votar candidatos	90
5.11 Función para votar candidatos en el Smart Contract	90
5.12 Contrato de Elecciones	93
5.13 Contrato de Migración	96

1 Introducción.

En la era digital actual, la transformación de los procesos electorales tradicionales hacia plataformas electrónicas se presenta como una necesidad imperativa para mejorar la eficiencia, transparencia y accesibilidad de las elecciones. Sin embargo, la implementación de sistemas de votación electrónica enfrenta desafíos significativos, como la seguridad, la integridad y la confianza del votante. En este contexto, la tecnología blockchain emerge como una solución innovadora que puede abordar estos desafíos, ofreciendo una plataforma segura, descentralizada y transparente para la gestión de elecciones.

La blockchain, conocida por su capacidad para proporcionar registros inmutables y verificables de transacciones, se ha aplicado con éxito en diversas áreas, desde las finanzas hasta la cadena de suministro. Sin embargo, su potencial en el ámbito electoral aún está en fases iniciales de exploración. La motivación de este proyecto radica en investigar y demostrar cómo la integración de blockchain en sistemas de votación electrónica puede mejorar significativamente la confianza y la transparencia de los procesos electorales.

Este proyecto también explora los principios y aplicaciones de la Web 3.0 (Web3), una evolución de la web que se basa en la descentralización y el uso de tecnologías como blockchain y smart contracts (contratos inteligentes). La Web3 ofrece una infraestructura más segura y eficiente para el desarrollo de aplicaciones descentralizadas, lo que es crucial para la implementación de sistemas de votación electrónica avanzados. La combinación de blockchain y los principios de la Web3 crea una plataforma robusta y moderna que puede revolucionar la forma en que se llevan a cabo las elecciones.

Además de la seguridad y transparencia, otro aspecto crucial que se aborda en este proyecto es la accesibilidad. La votación electrónica basada en blockchain puede proporcionar una mayor accesibilidad a votantes con discapacidades, a aquellos que residen en áreas remotas, y a ciudadanos que se encuentran en el extranjero. Mediante la implementación de interfaces de usuario intuitivas y accesibles, junto con mecanismos de autenticación seguros, se puede garantizar que todos los ciudadanos tengan la oportunidad de participar en el proceso electoral de manera justa y equitativa.

Este trabajo tiene como objetivo no solo presentar un análisis teórico de las ventajas de la blockchain en la votación electrónica, sino también ofrecer una implementación práctica que pueda servir como base para futuros desarrollos en este campo. La esperanza es que, con el tiempo, estas tecnologías puedan ser adoptadas globalmente, llevando a una democratización y modernización de los procesos electorales a nivel mundial.

1.1 Motivación y alcance.

Considero de especial importancia recalcar que aunque este trabajo puede no parecer directamente relacionado con mi carrera en Ingeniería en Sonido e Imagen en Telecomunicación, considero que la formación en ingeniería va más allá de la especialización en un solo campo. Estudiar una ingeniería te dota de una serie de aptitudes y habilidades, como el pensamiento analítico, la resolución de problemas y la capacidad de aprender nuevas tecnologías de manera rápida y eficiente. Estas competencias te permiten adaptarte y progresar profesionalmente en diversos campos, aprendiendo y aplicando conocimientos en áreas emergentes en un tiempo reducido. Mi motivación para este proyecto surge del deseo de ampliar y aplicar estos conocimientos y habilidades, explorando nuevas tecnologías y conceptos avanzados que pueden tener un impacto significativo en diversos sectores.

El alcance de este trabajo es establecer una base sólida de los conceptos fundamentales relacionados con la Web3 y la votación electrónica basada en blockchain. Para ello, se explicarán en detalle los conceptos de blockchain y contratos inteligentes, que son los pilares sobre los que se basa la Web3. Este apartado busca proporcionar una comprensión clara y detallada de cómo funcionan estas tecnologías y sus aplicaciones potenciales en el ámbito electoral. Asimismo, se revisarán y analizarán los trabajos y proyectos previos en el ámbito de las elecciones electrónicas y otras aplicaciones relevantes de blockchain, ofreciendo una visión comprensible y bien fundamentada de las tendencias y avances actuales en estas tecnologías.

Además, se explorarán los desafíos y las soluciones propuestas en la literatura para la implementación de sistemas de votación electrónica seguros y transparentes. Esto incluirá un análisis de los mecanismos de consenso en blockchain, la gestión de claves criptográficas y la anonimización de votos, todos ellos cruciales para garantizar la seguridad y la privacidad de los votantes. También se discutirán las implicaciones legales y éticas de la adopción de la votación electrónica, con un enfoque en cómo los marcos regulatorios pueden adaptarse para incorporar estas nuevas tecnologías.

Con este enfoque, se espera no solo desarrollar una aplicación práctica, sino también contribuir al conocimiento académico y técnico en estas áreas emergentes, facilitando una comprensión profunda y accesible de las tecnologías que sustentan la Web3 y su potencial transformador en diversos contextos. El objetivo final es demostrar que las habilidades adquiridas en una formación en ingeniería pueden aplicarse de manera efectiva a la resolución de problemas complejos en una variedad de campos, y que la adopción de tecnologías como blockchain puede tener un impacto positivo y duradero en la sociedad.

2 Objetivos

En el mundo actual, donde la tecnología está transformando continuamente diversas facetas de la vida cotidiana, la adopción de soluciones digitales en procesos críticos como las elecciones se vuelve cada vez más relevante. La digitalización no solo afecta cómo comunicamos y accedemos a la información, sino también cómo interactuamos con instituciones y participamos en procesos democráticos.

El acto de votar, fundamental para la democracia, ha permanecido en gran medida sin cambios durante décadas. Sin embargo, los desafíos contemporáneos, como el aumento de la desconfianza en los procesos electorales, el riesgo de fraudes y manipulaciones, y la necesidad de accesibilidad para todos los votantes, impulsan la necesidad de innovaciones tecnológicas que puedan modernizar y proteger el sistema de votación.

Para abordar estos desafíos, este proyecto tiene como objetivo principal explorar y entender las tecnologías emergentes que están liderando esta transformación, centrándose en blockchain, Web3 y smart contracts. A través de una comprensión profunda y actualizada de estas tecnologías, se busca demostrar cómo pueden aplicarse en sistemas de votación electrónica para mejorar la seguridad, transparencia y accesibilidad del proceso electoral. Como resultado final, se desarrollará una aplicación web del sistema de votación utilizando estas tecnologías mencionadas. Esta aplicación permitirá a los usuarios emitir sus votos de manera segura y verificar los resultados de forma transparente, promoviendo una mayor confianza en el proceso electoral.

1. Blockchain:

- **Propósito y Origen:** Investigar cómo y por qué se desarrolló la tecnología blockchain, y entender sus características esenciales que la hacen única, como la descentralización y la inmutabilidad.
- **Evolución y Aplicaciones:** Analizar la evolución de blockchain desde sus inicios hasta su estado actual, y examinar cómo se está aplicando en diversos sectores, con un enfoque en las potencialidades para mejorar la votación electrónica.

2. Web3:

- **Principios Básicos:** Definir qué es Web3 y cómo difiere de las generaciones anteriores de la web. Examinar sus principios fundamentales, como la descentralización y la propiedad de datos por parte de los usuarios.
- **Impacto y Desarrollo:** Estudiar cómo Web3 está transformando la interacción digital y el desarrollo de aplicaciones descentralizadas, y su potencial para ofrecer infraestructuras más seguras y eficientes.

3. Smart Contracts:

- **Concepto y Funcionamiento:** Comprender qué son los smart contracts, cómo funcionan y qué beneficios aportan. Investigar cómo estos contratos autoejecutables pueden automatizar procesos y garantizar el cumplimiento de acuerdos de manera segura.
- **Casos de Uso y Aplicaciones:** Explorar las diversas aplicaciones de los smart contracts, especialmente en contextos donde la automatización y la confianza son cruciales, como en la votación electrónica.

Para aplicar estas tecnologías en el ámbito electoral, es esencial evaluar cómo pueden mejorar y superar las limitaciones de los sistemas tradicionales de votación. Las áreas clave de enfoque incluyen:

- **Mejora de Procesos Tradicionales:** Evaluar cómo estas tecnologías pueden abordar y superar las limitaciones de los sistemas de votación tradicionales, mejorando aspectos críticos como la seguridad, la transparencia y la accesibilidad.
- **Desafíos y Soluciones:** Identificar los principales desafíos en la implementación de sistemas de votación electrónica basados en estas tecnologías y proponer soluciones viables para superarlos.
- **Innovación y Confianza:** Investigar cómo la integración de blockchain, Web3 y smart contracts puede aumentar la confianza del público en los procesos electorales, promoviendo una mayor participación ciudadana y garantizando la integridad del proceso electoral.
- **Aplicación Web:** Desarrollar una aplicación web del sistema de votación utilizando blockchain, smart contracts y Web3, demostrando la viabilidad y eficacia de estas tecnologías en un entorno práctico.

Este proyecto también tiene como objetivo contribuir al conocimiento académico y práctico en el campo de las tecnologías emergentes y su aplicación en sistemas críticos como las elecciones. Los aspectos clave incluyen:

- **Base Teórica y Práctica:** Establecer una sólida base teórica sobre estas tecnologías y su aplicación en la votación electrónica, complementada con una implementación práctica que demuestre su viabilidad.
- **Impacto en la Ingeniería:** Demostrar cómo las competencias adquiridas en una formación en ingeniería pueden aplicarse para innovar en áreas emergentes, contribuyendo al desarrollo de soluciones tecnológicas avanzadas que pueden tener un impacto positivo significativo en la sociedad.

3 Marco Teórico

3.1 Blockchain

3.1.1 Introducción a la Blockchain

Una blockchain es, en esencia, una base de datos distribuida entre diferentes participantes, protegida criptográficamente y organizada en bloques de transacciones que están matemáticamente relacionados entre sí. En términos más simples, se trata de una base de datos descentralizada que no puede ser alterada una vez que la información ha sido registrada. Uno de los aspectos más importantes de la blockchain es que permite que partes que no confían plenamente unas en otras puedan mantener un consenso sobre la existencia, el estado y la evolución de una serie de factores compartidos. Este consenso es fundamental porque permite que todos los participantes confíen en la información registrada en la blockchain. Este mecanismo de confianza tiene el potencial de transformar numerosos sectores clave de la industria y la sociedad, cambiando incluso nuestra forma de entender el mundo.

Desde un punto de vista técnico, el sistema de confianza y consenso de la blockchain se construye a partir de una red global de ordenadores que gestionan una gigantesca base de datos. Esta base de datos puede estar abierta a la participación de cualquiera, en cuyo caso hablamos de una blockchain pública, o bien limitada a ciertos participantes, lo que se conoce como blockchain privada. En cualquier caso, estas redes operan sin la necesidad de una entidad central que supervise o valide los procesos.

La primera blockchain fue la pública de Bitcoin, lanzada en enero de 2009. En su funcionamiento, juegan un papel importante conceptos como la "minería", que se inspira en la minería del oro y se refiere al proceso computacional necesario para asegurar su red mediante la Prueba de Trabajo (Proof of Work, PoW). Los mineros, al resolver complejos problemas matemáticos, validan nuevas transacciones y las agrupan en bloques que se añaden a la cadena. Este proceso asegura la inmutabilidad y la transparencia de la blockchain. Sin embargo, este método consume una gran cantidad de energía, lo que ha llevado al desarrollo de otros mecanismos de consenso como Proof of Stake (PoS), que busca reducir el consumo energético al seleccionar validadores en función de la cantidad de criptomonedas que poseen y están dispuestos a "apostar" como garantía.

Las blockchains soportan smart contracts, que son programas autoejecutables almacenados en la blockchain que se activan cuando se cumplen condiciones predefinidas. Estos contratos pueden automatizar y asegurar una variedad de procesos, desde transacciones financieras hasta la gestión de cadenas de suministro, sin necesidad de intermediarios. Los smart contracts amplían significativamente las aplicaciones de la tecnología blockchain, permitiendo la creación de aplicaciones descentralizadas (Dapps) que operan de manera autónoma y segura.

Además de Bitcoin, otras blockchains como Ethereum han expandido las capacidades de esta tecnología al introducir plataformas que permiten la creación y ejecución de smart contracts. Ethereum, lanzada en 2015, se ha convertido en una de las principales plataformas para el desarrollo de aplicaciones descentralizadas debido a su flexibilidad y amplio soporte comunitario.

A pesar de sus ventajas, la blockchain no está exenta de desafíos. La escalabilidad es una de las principales limitaciones, ya que el proceso de validación puede ser lento y costoso en términos de recursos. Esto ha llevado a la investigación y desarrollo de soluciones de escalabilidad como las *sidechains* y las tecnologías de segunda capa (Layer 2, L2), que buscan mejorar la capacidad y la velocidad de las transacciones sin comprometer la seguridad y la descentralización.

Otro desafío importante es la regulación. La naturaleza descentralizada de la blockchain puede entrar en conflicto con las normativas existentes en varios países, lo que plantea preguntas sobre cómo integrar estas tecnologías en el marco legal y regulatorio global. La protección de datos y la privacidad también son áreas de preocupación, especialmente en aplicaciones que manejan información sensible.

La blockchain no solo redefine la manera en que gestionamos y compartimos información, sino que también tiene el potencial de revolucionar diversos sectores, desde las finanzas hasta la administración pública, pasando por la salud y la logística, entre otros. Su capacidad para proporcionar un registro inmutable y verificable de transacciones y datos la convierte en una herramienta poderosa para asegurar la transparencia y la confianza en múltiples ámbitos de la sociedad moderna. A medida que se superen los desafíos técnicos y regulatorios, veremos una adopción aún mayor de esta tecnología transformadora (Preukschat y cols., 2017).

3.1.1.1 Elementos básicos de la Blockchain

Para entender el alcance de la tecnología blockchain es fundamental conocer los elementos básicos que la componen. Estos son los siguientes:

- **Nodo:** Un nodo puede ser cualquier dispositivo informático, desde un ordenador personal hasta un superordenador, dependiendo de la complejidad de la red. Todos los nodos deben ejecutar el mismo software o protocolo para poder comunicarse entre sí y formar parte de la red blockchain, ya sea pública, privada o híbrida. En una blockchain pública, los nodos pueden operar de manera anónima y no necesitan identificarse, mientras que en una blockchain privada, los nodos suelen conocerse entre sí y pueden tener configuraciones idénticas o específicas según los permisos otorgados.
- **Protocolo Estándar:** El protocolo de una blockchain es un software que define cómo los nodos de la red se comunican entre sí. Es comparable a otros protocolos bien conocidos como TCP/IP para internet o SMTP para el intercambio de correos electrónicos. El protocolo blockchain establece un estándar común para la comunicación y asegura que todos los nodos participantes puedan interactuar de manera eficiente y segura.

- **Red entre Pares (P2P):** La red P2P (*Peer-to-Peer*) consiste en una estructura donde los nodos están directamente conectados entre sí sin necesidad de un servidor central. Este tipo de red permite un intercambio de información descentralizado y eficiente. Un ejemplo bien conocido de una red P2P es BitTorrent, que permite a los usuarios compartir archivos directamente entre ellos. En una blockchain, esta red facilita la distribución de datos y la validación de transacciones de manera simultánea entre todos los nodos participantes.
- **Sistema Descentralizado:** A diferencia de los sistemas centralizados, donde una única entidad controla toda la información, en un sistema descentralizado como una blockchain, todos los nodos de la red comparten el control. No existe una jerarquía entre los nodos en una blockchain pública, lo que significa que todos los nodos son iguales y tienen los mismos derechos. En una blockchain privada, puede existir una jerarquía, donde ciertos nodos tienen más autoridad que otros. Este punto será desarrollado en el siguiente apartado.

De lo anterior, se desprende que una blockchain es un conjunto de ordenadores (o servidores) llamados "nodos" que, conectados en red, utilizan un mismo sistema de comunicación (el protocolo) con el objetivo de validar y almacenar la misma información registrada en una red P2P. Esta sería la estructura "física" de la blockchain, similar a la carrocería de un coche.

Pero, ¿qué impulsa realmente la blockchain? El "motor" de la blockchain es la combinación de todos estos elementos que garantizan que la información registrada no pueda ser modificada. Esto se logra a través de complejos algoritmos criptográficos y la capacidad colectiva de la red, que trabajan juntos para asegurar la irreversibilidad y la seguridad de la información.

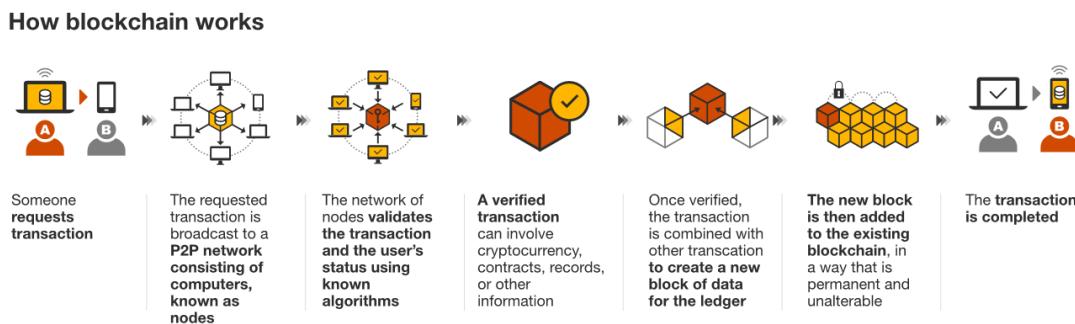


Figura 3.1: Como funciona la Blockchain (PwC, 2023).

La criptografía desempeña un papel crucial en este proceso, asegurando que cada bloque de transacciones esté encadenado al anterior mediante funciones hash criptográficas. Este encadenamiento hace que cualquier intento de modificar un bloque alteraría todos los bloques posteriores, lo cual sería inmediatamente detectable por la red. Los mecanismos de consenso, como PoW o PoS, aseguran que todos los nodos lleguen a un acuerdo sobre el estado de la blockchain, reforzando aún más la integridad y la seguridad del sistema.

La blockchain es una infraestructura compleja y robusta que combina tecnología avanzada y principios descentralizados para crear un sistema seguro y transparente para el almacenamiento y la transferencia de datos. Este enfoque innovador tiene el potencial de transformar numerosos sectores, desde las finanzas hasta la administración pública, proporcionando una base confiable para la gestión de información en la era digital (IBM, 2023; Preukschat y cols., 2017; Wikipedia contributors, 2024a).

3.1.2 Sistemas descentralizados

Hay muchas formas de implementar sistemas de software. Sin embargo, una de las decisiones fundamentales al implementar un sistema concierne a su arquitectura, es decir, la forma en que sus componentes están organizados y relacionados entre sí. Esta elección es crucial porque afecta directamente la eficiencia, escalabilidad y robustez del sistema. Los dos enfoques arquitectónicos principales para sistemas de software son centralizados y distribuidos o descentralizados. En sucesivos apartados se explica los conceptos basados en libro Blockchain Basics (Drescher, 2017).

En los sistemas de software centralizados, los componentes están ubicados alrededor de un componente central y conectados a él. Este componente central actúa como un coordinador principal que gestiona la comunicación y el procesamiento de datos. La simplicidad de esta estructura facilita su diseño, implementación y mantenimiento, ya que el control y la supervisión se concentran en un solo punto. No obstante, esta centralización también puede ser una desventaja, ya que crea un único punto de falla: si el componente central falla, todo el sistema puede colapsar. Además, a medida que el sistema crece, el componente central puede convertirse en un cuello de botella, limitando la capacidad de escalar el sistema.

En contraste, los componentes de los sistemas distribuidos forman una red de componentes conectados sin tener ningún elemento central de coordinación o control. Esta arquitectura permite una mayor escalabilidad y resiliencia, ya que la carga de trabajo y las responsabilidades están distribuidas entre varios nodos. Si un componente falla, los demás pueden continuar funcionando, lo que mejora la tolerancia a fallos. Sin embargo, diseñar y gestionar sistemas distribuidos es más complejo debido a la necesidad de asegurar la coherencia de datos y la seguridad a través de múltiples nodos, así como manejar la comunicación eficiente entre ellos.

La Figura 3.2 muestra estas dos arquitecturas opuestas. Los círculos en la figura representan nodos, y las líneas representan conexiones entre ellos. En este punto, no es importante conocer los detalles de lo que hacen estos componentes y qué información se intercambia entre los nodos. Lo importante es la existencia de estas dos formas diferentes de organizar sistemas de software. En el lado izquierdo, se ilustra una arquitectura distribuida donde los componentes están conectados entre sí sin tener un elemento central. Es importante ver que ninguno de los componentes está conectado directamente con todos los demás componentes. Sin embargo, todos los componentes están conectados entre sí al menos indirectamente, asegurando la comunicación a través de la red. En el lado derecho se ilustra una arquitectura centralizada donde cada componente está conectado a un componente central. Los componentes no están conectados entre sí directamente. Solo tienen una conexión directa con el componente central, lo que centraliza la gestión y el flujo de información en un solo nodo.

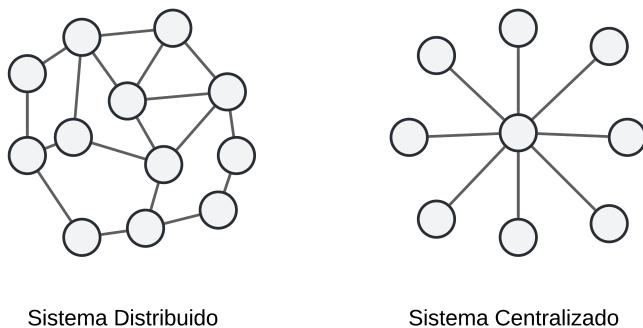


Figura 3.2: Arquitectura de sistema distribuido vs centralizado

3.1.2.1 Ventajas y desventajas de los sistemas descentralizados

- **Ventajas:** Las ventajas de los sistemas descentralizados son:

– Mayor Potencia de Cómputo

La potencia de cómputo de un sistema distribuido es el resultado de combinar la potencia de cómputo de todas los ordenadores conectados. Por lo tanto, los sistemas distribuidos típicamente tienen una potencia de cómputo superior en comparación con cada ordenador individual. Esta afirmación se ha comprobado cierta incluso al comparar sistemas distribuidos compuestos por computadoras de potencia de cómputo relativamente baja con superordenadores aislados. Esto se debe a la capacidad de los sistemas distribuidos para dividir y parallelizar tareas complejas entre múltiples nodos, aprovechando al máximo los recursos disponibles y logrando un procesamiento más eficiente y rápido. Además, esta configuración permite una escalabilidad casi ilimitada, ya que se pueden agregar más nodos al sistema para aumentar la capacidad de procesamiento según sea necesario.

– Reducción de Costes

El precio de los ordenadores convencionales, la memoria, el espacio en disco y los equipos de red ha caído drásticamente en los últimos 20 años. Dado que los sistemas distribuidos consisten en muchos ordenadores, los costes iniciales de los sistemas distribuidos son más altos que los costes iniciales de los ordenadores individuales. Sin embargo, los costes de creación, mantenimiento y operación de un superordenador siguen siendo mucho más altos que los de un sistema distribuido. Esto es particularmente cierto, ya que la sustitución de ordenadores individuales en un sistema distribuido se puede realizar sin un impacto significativo en el sistema en su conjunto. Además, los sistemas distribuidos ofrecen la ventaja de flexibilidad y escalabilidad, permitiendo actualizaciones y expansiones de manera gradual y económica, sin necesidad de inversiones masivas de una sola vez. La capacidad de reemplazar o actualizar componentes individuales sin detener el funcionamiento general del sistema contribuye significativamente a la reducción de costes a largo plazo.

– Mayor Fiabilidad

La mayor fiabilidad de un sistema distribuido se basa en el hecho de que toda la red de ordenadores puede seguir operando incluso cuando algunas máquinas individuales fallan. Un sistema distribuido no tiene un único punto de fallo. Si un elemento falla, los elementos restantes pueden hacerse cargo. Por lo tanto, un superordenador individual suele tener una fiabilidad menor que un sistema distribuido. Además, la capacidad de redistribuir tareas y cargas de trabajo entre los componentes sobrevivientes del sistema asegura una continuidad operativa y minimiza el impacto de fallos aislados. Esta característica inherente de los sistemas distribuidos los hace ideales para aplicaciones críticas donde la disponibilidad y la resistencia a fallos son esenciales.

– Capacidad de Crecimiento Natural

La potencia de cómputo de un sistema distribuido es el resultado de la potencia de cómputo agregada de sus componentes. Se puede aumentar la potencia de cómputo de todo el sistema conectando ordenadores adicionales al sistema. Como resultado, la potencia de cómputo del sistema completo puede incrementarse de manera incremental en una escala detallada. Esto apoya la forma en que la demanda de potencia de cómputo aumenta en muchas organizaciones. El crecimiento incremental de los sistemas distribuidos contrasta con el crecimiento de la potencia de cómputo de los ordenadores individuales. Los ordenadores individuales proporcionan una potencia idéntica hasta que son reemplazados por un ordenador más potente. Esto resulta en un crecimiento discontinuo de la potencia de cómputo, que rara vez es apreciado por los consumidores de servicios de cómputo. Además, el crecimiento escalonado de los sistemas distribuidos permite a las organizaciones adaptar sus recursos tecnológicos según sus necesidades actuales y futuras, evitando inversiones excesivas y maximizando la eficiencia operativa.

- **Desventajas:** Las desventajas de los sistemas descentralizados son:

– Sobrecarga de Coordinación

La potencia de cómputo de un sistema distribuido es el resultado de combinar la potencia de cómputo de todos los ordenadores conectados. Por lo tanto, los sistemas distribuidos típicamente tienen una potencia de cómputo superior en comparación con cada ordenador individual. Esta afirmación se ha comprobado cierta incluso al comparar sistemas distribuidos compuestos por computadoras de potencia de cómputo relativamente baja con superordenadores aislados. Esto se debe a la capacidad de los sistemas distribuidos para dividir y paralelizar tareas complejas entre múltiples nodos, aprovechando al máximo los recursos disponibles y logrando un procesamiento más eficiente y rápido. Además, esta configuración permite una escalabilidad casi ilimitada, ya que se pueden agregar más nodos al sistema para aumentar la capacidad de procesamiento según sea necesario.

– **Sobrecarga de Comunicación**

La coordinación requiere comunicación. Por lo tanto, los ordenadores que forman un sistema distribuido deben comunicarse entre sí. Esto requiere la existencia de un protocolo de comunicación y el envío, recepción y procesamiento de mensajes, lo cual, a su vez, consume esfuerzo y potencia de cómputo que no pueden dedicarse a la tarea de cómputo genuina, de ahí el término sobrecarga de comunicación. La necesidad de mantener una comunicación constante y eficiente entre los nodos implica un uso adicional de recursos, tanto en términos de ancho de banda como de capacidad de procesamiento. Esta sobrecarga puede afectar el rendimiento general del sistema si no se gestiona adecuadamente, convirtiéndose en un factor crítico en el diseño y la operación de sistemas distribuidos.

– **Dependencia de las Redes**

Cualquier tipo de comunicación requiere un medio. El medio es responsable de transferir información entre las entidades que se comunican entre sí. Los ordenadores en sistemas distribuidos se comunican mediante mensajes que se envían a través de una red. Las redes tienen sus propios desafíos y adversidades, que a su vez impactan la comunicación y coordinación entre los ordenadores que forman un sistema distribuido. Sin embargo, sin una red, no habría sistema distribuido, no habría comunicación y, por lo tanto, no habría coordinación entre los nodos, de ahí la dependencia de las redes. Esta dependencia implica que cualquier problema en la red, como latencia, pérdida de paquetes o fallos de conectividad, puede afectar significativamente el rendimiento y la fiabilidad del sistema distribuido. Por lo tanto, garantizar una infraestructura de red robusta y eficiente es crucial para el funcionamiento exitoso de estos sistemas.

– **Mayor Complejidad del Programal**

Resolver un problema de computación implica escribir programas y software. Debido a las desventajas mencionadas anteriormente, cualquier software en un sistema distribuido debe abordar problemas adicionales como la coordinación, la comunicación y el uso de redes. Esto aumenta la complejidad del software. La necesidad de gestionar la interacción entre múltiples nodos, garantizar la coherencia de los datos y manejar posibles fallos en la red añade capas adicionales de dificultad al desarrollo de software. Además, se requieren algoritmos especializados para la sincronización y la distribución de tareas, lo que complica aún más el proceso de programación y mantenimiento del sistema. Esta mayor complejidad puede resultar en un desarrollo más lento y en mayores costes de implementación y mantenimiento.

– **Problemas de Seguridad**

La comunicación a través de una red implica enviar y compartir datos que son críticos para la tarea de computación genuina. Sin embargo, enviar información a través de una red conlleva preocupaciones de seguridad, ya que entidades no confiables pueden utilizar la red para acceder y explotar la información. Por lo tanto,

cualquier sistema distribuido debe abordar los problemas de seguridad. Cuanto menos restringido sea el acceso a la red por la cual se comunican los nodos distribuidos, mayores serán las preocupaciones de seguridad para el sistema distribuido. Esto implica implementar medidas robustas de seguridad, como la encriptación de datos, autenticación de usuarios, y monitoreo constante de la red para detectar y mitigar posibles amenazas. La protección de la integridad y confidencialidad de los datos es crucial para mantener la confianza y el correcto funcionamiento del sistema distribuido.

3.1.2.2 Sistemas Distribuidos Peer-to-Peer (P2P)

Las redes P2P son un tipo especial de sistemas distribuidos. Consisten en ordenadores individuales (también llamados nodos) que ponen sus recursos computacionales (por ejemplo, potencia de procesamiento, capacidad de almacenamiento, datos o ancho de banda de red) directamente a disposición de todos los demás miembros de la red sin tener ningún punto central de coordinación. Los nodos en la red son iguales en cuanto a sus derechos y roles en el sistema. Además, todos ellos son tanto proveedores como consumidores de recursos.

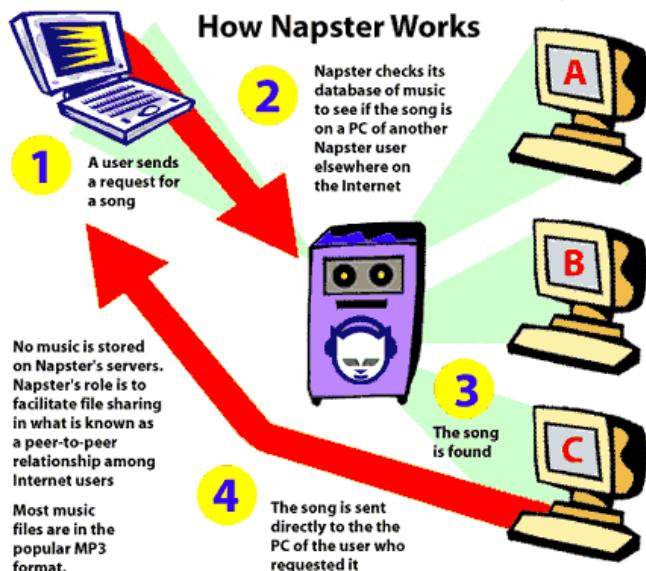


Figura 3.3: Ejemplo de red P2P (Napster) (RedesZone, 2024).

Los sistemas P2P tienen aplicaciones interesantes como el intercambio de archivos, la distribución de contenido y la protección de la privacidad. La mayoría de estas aplicaciones utilizan una idea simple pero poderosa: convertir los ordenadores de los usuarios en nodos que conforman todo el sistema distribuido. Como resultado, cuanto más usuarios o clientes utilizan el software, más grande y poderoso se vuelve el sistema.

Las redes P2P ofrecen varias ventajas, como la escalabilidad dinámica y la resistencia a fallos, ya que no dependen de un servidor central. Sin embargo, también enfrentan desafíos significativos, como la gestión eficiente de los recursos distribuidos, la seguridad de la información y la coordinación entre nodos sin un control centralizado. La capacidad de los nodos para auto-organizarse y adaptarse a las condiciones cambiantes de la red es crucial para el éxito de estos sistemas.

La relación entre los sistemas P2P puramente distribuidos y la blockchain es que los primeros utilizan la segunda como una herramienta para lograr y mantener la integridad. Por lo tanto, el argumento que explica la emoción y el potencial de la blockchain es el siguiente: Los sistemas P2P puramente distribuidos tienen un enorme potencial comercial, ya que pueden reemplazar los sistemas centralizados y cambiar industrias enteras debido a la desintermediación. Dado que los sistemas P2P puramente distribuidos pueden utilizar la blockchain para lograr y mantener la integridad, la blockchain se vuelve importante también. Sin embargo, el hecho principal que entusiasma a la gente es la desintermediación. La blockchain es solo un medio para un fin que ayuda a lograr eso.

La desintermediación se refiere a la eliminación de intermediarios en las transacciones, lo que puede reducir costos, aumentar la eficiencia y proporcionar mayor transparencia y seguridad. Esto es particularmente relevante en industrias como la financiera, donde la eliminación de intermediarios puede simplificar procesos y reducir tarifas. La blockchain, al proporcionar un registro seguro y transparente de transacciones, permite a los sistemas P2P mantener la integridad sin necesidad de un tercero confiable. Esta capacidad de operar de manera autónoma y segura sin intermediarios es lo que realmente impulsa el interés y el entusiasmo por la blockchain y los sistemas P2P.

3.1.2.3 El Propósito de la Blockchain

Al diseñar un sistema de software, se puede elegir qué estilo arquitectónico se utilizará, de manera similar a elegir un motor para un coche. La decisión arquitectónica puede tomarse independientemente de los aspectos funcionales de la capa de aplicación. Como resultado, se pueden crear sistemas distribuidos, así como sistemas centralizados, con una funcionalidad idéntica en la capa de aplicación. La arquitectura es solo un medio para un fin cuando se trata de implementar un sistema.

Cada uno de los dos conceptos arquitectónicos tiene sus propias ventajas y desventajas y su propia manera específica de hacer las cosas. Elegir una arquitectura específica tiene consecuencias en cómo se lograrán los aspectos funcionales y no funcionales de un sistema. En particular, ambos conceptos arquitectónicos tienen enfoques muy diferentes para garantizar la integridad. Y es aquí donde entra en escena la blockchain. La blockchain es una herramienta para lograr la integridad en los sistemas de software distribuidos. Por lo tanto, puede verse como una herramienta para lograr un aspecto no funcional de la capa de implementación.

El principal desafío que aborda la blockchain es lograr y mantener la integridad en un sistema peer-to-peer (P2P) puramente distribuido, que comprende un número desconocido de nodos con una fiabilidad y confiabilidad también desconocidas. Este desafío radica en la

ausencia de una autoridad central que coordine o supervise el sistema. La tarea es comparable a "pastorear gatos", donde organizar entidades independientes y descoordinadas es notoriamente difícil.

La confianza y la integridad son esenciales para el funcionamiento de los sistemas P2P. La integridad, en el contexto de los sistemas de software, se refiere a que un sistema sea seguro, completo, coherente, correcto y libre de corrupción y errores. La confianza, por otro lado, es la creencia firme de los humanos en la fiabilidad, veracidad o capacidad de alguien o algo sin necesidad de evidencia, prueba o investigación. La confianza se otorga por adelantado y aumentará o disminuirá en función de los resultados de las interacciones continuas.

En lo que respecta a los sistemas P2P, esto significa que las personas se unirán y continuarán contribuyendo a un sistema si confían en él y si los resultados de interactuar con el sistema de manera continua confirman y refuerzan su confianza. La integridad del sistema es necesaria para cumplir con las expectativas de los usuarios y reforzar su confianza en el sistema. Si la confianza de los usuarios no es reforzada por el sistema debido a una falta de integridad, los usuarios abandonarán el sistema, lo que eventualmente causará su terminación.

El problema central que la blockchain pretende resolver no es nuevo; de hecho, es un problema bien conocido y ampliamente discutido en la informática. Utilizando una metáfora del ámbito militar, el problema es conocido como el problema de los generales bizantinos. Este problema describe una situación en la que varios generales de un ejército bizantino están sitiando una ciudad y deben acordar un plan de ataque por medio de mensajes enviados a través de mensajeros, pero algunos de los generales pueden ser traidores que intentan evitar que los leales lleguen a un acuerdo.

La blockchain aborda este problema mediante el uso de algoritmos criptográficos y mecanismos de consenso que permiten a los nodos de una red distribuida llegar a un acuerdo sobre el estado del sistema de manera segura y confiable, incluso en presencia de nodos maliciosos o fallidos. Al hacerlo, la blockchain proporciona una forma de garantizar la integridad y la confianza en sistemas distribuidos donde las condiciones son inciertas y los participantes no pueden ser completamente confiables. Esta capacidad de operar de manera autónoma y segura sin intermediarios es lo que realmente impulsa el interés y el entusiasmo por la blockchain y los sistemas P2P.

La blockchain se puede definir de varias maneras: como una estructura de datos, un algoritmo, un conjunto de tecnologías o un término general para sistemas peer-to-peer puramente distribuidos. Como estructura de datos, se refiere a datos organizados en bloques conectados en una cadena. Como algoritmo, describe una secuencia de instrucciones para negociar el contenido informativo en un sistema distribuido. Como conjunto de tecnologías, combina la estructura de datos y el algoritmo con tecnologías criptográficas y de seguridad. Finalmente, como término general, se refiere a sistemas peer-to-peer de libros contables distribuidos que utilizan estas tecnologías para mantener la integridad.

3.1.3 Planificación de la Blockchain

Como punto de partida, los hechos principales sobre el sistema en consideración se pueden resumir de la siguiente manera:

El sistema será un sistema puramente distribuido peer-to-peer, compuesto por los recursos computacionales aportados por sus usuarios. Utilizará Internet como red para conectar los nodos individuales. Ni el número de nodos ni su confiabilidad y fiabilidad son conocidos. El objetivo del sistema peer-to-peer es la gestión de la propiedad de un bien digital (por ejemplo, puntos de bonificación por ventas o dinero digital). Por lo tanto las tareas claves serán:

1. Describir la Propiedad:

El primer paso en la planificación es definir claramente qué se está gestionando. Esto implica crear una representación digital precisa del bien o propiedad. La descripción debe ser lo suficientemente detallada como para que todos los participantes en la red puedan entender y reconocer el bien.

2. Proteger la Propiedad:

La seguridad es una prioridad. Se deben implementar técnicas de cifrado y autenticación para proteger la propiedad digital contra accesos no autorizados y fraudes. Esto asegura que solo los usuarios autorizados puedan acceder y transferir la propiedad, manteniendo la integridad y la confianza en el sistema.

3. Almacenar los Datos de las Transacciones:

Es crucial registrar cada transacción de manera segura y eficiente. El almacenamiento de datos debe garantizar la integridad y disponibilidad continua de la información, permitiendo que todas las transacciones sean rastreables y verificables en cualquier momento.

4. Preparar los Libros Contables para su Distribución:

En un entorno distribuido y no confiable, los libros contables deben estar preparados para ser compartidos entre todos los nodos de la red. Esto implica asegurar que los datos almacenados sean consistentes y estén verificados, permitiendo a cada nodo tener acceso a una copia precisa del libro contable.

5. Distribuir los Libros Contables:

La distribución efectiva de los libros contables es esencial. Cada nodo en la red debe tener acceso a una copia actualizada del libro contable. Esto asegura que todos los participantes tengan la misma información y puedan operar en base a datos precisos y actualizados.

6. Añadir Nuevas Transacciones a los Libros Contables:

Se deben definir mecanismos claros y seguros para añadir nuevas transacciones al libro contable. Este proceso debe garantizar que cada nueva transacción se integre sin comprometer la integridad y consistencia de los datos ya existentes.

7. Decidir Qué Libro Contable Representa la Verdad:

La planificación del blockchain implica una serie de tareas interconectadas que aseguran la creación de un sistema robusto y confiable para la gestión de la propiedad digital. Cada tarea se enfoca en aspectos críticos de seguridad, almacenamiento y distribución de datos, garantizando que el sistema pueda operar de manera efectiva en un entorno distribuido y no confiable.

3.1.3.1 Documentación de la Propiedad

El objetivo es documentar la propiedad de una manera transparente y comprensible. Cualquier persona que lea la documentación debería poder hacer una declaración inequívoca sobre la asociación de los bienes con sus propietarios.

El desafío es encontrar una forma de documentación que no solo reclame que alguien es propietario de algo, sino que también proporcione evidencia de propiedad y, por lo tanto, sirva como prueba de propiedad.

En lugar de describir el estado actual de la propiedad mediante datos de inventario (es decir, listando las posesiones actuales de todos los propietarios), se mantiene una lista de todas las transferencias de propiedad en un libro contable de manera continua. Cada transferencia de propiedad se describe con datos de transacción que indican claramente qué propietario transfiere la propiedad de qué ítem y a quién en qué momento. Toda la historia de datos de transacciones almacenada en un libro contable se convierte en una pista de auditoría que proporciona evidencia de cómo cada uno obtuvo su posesión. Por lo que las tareas claves serán:

1. Mantener un Historial de Transferencias:

- Utilizar un libro contable que registre todas las transferencias de propiedad.
- Cada transferencia se describe con datos específicos que incluyen los identificadores de las cuentas involucradas, la cantidad de bienes transferidos, el momento de la transacción, la tarifa pagada y la prueba de autorización del propietario.

2. Documentar la Transferencia de Propiedad:

Los datos necesarios para ejecutar una transferencia incluyen:

- Identificador de la cuenta que transfiere la propiedad.
- Identificador de la cuenta que recibe la propiedad.
- Cantidad de bienes a transferir.
- Momento de la transacción.
- Tarifa pagada al sistema.
- Prueba de autorización del propietario de la cuenta que transfiere.

3. Asegurar la Integridad del Historial:

- Mantener el orden de las transacciones en el historial es crucial para garantizar la coherencia y la integridad del sistema.

- Resolver discrepancias entre diferentes versiones del historial para determinar la versión verdadera y aceptada de las transacciones.

4. Garantizar la Autorización de las Transacciones:

- Cada transacción debe incluir prueba de que el propietario de la cuenta que transfiere la propiedad ha autorizado la transferencia.
- Proteger el sistema contra transacciones no autorizadas mediante técnicas de autenticación y cifrado.

Obtenemos como conclusión que la documentación de la propiedad en blockchain se centra en mantener un registro completo y ordenado de todas las transacciones de propiedad, asegurando que cada transferencia esté autorizada y que el historial de transacciones se mantenga íntegro y verificable.

3.1.4 Fundamentos Criptográficos de la Blockchain

La criptografía se basa en la idea fundamental de proteger los datos contra el acceso no autorizado. En el mundo digital, actúa de manera similar a las cerraduras de las puertas o las cajas fuertes de los bancos, las cuales protegen su contenido de personas no autorizadas. Así como las cerraduras y llaves físicas garantizan la seguridad en el mundo tangible, la criptografía utiliza claves para proteger la información digital.

El equivalente digital a cerrar una cerradura es el cifrado, mientras que el equivalente digital a abrir una cerradura es el descifrado. Por lo tanto, cuando hablamos de proteger datos utilizando criptografía, usamos los términos cifrado y descifrado para referirnos a la protección y desprotección de datos, respectivamente. Los datos cifrados se llaman texto cifrado. Para cualquiera que no sepa cómo descifrarlo, el texto cifrado parece una serie de letras y números sin sentido. Sin embargo, el texto cifrado es útil, pero solo para aquellos que poseen la clave necesaria para descifrarlo. El texto cifrado descifrado es idéntico a los datos originales que fueron cifrados. Así, el proceso completo de la criptografía puede resumirse como: comenzar con algunos datos, producir texto cifrado cifrando los datos originales con una clave criptográfica, preservar el texto cifrado o enviarlo a alguien, y finalmente recuperar los datos originales descifrando el texto cifrado con una clave criptográfica.

En criptografía, el cifrado transforma los datos originales en texto cifrado mediante el uso de una clave. Este proceso asegura que los datos solo puedan ser leídos por quienes tienen la clave correcta para descifrarlos. El descifrado, por su parte, convierte el texto cifrado de nuevo en su forma original utilizando la misma clave (en criptografía simétrica) o una clave diferente pero relacionada (en criptografía asimétrica).

El texto cifrado, aunque ininteligible para quienes no tienen la clave, mantiene su utilidad y valor, ya que puede ser transportado o almacenado de manera segura hasta que llegue a su destinatario legítimo. Una vez que el destinatario recibe el texto cifrado, puede utilizar la clave adecuada para descifrarlo y acceder a la información original, asegurando que los datos hayan permanecido secretos y sin alteraciones durante su transmisión o almacenamiento.



Figura 3.4: Ilustración esquemática de los conceptos criptográficos básicos y su terminología.

¿Qué ocurre si alguien intenta descifrar un texto cifrado utilizando una clave incorrecta? El resultado es un montón inútil de números, letras y signos que no revelan ninguno de los datos que se cifraron.

3.1.4.1 Fundamentos de las funciones Hash

Las huellas digitales humanas son impresiones únicas de los surcos de fricción en los dedos, utilizadas históricamente para identificar individuos de manera precisa. Se emplean en investigaciones criminales, identificación de delincuentes y exoneración de inocentes. Este concepto se traslada al ámbito digital a través de los valores hash criptográficos.

El valor hash criptográfico actúa como la huella digital de los datos, proporcionando una identificación única. La blockchain utiliza extensivamente este concepto para mantener la integridad y seguridad de los datos. Entender el hashing criptográfico es, por lo tanto, esencial para comprender cómo funciona la blockchain y asegurar que los datos sean fiables y no manipulables.

El objetivo principal del uso de funciones hash en blockchain es transformar cualquier tipo de dato en un número de longitud fija, sin importar el tamaño de los datos originales. Esto permite identificar y comparar datos de manera única y rápida, asegurando tanto la integridad como la seguridad en el sistema blockchain.

En un sistema distribuido P2P, donde se maneja una enorme cantidad de datos de transacciones, es crucial poder identificar estos datos de manera única y compararlos de forma eficiente. Así, el objetivo es utilizar huellas digitales (hashes) para identificar datos de transacciones, y potencialmente cualquier tipo de dato, de forma única.

Las funciones hash son algoritmos que convierten cualquier tipo de dato en un número de longitud fija, sin importar el tamaño de los datos de entrada. Estas funciones procesan una pieza de datos a la vez, generando un valor hash basado en los bits y bytes que componen la información original. Para cumplir con la longitud requerida, los valores hash pueden incluir ceros al inicio.

Existen numerosas funciones hash, cada una con características específicas, especialmente en cuanto a la longitud del valor hash que producen. Un subgrupo crucial dentro de estas funciones son las funciones hash criptográficas, diseñadas para generar huellas digitales únicas para cualquier tipo de dato.

Estas huellas digitales son fundamentales para garantizar la integridad y seguridad de los datos en diversas aplicaciones, especialmente en blockchain. Las funciones hash criptográficas tienen las siguientes propiedades:

- **Determinismo**

La función hash produce valores hash idénticos para datos de entrada idénticos. Cualquier discrepancia observada en los valores hash de los datos debe ser causada únicamente por discrepancias en los datos de entrada y no por el funcionamiento interno de la función hash.

- **Pseudoaleatorio**

Indica que el valor hash devuelto por una función hash cambia de manera impredecible cuando los datos de entrada se modifican. Incluso si los datos de entrada cambian mínimamente, el valor hash resultante será completamente diferente de manera impredecible. En otras palabras, el valor hash de los datos modificados siempre debe ser una sorpresa y no debe ser posible predecir el valor hash basado en los datos de entrada.

- **Función Unidireccional**

Una función unidireccional no permite rastrear los valores de entrada a partir de sus salidas. Esto significa que no se puede utilizar de manera inversa, es decir, es imposible recuperar los datos originales basándose en el valor hash. Los valores hash no proporcionan información sobre el contenido de los datos de entrada, de la misma manera que una huella dactilar aislada no proporciona información sobre la persona cuya huella se ha tomado. Las funciones unidireccionales también se consideran no invertibles.

- **Resistencia a Colisiones**

Una función hash se considera resistente a colisiones si es extremadamente difícil encontrar dos o más conjuntos de datos distintos que produzcan el mismo valor hash. En otras palabras, si la probabilidad de obtener un valor hash idéntico para diferentes conjuntos de datos es muy baja, entonces la función hash es resistente a colisiones. En este caso, se puede considerar que los valores hash creados por la función son únicos y, por lo tanto, utilizables para identificar datos. Si se obtiene un valor hash idéntico para diferentes conjuntos de datos, se enfrenta a una colisión de hash, que es el equivalente digital de tener dos personas con huellas dactilares idénticas.

Ser resistente a colisiones es fundamental para que los valores hash sean utilizables como huellas digitales. Aunque el funcionamiento interno de las funciones hash resistentes a colisiones está fuera del alcance de este documento, se puede asegurar que se ha invertido un gran esfuerzo en reducir el riesgo de producir colisiones de hash.

En la página de Blockchain Basics (2016) podemos ver un ejemplo claro de como utilizar las funciones hash:

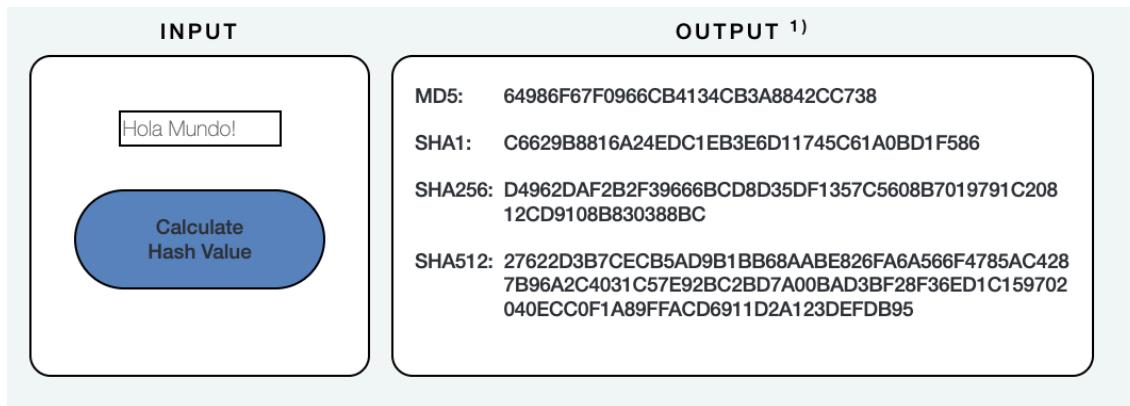


Figura 3.5: Cálculo de los valores hash de un texto.

Al hacer clic en el botón, el cuadro de salida a la derecha mostrará el valor hash del texto de entrada calculado utilizando cuatro funciones hash diferentes. Los valores hash se consideran a menudo como números hash, ya que utilizan no solo los dígitos del 0 al 9, sino también las letras de la A a la F para representar valores numéricos del 10 al 15. Estos números se denominan números hexadecimales y son ampliamente utilizados por los informáticos por diversas razones, las cuales no se discutirán aquí. Es importante destacar que los valores hash resultantes difieren debido a las distintas implementaciones de las funciones hash que los generan. Estos detalles se aceptan como dados, ya que no se pretende profundizar en el extenso tema de la implementación de las funciones hash en este contexto.

Las funciones hash aceptan un dato cada vez. No acepta un montón de datos independientes a la vez, pero, en realidad, a menudo necesitamos un único valor hash para una gran colección de datos. En concreto, la infraestructura de la cadena de bloques tiene que gestionar muchos datos de transacciones a la vez y necesita un único valor hash para todos ellos. Para esto se aplican diferentes patrones a las funciones hash:

- **Hashing independiente:** Aplicar la función hash a cada pieza de datos de manera independiente. Este método asegura que cada dato tenga su propio valor hash único sin depender de otros datos. En la figura 3.6 los recuadros blancos que contienen una palabra cada uno representan los datos que hay que hashear y los círculos grises muestran los valores hash correspondientes. Las flechas que apuntan de las cajas a los círculos ilustran esquemáticamente la transformación de los datos en valores hash.

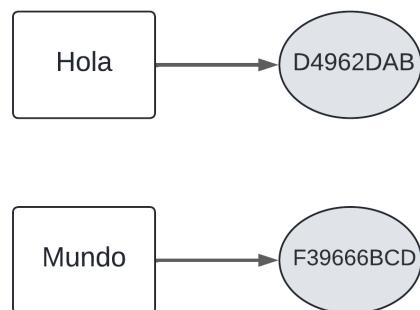


Figura 3.6: Ilustración esquemática del hash de diferentes datos de forma independiente.

- **Hashing repetitivo:** Aplicar repetidamente una función hash a su propio resultado. Este método puede aumentar la complejidad de descifrar los datos, añadiendo una capa extra de seguridad.



Figura 3.7: Cálculo de valores hash repetidamente.

- **Hashing combinado:** El hashing combinado tiene como objetivo generar un único valor hash para varias piezas de datos en una sola operación. Esto se logra combinando todas las piezas de datos independientes en una sola y calculando su valor hash a partir de esta combinación. Esta técnica es especialmente útil cuando se necesita obtener un valor hash único para un conjunto de datos disponibles en un momento determinado.

No obstante, el hashing combinado presenta algunas desventajas. La combinación de datos requiere una cantidad significativa de poder de cómputo, tiempo y espacio de memoria, por lo que su uso debe limitarse a cuando las piezas de datos individuales son pequeñas. Además, una limitación importante es que los valores hash de las piezas individuales no se pueden obtener, ya que solo se calcula el hash de los datos combinados.

La figura 3.8 muestra el concepto de hashing combinado. Las palabras individuales se combinan primero en una palabra con un espacio de letras entre ellas y la frase resultante se somete a hash después. El valor hash resultante mostrado es idéntico al primer valor hash de la figura 3.7.

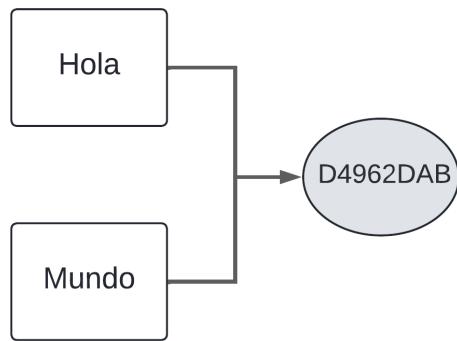


Figura 3.8: Combinación de datos y cálculo del valor hash.

- **Hashing secuencial:** Actualizar incrementalmente un valor hash a medida que llegan nuevos datos, combinando el valor hash existente con los nuevos datos. Este método es útil para procesar flujos de datos en tiempo real.

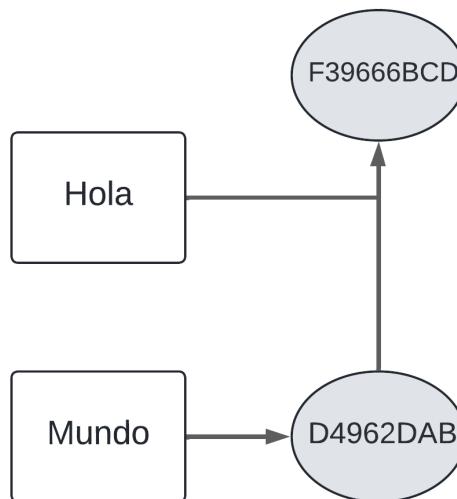


Figura 3.9: Cálculo de los valores hash de un texto de manera secuencial.

- **Hashing jerárquico:** Crear una jerarquía de valores hash aplicando hashing combinando a pares de valores hash, formando una estructura de árbol con un único valor hash en la cima. Este método, conocido como árbol de *Merkle*, se utiliza ampliamente en blockchain para verificar grandes volúmenes de datos de manera eficiente.

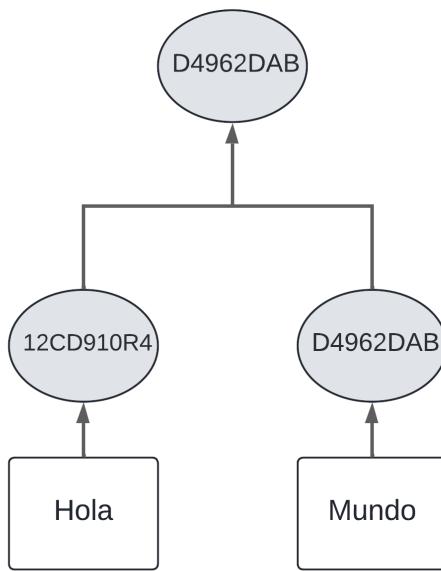


Figura 3.10: Cálculo de los valores hash de un texto jerarquicamente.

3.1.4.2 Puzzles Hash

Los valores hash no solo son útiles para operaciones básicas como comparar, referenciar y almacenar datos de manera segura y eficiente. También pueden ser utilizados para permitir que los ordenadores desafíen a otros ordenadores con acertijos complejos. Aunque pueda parecer inusual, este uso de los valores hash es esencial en la tecnología blockchain.

En algunas situaciones, es necesario crear acertijos que requieran una cantidad significativa de recursos computacionales para ser resueltos. Estos acertijos no deben poder resolverse mediante conocimientos previos o razonamiento lógico, como en una prueba de inteligencia o de conocimientos. La única forma de resolver estos acertijos es mediante la pura potencia de cálculo y trabajo computacional arduo.

Un candado de combinación es un tipo de cerradura que necesita una secuencia específica de números para ser abierto. Si no se conoce la secuencia, se probarían todas las combinaciones posibles hasta encontrar la correcta. Este proceso garantiza que el candado se abrirá, pero requiere mucho tiempo. Probar todas las combinaciones posibles no depende del conocimiento o razonamiento, sino de la diligencia y el esfuerzo. Los acertijos hash son similares a

esta tarea, pero en el ámbito digital, donde se deben probar numerosas combinaciones hasta encontrar la correcta.

Los acertijos hash solo pueden ser resueltos mediante prueba y error. Esto requiere que se adivine un número aleatorio (nonce), se calcule el valor hash de los datos combinados con la función hash requerida y se evalúe el valor hash resultante según las restricciones establecidas. Si las restricciones son cumplidas por el valor hash, el acertijo hash habrá sido resuelto; de lo contrario, se continuará con otro nonce hasta que eventualmente se resuelva el acertijo. El nonce que, al combinarse con los datos dados, produce un valor hash que cumple con las restricciones es llamado la solución. Ese nonce particular siempre debe ser presentado al afirmar que se ha resuelto un acertijo hash.

En el ejemplo de la figura 3.10 hecho en la web de Blockchain Basics (2016) podemos ver de manera sencilla el funcionamiento. Al introducir los datos y resolver el puzzle, nos devuelve un output donde las función hash con dos ceros delante (*Leading zeros* en la web) es la más corta. *Nonce* nos indica los pasos que se han dado para llegar a la solución.

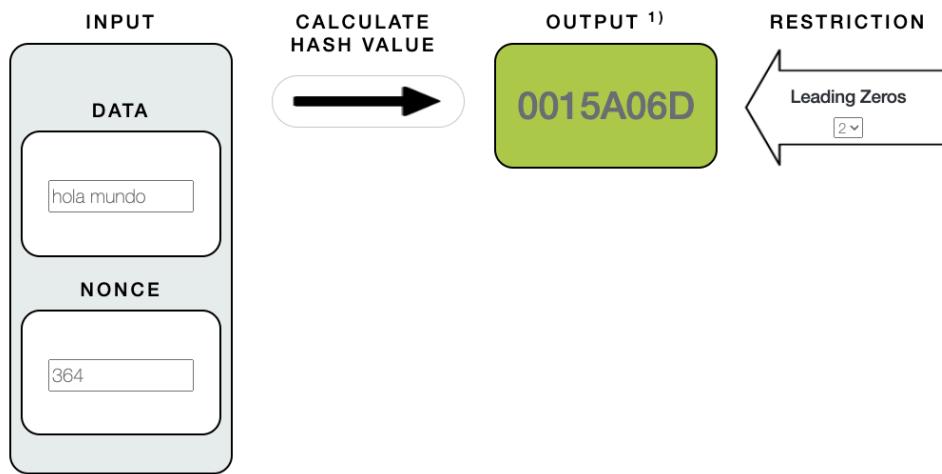


Figura 3.11: Ejemplo de puzzle hash.

Requerir que el valor hash cumpla con una cierta restricción es el núcleo del acertijo hash. Por lo tanto, ni la restricción ni su descripción son arbitrarias. En su lugar, la restricción utilizada por los acertijos hash está estandarizada para que los ordenadores puedan desafiar a otros ordenadores con estos acertijos. En el contexto de los acertijos hash, las restricciones a menudo se llaman dificultad o nivel de dificultad.

La dificultad se expresa como un número natural y se refiere al número de ceros iniciales que debe tener el valor hash. Por ejemplo, una dificultad de 1 significa que el valor hash debe tener (al menos) un cero inicial, mientras que una dificultad de 10 significa que el valor hash debe tener al menos 10 ceros iniciales. Cuanto mayor sea el nivel de dificultad, más ceros iniciales se requieren y más complicado será el acertijo hash. Cuanto más complicado sea el

acertijo hash, más potencia computacional o tiempo se necesitarán para resolverlo.

Los acertijos hash dependen de que las funciones hash sean unidireccionales, lo que impide resolverlos yendo del resultado deseado a la entrada necesaria. Estos acertijos solo se pueden resolver mediante prueba y error, consumiendo mucha potencia de cálculo y tiempo. El nivel de dificultad determina cuántos intentos se necesitan en promedio para encontrar la solución, afectando los recursos computacionales y el tiempo requerido. Las funciones hash son deterministas y rápidas, facilitando la verificación de la solución una vez encontrada. Si el valor hash calculado no cumple las restricciones, no es culpa de la función hash, sino simplemente porque el acertijo no ha sido resuelto aún.

Los usos dentro de la blockchain son:

- **Almacenamiento de datos de transacciones de manera sensible a los cambios:** El hashing asegura que cualquier modificación en los datos de transacción resulte en un cambio significativo en el valor hash, lo que permite detectar alteraciones.
- **Como una huella digital de los datos de transacción:** Los valores hash actúan como identificadores únicos de los datos de transacción, garantizando su integridad y autenticidad.
- **Como una forma de generar gastos computacionales para cambiar la estructura de datos de la blockchain:** El hashing se utiliza para introducir un gasto computacional significativo, haciendo que las modificaciones en la estructura de datos de la blockchain sean computacionalmente costosas y, por tanto, desalentando alteraciones no autorizadas.

3.1.4.3 Criptografía Asimétrica.

La blockchain utiliza la criptografía asimétrica para lograr dos objetivos principales:

- **Identificación de usuarios:** La blockchain necesita identificar a los usuarios o cuentas de usuario para mantener la relación entre el propietario y su propiedad. Para esto, la blockchain utiliza el enfoque de criptografía asimétrica de público a privado para identificar cuentas de usuario y transferir la propiedad entre ellas.

Los números de cuenta en la blockchain son, en realidad, claves criptográficas públicas. Por lo tanto, los datos de las transacciones utilizan estas claves criptográficas públicas para identificar las cuentas involucradas en la transferencia de propiedad. En este sentido, la blockchain trata a las cuentas de usuario de manera similar a buzones de correo: tienen una dirección públicamente conocida y cualquiera puede enviarles mensajes.

- **Autorizar transacciones:** Los datos de las transacciones deben incluir siempre un elemento que sirva como prueba de que el propietario de la cuenta que transfiere la propiedad está de acuerdo con la transferencia descrita. El flujo de información implicado en este acuerdo comienza en el propietario de la cuenta que transfiere la propiedad y debe llegar a todos los que inspeccionen los datos de la transacción. Este tipo de flujo de

información es similar al caso de uso de la criptografía asimétrica de privado a público.

El propietario de la cuenta que transfiere la propiedad crea un texto cifrado con su clave privada. Todos los demás pueden verificar esta prueba de acuerdo utilizando la clave criptográfica pública, que corresponde al número de la cuenta que transfiere la propiedad. De esta manera, se asegura que solo el propietario legítimo pueda autorizar la transferencia y que cualquier persona pueda verificar la validez de la autorización sin comprometer la seguridad de la clave privada. Este procedimiento es conocido como firma digital.

Debemos entender que existen dos claves o llaves: una clave amarilla y una clave negra. Juntas forman el par de claves correspondientes. El mensaje original se cifra con la clave negra, lo que produce un texto cifrado representado por una caja negra con letras blancas. El mensaje original también se puede cifrar con la segunda clave, lo que produce un texto cifrado diferente representado por una caja blanca con letras negras. Los colores de las cajas que representan el texto cifrado y los colores de las claves utilizadas para producirlos son idénticos para resaltar su relación: la clave negra produce texto cifrado negro, mientras que la clave blanca produce texto cifrado blanco.

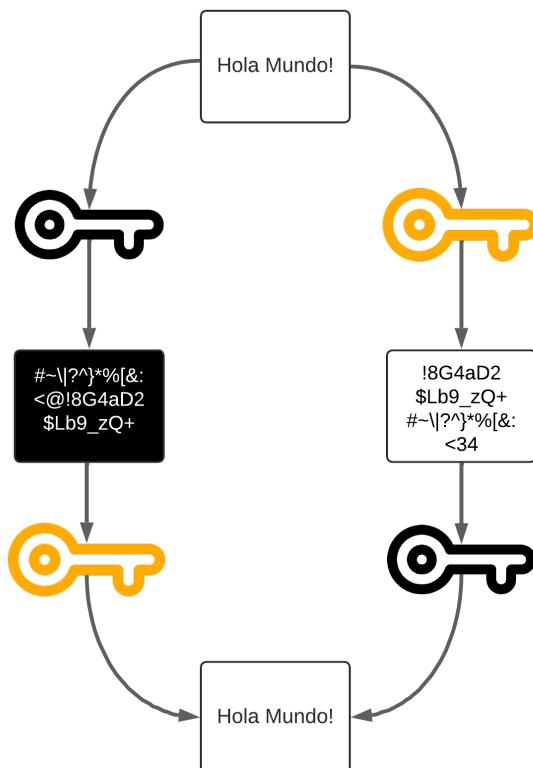


Figura 3.12: Funcionamiento de la criptografía asimétrica.

El truco de la criptografía asimétrica es que nunca se puede descifrar el texto cifrado con la clave que se utilizó para crearlo. La decisión sobre qué clave usar para el cifrado y cuál para el descifrado depende de ti. Puedes intercambiar los roles de las claves como deseas para cada nuevo conjunto de datos que quieras cifrar, pero siempre debes conservar ambas claves para realizar tanto el cifrado como el descifrado. Si solo tienes una de las claves, tu capacidad es limitada. Aunque siempre puedes crear texto cifrado aplicando tu clave a los datos, no puedes descifrarlo porque te falta la clave complementaria. Sin embargo, puedes descifrar el texto cifrado que fue creado con la clave complementaria correspondiente. Una clave aislada es como una calle de sentido único: puedes recorrerla, pero nunca puedes regresar por la misma calle.

La criptografía asimétrica se utiliza en la blockchain para firmar digitalmente mensajes y verificar su autenticidad. Al enviar un mensaje "¡Hola Mundo!", se genera un valor hash del saludo, que luego se cifra con una clave privada, creando una firma digital única. Este mensaje, junto con su firma digital, se envía al mundo, y cualquier receptor puede verificar su autenticidad calculando el valor hash del saludo y descifrando la firma digital con la clave pública correspondiente. Si los valores hash coinciden, se confirma que el mensaje es auténtico y no ha sido alterado. Si el mensaje es modificado, los valores hash no coincidirán, indicando que el mensaje no fue autorizado por el remitente original, protegiendo así la integridad y autenticidad del contenido.

3.1.5 Estructura de Datos de la Blockchain

Con los conceptos vistos podemos entender la estructura de datos de la blockchain. La estructura de datos de la blockchain se caracteriza por ser un registro seguro y ordenado de transacciones distribuidas a través de una red de nodos. Esta estructura está compuesta por bloques, donde cada bloque contiene un conjunto de transacciones y un encabezado.

- **Bloques:** Los bloques son las unidades básicas de la blockchain, cada uno de los cuales contiene un conjunto de transacciones. Un bloque típicamente contiene un encabezado y un cuerpo. El cuerpo incluye las transacciones, mientras que el encabezado contiene metadatos importantes para la integridad y el enlace de la cadena. Los bloques son generados por los mineros y, una vez verificados, se agregan de manera inmutable a la cadena, formando un registro permanente y auditado de todas las transacciones.
- **Encabezado de Bloques:** El encabezado del bloque incluye varios campos críticos. El hash del bloque previo es un hash del encabezado del bloque anterior en la cadena, creando un enlace criptográfico entre los bloques. Esta referencia asegura que cualquier cambio en un bloque alterará todos los bloques posteriores, manteniendo así la integridad de la cadena. La raíz del árbol de Merkle es un hash que resume todas las transacciones en el bloque, permitiendo verificar que las transacciones no han sido alteradas sin necesidad de verificar cada una individualmente. El nonce es un número aleatorio utilizado solo una vez que los mineros deben encontrar para que, combinado con los otros datos del encabezado, produzca un hash con ciertas propiedades, como un número específico de ceros iniciales. La dificultad es un parámetro que define cuán

difícil es encontrar un nonce válido, ajustándose periódicamente para mantener el tiempo medio de creación de bloques constante. Finalmente, la marca de tiempo registra el momento en que el bloque fue creado, ayudando a mantener un orden cronológico de los bloques. Además, el encabezado puede incluir versiones de software y otros parámetros de red que aseguran la compatibilidad y actualización de la red blockchain.

- **Árbol de Merkle:** Un árbol de Merkle es una estructura de datos que permite la verificación eficiente y segura del contenido de grandes conjuntos de datos. Cada hoja del árbol es un hash de una transacción, mientras que los nodos padres son hashes de sus nodos hijos, culminando en la raíz del árbol, que es un resumen de todas las transacciones del bloque. Esto permite verificar que una transacción está incluida en un bloque sin necesidad de verificar todas las transacciones del bloque, mejorando la eficiencia. Los árboles de Merkle son fundamentales para las auditorías rápidas y la sincronización de datos en la red blockchain, permitiendo a los nodos verificar partes del bloque sin tener que descargar el bloque completo.

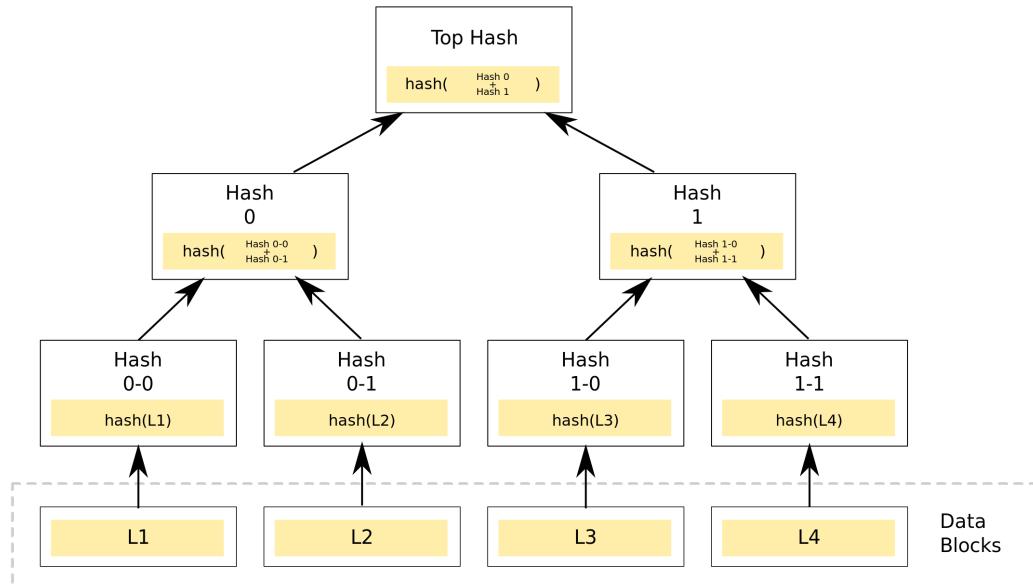


Figura 3.13: Ejemplo de árbol de Merkle (Wikipedia, 2022).

- **Proceso de Minería:** La minería es el proceso de agregar nuevos bloques a la blockchain. Los mineros compiten para resolver un problema criptográfico que implica encontrar un nonce que haga que el hash del encabezado del bloque cumpla con los criterios de dificultad. Este proceso, PoW, asegura que la creación de cada nuevo bloque requiere un esfuerzo computacional significativo, lo que previene ataques de spam y asegura que los bloques se creen de manera ordenada y descentralizada. Los mineros son recompensados con nuevas monedas (recompensa de bloque) y/o tarifas de transacción por su trabajo, incentivando la participación en el mantenimiento de la red. La minería

también incluye la validación de transacciones y la propagación de nuevos bloques a través de la red.

- **Cadena de Bloques:** La blockchain es una cadena continua de bloques donde cada nuevo bloque se agrega al final. Cada bloque está ligado al anterior mediante el hash del bloque previo, creando una estructura segura y lineal que es resistente a la manipulación. El último bloque agregado se conoce como la "cabeza" de la cadena, y es a partir de este punto que se agregan nuevos bloques. La longitud y la integridad de la cadena son verificadas constantemente por los nodos de la red, asegurando que todos los participantes tengan una copia sincronizada y actualizada de la blockchain.
- **Sensibilidad a los Cambios:** Cualquier cambio en los datos de un bloque altera su hash, lo que afecta todos los bloques posteriores. Esta característica asegura que los datos en la blockchain sean inmutables, ya que modificar un bloque requeriría recalcular los hashes de todos los bloques posteriores, una tarea extremadamente difícil y costosa. Esta propiedad de inmutabilidad es lo que hace que la blockchain sea una tecnología confiable para aplicaciones que requieren un registro histórico preciso y seguro, como la trazabilidad de productos, registros financieros y contratos inteligentes.
- **Distribución en Red:** La blockchain se mantiene a través de una red de nodos distribuidos que colaboran en la validación y propagación de bloques. Cada nodo tiene una copia completa de la blockchain, lo que asegura que no haya un punto central de control o falla. La distribución de la blockchain en múltiples nodos hace que la red sea resistente a ataques y fallos, ya que comprometer la integridad de la cadena requeriría comprometer una mayoría de los nodos en la red. Los nodos también participan en el consenso, un mecanismo que asegura que todos los nodos acuerden el estado actual de la blockchain, previniendo bifurcaciones y asegurando la coherencia de los datos a través de la red.

3.1.6 Aplicaciones de la Blockchain

La blockchain es agnóstica respecto a los datos que almacena, lo que significa que puede almacenar cualquier tipo de datos. Por lo tanto, el rango de datos almacenados y las áreas de aplicación de la blockchain son tan amplias y diversas como las actividades humanas mismas. A continuación se presenta una selección de áreas de aplicación concretas en las que la blockchain ya se utiliza o puede utilizarse próximamente:

- **Pagos:** Gestión de la propiedad y transferencia de monedas fiduciarias digitales. La blockchain permite transacciones rápidas y seguras, reduciendo los costos asociados con los métodos de transferencia tradicionales.
- **Criptomonedas:** Gestión de la propiedad y creación de instrumentos de pago digitales que existen independientemente de cualquier gobierno, banco central u otra institución central. Ejemplos incluyen Bitcoin y Ethereum.
- **Micropagos:** Transferencia de pequeñas cantidades de dinero que serían demasiado costosas utilizando medios de transferencia tradicionales. La blockchain facilita estas transacciones al reducir significativamente los costos de transacción.

- **Activos Digitales:** Gestión de la creación, propiedad y transferencia de elementos digitales que tienen valor intrínseco o representan bienes valiosos en el mundo real. Esto incluye tokens no fungibles (NFTs) y otros activos digitales.
- **Identidad Digital:** Proporcionar pruebas de identidad y autenticación basadas en elementos digitales únicos. La blockchain puede asegurar identidades digitales inmutables y verificables. En los artículos *Addressing the Challenges of Biological Passport Through Blockchain Technology* (Osuna y cols., 2023) y *Corporate Digital Identity Based on Blockchain* (Balastegui-García y cols., 2023) se demuestra que la tecnología blockchain puede proporcionar una solución robusta y segura para los desafíos asociados con los pasaportes biológicos, mejorando la integridad, la transparencia y la privacidad de los datos.
- **Servicios de Notaría:** Digitalización, almacenamiento y verificación de documentos o contratos y pruebas de propiedad o transferencia. La blockchain puede registrar de manera segura transacciones y contratos, proporcionando una prueba inmutable de la existencia y el estado de los documentos.
- **Cumplimiento y Auditoría:** Auditoría de actividades empresariales de personas u organizaciones en industrias reguladas en un seguimiento de auditoría. La transparencia y la inmutabilidad de la blockchain facilitan la supervisión y el cumplimiento normativo.
- **Impuestos:** Cálculo y recaudación de impuestos basados en transacciones o en la mera propiedad, reduciendo la evasión fiscal o la doble tributación. La blockchain puede automatizar el proceso de recolección de impuestos y asegurar su precisión.
- **Votación:** Creación, distribución y conteo de papeletas de voto digitales. La blockchain puede proporcionar un sistema de votación seguro y transparente, garantizando la integridad del proceso electoral. El artículo *Practical I-Voting on Stellar Blockchain* (Barański y cols., 2020) proporciona una solución innovadora y práctica para la votación electrónica, utilizando tecnologías avanzadas para asegurar la integridad y privacidad del proceso electoral.
- **Gestión de Registros:** Creación y almacenamiento de registros médicos. La blockchain puede asegurar la privacidad y la integridad de los registros médicos, facilitando el acceso y la gestión de la información de salud.
- **Otros Casos de Uso:** El artículo *The Case of rWallet: A Blockchain-Based Tool to Navigate Some Challenges Related to Irregular Migration* (Visvizi y cols., 2023) presenta una herramienta basada en blockchain llamada rWallet. Esta herramienta se propone como una solución para abordar varios desafíos relacionados con la migración irregular. El artículo *Blockchain-Based Framework for Traffic Event Verification in Smart Vehicles* (Pujol y cols., 2024) presenta un marco basado en blockchain para la verificación de eventos de tráfico en vehículos inteligentes. Este marco tiene como objetivo asegurar la integridad y autenticidad de los mensajes de eventos reportados por los vehículos en un entorno de ciudad inteligente.

Estas aplicaciones demuestran la versatilidad y el potencial transformador de la tecnología blockchain en diversos sectores. Desde la simplificación de transacciones financieras hasta la

protección de identidades digitales y la mejora de procesos electorales, la blockchain está posicionada para impactar significativamente en múltiples aspectos de la vida diaria y los negocios.

3.2 Smart Contracts

Nick Szabo fue de los primeros en definir este término.

Smart Contract: Un contrato inteligente es un protocolo de transacción informatizado que ejecuta los términos de un contrato. Los objetivos generales del diseño de contratos inteligentes son satisfacer las condiciones contractuales habituales (como condiciones de pago, gravámenes, confidencialidad e incluso ejecución), minimizar las excepciones tanto maliciosas como accidentales y reducir al mínimo la necesidad de intermediarios de confianza. Los objetivos económicos relacionados incluyen la reducción de las pérdidas por fraude, los costes de arbitraje y ejecución, y otros costes de transacción (Szabo, 1994).

Se consideran los protocolos de efectivo digital como ejemplos claros de contratos inteligentes. Permiten realizar pagos en línea mientras mantienen las características deseadas del dinero en efectivo físico: que no se pueda falsificar, que sea confidencial y divisible. Al observar de nuevo los protocolos de efectivo digital dentro del contexto más amplio del diseño de contratos inteligentes, se puede ver que estos protocolos pueden usarse para implementar una gran variedad de valores electrónicos al portador, no solo dinero en efectivo. También se reconoce que, para llevar a cabo una transacción completa entre cliente y vendedor, se necesita más que solo el protocolo de efectivo digital; se requiere un protocolo que garantice la entrega del producto si se realiza el pago, y viceversa.

Los sistemas comerciales actuales utilizan diversas técnicas para lograr esto, como el correo certificado, el intercambio cara a cara, la dependencia de la historia crediticia y las agencias de cobro para extender crédito, etc. Los contratos inteligentes tienen el potencial de reducir significativamente el fraude y los costos de ejecución de muchas transacciones comerciales. Los protocolos de efectivo digital emplean varios de los nuevos elementos que surgen de los campos de la criptografía y la informática. Aunque la mayoría de estos componentes aún no se han utilizado ampliamente para facilitar acuerdos contractuales, su potencial es enorme. Estos subprotocolos incluyen el acuerdo bizantino, la encriptación simétrica y asimétrica, las firmas digitales, las firmas ciegas, el método de cortar y elegir, el compromiso de bits, los cálculos seguros multipartitos, el intercambio secreto, la transferencia sin conocimiento y el cálculo seguro multipartito (Szabo, 1994).

Se podría notar que los objetivos de los contratos inteligentes, como la visibilidad, la ejecutabilidad en línea, la verificabilidad y la privacidad, resultan en dos direcciones opuestas. La privacidad ejerce una fuerza de control sobre los contratos, buscando minimizar la exposición a partes externas. En contraposición, los otros tres objetivos, visibilidad, ejecutabilidad y verificabilidad, requieren que se entregue acceso a los datos contractuales a los participantes o auditores. Por lo tanto, se debe encontrar un punto óptimo donde se proporcione la menor

cantidad de información y control posible a las partes externas, pero que aún permita la posibilidad de verificar, observar y hacer cumplir el contrato.

La primera blockchain en utilizar smart contracts fue Ethereum. Ethereum fue conceptualizada por Vitalik Buterin a finales de 2013 y su red principal se lanzó oficialmente el 30 de julio de 2015. Fue diseñada específicamente para soportar smart contracts, permitiendo a los desarrolladores crear y desplegar aplicaciones descentralizadas (dApps). La plataforma Ethereum es una blockchain general que incluye una máquina virtual (Ethereum Virtual Machine, EVM) para ejecutar contratos inteligentes. Dado que el entorno existe únicamente en la blockchain en forma de una máquina virtual, los contratos inteligentes están completamente aislados de la red, el sistema de archivos u otros procesos en las máquinas de los nodos. Para escribir contratos inteligentes en Ethereum, se creó un lenguaje de alto nivel y completo en Turing llamado Solidity. Este lenguaje, aunque similar en sintaxis a JavaScript, se escribe en un estilo completamente diferente. Una vez que un contrato se ha escrito en Solidity, se compila en bytecode de EVM y luego se despliega en una dirección específica de Ethereum. Para desplegar e interactuar con contratos inteligentes en Ethereum, se utiliza una biblioteca RPC de JavaScript especial junto con una API web.

El desarrollo de contratos inteligentes comenzó con Ethereum y Solidity, y aún es una disciplina en desarrollo. El lenguaje Solidity tiene una serie de peculiaridades conocidas y una lista de cambios futuros, lo que significa que el código escrito actualmente puede no ser completamente funcional con la próxima actualización. A pesar de su uso relativamente reciente, se han recopilado algunas prácticas recomendadas específicas para el desarrollo de contratos inteligentes en Solidity.

3.2.1 Herramientas Para el Desarrollo Blockchain

3.2.1.1 Solidity

Solidity es un lenguaje de programación de alto nivel diseñado específicamente para escribir contratos inteligentes que se ejecutan en la Ethereum Virtual Machine (EVM). Desarrollado por los colaboradores del proyecto Ethereum, Solidity se lanzó oficialmente en 2014 y ha sido el lenguaje principal para el desarrollo de aplicaciones descentralizadas (dApps) y contratos inteligentes en la blockchain de Ethereum. Influenciado por lenguajes como JavaScript, Python y C++, Solidity es conocido por su sintaxis accesible y su capacidad para manejar tareas complejas de manera eficiente en un entorno de blockchain.

- **Características y Sintaxis de Solidity** Solidity es un lenguaje tipado estáticamente, lo que significa que los tipos de variables deben ser definidos explícitamente y no pueden cambiar durante la ejecución del contrato. Esta característica ayuda a detectar errores durante la fase de compilación, mejorando la fiabilidad del código. La sintaxis de Solidity es similar a la de JavaScript, lo que facilita su aprendizaje para los desarrolladores familiarizados con lenguajes de programación web. Un contrato en Solidity comienza con la declaración del pragma para especificar la versión del compilador a utilizar, seguida por la definición del contrato en sí, donde se declaran variables de estado, funciones, modificadores y eventos.

Código 3.1: Ejemplo de sintaxis en Solidity

```

1      pragma solidity ^0.8.0;
2
3      contract SimpleStorage {
4          uint storedData;
5
6          function set(uint x) public {
7              storedData = x;
8          }
9
10         function get() public view returns (uint) {
11             return storedData;
12         }
13     }
14 }
```

En el ejemplo anterior, SimpleStorage es un contrato básico que permite almacenar y recuperar un número entero. La declaración pragma solidity 0.8.0 especifica que el contrato debe ser compilado con la versión 0.8.0 o superior del compilador de Solidity.

- **Tipos de Datos y Estructuras en Solidity** Solidity soporta una variedad de tipos de datos primitivos como enteros (int, uint), booleanos (bool), direcciones (address), y cadenas de texto (string). Además, permite la creación de estructuras (structs) para definir tipos de datos personalizados y arreglos (arrays) para almacenar colecciones de datos. Los mapas (mappings) son otra característica clave, permitiendo la asociación de claves y valores, similar a los diccionarios en Python.

Código 3.2: Ejemplo de datos en Solidity

```

1
2 struct Person {
3     uint age;
4     string name;
5 }
6
7 mapping(address => Person) public people;
```

En este ejemplo, se define una estructura Person con campos age y name, y un mapping que asocia direcciones (address) a personas (Person).

- **Funciones y Modificadores** Las funciones en Solidity pueden ser de varios tipos: públicas (public), privadas (private), internas (internal) y externas (external), cada una con diferentes niveles de acceso. Las funciones view y pure son funciones de solo lectura, donde view puede leer el estado del contrato y pure ni siquiera puede leer el estado. Los modificadores de funciones son bloques de código reutilizables que se ejecutan antes de la función que modifican, utilizados para aplicar restricciones o lógica común.

Código 3.3: Ejemplo de funciones y modificadores en Solidity

```

1
2 modifier onlyOwner {
3     require(msg.sender == owner, "Not the contract owner");
4     _;
5 }
6
7 function set(uint x) public onlyOwner {
8     storedData = x;
```

```

9 }
10
11     mapping(address => Person) public people;

```

En este ejemplo, el modificador `onlyOwner` asegura que solo el propietario del contrato pueda llamar a la función `set`.

- **Eventos y Logs** Los eventos en Solidity permiten la comunicación con aplicaciones fuera de la blockchain, registrando actividades y cambios de estado en el contrato. Los eventos se almacenan en los logs de transacciones y pueden ser escuchados por aplicaciones front-end para reaccionar a cambios en el contrato.

Código 3.4: Ejemplo de funciones y modificadores en Solidity

```

1
2 event DataStored(uint data);
3
4 function set(uint x) public {
5     storedData = x;
6     emit DataStored(x);
7 }

```

El evento `DataStored` se emite cada vez que se llama a la función `set`, permitiendo a las aplicaciones escuchar y reaccionar a este cambio.

- **Seguridad y Buenas Prácticas** La seguridad es un aspecto crucial en el desarrollo de contratos inteligentes debido a la naturaleza inmutable de la blockchain. Los desarrolladores deben seguir buenas prácticas como el uso de bibliotecas seguras (e.g., OpenZeppelin), la realización de auditorías de seguridad, y la implementación de patrones de diseño seguros como los patrones de guardián y el uso de `require` y `assert` para validaciones.
- **Gestión de Gas** El gas es la unidad de medida utilizada para cuantificar el costo computacional de las transacciones y la ejecución de contratos inteligentes en Ethereum. Cada operación en Solidity tiene un costo de gas asociado, y los usuarios deben pagar este costo para ejecutar contratos. Esto asegura que los recursos de la red se utilicen de manera eficiente y previene ataques de denegación de servicio (DoS).
- **Futuro de Solidity** El equipo de desarrollo de Solidity está continuamente mejorando el lenguaje, con actualizaciones que abordan problemas de seguridad, añaden nuevas características y mejoran la eficiencia del compilador. Con el crecimiento continuo del ecosistema Ethereum y la adopción de aplicaciones descentralizadas, Solidity seguirá siendo una herramienta fundamental para los desarrolladores de blockchain. Las futuras versiones del lenguaje y las mejoras en la EVM prometen hacer de Solidity un lenguaje aún más poderoso y seguro para la creación de contratos inteligentes.

En resumen, Solidity es un lenguaje esencial para el desarrollo de contratos inteligentes en la blockchain de Ethereum. Su diseño y características permiten a los desarrolladores crear aplicaciones descentralizadas complejas y seguras, con una comunidad activa y un ecosistema robusto que apoya su uso y evolución continua.

3.2.1.2 Ganache

Ganache es una herramienta desarrollada por Truffle Suite que permite a los desarrolladores crear una blockchain local para probar y depurar contratos inteligentes y aplicaciones descentralizadas (dApps) en un entorno controlado y seguro. Ganache emula una red Ethereum completa en su máquina local, proporcionando una manera rápida y eficiente de desarrollar y probar contratos inteligentes sin la necesidad de conectarse a la red principal de Ethereum o a una red de prueba pública. Esta herramienta es esencial para el ciclo de desarrollo de contratos inteligentes, ya que facilita la experimentación y la validación de código antes de su despliegue en una red real.

- **Configuración y Uso de Ganache** La configuración de Ganache es sencilla y directa. Después de descargar e instalar Ganache, los desarrolladores pueden iniciar una nueva blockchain local con solo un clic en la GUI o ejecutando un comando en la CLI. Ganache genera automáticamente varias cuentas prefinanciadas con Ether, lo que facilita la realización de transacciones y pruebas sin preocuparse por el suministro de fondos. Los desarrolladores pueden configurar parámetros como el número de cuentas, el saldo inicial de Ether, el número de bloques minados y el costo del gas. Esta flexibilidad permite la personalización del entorno de desarrollo para que coincida con los escenarios específicos de prueba.

Una vez que Ganache está en funcionamiento, los desarrolladores pueden conectarse a esta blockchain local utilizando herramientas como Truffle o Hardhat. Por ejemplo, en un proyecto Truffle, los desarrolladores pueden configurar la conexión a Ganache en el archivo de configuración *truffle-config.js*:

Código 3.5: Ejemplo de configuración en Ganache

```

1 module.exports = {
2   networks: {
3     development: {
4       host: "127.0.0.1", // Localhost (default: none)
5       port: 7545, // Standard Ethereum port (default: none)
6       network_id: "*", // Any network (default: none)
7     },
8   },
9 },
10 // otras configuraciones
11};

```

Con esta configuración, los desarrolladores pueden compilar, desplegar y probar sus contratos inteligentes en la blockchain local proporcionada por Ganache.

- **Beneficios de Usar Ganache** Uno de los principales beneficios de usar Ganache es la velocidad y la eficiencia que ofrece en el desarrollo y pruebas de contratos inteligentes. Al eliminar la necesidad de esperar confirmaciones de transacciones en una red pública, los desarrolladores pueden iterar rápidamente sobre su código y detectar errores más fácilmente. Ganache también proporciona un entorno controlado donde los desarrolladores pueden simular diversas condiciones de red y casos de uso, como cambios en el costo del gas o la congestión de la red, lo que permite una prueba más exhaustiva y robusta de los contratos.

Otro beneficio importante es la capacidad de Ganache para depurar contratos inteligentes. Con su integración con herramientas como Truffle y Hardhat, Ganache permite la ejecución de pruebas automatizadas y la depuración detallada de transacciones y estados de los contratos. Esto ayuda a identificar y corregir errores en el código antes de su despliegue en una red pública, mejorando la seguridad y la fiabilidad de las dApps.

- **Integración con Otras Herramientas de Desarrollo** Ganache se integra perfectamente con otras herramientas de desarrollo del ecosistema Ethereum. Truffle, una suite de desarrollo completa para Ethereum, ofrece herramientas para la compilación, despliegue y pruebas de contratos inteligentes, y puede utilizar Ganache como blockchain local para estas tareas. Hardhat es otra herramienta popular que se integra bien con Ganache, proporcionando un entorno de desarrollo flexible y potente para contratos inteligentes. Estas integraciones permiten a los desarrolladores configurar flujos de trabajo eficientes y efectivos, utilizando Ganache para pruebas locales rápidas y otras herramientas para pruebas más avanzadas y despliegues.
- **Limitaciones de Ganache** Aunque Ganache es una herramienta poderosa, también tiene algunas limitaciones. Dado que Ganache emula una blockchain local, no puede replicar completamente las condiciones de una red pública como Ethereum Mainnet, incluyendo la latencia de red, los ataques de red y la dinámica del mercado de gas. Por lo tanto, es crucial que los desarrolladores realicen pruebas adicionales en redes de prueba públicas (como Ropsten, Rinkeby o Goerli) antes de desplegar sus contratos en la Mainnet. Además, Ganache no soporta algunos de los mecanismos avanzados de consenso y características específicas de otras blockchains, lo que puede limitar su uso para desarrollos que se centran en plataformas distintas a Ethereum.

3.2.1.3 Truffle

Truffle es un entorno de desarrollo avanzado para Ethereum que ofrece un conjunto completo de herramientas diseñadas para facilitar la creación, compilación, despliegue y pruebas de contratos inteligentes. Desarrollado por Truffle Suite, Truffle es ampliamente utilizado en el ecosistema Ethereum debido a su capacidad para simplificar y automatizar muchas de las tareas involucradas en el desarrollo de aplicaciones descentralizadas (dApps). Desde su lanzamiento, Truffle se ha convertido en una de las herramientas más populares y robustas para desarrolladores de blockchain, proporcionando un marco estructurado y eficiente para gestionar proyectos complejos.

- **Características de Truffle** Truffle ofrece una amplia gama de características que lo convierten en una herramienta esencial para el desarrollo de contratos inteligentes y dApps. Una de las características principales es su capacidad para gestionar el ciclo de vida completo del desarrollo de contratos inteligentes, desde la escritura y compilación del código hasta el despliegue en diversas redes blockchain. Truffle también proporciona un sistema de migraciones, que permite a los desarrolladores gestionar el despliegue de contratos inteligentes de manera controlada y repetible, asegurando que todas las dependencias y configuraciones se manejen correctamente.

Otra característica destacada de Truffle es su suite de pruebas automatizadas, que permite a los desarrolladores escribir y ejecutar pruebas unitarias y de integración

para sus contratos inteligentes. Utilizando frameworks de pruebas como Mocha y Chai, Truffle facilita la creación de pruebas detalladas que ayudan a asegurar la funcionalidad y seguridad del código. Además, Truffle incluye una consola interactiva que permite a los desarrolladores interactuar directamente con sus contratos desplegados, facilitando la depuración y el análisis de contratos en tiempo real.

- **Configuración y Uso de Truffle** La configuración de Truffle es sencilla y directa. Después de instalar Truffle a través de npm (Node Package Manager), los desarrolladores pueden inicializar un nuevo proyecto utilizando el comando `truffle init`, que crea una estructura de proyecto predeterminada con carpetas para contratos, migraciones y pruebas. El archivo de configuración principal, `truffle-config.js`, permite a los desarrolladores especificar detalles de la red, configuraciones de compilación y otros parámetros necesarios para el desarrollo y despliegue de contratos.
Una vez configurado el proyecto, los desarrolladores pueden escribir sus contratos inteligentes en la carpeta `contracts` y definir las migraciones en la carpeta `migrations`. Truffle facilita la compilación de contratos con el comando `truffle compile`, y el despliegue a una red blockchain con `truffle migrate`. La capacidad de Truffle para gestionar migraciones asegura que los contratos se desplieguen en el orden correcto y que las dependencias entre contratos se resuelvan automáticamente.
- **Integración con Ganache** Truffle se integra perfectamente con Ganache, proporcionando un entorno de desarrollo y pruebas completo y eficiente. Ganache puede utilizarse como una blockchain local para pruebas rápidas y seguras, mientras que Truffle gestiona la compilación, despliegue y pruebas de contratos inteligentes en esta blockchain. Esta integración permite a los desarrolladores iterar rápidamente sobre su código, detectar errores y optimizar sus contratos antes de desplegarlos en una red pública.
- **Beneficios de Usar Truffle** Truffle ofrece numerosos beneficios que lo convierten en una herramienta indispensable para desarrolladores de Ethereum. Su capacidad para gestionar el ciclo de vida completo del desarrollo de contratos, desde la escritura y pruebas hasta el despliegue y migraciones, simplifica el proceso y reduce la posibilidad de errores. La suite de pruebas automatizadas de Truffle asegura que los contratos sean robustos y seguros, mientras que la integración con herramientas como Ganache facilita el desarrollo y pruebas en un entorno local controlado.

3.3 Web 3.0

Web3, representa la próxima generación de internet que promueve la descentralización, la propiedad de datos por parte de los usuarios y una mayor seguridad a través de tecnologías emergentes como blockchain y contratos inteligentes. A diferencia de la Web 2.0, que se caracteriza por plataformas centralizadas y la propiedad de datos por parte de grandes corporaciones, Web3 busca empoderar a los usuarios, dándoles control directo sobre sus datos y facilitando interacciones peer-to-peer sin intermediarios. Esto implica un cambio de paradigma significativo, donde los usuarios no solo consumen contenido, sino que también pueden poseer y gestionar sus datos y activos digitales.

3.3.1 Principios Fundamentales de Web3

En Web3, los datos y servicios no están controlados por entidades centralizadas. En lugar de ello, la información se almacena en redes descentralizadas de nodos (ordenadores) que operan de manera autónoma y colaborativa. Esta estructura reduce el riesgo de censura y abuso de poder por parte de entidades centralizadas, permitiendo que las aplicaciones descentralizadas (dApps) funcionen sobre estas redes y aprovechen blockchain para garantizar la inmutabilidad y seguridad de las transacciones y datos. Además, en la Web 2.0, los datos generados por los usuarios, como publicaciones, fotos y preferencias, son controlados y monetizados por las plataformas que los alojan. Web3 cambia este paradigma al permitir que los usuarios sean los verdaderos propietarios de sus datos, utilizando tecnologías como identidades descentralizadas (DIDs) y sistemas de almacenamiento distribuidos (e.g., IPFS), permitiendo a los usuarios controlar quién tiene acceso a su información y cómo se utiliza. Web3 también promueve la soberanía del usuario, permitiendo a las personas interactuar directamente entre sí sin intermediarios, logrado mediante el uso de contratos inteligentes que pueden automatizar acuerdos y transacciones de manera segura y transparente, eliminando intermediarios, reduciendo costos, aumentando la eficiencia y abriendo nuevas oportunidades para la innovación en diversos sectores.

3.3.2 Características Clave de Web3

Las interacciones en Web3 se realizan directamente entre pares (usuarios) sin la necesidad de un intermediario centralizado, facilitadas mediante redes peer-to-peer y tecnologías de contratos inteligentes. Ejemplos de aplicaciones P2P incluyen plataformas de intercambio de criptomonedas, redes sociales descentralizadas y mercados de NFT (tokens no fungibles). La naturaleza descentralizada de Web3 mejora la resiliencia del sistema, ya que no hay un único punto de fallo; las redes distribuidas son más resistentes a ataques ciberneticos y fallos de infraestructura, y la criptografía avanzada se utiliza para asegurar las transacciones y los datos, protegiendo la privacidad de los usuarios y garantizando la integridad de la información. Además, todas las transacciones y operaciones en una blockchain pública son transparentes y pueden ser verificadas por cualquier persona, lo que aumenta la confianza en el sistema y permite auditar de manera independiente todas las actividades realizadas. Los contratos inteligentes ejecutan de manera automática y transparente las condiciones predefinidas, eliminando la posibilidad de manipulación o fraude. Web3 también introduce el concepto de economía tokenizada, donde los activos digitales (tokens) pueden representar valor, derechos de propiedad, votos en un sistema de gobernanza, entre otros, y pueden ser intercambiados de manera sencilla y segura en plataformas descentralizadas, facilitando nuevos modelos económicos y de negocio.

3.3.2.1 Carteras Digitales

Las carteras digitales son fundamentales para la interacción con la Web3, actuando como herramientas que permiten a los usuarios gestionar sus criptomonedas, activos digitales y credenciales de identidad de manera segura y descentralizada. Estas carteras pueden ser software (como MetaMask y Trust Wallet) o hardware (como Ledger y Trezor), cada una ofreciendo diferentes niveles de seguridad y usabilidad. Las carteras software son aplicaciones

que se instalan en dispositivos móviles o navegadores web, proporcionando una interfaz fácil de usar para realizar transacciones y gestionar activos. MetaMask, por ejemplo, es una extensión del navegador que permite a los usuarios interactuar con dApps directamente desde su navegador, simplificando el proceso de conexión y transacción en la Web3. Por otro lado, las carteras hardware son dispositivos físicos que almacenan las claves privadas de los usuarios fuera de línea, protegiéndolas contra ataques y accesos no autorizados, lo que las hace ideales para la custodia a largo plazo de grandes cantidades de activos.

Metamask: Es una cartera digital de criptomonedas y una extensión de navegador que permite a los usuarios interactuar con la blockchain de Ethereum y otras redes compatibles. Desarrollada por ConsenSys, MetaMask proporciona una interfaz sencilla y segura para gestionar criptomonedas, realizar transacciones y conectarse con aplicaciones descentralizadas (dApps). Desde su lanzamiento en 2016, MetaMask se ha convertido en una de las herramientas más populares en el ecosistema Ethereum, facilitando el acceso de millones de usuarios a la Web3.

- **Características de MetaMask** MetaMask ofrece una serie de características que lo hacen indispensable para los usuarios de la Web3. Una de las principales características es la gestión de múltiples cuentas y direcciones, permitiendo a los usuarios crear y gestionar varias identidades en la blockchain desde una sola interfaz. MetaMask también incluye un generador de claves y una bóveda segura para almacenar claves privadas, asegurando que solo los usuarios tengan acceso a sus fondos y datos. Además, MetaMask soporta la importación y exportación de claves privadas, permitiendo a los usuarios recuperar sus cuentas en caso de pérdida o cambiar de dispositivos.

Otra característica destacada de MetaMask es su capacidad para conectarse con dApps directamente desde el navegador. MetaMask inyecta un objeto web3 en cada página web que el usuario visita, permitiendo que las dApps detecten automáticamente y se conecten a la cuenta del usuario. Esto facilita la interacción con aplicaciones descentralizadas, permitiendo a los usuarios realizar transacciones y firmar mensajes directamente desde la extensión del navegador.

- **Configuración y Uso de MetaMask** Configurar MetaMask es un proceso sencillo y directo. Después de instalar la extensión en un navegador compatible (Chrome, Firefox, Brave o Edge), los usuarios deben crear una nueva cartera o importar una existente. Al crear una nueva cartera, MetaMask genera una frase de recuperación de 12 palabras que los usuarios deben guardar de manera segura. Esta frase de recuperación es crucial para recuperar el acceso a la cartera en caso de pérdida del dispositivo.

Una vez configurada la cartera, los usuarios pueden gestionar sus criptomonedas directamente desde la interfaz de MetaMask. La pantalla principal muestra el saldo de la cuenta y proporciona botones para enviar y recibir criptomonedas. Los usuarios pueden añadir tokens personalizados a su cartera ingresando la dirección del contrato del token, lo que permite gestionar una variedad de activos digitales desde una sola interfaz.

MetaMask también permite a los usuarios conectarse a diferentes redes de Ethereum, incluyendo Mainnet, redes de prueba (Ropsten, Rinkeby, Goerli) y redes personalizadas. Esta flexibilidad es útil para los desarrolladores que necesitan probar sus contratos inteligentes y dApps en diferentes entornos antes de desplegarlos en la red principal.

- **Conexión con dApps** Una de las funcionalidades más importantes de MetaMask es su capacidad para conectarse con dApps. Cuando un usuario visita una dApp que requiere interacción con la blockchain, MetaMask solicita permiso para conectarse a la cuenta del usuario. Una vez conectado, la dApp puede solicitar transacciones, firmas de mensajes y otras interacciones que el usuario puede aprobar o rechazar a través de la interfaz de MetaMask.

Este modelo de interacción asegura que los usuarios tengan un control total sobre sus transacciones y datos, proporcionando una capa adicional de seguridad y transparencia. Por ejemplo, en una plataforma de intercambio descentralizado (DEX) como Uniswap, MetaMask permite a los usuarios intercambiar tokens directamente desde su cartera, sin necesidad de confiar en un tercero para custodiar sus activos.

- **Seguridad en MetaMask** La seguridad es una prioridad en MetaMask, y la herramienta implementa varias medidas para proteger los fondos y datos de los usuarios. Las claves privadas se almacenan de manera segura en el dispositivo del usuario y nunca se envían a servidores externos. La frase de recuperación de 12 palabras es crucial para recuperar el acceso a la cartera y debe ser guardada de manera segura, ya que cualquier persona con acceso a esta frase puede controlar la cartera.

MetaMask también proporciona opciones de configuración avanzadas, como la habilitación de notificaciones de seguridad y la configuración de límites de gas para transacciones, lo que ayuda a prevenir errores y proteger a los usuarios contra intentos de fraude o phishing. La educación continua y la concienciación sobre las mejores prácticas de seguridad también son fundamentales, y MetaMask ofrece recursos y guías para ayudar a los usuarios a mantenerse seguros en el ecosistema Web3.

- **Beneficios de Usar MetaMask** MetaMask ofrece numerosos beneficios que lo hacen una herramienta esencial para los usuarios de Ethereum y Web3. Su facilidad de uso y configuración rápida permiten a los usuarios nuevos ingresar al mundo de las criptomonedas y las dApps con facilidad. La capacidad de gestionar múltiples cuentas y conectarse a diferentes redes proporciona flexibilidad y conveniencia para usuarios y desarrolladores. Además, la integración directa con dApps a través del navegador permite una experiencia de usuario fluida y segura.
- **Integración con Otras Herramientas y Plataformas** MetaMask se integra perfectamente con una amplia gama de herramientas y plataformas en el ecosistema Ethereum. Es compatible con servicios como Infura para facilitar la conectividad a la blockchain de Ethereum y puede utilizarse junto con Truffle y Ganache para desarrollar, probar y desplegar contratos inteligentes. Además, muchas dApps populares en el espacio DeFi, como Aave, Compound y MakerDAO, están diseñadas para funcionar sin problemas con MetaMask, proporcionando una experiencia de usuario coherente y segura.

3.3.2.2 Identidad Descentralizada

La identidad descentralizada es un concepto central en Web3 que permite a los usuarios controlar sus identidades digitales sin depender de proveedores centralizados. Los sistemas

de identidad descentralizada, como Decentralized Identifiers (DIDs) y Verifiable Credentials, permiten a los usuarios crear y gestionar identidades digitales que son independientes de cualquier autoridad central. DIDs son identificadores únicos que no están vinculados a un registro centralizado y pueden ser verificados de manera independiente. Estos identificadores permiten a los usuarios demostrar su identidad sin tener que confiar en terceros centralizados. Las Verifiable Credentials son credenciales digitales que pueden ser emitidas por cualquier entidad y verificadas de manera descentralizada, proporcionando un mecanismo para que los usuarios prueben atributos de su identidad (como edad, nacionalidad o calificaciones académicas) sin revelar información adicional. Estas tecnologías tienen el potencial de transformar cómo se gestionan y verifican las identidades en línea, aumentando la privacidad y el control de los usuarios sobre sus datos personales.

3.3.2.3 Seguridad de los Smart Contracts

La seguridad es un aspecto crítico en el desarrollo y ejecución de contratos inteligentes. Los contratos inteligentes deben ser diseñados y auditados cuidadosamente para evitar vulnerabilidades y ataques. Una de las vulnerabilidades más conocidas es el ataque de reentrancy, que permitió el hackeo de The DAO en 2016, resultando en la pérdida de millones de dólares en Ether. Para mitigar riesgos, los desarrolladores utilizan prácticas de programación segura, herramientas de análisis estático y dinámico, y auditorías de seguridad por terceros. Plataformas como ConsenSys Diligence y CertiK ofrecen servicios de auditoría de contratos inteligentes para identificar y corregir vulnerabilidades. Además, la comunidad de desarrolladores de Ethereum ha desarrollado patrones de diseño y bibliotecas, como OpenZeppelin, que proporcionan implementaciones seguras de contratos comunes.

3.3.2.4 Desarrollo y Herramientas para Smart Contracts

El desarrollo de contratos inteligentes requiere herramientas y entornos especializados que faciliten la escritura, prueba y despliegue de contratos en la blockchain. Truffle es un popular entorno de desarrollo que proporciona una suite completa para el desarrollo de contratos inteligentes, incluyendo herramientas para compilación, despliegue y pruebas. Hardhat es otra herramienta avanzada que permite a los desarrolladores gestionar y automatizar tareas de desarrollo de contratos inteligentes, proporcionando un entorno flexible y personalizable. Remix es un IDE basado en navegador que permite a los desarrolladores escribir, probar y depurar contratos inteligentes directamente en el navegador. Además, plataformas como Infura y Alchemy proporcionan infraestructura backend para aplicaciones Web3, permitiendo a los desarrolladores conectarse a la blockchain de manera eficiente y escalable. Las prácticas recomendadas para el desarrollo de contratos inteligentes incluyen el uso de bibliotecas seguras, la realización de auditorías de seguridad y la implementación de pruebas exhaustivas para asegurar la robustez y seguridad del código.

3.3.2.5 Mecanismos de Ejecución y Verificación

La Ethereum Virtual Machine (EVM) es fundamental para la ejecución de contratos inteligentes en la blockchain de Ethereum. La EVM es una máquina virtual completa de Turing que ejecuta bytecode específico de Ethereum, permitiendo la implementación de contratos

inteligentes y aplicaciones descentralizadas. Para realizar cualquier operación en la EVM, los usuarios deben pagar "gas", que es una medida del costo computacional de la operación. El gas asegura que los recursos de la red se utilicen de manera justa y previene que operaciones costosas o maliciosas afecten el rendimiento de la red. El costo del gas varía según la demanda de la red y la complejidad de la operación. Otros mecanismos de ejecución incluyen la Solana Runtime, que permite la ejecución de contratos inteligentes en la blockchain de Solana a alta velocidad y con bajas tarifas de transacción, y la WASM (WebAssembly) utilizada en plataformas como EOS y Polkadot para ejecutar contratos inteligentes de manera eficiente y segura.

3.3.2.6 Lenguajes de Programación para Smart Contracts

Solidity es el lenguaje de programación más utilizado para escribir contratos inteligentes en Ethereum. Es un lenguaje de alto nivel, influenciado por JavaScript, Python y C++, diseñado específicamente para la creación de contratos inteligentes. Solidity permite a los desarrolladores escribir programas que se ejecutan en la EVM, gestionando la lógica de las transacciones y las interacciones entre contratos. Además de Solidity, existen otros lenguajes como Vyper, que es más sencillo y está diseñado para mejorar la seguridad de los contratos inteligentes en Ethereum. En otras plataformas, como Hyperledger Fabric, se utiliza Chaincode, que puede ser escrito en lenguajes como Go, Java y JavaScript, proporcionando flexibilidad y familiaridad para los desarrolladores empresariales. En Cardano, se utiliza Plutus, un lenguaje basado en Haskell, conocido por su enfoque en la seguridad y la corrección formal del código.

3.3.2.7 Beneficios y Limitaciones de los Smart Contracts

Los contratos inteligentes ofrecen numerosos beneficios, incluyendo la automatización, transparencia, reducción de costos y eliminación de intermediarios. Al automatizar la ejecución de acuerdos, los contratos inteligentes reducen la necesidad de intermediarios, disminuyendo costos y riesgos de manipulación. La transparencia de los contratos inteligentes, que se ejecutan en una blockchain pública, permite a todas las partes verificar de manera independiente la ejecución de los acuerdos. Sin embargo, también presentan limitaciones, como problemas de escalabilidad, costes asociados al gas, desafíos de integración y cuestiones de privacidad. La escalabilidad es un desafío significativo, ya que las blockchains actuales tienen una capacidad limitada para procesar transacciones, lo que puede resultar en congestión y altos costos de gas durante períodos de alta demanda. Además, la integración de contratos inteligentes con sistemas tradicionales puede ser compleja y requerir soluciones personalizadas. Las cuestiones de privacidad también son una preocupación, ya que las transacciones en la blockchain son públicas por defecto, lo que puede no ser adecuado para todas las aplicaciones.

4 Estado del Arte

En esta sección, trataremos el estado del arte. En la figura 4.1, se presenta una cronología general de los diferentes apartados relacionados con este tema. Desglosaremos la historia para explicar cada aspecto en profundidad.

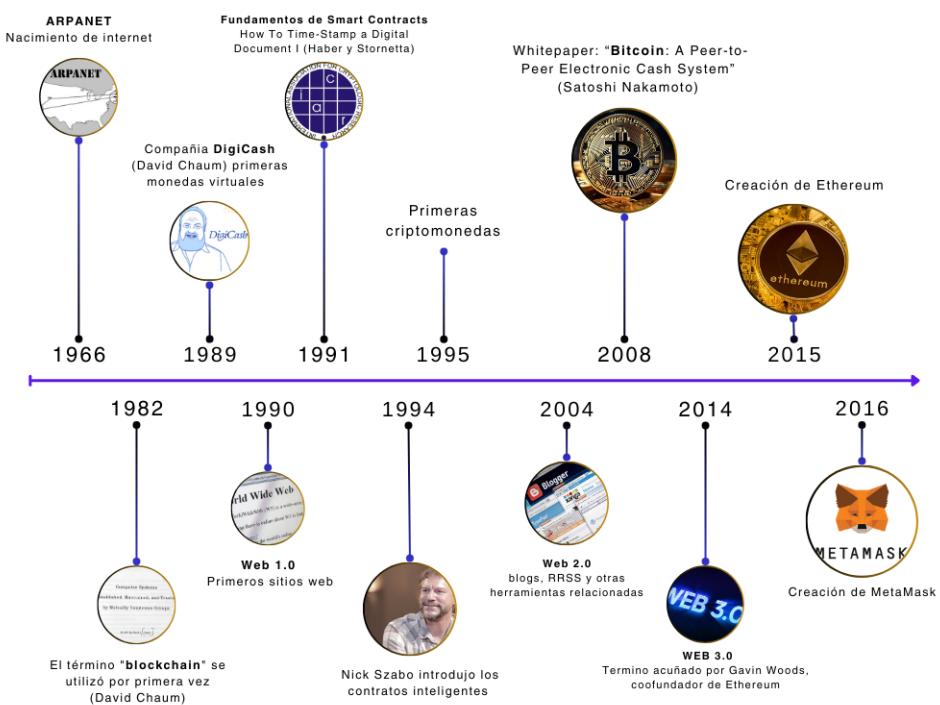


Figura 4.1: Cronología de la Web3.

4.1 Blockchain

Hace casi una década, Satoshi Nakamoto, la persona o grupo desconocido detrás de Bitcoin, describió cómo la tecnología blockchain, una estructura enlazada distribuida de igual a igual, podría utilizarse para resolver el problema de mantener el orden de las transacciones y evitar el problema del doble gasto. Bitcoin ordena las transacciones y las agrupa en una estructura de tamaño limitado llamada bloques, que comparten la misma marca de tiempo. Los nodos de la red (mineros) son responsables de enlazar los bloques entre sí en orden cronológico, con cada bloque conteniendo el hash del bloque anterior para crear una blockchain. Así, la estructura de blockchain logra contener un registro robusto y auditible

de todas las transacciones. En este apartado trataremos el estado del arte basado en un estudio financiado por la Comisión Europea en el programa Horizon 2020 (Casino y cols., 2019).

Las blockchains han introducido serias disrupciones en los procesos comerciales tradicionales, ya que ahora pueden operar de manera descentralizada con el mismo nivel de certeza. Las características inherentes de la arquitectura y el diseño de blockchain proporcionan propiedades como transparencia, robustez, auditabilidad y seguridad. Una blockchain es ideal en el sector bancario, ya que los bancos pueden cooperar bajo la misma blockchain y gestionar las transacciones de sus clientes. De esta manera, más allá de la transparencia, la blockchain facilita la auditoría de transacciones. Las empresas invierten en esta tecnología ya que ven el potencial de descentralizar sus arquitecturas y minimizar sus costos de transacción al hacerse intrínsecamente más seguras, transparentes y, en algunos casos, más rápidas. Por lo tanto, las blockchains no son solo una moda.

Los investigadores y desarrolladores ya son conscientes de las capacidades de la nueva tecnología y exploran diversas aplicaciones en una amplia gama de sectores. Basado en la audiencia a la que se dirige, se pueden distinguir tres generaciones de blockchains: Blockchain 1.0 que incluye aplicaciones que permiten transacciones de criptomonedas digitales; Blockchain 2.0 que incluye contratos inteligentes y un conjunto de aplicaciones que se extienden más allá de las transacciones de criptomonedas; y Blockchain 3.0 que incluye aplicaciones en áreas más allá de las dos versiones anteriores, como el gobierno, la salud, la ciencia y el IoT. Se destaca el creciente interés de la comunidad académica donde se identifica tres corrientes de investigación clave: la clasificación de la gama de aplicaciones basadas en blockchain en una amplia gama de sectores; la idoneidad de la tecnología blockchain para crear valor en estos sectores teniendo en cuenta las diversas limitaciones que presenta esta tecnología; y la orientación a los investigadores proporcionando una hoja de ruta de avenidas de investigación prometedoras, desafíos y oportunidades para las cuales se necesita más investigación.



Figura 4.2: Capitalización de mercado de las principales criptomonedas según la web *Statista* (Roa, 2021).

4.1.1 Historia de la Blockchain

El término "blockchain" se utilizó por primera vez en 1982. Fue en una disertación académica en la Universidad de Berkeley, obra de David Chaum, titulada "Sistemas informáticos establecidos, mantenidos y confiables por grupos mutuamente sospechosos".

Aunque la utilidad principal de esta tecnología no era originalmente el soporte de monedas digitales, Chaum vio de inmediato su potencial en este ámbito. En 1989, lanzó la compañía DigiCash, responsable de las primeras monedas virtuales en 1995. Sin embargo, Chaum no logró convencer a las entidades bancarias de sus beneficios y la falta de infraestructura adecuada en ese momento llevó al fracaso del proyecto.

Blockchain volvió a cobrar relevancia en 2008, en foros de Internet, donde comenzó a circular el artículo "Bitcoin: un sistema de efectivo electrónico de persona a persona", firmado por Satoshi Nakamoto. Hasta la fecha, se desconoce si este nombre corresponde a un individuo anónimo o a un colectivo, pero se le atribuye la creación del blockchain moderno, basado en gran medida en las propuestas técnicas de Chaum, con algunas modificaciones.

En 2015, un grupo de contribuyentes del proyecto Bitcoin creó Ethereum. Hasta ese momento, la tecnología blockchain se utilizaba para soportar criptomonedas, pero Ethereum incluye código fuente ejecutable, lo que la hace mucho más flexible y versátil, permitiendo también el alojamiento de NFT y una variedad de aplicaciones descentralizadas (DApps) (Orange, 2022).

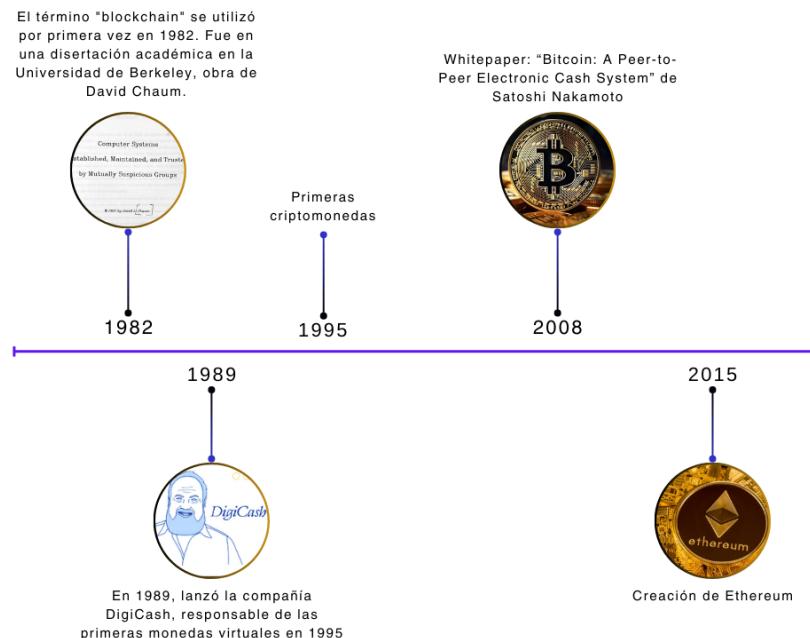


Figura 4.3: Cronología de la Blockchain.

4.1.2 Clasificación de Redes Blockchain

La literatura actual clasifica las redes blockchain de varias maneras. Estas categorías se forman según la gestión y los permisos de la red, dividiéndose en públicas, privadas y federadas. En las blockchains públicas (sin permiso), cualquier persona puede unirse como un nuevo usuario o nodo minero. Además, todos los participantes pueden realizar operaciones como transacciones o contratos. En las blockchains privadas, que junto con las federadas pertenecen a la categoría de blockchain con permiso, generalmente se define una lista blanca de usuarios permitidos con características y permisos particulares sobre las operaciones de la red. Dado que el riesgo de ataques Sybil¹ es casi insignificante en estos casos, las redes blockchain privadas pueden evitar mecanismos costosos de Prueba de Trabajo (PoW). En su lugar, se podría adoptar una gama más amplia de protocolos de consenso basados en desincentivos.

Una blockchain federada o institucional es una combinación híbrida de blockchains públicas y privadas. Aunque comparte un nivel similar de escalabilidad y protección de la privacidad con la blockchain privada, su principal diferencia radica en que se selecciona un conjunto de nodos, llamados nodos líderes, en lugar de una entidad única para verificar los procesos de transacción. Esto permite un diseño parcialmente descentralizado donde los nodos líderes pueden otorgar permisos a otros usuarios. La tabla 4.1 resume las principales características de cada red blockchain en cuanto a eficiencia, seguridad y mecanismos de consenso.

Implementaciones bien conocidas de blockchains públicas incluyen Bitcoin, Ethereum, Litecoin y, en general, la mayoría de las criptomonedas. Una de sus principales ventajas es la falta de costos de infraestructura: la red se auto-sustenta y es capaz de mantenerse por sí misma, reduciendo drásticamente los costos de gestión. En las blockchains privadas, las principales aplicaciones son la gestión de bases de datos, la auditoría y, en general, soluciones que demandan alto rendimiento. Multichain es un ejemplo de una plataforma abierta para construir y desplegar blockchains privadas. Finalmente, las blockchains federadas se utilizan principalmente en los sectores bancario e industrial. Este es el caso del proyecto Hyperledger, que desarrolla marcos de blockchain basados en permisos entre industrias. Recientemente, Ethereum también ha proporcionado herramientas para construir blockchains federadas. Otros proyectos como Cardano son bastante ambiciosos y buscan ofrecer más funcionalidad.

A medida que la tecnología blockchain continúa evolucionando, se están explorando nuevas aplicaciones y modelos de negocio. Por ejemplo, las finanzas descentralizadas (DeFi) están ganando popularidad al permitir transacciones financieras sin intermediarios tradicionales, utilizando contratos inteligentes en plataformas como Ethereum. Además, las organizaciones autónomas descentralizadas (DAO) están emergiendo como una forma innovadora de gobernanza, permitiendo a las comunidades tomar decisiones colectivas mediante la votación en la blockchain. Estos desarrollos no solo están revolucionando la forma en que se gestionan las finanzas y la gobernanza, sino que también están impulsando la adopción de la blockchain en diversos sectores, desde el arte digital (a través de los NFTs) hasta la cadena de suministro y

¹Un ataque Sybil ocurre cuando un adversario crea múltiples identidades falsas en una red distribuida para ganar influencia desproporcionada y manipular el sistema. Este tipo de ataque se mitiga mediante mecanismos de consenso como Proof of Work, Proof of Stake, y verificación de identidad.

la gestión de identidades. La capacidad de la blockchain para ofrecer transparencia, seguridad y descentralización sigue abriendo nuevas posibilidades y desafíos, prometiendo transformar profundamente la economía global y las estructuras sociales.

Propiedad	Pública	Privada	Institucional
Mecanismo de Consenso	<ul style="list-style-type: none"> • PoW costoso • Todos los mineros 	<ul style="list-style-type: none"> • PoW ligero • Organización centralizada 	<ul style="list-style-type: none"> • PoW ligero • Conjunto de nodos líderes
Anonimato de Identidad	<ul style="list-style-type: none"> • (Pseudo) Anónimo • ¿Malicioso? 	<ul style="list-style-type: none"> • Usuarios identificados • Confiable 	<ul style="list-style-type: none"> • Usuarios identificados • Confiable
Eficiencia y Consumo del Protocolo	<ul style="list-style-type: none"> • Baja eficiencia • Alto consumo de energía 	<ul style="list-style-type: none"> • Alta eficiencia • Bajo consumo de energía 	<ul style="list-style-type: none"> • Alta eficiencia • Bajo consumo de energía
Inmutabilidad	<ul style="list-style-type: none"> • Casi imposible 	<ul style="list-style-type: none"> • Ataques de colusión 	<ul style="list-style-type: none"> • Ataques de colusión
Propiedad y Gestión	<ul style="list-style-type: none"> • Pública • Sin permisos 	<ul style="list-style-type: none"> • Centralizada • Lista blanca con permisos 	<ul style="list-style-type: none"> • Semi-centralizada • Nodos con permisos
Aprobación de Transacciones	<ul style="list-style-type: none"> • Orden de minutos 	<ul style="list-style-type: none"> • Orden de milisegundos 	<ul style="list-style-type: none"> • Orden de milisegundos

Tabla 4.1: Clasificación y características principales de las redes blockchain.

Las empresas de diversos sectores están entusiasmadas con la tecnología blockchain por su capacidad para impulsar la transformación digital y resolver problemas reales. Sin embargo, muchos especialistas en TI no comprenden completamente las razones para usar blockchain, especialmente en la gestión de datos. Si no es necesario almacenar datos o si solo se prevé un escritor en el sistema, una base de datos tradicional puede ser más adecuada debido a su rendimiento superior. Blockchain es ideal cuando se requiere una transacción entre fuentes no confiables o un registro histórico permanente. Antes de adoptar soluciones basadas en blockchain, se debe evaluar la idoneidad de la tecnología frente a los requisitos del caso de uso. Las bases de datos son naturalmente mutables y tienen roles administrativos que pueden alterar completamente el contenido y la estructura de la información. Se ha desarrollado un marco para evaluar la idoneidad de las soluciones basadas en blockchain en comparación con las bases de datos tradicionales en áreas como las suposiciones de confianza, los requisitos del contexto, las características de rendimiento y los mecanismos de consenso requeridos.

Características/Requisitos	Escalabilidad	Privacidad	Interoperabilidad	Auditoría	Latencia	Visibilidad
Finanzas	Sí	Depende	Sí	Sí	Sí	Sí
Servicios de Ciudadanía	Sí	Sí	Sí	Sí	Sí	Sí
Verificación de Integridad	Sí	Sí	Sí	Sí	Sí	Sí
Gobernanza	Depende		Sí	Sí		
IoT	Sí		Sí	Sí	Depende	
Salud	Sí	Sí	Sí	Sí		
Privacidad y Seguridad	Sí	Sí	Sí	Sí		
Negocios	Depende	Depende	Sí	Sí	Depende	
Educación	Sí	Sí	Sí	Sí		
Gestión de Datos	Sí		Sí	Sí	Depende	Sí

Tabla 4.2: Características/requisitos que permiten/requieren cada familia de aplicaciones blockchain. "Sí" indica que este requisito es obligatorio, mientras que "Depende" indica que depende del caso (Casino y cols., 2019).

4.1.3 Direcciones Futuras

Según el artículo The Revolution of Blockchain: State-of-the-Art and Research Challenges (Namasudra y cols., 2021) , la demanda de la tecnología Blockchain está en continuo aumento. La tecnología de libro mayor distribuido, junto con la velocidad y la transparencia, son algunas de las principales razones que explican este rápido incremento en su demanda.

En 2017, el mercado de Blockchain alcanzó los 411,5 millones de dólares, y con una tasa de crecimiento anual del 79,6%, se proyecta que llegará a 7683,7 millones para 2022. Las transacciones de pagos digitales fueron de 2672 mil millones en 2017, con un 3% de participación de la red P2P, y se espera que alcancen 4644 mil millones para 2021, con un 4% de participación P2P. El número de usuarios de la red P2P aumentará de 180 millones en 2017 a aproximadamente 190 millones en 2021, destacando el papel crucial de Blockchain en estas redes.

El creciente reconocimiento del potencial de la tecnología Blockchain está impulsando inversiones significativas, especialmente en sectores como finanzas, salud y cadena de suministro. Blockchain ofrece soluciones seguras y transparentes para la gestión de datos y transacciones, y está fomentando la creación de nuevos modelos de negocio. Las startups están explorando aplicaciones innovadoras como contratos inteligentes y sistemas de votación digital, subrayando la versatilidad y el impacto transformador de Blockchain. Este interés global está acelerando el desarrollo y adopción de esta tecnología, iniciando una nueva era de innovación digital.

Región	Financiación (USD)	Participación en la financiación industrial total en la región (%)	Hallazgos
Asia	210 millones	0.25	Asia ya ha comenzado a invertir en tecnología Blockchain.
Resto del Mundo	41 millones	0.29	Australia, Medio Oriente, África y Sudamérica han invertido solo el 2.5% de la financiación global.
Norte América	1.53 mil millones	0.32	EE.UU., Canadá y México han invertido mucho en las empresas emergentes de Blockchain.
Europa	1.03 mil millones	1.25	Reino Unido tiene la mayor participación de financiación, seguido por Irlanda y Países Bajos.

Tabla 4.3: Inversión en tecnología Blockchain en diferentes regiones en 2017(Namasudra y cols., 2021).

Con la creciente demanda de sistemas descentralizados, se considera que la tecnología Blockchain puede ser una excelente alternativa para cualquier sistema o red que maneje datos y transacciones. Esta tecnología ofrece una solución innovadora que permite registrar y verificar transacciones de manera segura y transparente sin necesidad de un intermediario centralizado. Este enfoque descentralizado no solo reduce los riesgos de fraude y manipulación de datos, sino que también aumenta la confianza entre las partes involucradas al proporcionar un registro inmutable de todas las transacciones.

La robusta seguridad que ofrece Blockchain se debe a su estructura criptográfica avanzada y a su mecanismo de consenso distribuido, que hacen extremadamente difícil que actores malintencionados alteren los datos sin ser detectados. Esta seguridad intrínseca hace que Blockchain sea ideal para aplicaciones que requieren alta integridad y protección de datos sensibles.

Se espera que la tecnología Blockchain se desarrolle en los siguientes dominios principales:



Figura 4.4: Mapa mental de los distintos tipos de aplicaciones de blockchain (Casino y cols., 2019).

4.1.3.1 Desafíos

Los desafíos potenciales de la Blockchain son:

1. **Seguridad de la Clave Privada:** En la tecnología Blockchain, se asigna una clave privada única y secreta a cada nodo o usuario del sistema. Si la aleatoriedad de la clave privada es baja, un atacante puede obtenerla y comprometer todos los detalles del usuario, siendo difícil identificar al atacante debido a la naturaleza descentralizada de la red.
2. **Vulnerabilidad del 51%:** La aprobación de al menos el 51% de los nodos es necesaria para validar cualquier transacción en una red Blockchain. Si un solo ente controla el 51% de los nodos, puede manipular la red para validar transacciones inválidas. Esto ocurrió en 2014 cuando el grupo de minería ghash.io alcanzó el 42% del poder computacional de Bitcoin, provocando temor entre los mineros.
3. **Doble Gasto:** El doble gasto se refiere a usar la misma criptomoneda en múltiples transacciones explotando el intervalo de tiempo entre la iniciación y confirmación de las transacciones, permitiendo que la primera transacción se complete antes de que se declare inválida la segunda.
4. **Fugas de Privacidad en Transacciones:** Aunque se toman medidas para proteger la privacidad en Blockchain, como el uso de monedas de mezcla en Monero, estas medidas tienen limitaciones. En 2017, se descubrió que el 66% de las transacciones en Monero no incluían monedas de mezcla, lo que compromete la privacidad de los usuarios.
5. **Actividades Criminales:** Las direcciones de Bitcoin no están vinculadas a identidades reales, lo que facilita su uso para actividades ilegales como el ransomware WannaCry, que exigía pagos en Bitcoin. Los contratos inteligentes también pueden ser utilizados para actividades maliciosas, como el robo de contraseñas.
6. **Ataques:** La tecnología Blockchain está sujeta a diversos ataques, como el de minería egoísta, ataques DAO, ataques de vida, secuestro de BGP y ataques de balance.
7. **Estandarización:** La falta de estandarización en la red Blockchain permite a los desarrolladores trabajar sin restricciones, lo que crea problemas para los departamentos de TI. La estandarización podría facilitar la colaboración entre empresas y la integración con sistemas existentes.
8. **Costo Ambiental:** Implementar Blockchain es costoso debido a la necesidad de software especializado y personal calificado. Además, el alto consumo de energía requerido para mantener las redes Blockchain plantea preocupaciones ambientales.
9. **Consumo de Energía:** La ejecución de algoritmos complejos para validar transacciones en Blockchain consume mucha energía. Redes como Bitcoin y Ethereum requieren resolver problemas matemáticos complejos, lo que demanda gran cantidad de energía.
10. **Lentitud y Complejidad:** La naturaleza distribuida y la operación de cifrado hacen que Blockchain sea más lento que los métodos tradicionales de pago. Las transacciones

en Bitcoin pueden tardar hasta una hora en completarse, lo que es ineficiente para aplicaciones de tiempo real.

11. **Percepción Pública:** La gente común aún desconoce los beneficios potenciales de la tecnología Blockchain. A menudo se asocia Blockchain con Bitcoin y actividades ilegales, lo que genera una percepción negativa. Es crucial educar al público sobre las diferencias y beneficios de Blockchain más allá de las criptomonedas.

4.1.4 Conclusiones

Aunque las aplicaciones de la tecnología Blockchain están siendo ampliamente implementadas, aún quedan muchos problemas por abordar. Al hacerlo, las blockchains no solo se volverán más escalables y eficientes, sino también más duraderas. Las características que ofrecen no son únicas si se evalúan individualmente, y la mayoría de los mecanismos en los que se basan son conocidos desde hace años. Sin embargo, la combinación de todas estas características las hace ideales para muchas aplicaciones, lo que justifica el intenso interés por parte de diversas industrias.

A medida que las blockchains maduren, se espera que sus aplicaciones penetren en más industrias y dominios de los que se han cubierto hasta ahora. No obstante, aunque muchos intentan proponer las blockchains como una panacea y una alternativa a las bases de datos tradicionales, esto está lejos de ser cierto. Como ya se ha discutido, hay muchos escenarios en los que se deben utilizar bases de datos tradicionales en su lugar. Además, se han identificado las características individuales que se requieren principalmente para cada dominio de aplicación. Esto facilita la elección de la blockchain adecuada y los mecanismos correspondientes para adaptar la blockchain a las necesidades reales de la aplicación.

4.2 Smart Contracts

Los contratos inteligentes, como una funcionalidad adicional a la blockchain, han recibido una atención creciente recientemente. Son programas ejecutables cuya instancia y estado se almacenan en la blockchain. Por lo tanto, los contratos inteligentes y la blockchain permiten un protocolo confiable, rastreable e irreversible sin la necesidad de terceros confiables que generalmente constituyen un punto único de falla. Si un usuario crea y distribuye un contrato inteligente, otros podrán interactuar con él mientras que la blockchain subyacente asegura una ejecución confiable. En este apartado desglosaré las partes del estudio Recent Advances in Smart Contracts: A Technical Overview and State of the Art (Kemmoe y cols., 2020)

4.2.1 Historia de los Smart Contracts

Aunque las tecnologías de contratos inteligentes han comenzado a recibir más atención tanto de la industria como de la academia, existe una larga historia que va desde las criptomonedas hasta la blockchain y los contratos inteligentes. En 1991, se propuso un sistema difícil de manipular para estampar digitalmente documentos (Haber y Stornetta, 1991). En

este esquema, a un documento digital se le emitía un certificado que contenía la fecha de creación y la información sobre certificados emitidos previamente para otros documentos digitales, creando así un enlace que permitía recuperar y probar la fecha de creación de un documento.

Posteriormente, en 2008, se propuso un sistema de pago descentralizado llamado Bitcoin (Nakamoto, 2009), donde el historial de transacciones se almacena en un libro mayor distribuido hecho de bloques. Cada bloque contiene un conjunto de transacciones, un nonce, una marca de tiempo y un hash del bloque anterior, similar al enfoque utilizado en 1991. Esta idea fue la chispa que originó lo que hoy se conoce como blockchain. Una blockchain es una estructura de datos distribuida hecha de bloques de datos en la que un bloque está vinculado a otro a través de su valor hash. Tiene dos operaciones básicas: leer (para leer el contenido de los bloques) y agregar (para agregar nuevos bloques). Aunque un adversario podría intentar eliminar o modificar algunos bloques, esto requeriría cálculos extremadamente costosos, lo que lo hace inviable. La propiedad de lectura y agregación de la blockchain la convirtió en un medio adecuado para desarrollar diferentes sistemas de pago descentralizados, eliminando la necesidad de instituciones centralizadas como los bancos. Cualquier usuario puede transferir monedas a otros y participar en el proceso de verificación de transacciones.

Con el crecimiento de Bitcoin, se reconoció que la blockchain podría usarse para desarrollar soluciones descentralizadas en otros dominios, como la industria alimentaria, la salud, etc. Sin embargo, la arquitectura de Bitcoin, que solo soporta scripts para verificar y validar transacciones de moneda, no era suficiente para soportar tales aplicaciones. Por ello, se introdujo el protocolo de contratos inteligentes en la blockchain. En 1994, Nick Szabo introdujo los contratos inteligentes (Szabo, 1994), que son programas informáticos que replican las acciones descritas en los contratos físicos/tradicionales. Posteriormente, en 1996, se definieron los siguientes objetivos de un contrato inteligente: observabilidad, verificabilidad, privacidad y ejecutabilidad. Con su propiedad de lectura y agregación, Bitcoin y su lenguaje de scripts demostraron que la blockchain es una plataforma adecuada para implementar contratos inteligentes. En una blockchain, un contrato inteligente no puede ser modificado, puede ser fácilmente observado, verificado, autoejecutable y, dependiendo del modo de acceso de la blockchain, se puede lograr la privacidad. A pesar de que su lenguaje de scripts solo permite verificar y validar transacciones, Bitcoin puede considerarse la primera implementación de un contrato inteligente en la blockchain. Más tarde, en 2015, se creó Ethereum, una plataforma blockchain que presenta un sistema de pago descentralizado y un lenguaje Turing completo, lo que permite el desarrollo de una amplia variedad de contratos inteligentes en una blockchain.

La figura 4.5 ilustra la cronología de los Smart Contracts para manejar la información de una manera más visual y organizada.

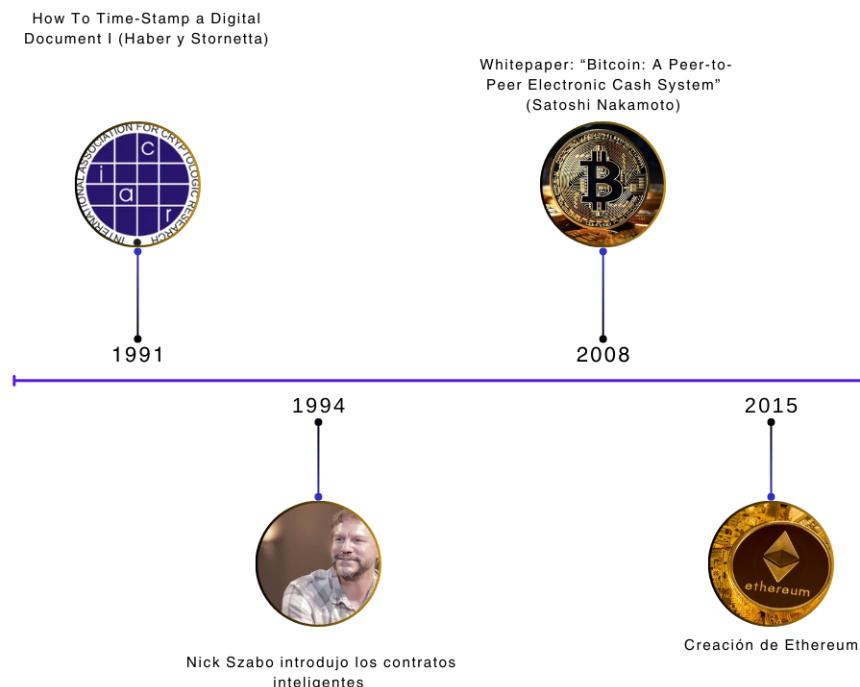


Figura 4.5: Cronología de los Smart Contracts.

4.2.2 Marco de Investigación

Los contratos inteligentes y la tecnología blockchain ofrecen un gran potencial para resolver problemas en diversos campos. Sin embargo, la falta de una estructura concreta ha llevado a los desarrolladores a crear continuamente nuevas aplicaciones y plataformas. Durante este proceso, a veces se descartan las ventajas o se recrean los defectos de versiones anteriores. Esta situación promueve una especie de anarquía en el campo, similar a los inicios de las redes informáticas antes de la presentación del modelo OSI, cuando cada ingeniero creaba diferentes modelos de red que no podían comunicarse entre sí.

Para organizar mejor la investigación sobre la tecnología de contratos inteligentes, se propone una arquitectura en capas. Esta arquitectura puede servir también como referencia para desarrollar futuras plataformas de blockchain y contratos inteligentes. La arquitectura propuesta se compone de seis capas:

- **Capa de Aplicaciones:** Representa la aplicación de contratos inteligentes en dominios específicos como las finanzas, la industria alimentaria, la salud, la logística y muchos otros. Aquí se desarrollan las soluciones que aprovechan las capacidades únicas de los contratos inteligentes para ofrecer servicios innovadores y eficientes.
- **Capa de Manifestaciones:** Representa las diferentes formas que pueden tomar las aplicaciones de contratos inteligentes, como las organizaciones autónomas descentralizadas (DAO), las cuales operan sin intervención humana directa, siguiendo únicamente

las reglas codificadas en sus contratos inteligentes.

- **Capa de Inteligencia:** Encapsula elementos que pueden hacer que un contrato inteligente sea capaz de aprender y adaptarse a situaciones presentadas, utilizando tecnologías como la inteligencia artificial y el aprendizaje automático para mejorar la toma de decisiones y la eficiencia operativa.
- **Capa de Operaciones:** Se centra en las operaciones relacionadas con la forma de un contrato inteligente, incluyendo la creación, ejecución, y terminación de contratos, así como la gestión de eventos y condiciones dentro del contrato.
- **Capa de Contratos:** Encapsula la lógica del contrato inteligente, es decir, cómo van a funcionar, incluyendo las reglas y condiciones que deben cumplirse para que se ejecuten las acciones definidas en el contrato.
- **Capa de Infraestructuras:** Consiste en todas las infraestructuras que apoyan el desarrollo de contratos inteligentes y sus aplicaciones, incluyendo la red blockchain subyacente, herramientas de desarrollo, plataformas de despliegue, y mecanismos de seguridad.

Además, se propone un conjunto de desafíos que enfrentan las implementaciones actuales de blockchain y contratos inteligentes, así como una serie de tendencias de desarrollo futuras. Estos desafíos incluyen la escalabilidad, la interoperabilidad entre diferentes sistemas blockchain, la privacidad y seguridad de los datos, y la eficiencia energética. Las tendencias futuras apuntan hacia la creación de estándares globales, la mejora de las técnicas de consenso, y la integración de tecnologías emergentes como la computación cuántica.

4.2.3 Direcciones Futuras

Tras un análisis de los artículos relacionados, se sugieren las siguientes direcciones futuras para la investigación:

- **Aplicaciones de Contratos Inteligentes:** Los contratos inteligentes proporcionan protocolos verificables y confiables sin la necesidad de un tercero de confianza. Debido a esta propiedad, pueden ser utilizados en muchas aplicaciones que involucran dispositivos con recursos limitados, como IoT, redes vehiculares ad-hoc (VANET), productos de hogar inteligente, entre otros. Dado que estos dispositivos tienen una capacidad limitada para almacenar protocolos verificados, incluyendo herramientas criptográficas, los contratos inteligentes serán una fuente potencial de rica funcionalidad.
- **Paralelización del Código:** A pesar de los esfuerzos por proponer marcos de ejecución paralela de contratos inteligentes, se cree que la paralelización en la ejecución de contratos inteligentes puede extenderse aún más para permitir el paralelismo de tareas y un grado limitado de paralelismo de datos. Por ejemplo, en el caso del paralelismo de tareas, un contrato inteligente que maneja una solicitud de préstamo puede tener que verificar si un cliente tiene suficiente dinero para el pago inicial y el seguro antes de proceder, lo que podría realizarse simultáneamente por diferentes hilos si el código lo permite. En el caso de un grado limitado de paralelismo de datos, un bucle que

solo lee de una ubicación de memoria dada y no sufre de dependencia de bucle puede ser ejecutado por múltiples hilos, cada uno manejando un número dado de iteraciones. Los marcos tradicionales como OpenMP, OpenCL o CUDA pueden considerarse como plantillas potenciales para producir una posible solución. Además, dado que dicha paralelización solo está al nivel del código del contrato inteligente, teóricamente, no hay riesgo de un problema con el protocolo de consenso. Por lo tanto, la existencia de un marco así podría mejorar el tiempo de ejecución de una transacción, mejorar la utilización de la CPU de los mineros y posiblemente reducir el costo de ejecución. Esto también podría ser una posible solución para resolver el problema planteado sobre las transacciones con un tiempo de ejecución significativo.

- **Ejecución Distribuida de Contratos Inteligentes por Dispositivos IoT:** Se observó que cuando los IoT se utilizan junto con contratos inteligentes, principalmente realizan transacciones de push (subir datos a una blockchain a través de un contrato inteligente) o transacciones de pull (obtener datos de una blockchain a través de un contrato inteligente). No se utilizan para ejecutar contratos inteligentes, en parte debido a su baja potencia de cómputo. Por lo tanto, se piensa que distribuir la ejecución de un programa de contrato inteligente entre un conjunto de nodos en lugar de tener un solo nodo ejecutando todo el programa no solo permitirá la ejecución de un contrato inteligente por un conjunto de IoTs, sino que también abrirá la puerta para la implementación de soluciones basadas en contratos inteligentes en entornos más restringidos.
- **Revocación de Certificados y Tokens:** Se observó que los esquemas que emiten certificados o tokens no proporcionan un mecanismo para revocarlos (los certificados y tokens pueden tener una duración de validez ilimitada) o centralizan el mecanismo de revocación delegándolo a entidades de confianza (lo que puede dejar un esquema abierto a ataques adversarios). Utilizar un esquema de firma basado en atributos en el que la fecha de expiración de un certificado/token sea uno de los atributos podría ser una posible solución. Con un esquema así, se podrá probar que un certificado/token sigue siendo válido si se verifica la firma asociada. Se propuso un esquema de encriptación que utiliza este enfoque para revocar claves delegadas.
- **Generación de Números Aleatorios:** Se propuso un PRNG para contratos inteligentes basado en un esquema de contribución. Sin embargo, una solicitud resulta en el retorno de un bit. Como posible mejora, se propone el uso del resultado devuelto después de aplicar XOR al valor hash de los números retenidos enviados por participantes honestos. Dependiendo de la función hash utilizada, esto puede proporcionar un número aleatorio de alta entropía en una sola ronda.
- **Ejecución de Contratos Inteligentes con NIZK como Prueba de Corrección:** Se observó que, a menos que el tipo de acceso de la plataforma blockchain sea privado, es difícil conservar la confidencialidad de los datos manejados. Incluso con acceso privado, la confidencialidad total de los datos no está garantizada ya que ciertas entidades de confianza aún tienen acceso a ellos. Por lo tanto, para proporcionar una confidencialidad más fuerte mientras se permite a cualquier parte verificar la corrección de las operaciones, se cree que una plataforma blockchain en la que la ejecución de contratos

inteligentes resulte en la producción de una prueba NIZK podría ser una posible solución. Esto permitiría a cualquier parte verificar que un contrato inteligente se ejecutó correctamente sin necesidad de acceder a los datos.

4.3 Web 3.0

Se considera que uno de los principios clave de Web3 es la transición de un ecosistema de internet dominado por unas pocas grandes empresas a uno en el que los usuarios individuales tienen más control sobre sus interacciones y datos en línea. En el modelo actual, las grandes corporaciones, como Google, controlan el tráfico y las búsquedas, gestionando y monetizando los datos de los usuarios.

Se permite una internet más distribuida con Web3, donde las interacciones sociales de los participantes no están gobernadas por unas pocas entidades gigantes. En cambio, los usuarios adquieren mayor autonomía sobre sus datos e interacciones. La propiedad de los datos se desplaza significativamente hacia los usuarios, marcando una desviación de los modelos centralizados actuales. Este cambio podría anunciar el surgimiento de nuevas corporaciones que faciliten interacciones descentralizadas y promuevan un entorno donde los datos y el control no están monopolizados.

Aunque se está en las primeras etapas de esta transición y es difícil predecir su impacto exacto, la idea central es que la concentración de atención y la propiedad de los datos entre unas pocas superpotencias tecnológicas disminuirá a medida que más aplicaciones adopten principios descentralizados.

A nivel técnico, la tecnología blockchain es el habilitador clave de esta descentralización. Blockchain proporciona un registro seguro e inmutable que soporta transacciones entre pares sin necesidad de intermediarios. Esto asegura que ninguna entidad tenga control sobre toda la red, mejorando la seguridad y la confianza en las interacciones en línea.

Se espera que Web3 evolucione para crear una internet donde los usuarios estén empoderados para controlar sus propios datos e interacciones, reduciendo la dominancia de unas pocas grandes corporaciones y fomentando un entorno digital más democrático.

4.3.1 Historia de la Web 3.0

Para entender plenamente en qué se ha convertido el internet, es necesario primero comprender sus inicios. Lo que comenzó como una red cerrada para científicos y académicos ha evolucionado significativamente. A pesar de los notables avances tecnológicos, una serie de errores ha seguido este progreso. La web actual está plagada de problemas que solo pueden apreciarse completamente al analizar sus raíces. Antes de explorar Web3, es fundamental revisar el pasado y comprender cómo se llegó al internet contemporáneo que se conoce, aprecia y, en ocasiones, se critica.

4.3.1.1 Nacimiento de Internet

Se considera que los orígenes del internet se remontan a 1966, cuando el Departamento de Defensa de los Estados Unidos comenzó a trabajar en un proyecto conocido como ARPANET.

ARPANET, la Red de la Agencia de Proyectos de Investigación Avanzada, tenía como objetivo crear un sistema para el acceso remoto y la comunicación entre computadoras. La primera iteración de ARPANET era una red descentralizada y de igual a igual, compuesta por un puñado de ordenadores en universidades estadounidenses.

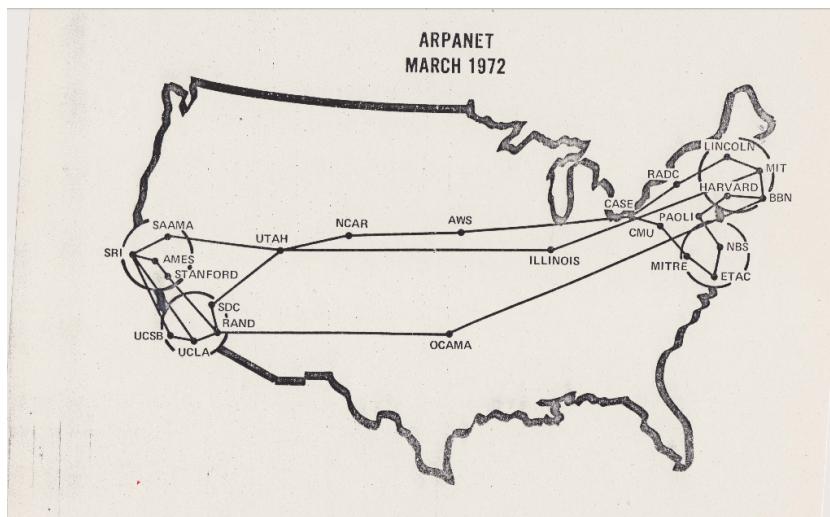


Figura 4.6: Mapa de Arpanet 1972 (Wikipedia contributors, 2023).

Aunque ARPANET recibe mucho crédito por ser pionero en el internet, no estaban solos. Redes similares se estaban creando en otras partes del mundo al mismo tiempo, pero no podían comunicarse entre sí. Esto se debía principalmente a diferencias en cómo estas redes formateaban la información y asignaban direcciones web. En lugar de la web global que se tiene hoy, estas primeras redes existían como intranets separadas.

Se puede considerar que el "internet" de esa época existía como pequeños pueblos, todos autosuficientes y desconocidos entre sí. Los miembros del mismo pueblo podían intercambiar información entre ellos, pero no tenían forma de comunicarse con los forasteros. Incluso si lograban contactar con un pueblo vecino, no hablarían el mismo idioma. Esto creó una serie fragmentada de pequeñas redes dispersas por todo el mundo.

Esto cambió en la década de 1970, principalmente con la introducción de TCP/IP. TCP/IP, o Protocolo de Control de Transmisión/Protocolo de Internet, estableció estándares para formatear información y estructurar direcciones web. Al adoptar estas nuevas directrices, las redes que anteriormente estaban separadas se volvieron compatibles entre sí. Todos estos pequeños pueblos aprendieron el mismo idioma y abrieron sus puertas entre ellos.

A medida que más y más redes adoptaron los estándares TCP/IP, comenzaron a formar una gran red. Este fue el nacimiento del internet.

4.3.1.2 Web 1.0

El concepto de la web como un sistema de información accesible para usuarios individuales surgió en la década de 1990 cuando los primeros sitios web y (pocos) desarrolladores web comenzaron a subir páginas estáticas.

El sistema fue inventado en 1989 y su implementación práctica fue pionera en 1990 por Tim Berners-Lee. La web en esta forma incipiente se concibió como una colección de documentos que podían ser accesibles a través de una red interconectada de computadoras (internet) que intercambiaban información entre sí. Berners-Lee también fue pionero en otros elementos fundamentales de la web moderna:

- **HTML (Hyper Text Markup Language):** El lenguaje de marcado utilizado por los navegadores web (aplicaciones que permiten a los usuarios acceder a la Web) para mostrar y renderizar páginas web.
- **URI/URLs (Unique Resource Identifier/Locator):** Las secuencias de caracteres y direcciones únicas que apuntan a la ubicación de una página web HTML dada.
- **HTTP (HyperText Transfer Protocol):** El marco que utilizan las computadoras para solicitar y enviar información entre sí.

En otras palabras, HTML es una colección de casas, las URIs/URLs son las direcciones de las casas y HTTP representa la carretera que las conecta en el vecindario.

Todo esto surgió en 1990, pero tomó toda la década para que la World Wide Web realmente despegara en popularidad, ya que esta arquitectura no fue concebida originalmente para ser utilizada por los consumidores.

A principios de la década de 1990, había menos de 100 páginas en línea. Al final del siglo XX, la web creció en adopción entre la población general después de que muchas empresas desarrollaran y lanzaran sus propios navegadores web y motores de búsqueda, señalando así un cambio tecnológico y económico masivo en menos de 10 años.

4.3.1.3 Web 2.0

La segunda fase de la web vio el auge de las compras en línea y el contenido generado por los usuarios. El siglo XXI fue testigo del boom del comercio electrónico y de gigantes de las redes sociales como Amazon, eBay, Twitter, YouTube y Facebook (ahora Meta).

Además, personas sin experiencia técnica comenzaron a desarrollar sus propias páginas y sitios web para blogs, vlogs, promoción personal y negocios. Esta fase transformó la web en un espacio interactivo donde los usuarios no solo consumían contenido, sino que también lo

creaban y compartían.

Se reconocen varias características clave en Web 2.0 que marcaron su evolución respecto a su predecesora. La experiencia del usuario se volvió mucho más rica gracias al desarrollo de contenido dinámico que permite a cualquier persona hacer clic, etiquetar, comentar e interactuar con las páginas web. Por ejemplo, la mayoría de los sitios web ahora muestran barras de navegación que permiten a los usuarios explorar secciones específicas o páginas web. Las plataformas de redes sociales también permiten a los usuarios etiquetar a otros para notificarles sobre actividades específicas, como publicaciones o comentarios, creando experiencias interactivas en el núcleo de estas plataformas.

El papel del usuario cambió de ser un mero consumidor de páginas web a convertirse en creador y participante activo en experiencias de redes sociales y comercio electrónico.

El número de usuarios de internet en todo el mundo se disparó a medida que la web se hizo accesible a la población en general, en lugar de estar restringida a un grupo de tecnólogos. Este aumento fue liderado por el desarrollo tecnológico en el poder de cómputo y las tecnologías de redes que permitieron a los usuarios web acceder a contenido en línea desde múltiples dispositivos (teléfonos, computadoras personales, tabletas). El lanzamiento del iPhone en 2008 es el ejemplo principal de esta transformación.

Las redes sociales y los gigantes tecnológicos en motores de búsqueda y comercio electrónico ganaron prominencia y tomaron una participación importante en el control del mercado en línea de los usuarios. Google llegó a ser sinónimo de buscar información en línea, mientras que Facebook se convirtió en el lugar de facto para conectarse en línea con la red social de uno. Amazon se convirtió en la plataforma central de comercio electrónico para usuarios de todo el mundo. La prominencia de estas plataformas revolucionó la mayoría de los comportamientos en línea de los usuarios.

4.3.1.4 Web 3.0

Web3, es una idea para una nueva iteración de la World Wide Web que incorpora conceptos como la descentralización, tecnologías blockchain y economías basadas en tokens. Algunos tecnólogos y periodistas lo han contrastado con Web 2.0, en la que los datos y el contenido están centralizados en un pequeño grupo de empresas, a veces referidas como "Big Tech". El término "Web3" fue acuñado en 2014 por el cofundador de Ethereum, Gavin Wood, y la idea ganó interés en 2021 entre los entusiastas de las criptomonedas, grandes empresas tecnológicas y firmas de capital de riesgo. Los conceptos de Web3 fueron representados por primera vez en 2013. (Wikipedia contributors, 2024b). MetaMask fue creado en 2016 por ConsenSys.

Se han expresado preocupaciones por parte de los críticos sobre la centralización de la riqueza en un pequeño grupo de inversores e individuos, o una pérdida de privacidad debido a una recopilación de datos más expansiva. Multimillonarios como Elon Musk y Jack Dorsey han argumentado que Web3 solo sirve como una palabra de moda o término de marketing.

Uno de los principios fundamentales de Web3 es la transición de un ecosistema de internet donde el tráfico y las búsquedas están dominados por unas pocas grandes empresas a uno nuevo donde los usuarios individuales de internet tienen más control sobre el flujo de comunicación en línea.

Se permite una internet más distribuida con Web3, donde el gráfico social de los participantes es más rico y las interacciones no son administradas ni controladas por unas pocas grandes corporaciones, como Google.

La propiedad de los datos se trasladará cada vez más al usuario y esta tercera fase de la Web puede señalar el advenimiento de una nueva ola de corporaciones que faciliten interacciones descentralizadas.

Aunque todavía es temprano y es difícil saber exactamente cómo será, el punto principal aquí es que la cuota de atención y propiedad de datos incrustada en unas pocas superpotencias tecnológicas disminuirá si más y más aplicaciones dependen de principios descentralizados.

La descentralización es un pilar conceptual clave de Web3, habilitado a nivel técnico por la tecnología blockchain.

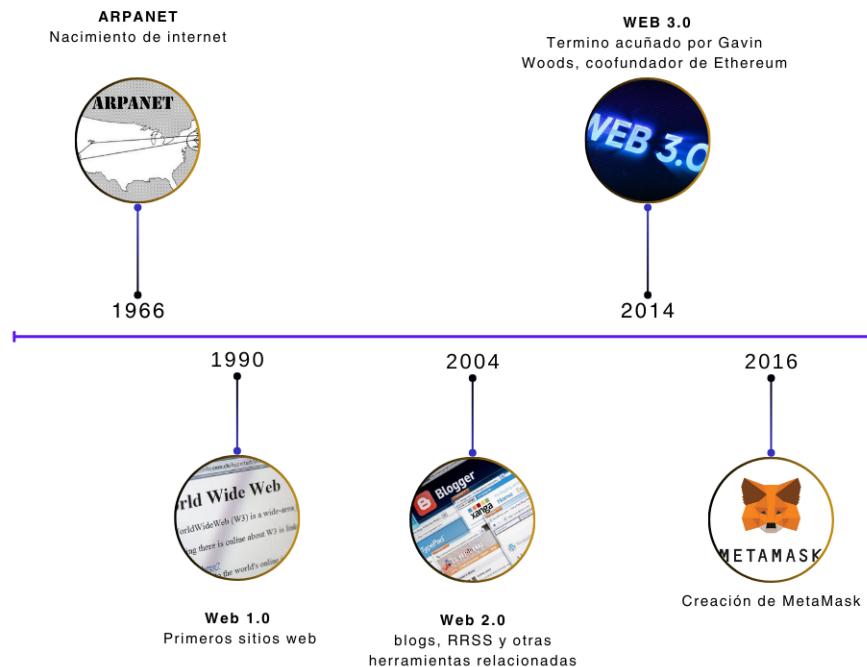


Figura 4.7: Cronología de la Web3.

5 Desarrollo Web3 de la Plataforma Electoral

En alineación con los objetivos establecidos, este apartado se centrará en el desarrollo de una aplicación web como ejercicio práctico para consolidar los conocimientos adquiridos en el marco teórico sobre las diferentes tecnologías mencionadas, específicamente Blockchain, Web3 y Smart Contracts. Este proyecto tiene como finalidad demostrar el funcionamiento de estas tecnologías, sin ahondar en los conceptos de la creación de una página web en sí, ya que no es el propósito principal de este Trabajo de Fin de Grado.

Se detallarán las herramientas y tecnologías utilizadas en la implementación de la aplicación, abordando únicamente aquellos aspectos que se encuentren dentro del alcance de este trabajo. La aplicación desarrollada se alojará en mi repositorio privado de GitHub (<https://github.com/caplopez>) hasta la presentación oficial del proyecto. A pesar de mantener la privacidad del repositorio hasta su exposición, se proporcionarán todos los códigos necesarios para comprender las distintas funcionalidades relacionadas con las tecnologías mencionadas.

Además, se incluirán descripciones detalladas de cómo se integran y funcionan estas tecnologías dentro de la aplicación web, ofreciendo una visión clara y precisa de su implementación. Este enfoque permitirá a los lectores obtener una comprensión práctica y aplicada de Blockchain, Web3 y Smart Contracts, demostrando su potencial y aplicación real en el desarrollo web.

La documentación proporcionará una guía paso a paso de los procesos seguidos, así como una explicación de las decisiones técnicas tomadas durante el desarrollo del proyecto, garantizando así una comprensión integral de las tecnologías y su implementación práctica.

5.1 Estudio Comparativo

En este apartado se expondrán diferentes trabajos realizados en el área de las tecnologías mencionadas. El objetivo es replicar el proceso creativo inherente a la creación de una aplicación. Como se menciona en la introducción de este trabajo, se pretende aprovechar al máximo las capacidades adquiridas durante el grado, y una de ellas es la investigación.

Es muy pertinente basarse en la experiencia obtenida para la creación de un nuevo proyecto en cualquiera de las áreas del conocimiento; por lo tanto, es pertinente basarnos en ideas, proyectos o propuestas realizadas con anterioridad. Este enfoque no solo permite una comprensión más profunda de las tecnologías, sino que también facilita la identificación de buenas prácticas y la implementación de soluciones probadas.

La exposición breve de estos trabajos servirá como una fuente de inspiración y un referente clave para la creación de la aplicación web propuesta, asegurando que se aprovechen al máximo las capacidades investigativas y creativas desarrolladas durante el grado.

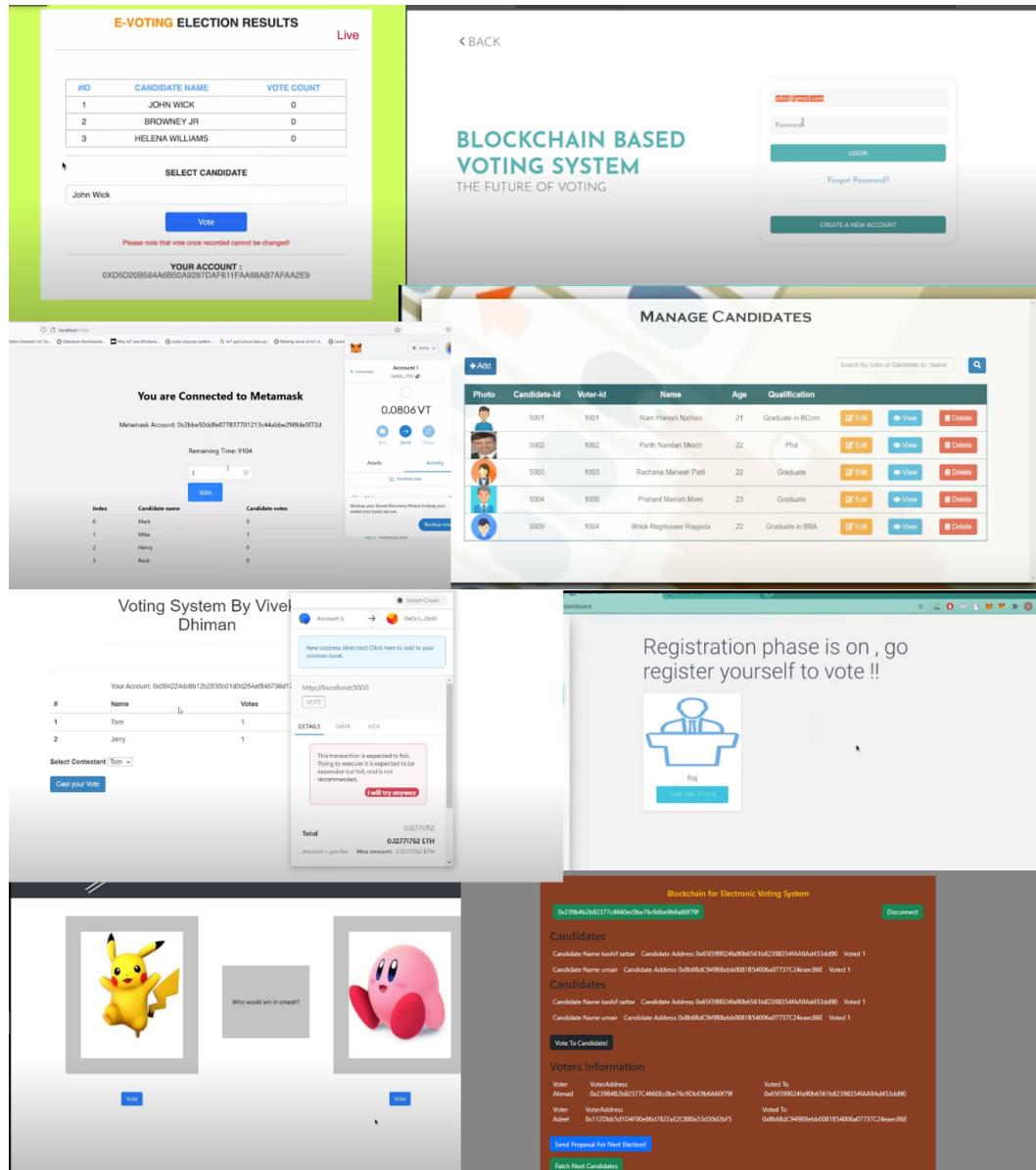


Figura 5.1: Ideas para la aplicación encontradas en la web.

Es evidente que existen múltiples ideas disponibles en la web que pueden servir como fuente de inspiración, y resulta altamente pertinente aprovecharlas. Las diversas ideas han sido identificadas en una variedad de fuentes, tales como repositorios en GitHub, páginas de proyectos propios de los contribuidores, y canales de YouTube.

Estas fuentes diversas proporcionarán una base sólida para contrastar las ideas propias con implementaciones reales. La revisión y análisis de estos proyectos permitirán evaluar la viabilidad y efectividad de las propuestas, asegurando que se fundamenten en soluciones prácticas y probadas. Además, este enfoque facilitará la identificación de mejores prácticas y métodos innovadores que pueden ser adaptados y mejorados en el desarrollo del proyecto actual.

En este sentido, la consulta de diferentes fuentes (KashifCh-eth, n.d; SamarthGhante, n.d; GitHub, n.d; Shifaj22, n.d; Krish-Depani, n.d; Baimamboukar, n.d) no solo enriquecerá el proceso creativo, sino que también garantizará que las decisiones técnicas y conceptuales estén bien informadas y alineadas con los estándares actuales de la industria. La integración de estas ideas contribuirá significativamente a la robustez y calidad del proyecto final, reflejando un enfoque riguroso y fundamentado en la investigación y el análisis crítico.

5.2 Herramientas Para el Desarrollo de la Dapp

Una vez realizado el estudio comparativo, se pueden tomar como punto de partida algunas herramientas identificadas en dicho análisis. Tal como se ha mencionado en esta sección, el objetivo es profundizar en las diferentes tecnologías blockchain, por lo que nos centraremos en estas al inspirarnos en la fase del estudio comparativo. A lo largo de este trabajo, se han abordado distintos frameworks que no están directamente relacionados con el desarrollo Web3. Sin embargo, como se ha indicado, no es objeto de este trabajo profundizar en ellos, por lo que se mencionarán de manera superficial.

El estudio comparativo ha revelado diversas herramientas y tecnologías que pueden ser cruciales para el desarrollo y la implementación de aplicaciones basadas en blockchain. Al enfocarnos en estas tecnologías, podemos asegurar que el proyecto aproveche las mejores prácticas y soluciones más adecuadas disponibles en el campo. Es importante destacar que, aunque se han explorado múltiples frameworks, el enfoque principal sigue siendo el desarrollo y la aplicación de tecnologías blockchain en un contexto Web3. Este enfoque garantizará que el trabajo se mantenga alineado con los objetivos iniciales y aporte un valor significativo en el área de estudio propuesta.

- **Solidity:** Es un lenguaje de programación utilizado principalmente para escribir contratos inteligentes en la plataforma Ethereum. Estos contratos inteligentes son programas que se ejecutan automáticamente cuando se cumplen ciertas condiciones, lo que permite la automatización de transacciones y acuerdos en la blockchain de Ethereum.
 - Escribir los contratos inteligentes como Elecciones.sol, que gestiona la lógica de la votación, la adición de candidatos, y el registro de usuarios.
 - **Truffle Suite:** Conjunto de herramientas de desarrollo para la blockchain de Ethereum, que facilita la escritura, prueba y despliegue de contratos inteligentes. Este framework incluye diversas funcionalidades que optimizan el proceso de desarrollo en Ethereum.
-

- Truffle Compile: Para compilar los contratos inteligentes escritos en Solidity.
 - Truffle Migrate: Desplegar los contratos compilados en una red de Ethereum (local o pública).
 - Truffle Test: Probar los contratos inteligentes para asegurarse de que funcionan como se espera.
- **Ganache:** Blockchain local utilizada para pruebas y desarrollo. Permite a los desarrolladores crear una blockchain simulada en su máquina local, donde pueden desplegar y probar contratos inteligentes sin incurrir en costos.
 - Crear una blockchain local donde se despliegan los contratos inteligentes para desarrollo y pruebas sin costo.
 - Permitir el seguimiento y depuración de las transacciones de prueba.
 - **Web3.js:** Biblioteca de JavaScript que permite interactuar con la blockchain de Ethereum desde el navegador o Node.js. Facilita la conexión entre las aplicaciones web y los contratos inteligentes desplegados en la blockchain.
 - Conectar la interfaz de usuario con los contratos inteligentes desplegados.
 - Manejar las transacciones y eventos en la blockchain.
 - Interactuar con MetaMask para autenticar y firmar transacciones.
 - **MetaMask:** Cartera digital que permite a los usuarios gestionar sus cuentas de Ethereum y firmar transacciones. Actúa como un puente entre los navegadores web y la blockchain de Ethereum, facilitando las interacciones seguras y descentralizadas.
 - Autenticar a los usuarios y gestionar sus cuentas de Ethereum.
 - Firmar transacciones de votación, registro y gestión de candidatos.
 - **Bootstrap, jQuery, NodeJS, npm:** Creación de interfaces web responsivas y dinámicas, simplificar la manipulación del DOM y gestionar eventos, así como ejecutar scripts de desarrollo y gestionar dependencias del proyecto. Estas herramientas aseguran una experiencia de usuario atractiva y una infraestructura de desarrollo eficiente.
 - Crear una interfaz de usuario atractiva y funcional.
 - Facilitar interacciones dinámicas en la página.
 - Ejecutar scripts de desarrollo y gestionar dependencias del proyecto.
 - Lite Server: Servidor de desarrollo ligero para servir los archivos estáticos del proyecto.
 - **Infura:** Proporciona acceso a la blockchain de Ethereum sin necesidad de ejecutar un nodo completo. Ofrece una API que permite a las aplicaciones interactuar con la blockchain de Ethereum de manera eficiente.
 - Conectarse a la red pública de Ethereum o redes de prueba cuando no se está utilizando Ganache.
 - Facilitar la interacción con la blockchain pública de Ethereum de manera eficiente.

En resumen, en la aplicación se utilizan diversas tecnologías clave para su desarrollo. Solidity, un lenguaje de programación para contratos inteligentes en Ethereum, se emplea para escribir contratos como Elecciones.sol, gestionando votaciones y candidatos. Truffle Suite facilita el proceso de compilar, desplegar y probar estos contratos inteligentes. Ganache proporciona una blockchain local para realizar pruebas sin costos. Web3.js permite la interacción entre la interfaz de usuario y los contratos inteligentes, gestionando transacciones. MetaMask se utiliza para autenticar usuarios y firmar transacciones. Bootstrap y jQuery se emplean para crear una interfaz de usuario atractiva, funcional y dinámica. Node.js junto con npm se utilizan para ejecutar scripts de desarrollo y gestionar las dependencias del proyecto, mientras que Infura permite la conexión a la red pública de Ethereum sin necesidad de ejecutar un nodo completo, facilitando la interacción eficiente con la blockchain.

En la tabla 5.1 se resumen de modo más visual las herramientas. Estas tecnologías no solo optimizan el desarrollo, sino que también garantizan una implementación segura y eficiente. Además, su integración asegura que la aplicación sea escalable y mantenga un alto rendimiento en diversas condiciones.

Tecnología	Descripción	Uso en la App
Solidity	Lenguaje de programación para contratos inteligentes en Ethereum.	Escribir contratos inteligentes como Elecciones.sol para gestionar votaciones y candidatos.
Truffle Suite	Framework de desarrollo para Ethereum.	Compilar, desplegar y probar contratos inteligentes.
Ganache	Blockchain local para pruebas.	Desplegar y probar contratos inteligentes sin costos.
Web3.js	Biblioteca JavaScript para interactuar con Ethereum.	Conectar la interfaz de usuario con contratos inteligentes y manejar transacciones.
MetaMask	Cartera digital para gestionar cuentas de Ethereum.	Autenticar usuarios y firmar transacciones de votación.
Bootstrap	Framework CSS para diseño web responsive.	Crear una interfaz de usuario atractiva y funcional.
jQuery	Biblioteca JavaScript para manipulación del DOM.	Manejar eventos de la interfaz de usuario y facilitar interacciones dinámicas.
Node.js & npm	Entorno de ejecución para JavaScript y gestor de paquetes.	Ejecutar scripts de desarrollo y gestionar dependencias del proyecto.
Infura	Acceso a la blockchain de Ethereum sin un nodo completo.	Conectar a la red pública de Ethereum y facilitar la interacción con la blockchain.

Tabla 5.1: Resumen de herramientas utilizadas en la aplicación.

5.3 Desarrollo Web3

Una vez identificadas las herramientas que se utilizarán para el desarrollo del proyecto, se procede a la descarga e instalación de los distintos programas siguiendo el orden descrito anteriormente. Con todas las herramientas instaladas, se crea el directorio del proyecto, denominado "Plataforma_Electoral". Este directorio contendrá todos los archivos necesarios para la Dapp.

A continuación, se detalla la estructura del directorio una vez finalizado el proyecto:

```

Plataforma_Electoral/
  build/
    contracts/ ... Archivos JSON generados automaticamente
                al hacer uso de truffle compile

  contracts/ ... Archivos de los diferentes Smart Contracts
    Elecciones.sol
    Migraciones.sol

  migrations/ ... Archivos de despliegue de los Smart Contracts
    1_Elecciones.js.
    2_Migraciones.js

  node_modules/ ... Archivos de los diferentes modulos nodeJS

  src/ ... Carperta contenedora de estilos,fuentes, imagenes...
    js/
      app.js ... Archivo JSON principal
      index.html ... Uno de los archivos .html

  bs-config.json
  package.json
  truffle-config.js ... Archivo de configuración para Ganache

```

5.4 Ganache y Truffle Suite

En el apartado 3.2.1.2 de este trabajo se detalla el uso de Ganache, una blockchain local utilizada para pruebas y desarrollo. En dicho apartado se proporciona el código de configuración inicial para su implementación que se obtiene al crear un proyecto con la herramienta **truffle**. No obstante, se ha procedido a optimizar dicho código para mejorar su eficiencia y funcionalidad de tal forma que habilitar el optimizador con **runs**: 200 es una buena práctica para producción, ya que reduce el tamaño del bytecode y los costos de gas en la blockchain:

Código 5.1: Configuración de Ganache

```

1 module.exports = {
2     networks: {
3         development: {
4             host: "127.0.0.1",
5             port: 7545,
6             network_id: "*" // Match any network id
7         },
8     },
9     compilers: {
10         solc: {
11             version: "0.8.0",
12             settings: {
13                 optimizer: {
14                     enabled: true,
15                     runs: 200
16                 }
17             }
18         }
19     }
20 };
21 };

```

Seguidamente, en Ganache se debe importar el archivo de configuración optimizado. La correcta importación del archivo de configuración permite una gestión precisa de las cuentas, balances y parámetros de gas, lo cual es esencial para la validación efectiva de las funcionalidades de la Dapp.

Para importar el archivo de configuración en Ganache, se deben seguir los siguientes pasos:

1. Iniciar Ganache y crear un nuevo workspace: Abrir la aplicación Ganache en su entorno local y seleccionar la opción de "New Workspace" (En mi caso ya había creado uno y en la figura 5.1 ya sale como "Plataforma Electoral").

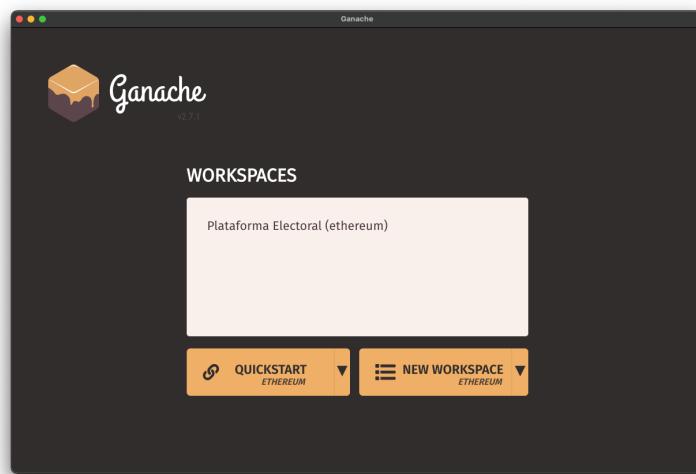


Figura 5.2: Página principal de Ganache.

2. Seleccionar la Opción de Importación: Navegar al menú de configuración de Ganache y

seleccionar la opción para importar un archivo de configuración.

3. Cargar el Archivo de Configuración: Ubicar y seleccionar el archivo de configuración optimizado previamente mencionado.

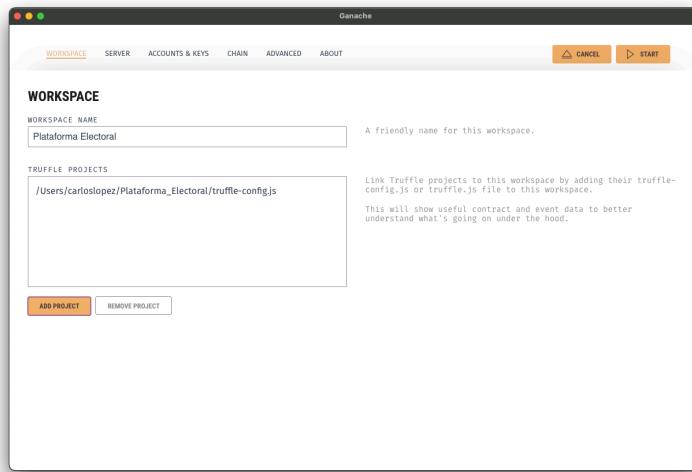


Figura 5.3: Página de creación del Workspace.

4. Verificar la Configuración Importada: Confirmar que los parámetros, cuentas y balances han sido correctamente importados según las especificaciones optimizadas.¹

ADDRESS	BALANCE	TX COUNT	INDEX	
0xA83598a1B31379837AaE3FDF342E080Ee9FE34e	100.00 ETH	0	0	🔗
0x1FeF69eD14242865741dfcc6F4b348bCd9949fc0	100.00 ETH	0	1	🔗
0x2180DC89E34C7258Df070F9f352124A740B9e9f5F	100.00 ETH	0	2	🔗
0xa4e0035f7f72c0be1EedDdf55bd474DE8Ec93628	100.00 ETH	0	3	🔗
0x4faf090E85ba570177ec6439500C876A0bc86ef6	100.00 ETH	0	4	🔗
0xa8485aFA4CEf3eAbcf9a87f27659A9B0E631246A	100.00 ETH	0	5	🔗
0xED0166CFc0FA90FdAc7b64A7F88Cb091261FFEc2	100.00 ETH	0	6	🔗

Figura 5.4: Página principal del Workspace con las diferentes cuentas creadas.

¹El workspace del proyecto final contiene cuentas diferentes ya que estas capturas se realizaron durante la fase de prueba y experimentación inicial.

Al crear un nuevo workspace en Ganache, se nos proporciona acceso a varias ventanas adicionales para la configuración avanzada del entorno de trabajo. Estas opciones permiten una personalización detallada que asegura un entorno de pruebas optimizado y alineado con las necesidades específicas del proyecto.

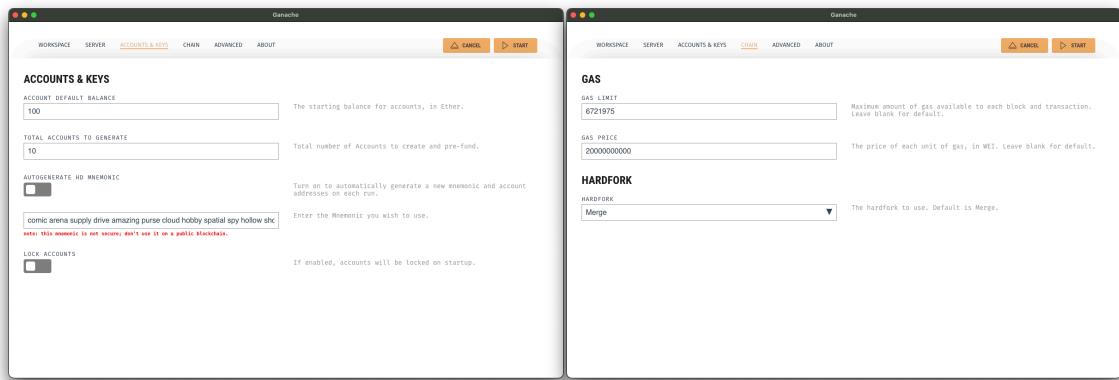


Figura 5.5: Configuración de cuentas

Figura 5.6: Configuración de la cadena

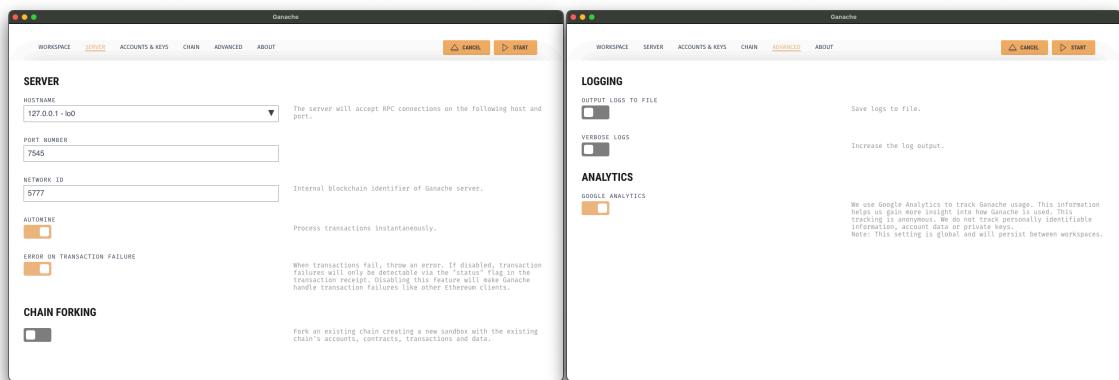


Figura 5.7: Configuración del servidor

Figura 5.8: Otras configuraciones

Entre las configuraciones disponibles, se incluyen:

- **Ajustes del Servidor:** Permiten modificar parámetros como el puerto en el que el servidor de Ganache estará escuchando y la configuración de red, asegurando que el entorno de pruebas sea accesible y funcional.
- **Gestión de Cuentas:** Facilita la configuración de las cuentas disponibles en el workspace, incluyendo el balance inicial de cada cuenta, el total de cuentas disponibles y la clave mnemónica que se utiliza para la generación de estas cuentas. Esto es crucial para simular diferentes escenarios de prueba.
- **Precio del Gas:** Permite ajustar el precio del gas, que es una medida de la cantidad

de trabajo que un nodo debe realizar para procesar una transacción. Ajustar este parámetro es vital para simular condiciones reales de red y evaluar el rendimiento de los contratos inteligentes bajo diferentes cargas de trabajo.

- **Configuración del Hardfork:** Permite seleccionar la versión del protocolo Ethereum (hardfork) que se utilizará en el workspace. Esta opción es fundamental para garantizar que los contratos inteligentes sean compatibles con las versiones específicas del protocolo Ethereum que se utilizarán en producción.

En la figura 5.9 se pueden observar las diferentes ventanas que proporciona el workspace de Ganache, así como información detallada del servidor, gas, bloques minados y otros datos relevantes. Estas ventanas permanecen vacías hasta el momento en que se utiliza el comando `truffle compile` en nuestra aplicación.



Figura 5.9: Vista de las diferentes ventanas en Ganache.

Al ejecutar `truffle compile`, se compilan los contratos inteligentes, lo que resulta en la aparición de información relevante en las ventanas de Ganache, indicando que los contratos están presentes en la blockchain, aunque aún no han sido desplegados. En este proceso se asegura que los contratos estén correctamente escritos y listos para ser implementados en la red.

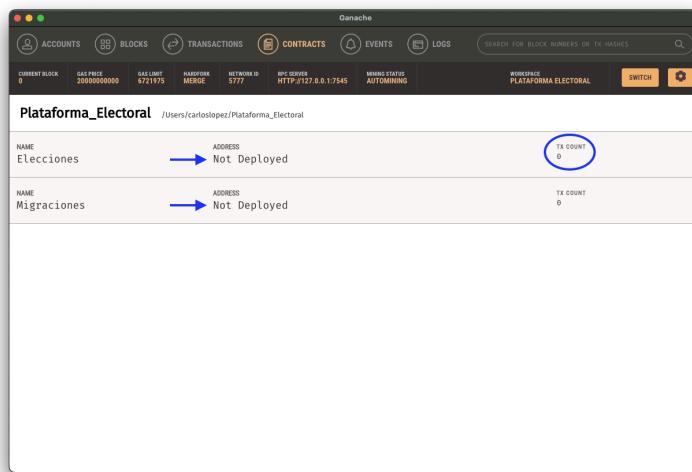


Figura 5.10: Contratos compilados.

El siguiente paso consiste en utilizar el comando `truffle migrate`, el cual se encargará de

desplegar los contratos inteligentes en la red de Ganache. Este comando utiliza los archivos ubicados en la carpeta migrations (1_Elecciones.js y 2_Migraciones.js) para gestionar el proceso de migración y despliegue de los contratos.

Código 5.2: Código para el despliegue del contrato "Migraciones.sol"

```

1 var Migraciones = artifacts.require("./Migraciones.sol");
2
3 module.exports = function(deployer) {
4   deployer.deploy(Migraciones);
5 };
6;
```

Código 5.3: Código para el despliegue del contrato "Elecciones.sol"

```

1
2 const Elecciones = artifacts.require("Elecciones");
3
4 module.exports = function (deployer) {
5   deployer.deploy(Elecciones);
6};
```

Al ejecutar el comando truffle migrate, la terminal proporciona información relevante acerca de los contratos inteligentes que se están desplegando en la red de Ganache. Esta información incluye detalles cruciales que permite verificar y validar el proceso de despliegue.

```

Starting migrations...
> Network name: 'development'
> Network id: 5777
> Block gas limit: 6721975 (0x6691b7)

1_migracion.js
Deploying 'Migraciones'
> transaction hash: 0x77a12be6e5d69aeeccb1f09b9d91570699e31809f988d06cbece8e5b3498bc32353
> Blocks: 0 Seconds: 0
> contract address: 0xcb3902b76785595dc4D7F8382D095264aB32e2a50
> block number: 1
> block timestamp: 1728866558
> account: 0xf5AC0fcb0F908d3338a69A3e522cc019Ae0eA4
> balance: 99,999,2314462
> gas used: 0 (0x379e1)
> gas price: 3,375 gwei
> value sent: 0 ETH
> total cost: 0.000768855375 ETH
> Saving artifacts
> Total cost: 0.000768855375 ETH

2_contratos.js
Deploying 'Elecciones'
> transaction hash: 0xb4f64fd9772ab21fe305c9b9dbcbef75d1bda199a6f764c299cdcb13e282fe03
> Blocks: 0 Seconds: 0
> contract address: 0x9f641d0908C4E5a1f4229a20854b97F63339F97
> block number: 2
> block timestamp: 1728866558
> account: 0xf5AC0fcb0F908d3338a69A3e522cc019Ae0eA4
> balance: 99,995,22725950369536
> gas used: 1223293 (0x12aa7d)
> gas price: 3,27303848 gwei
> value sent: 0 ETH
> total cost: 0.00400388506131464 ETH
> Saving artifacts
> Total cost: 0.00400388506131464 ETH
```

Figura 5.11: Información detallada del despliegue de los contratos.

Desglosemos la información²:

Información General de la Red

- **Network name:** development - Indica que la red de desarrollo local (Ganache) está siendo utilizada.
- **Network id:** 5777 - Identificador de la red de Ganache.
- **Block gas limit:** 6721975 - El límite de gas por bloque en la red de Ganache.

Despliegue de 1_migracion.js

- **Deploying 'Migraciones':** Informa que se está desplegando el contrato denominado 'Migraciones'.
 - **transaction hash:** 0x77a...32353 - El hash de la transacción de despliegue.
 - **Blocks:** 0 - Indica que el contrato se desplegó en el bloque génesis (bloque 0).
 - **contract address:** 0xeb3...e3e250 - Dirección en la blockchain donde el contrato ha sido desplegado.
 - **block number:** 1 - Número del bloque en el cual se incluyó la transacción.
 - **block timestamp:** 1712860656 - Marca de tiempo del bloque.
 - **account:** 0xbF5ACf0bF098d338a69aE522cc019aAe0eA4 - Cuenta que realizó la transacción de despliegue.
 - **balance:** 99.99921144625 - Balance restante de la cuenta después del despliegue.
 - **gas used:** 227809 - Cantidad de gas utilizada para el despliegue.
 - **gas price:** 3.375 gwei - Precio del gas en gwei.
 - **value sent:** 0 ETH - Valor en ETH enviado con la transacción (ninguno en este caso).
 - **total cost:** 0.000768855375 ETH - Costo total de la transacción en ETH.

Despliegue de 2_contratos.js

- **Deploying 'Elecciones':** Informa que se está desplegando el contrato denominado 'Elecciones'.
 - **transaction hash:** 0xb4f64fd9f77ab21fe35c9b9bdbe75d1bda1998a6f746c299cdcb13e82f7e03 - El hash de la transacción de despliegue.
 - **Blocks:** 0 - Indica que el contrato se desplegó en el bloque génesis (bloque 0).
 - **contract address:** 0x9F641a090B4e5a1f4229e2D854B97F63339F97 - Dirección en la blockchain donde el contrato ha sido desplegado.

²Nótese que en la captura realizada no concuerdan los nombres de los archivos .js del despliegue con los mencionados "1_Elecciones.js" y "2_Migraciones" ya que se realizó la captura durante la fase de prueba. A efectos prácticos, la información arrojada nos sirve con el objetivo de un entendimiento profundo de la información.

- **block number:** 2 - Número del bloque en el cual se incluyó la transacción.
- **block timestamp:** 1712860656 - Marca de tiempo del bloque.
- **account:** 0xbF5ACf0bF098d338a69aE522cc019aAe0eA4 - Cuenta que realizó la transacción de despliegue.
- **balance:** 99.9952722595638536 - Balance restante de la cuenta después del despliegue.
- **gas used:** 1229323 - Cantidad de gas utilizada para el despliegue.
- **gas price:** 3.2733848 gwei - Precio del gas en gwei.
- **value sent:** 0 ETH - Valor en ETH enviado con la transacción (ninguno en este caso).
- **total cost:** 0.0040038506131464 ETH - Costo total de la transacción en ETH.

Los bloques que contienen las transacciones de despliegue de los contratos inteligentes se registran como transacciones en la blockchain local de Ganache, y usualmente, estas transacciones son realizadas desde la primera cuenta generada por Ganache. Esta primera cuenta suele tener un balance inicial elevado para cubrir los costos de gas asociados con el despliegue de contratos. Suele ser la que se utiliza por defecto para ejecutar las transacciones de despliegue, a menos que se especifique lo contrario en la configuración de Truffle o en el script de migración. Veremos más adelante como manejar esta cuenta que va ser la del administrador en la Dapp.

ADDRESS	BALANCE	TX COUNT	INDEX	
0x9666a2B5993afb647816650fa6c185eBCD0401F1	100.00 ETH	2	0	
0xc6a8Ca9683A205f483ae1d3611F733ff1dB277fc	100.00 ETH	0	1	
0x09886b78b9b8777d8150899CF671662FB7702df0	100.00 ETH	0	2	
0xD170E0477a5cf5fC6a16429550D692C7667F47	100.00 ETH	0	3	
0x4741533DB8d37378Ee82320040083E8c454dC521	100.00 ETH	0	4	
0xc7C85C5a44Cc43C339dC0129E7C98e8811F19B4E	100.00 ETH	0	5	
0xFDDDBD3CB28594Fc486152B02Db67CFeCb65c78d	100.00 ETH	0	6	

Figura 5.12: Registro de cuentas donde aparece las transacciones (2) de los contratos deplegados.

En la ventana "Transacciones", la misma información que aparecía en la terminal se presenta de manera más visual y clara. En la figura 5.13, se destaca en un recuadro rojo la información que indica que estos contratos han sido creados con éxito.

Además, podemos confirmar que los contratos se crean en la misma cuenta que, por defecto, es la primera que aparece en Ganache, como se muestra en la figura 5.12 (0x9666a...401F1).

También se detalla el gas utilizado para la transacción, que en esencia es la firma del contrato, así como la dirección del contrato creado y el hash de la transacción. Este hash sirve como una huella digital de la transacción, asegurando su integridad y permitiendo su verificación y seguimiento.

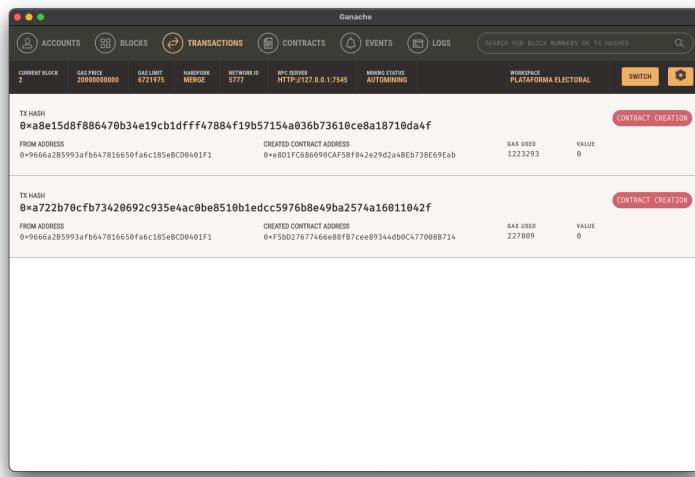


Figura 5.13: Registro de las transacciones al desplegar los contratos inteligentes.

Podemos observar cómo, en la ventana de contratos, donde anteriormente aparecía "Not Deployed" y 0 transacciones (figura 5.10), ahora se muestra "Deployed" junto con la dirección del contrato. Además, podemos visualizar el contenido de cada contrato al hacer clic en cada uno de ellos. Aparece una ventana con toda la información relevante del contrato. Como se muestra en la figura 5.13, aparece una dirección de contrato que coincide en esta ventana (0xe8D1F...69Eab y 0xF5bD2...8B714).

Cabe destacar que también se muestra un hash de la transacción de creación del contrato, que es diferente al hash de la transacción mencionado en la figura 5.13. Esto se debe a que el tx hash es el hash de la transacción que se genera cuando se envía la solicitud para desplegar el contrato inteligente, mientras que el creation tx hash es el hash específico asociado con la creación del contrato inteligente. El tx hash identifica de manera única la transacción en la blockchain, incluyendo detalles como la dirección de la cuenta que realizó la transacción, la cantidad de gas utilizada, y la dirección del contrato desplegado. Por otro lado, el creation tx hash proporciona un identificador único para la transacción que resultó en la creación del contrato, permitiendo un seguimiento preciso de esta operación dentro del contexto del contrato mismo.

Podemos observar, además, cómo nos aparece el contenido del contrato, el cual explicaremos más adelante. Esta ventana de detalles proporciona una visión completa del contrato inteligente, incluyendo su código fuente, las variables almacenadas y las funciones disponibles. También permite la interacción directa con el contrato, facilitando la ejecución de funciones específicas. Además, ofrece información sobre las transacciones y el estado actual del contrato.

The figure consists of two side-by-side screenshots of the Ganache interface. Both screenshots show the same basic layout with tabs for Accounts, Blocks, Transactions, Contracts, Events, Logs, and a search bar. The left screenshot is titled "Migraciones" and the right one is titled "Elecciones". Under the "TRANSACTIONS" tab, both show a single transaction each. The transaction details are as follows:

Migraciones Transaction:

```

AD06E8B
0x750D27677A66E8F87Cee8934d4b8C47700B8714
BALANCE 0, 00 ETH
0x9a72709c0f0b734c28652c9354AC0BE85108EdCC5976884a98A2574A1601804f
0x9a72709c0f0b734c28652c9354AC0BE85108EdCC5976884a98A2574A1601804f

```

Elecciones Transaction:

```

AD06E8B
0x4e01fC686098CAF58FBa2e29d2a4BEB738E69Eab
BALANCE 0, 00 ETH
0x9a72709c0f0b734c28652c9354AC0BE85108EdCC5976884a98A2574A1601804f
0x9a72709c0f0b734c28652c9354AC0BE85108EdCC5976884a98A2574A1601804f

```

Below the transactions, there are sections for "STORAGE" and "EVENTS" which are currently empty.

Figura 5.14: Información detallada del contrato Migraciones.

Figura 5.15: Información detallada del contrato Elecciones.

En la pestaña de bloques, se observa cómo se han minado dos bloques adicionales al bloque génesis. Cada uno de estos bloques corresponde a una transacción específica. Además, se puede notar que existe una trazabilidad temporal.

Al ingresar en cada bloque, se muestra la misma información que aparece en la pestaña de transacciones. Esto demuestra el ciclo completo de trazabilidad e inmutabilidad, características fundamentales que se han mencionado a lo largo de este trabajo.

Una vez que estos datos se almacenan en la blockchain, cualquier persona puede acceder a ellos, garantizando la transparencia. Esto no solo representa una ventaja competitiva significativa, sino que también beneficia a la sociedad en general, permitiendo a las personas de a pie verificar la autenticidad de la información y confiar en la integridad de los datos almacenados.

The figure consists of two side-by-side screenshots of the Ganache interface. The left screenshot shows a list of three blocks: BLOCK 2, BLOCK 1, and BLOCK 0. The right screenshot shows detailed information for BLOCK 2.

Block 2 Details:

BLOCK	MINED ON	GAS USED
2	2024-07-13 12:57:27	1223293
1	2024-07-13 12:57:27	257899
0	2024-07-13 12:57:05	0

Block 2 Transaction:

FROM ADDRESS	CREATED CONTRACT ADDRESS	GAS USED	VALUE
0x9a66a28593a3fb647816659fadC154bC00401F1	0x9950853a51cfcc1a900f1d42a9cb0b694e36684e2202832c40a4c78a4b78d173	1223293	0

Figura 5.16: Vista general de la cadena de bloques.

Figura 5.17: Información detallada del Bloque 2.

5.5 Metamask

El siguiente paso para poder interactuar con la DApp es crear una cuenta en MetaMask. Por motivos didácticos, asumiremos que este proceso ya se ha completado adecuadamente y pasaremos directamente a añadir la red de Ganache a MetaMask.

En MetaMask, debemos seguir la ruta: Redes > Agregar una red > Agregar una red manualmente. Una vez en esta pestaña, insertamos los datos de la red de Ganache. MetaMask necesita una URL RPC para conectarse a un nodo Ethereum. Esta URL puede ser proporcionada por un nodo local como Ganache, un proveedor de servicios como Infura, o directamente a un nodo completo que se esté ejecutando.

Ganache expone un servidor RPC que simula un nodo Ethereum localmente. La URL RPC típica de Ganache es `http://localhost:7545`. Además, debemos insertar el identificador de la cadena (Chain ID) que, en este caso, es 1337. Estos datos aparecen en Ganache en todo momento debajo de las diferentes ventanas seleccionables. Es importante notar que el Chain ID no es lo mismo que el Network ID, que en Ganache aparece como 5777.

- **Network ID:** Es un identificador numérico utilizado por Ethereum y otras blockchains para identificar una red específica. Es utilizado principalmente por los nodos de la red para conectarse entre sí y asegurarse de que están en la misma red.
- **Chain ID:** Es un identificador adicional introducido con EIP-155 que ayuda a prevenir ataques de repetición entre diferentes redes de Ethereum (por ejemplo, Mainnet, Ropsten, Kovan). Cada red tiene un Chain ID único.

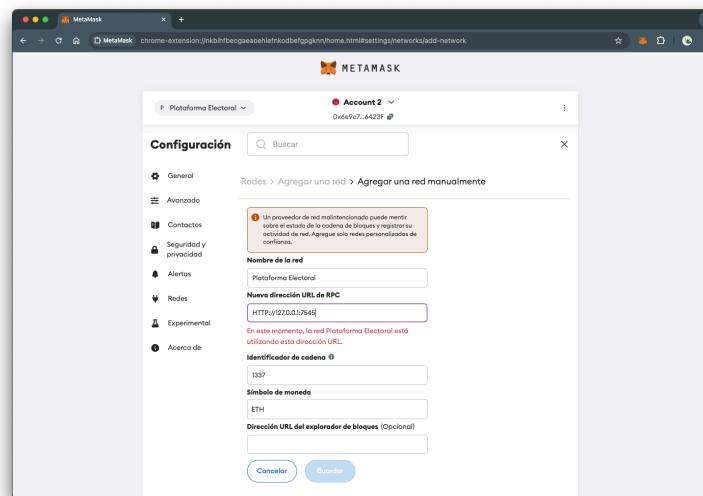


Figura 5.18: Agregando una red manualmente en Metamask.

Una vez añadida la red, procederemos a añadir las diferentes cuentas que nos proporciona Ganache. Al hacer clic en la pestaña superior de MetaMask, donde se muestra la cuenta con

la que estamos conectados, se abrirá una ventana que presenta un botón para añadir una cuenta. Inicialmente, aparece como si ya estuviéramos conectados con una cuenta, pero esta cuenta es la predeterminada de MetaMask y no corresponde con ninguna de las cuentas de Ganache. Por lo tanto, debemos añadir cada una manualmente. Para ello, seleccionamos la opción "Importar cuenta" y luego utilizamos la clave privada proporcionada por Ganache para cada cuenta. De esta forma, aseguramos que todas las cuentas de Ganache se añaden correctamente a MetaMask, permitiendo la interacción adecuada con la red de desarrollo.

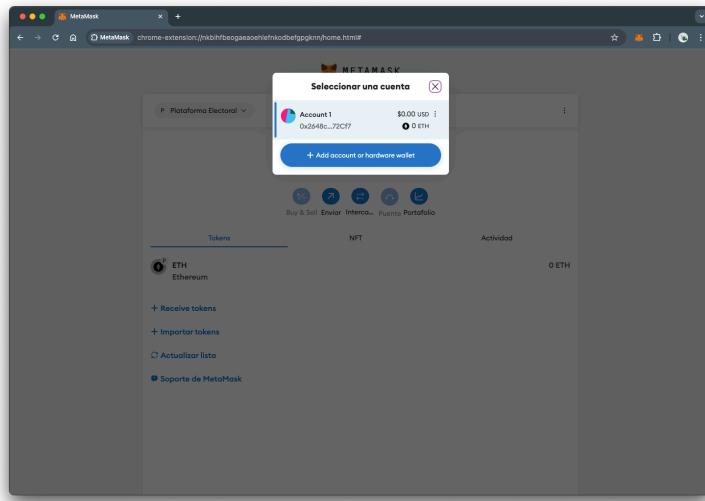


Figura 5.19: Registro de cuentas de Metamask.

En Ganache, en cada cuenta, aparece a la izquierda un botón con el símbolo de una llave (Figura 5.20). Al hacer clic en este botón, se abre una pestaña que contiene la clave privada de la cuenta. Esta clave privada será utilizada para añadir la cuenta en MetaMask. Para importar la cuenta en MetaMask, seguimos el procedimiento mencionado anteriormente y pegamos la clave privada obtenida de Ganache.

Una vez realizado todo esto, la cuenta añadida aparecerá en MetaMask con un balance inferior a 100 ETH, ya que se ha gastado una parte al desplegar los contratos. A continuación, también añadiremos una cuenta de usuario para poder realizar las pruebas correspondientes. Este proceso se sigue de la misma manera: obteniendo la clave privada de la cuenta de usuario en Ganache y utilizándola para importar la cuenta en MetaMask. Comprobamos que efectivamente la cuenta de Usuario tiene un balance de 100 ETH ya que no ha realizado ninguna transacción de momento (Figuras 5.21 y 5.22).

Con estas cuentas configuradas, estaremos listos para interactuar con la DApp y verificar que todas las funcionalidades operan correctamente en el entorno de desarrollo.

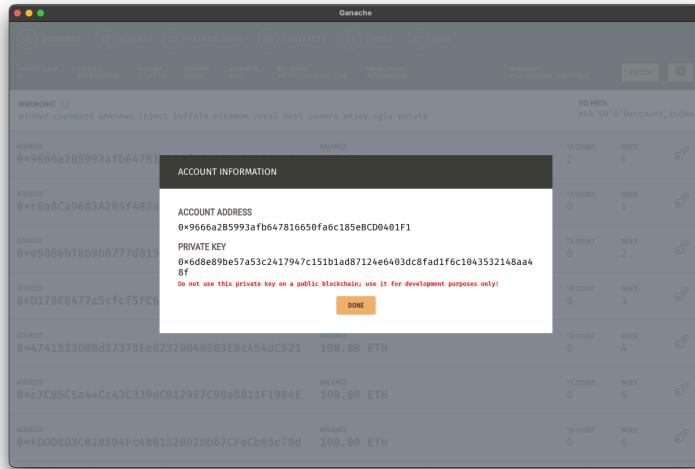


Figura 5.20: Pestaña con la clave privada de cada cuenta en Ganache.

Figura 5.21: Cuenta de Administración.

Figura 5.22: Cuenta de Usuario.

5.6 Plataforma Electoral

Una vez realizados todos estos pasos, podemos pasar a la DApp e interactuar correctamente con ella. Es importante destacar que el objetivo principal de este trabajo no es explicar en detalle el desarrollo de una página web. Por lo tanto, procederemos directamente a explicar en profundidad las diferentes interacciones con la blockchain que la DApp creada permite.

Esta sección se centrará en cómo la DApp utiliza contratos inteligentes desplegados, cómo se gestionan las transacciones en la blockchain y cómo los usuarios pueden interactuar con estas funcionalidades de manera segura y eficiente.

Una vez tenemos todo listo, estamos preparados para compilar nuestra DApp. Utilizamos el comando `npm start` para compilar el proyecto. Este comando inicia el servidor de desa-

rrollo y compila el código fuente de la DApp, preparándolo para ser ejecutado en el navegador.

Al abrir nuestra DApp en el navegador, lo primero que aparece es un pop-up de la extensión de MetaMask. Este pop-up solicita que iniciemos sesión con una de las cuentas que hemos añadido anteriormente en MetaMask. MetaMask detecta automáticamente que la DApp requiere autenticación y nos proporciona una interfaz segura para seleccionar y autenticar una de nuestras cuentas.

Una vez iniciada la sesión en MetaMask, la DApp puede interactuar con la blockchain utilizando la cuenta seleccionada. Esto permite realizar transacciones, firmar mensajes y ejecutar funciones de contratos inteligentes, asegurando que todas las acciones se realicen de manera segura y verificable. La integración con MetaMask es esencial para garantizar que los usuarios puedan interactuar con la DApp de forma fluida y segura, aprovechando todas las capacidades de la blockchain.

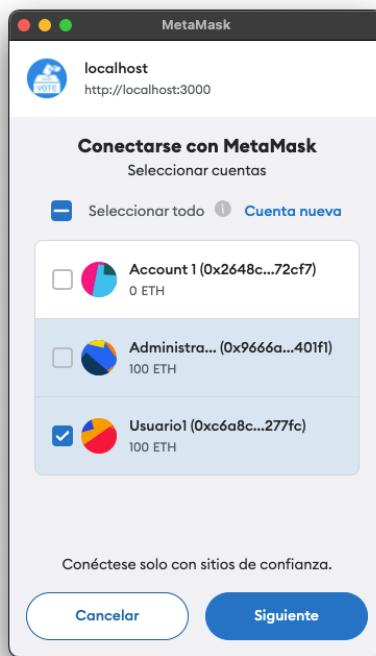


Figura 5.23: Inicio de sesión en Metamask al inicializar la Dapp.

La inicialización de la conexión a MetaMask se realiza en la función initWeb3 dentro de app.js. Este proceso implica la configuración de Web3 con el proveedor de MetaMask y la solicitud de acceso a las cuentas del usuario.

Código 5.4: Inicialización de la conexión a MetaMask

```
1 initWeb3: async function () {
```

```

3  console.log("App.initWeb3 called");
4  if (window.ethereum) {
5      App.web3Provider = window.ethereum;
6      try {
7          // Solicitar acceso a las cuentas del usuario
8          await window.ethereum.request({ method: 'eth_requestAccounts' });
9          document.querySelector('.loading').style.display = 'none'; // Ocultar el círculo de carga
10     } catch (error) {
11         console.error("El usuario denegó el acceso a la cuenta");
12     }
13 } else if (window.web3) {
14     App.web3Provider = window.web3.currentProvider;
15 } else {
16     App.web3Provider = new Web3.providers.HttpProvider('http://localhost:7545');
17 }
18 web3 = new Web3(App.web3Provider);
19 web3.eth.defaultAccount = web3.eth.accounts[0];
20 console.log("Web3 initialized");
21 return App.initContract();
22 }

```

Estos pasos del código aseguran que la aplicación esté conectada a MetaMask y pueda interactuar con la blockchain utilizando las cuentas de MetaMask:

1. Verificar la disponibilidad de `window.ethereum`:

- Comprueba si MetaMask está instalado en el navegador.

2. Configurar el proveedor de Web3:

- Si MetaMask está disponible, asigna `window.ethereum` a `App.web3Provider`.

3. Solicitar acceso a las cuentas de MetaMask:

- Llama a `window.ethereum.request({ method: 'eth_requestAccounts' })` para obtener permiso del usuario.

4. Manejo de la compatibilidad con versiones anteriores:

- Si `window.ethereum` no está disponible, verifica `window.web3` para versiones anteriores de MetaMask y asigna `window.web3.currentProvider` a `App.web3Provider`.

5. Configurar un proveedor HTTP de respaldo:

- Si ni `window.ethereum` ni `window.web3` están disponibles, usa `Web3.providers.HttpProvider` como proveedor de respaldo.

6. Inicializar Web3:

- Crea una nueva instancia de Web3 con el proveedor configurado.
- Establece la cuenta por defecto a la primera cuenta en la lista de cuentas de Web3.

7. Inicializar el contrato inteligente:

- Llama a `App.initContract()` para cargar y configurar los contratos inteligentes.

En la aplicación, la distinción entre si una cuenta es administrador o usuario se hace en la función `render` del archivo `app.js`. Aquí se verifica si la cuenta actual es la cuenta del administrador del contrato.

Código 5.5: Función render

```

1
2 render: function () {
3     console.log("App.render called");
4     var eleccionesInstance;
5     var loader = $("#loader");
6     var content = $("#content");
7
8     loader.show();
9     content.hide();
10
11    web3.eth.getCoinbase(function (err, cuenta) {
12        if (err === null) {
13            App.cuenta = cuenta;
14            $("#accountAddress").html("Tu cuenta: " + cuenta);
15        }
16    });
17
18    App.contracts.Elecciones.deployed().then(function(instance) {
19        eleccionesInstance = instance;
20        console.log("Contrato Elecciones desplegado");
21        if (typeof eleccionesInstance.administrador === "function") {
22            return eleccionesInstance.administrador();
23        } else {
24            console.error("administrador no es una función");
25        }
26    }).then(function (administrador) {
27        if (administrador && administrador !== App.cuenta) {
28            document.querySelector('.buy-tickets').style.display = 'none';
29        }
30        return eleccionesInstance.cantidadCandidatos();
31    }).then(function(cantidadCandidatos) {
32        var candidatesResults = $("#candidatesResults");
33        candidatesResults.empty();
34
35        var candidatesSelect = $('#candidatesSelect');
36        candidatesSelect.empty();
37
38        for (var i = 1; i <= cantidadCandidatos; i++) {
39            eleccionesInstance.candidatos(i).then(function(candidate) {
40                var id = candidate[0];
41                var fname = candidate[1];
42                var lname = candidate[2];
43                var idNumber = candidate[3];
44                var voteCount = candidate[4];
45
46                var candidateTemplate = '<tr><th>${id}</th><td>${fname} ${lname}</td><td>${←
47                ↪ idNumber}</td><td>${voteCount}</td></tr>';
48                candidatesResults.append(candidateTemplate);
49
50                var candidateOption = '<option value=' + ${id} + '>${fname} ${lname}</option>';
51                candidatesSelect.append(candidateOption);
52            });
53        }
54        return eleccionesInstance.votantes(App.cuenta);
55    }).then(function(haVotado) {
56        if (haVotado) {
57            $('form').hide();
58            $("#index-text").html("¡Has iniciado sesión con éxito!");
59            $("#new-candidate").html("No se pueden agregar nuevos candidatos. El proceso de elección ←
60            ↪ ya ha comenzado.");
61            $("#vote-text").html("Voto emitido con éxito para el candidato " + localStorage.getItem("←
62            ↪ votadoPorID"));
63        }
64    });
65}

```

```

61     loader.hide();
62     content.show();
63     return eleccionesInstance.cantidadUsuarios();
64   }).then(function (cantidadUsuarios) {
65     var voterz = $("#voterz");
66     voterz.empty();
67
68     for (var i = 1; i <= cantidadUsuarios; i++) {
69       eleccionesInstance.usuarios(i).then(function (user) {
70         var firstName = user[0];
71         var lastName = user[1];
72         var idNumber = user[2];
73         var email = user[3];
74         var address = user[5];
75
76         let voterTemplate = '<tr><td>${firstName} ${lastName}</td><td>${idNumber}</td><td>${email}</td><td>${address}</td></tr>';
77         voterz.append(voterTemplate);
78     });
79   }
80
81   if (localStorage.getItem("finalizarEleccion") === "1") {
82     $('form').hide();
83     $("#index-text").html("No hay elecciones activas en este momento");
84     $("#vote-text").html("No hay votaciones activas");
85     document.querySelector('.addCandidateForm').style.display = 'block';
86     document.querySelector('.vot').style.display = 'none';
87   }
88 }).catch(function(error) {
89   console.warn(error);
90 });
91 },

```

Esta función realiza diversas tareas, pero entre ellas hace una distinción crucial entre administrador y usuario. Es importante hacer esta distinción ya que al usuario no se le mostrará el panel de administración. Para implementar esta funcionalidad, es necesario consultar el contrato inteligente para verificar si se ha iniciado sesión en MetaMask con la cuenta de administración o con una cuenta de usuario.

El contrato inteligente contiene una variable que identifica al administrador. Al iniciar la DApp, se realiza una llamada al contrato para obtener esta información. Si la cuenta conectada en MetaMask corresponde a la del administrador, se muestra el panel de administración; de lo contrario, solo se muestran las funcionalidades accesibles para los usuarios. Este enfoque garantiza que solo el administrador tenga acceso a las funciones de gestión y configuración, mientras que los usuarios pueden interactuar con la DApp de manera segura y controlada.

Esta separación de roles es fundamental para mantener la integridad y seguridad de la DApp, asegurando que las funciones críticas solo sean accesibles para las cuentas autorizadas. Al implementar esta verificación, se protege la DApp contra accesos no autorizados y se garantiza que cada usuario tenga acceso solo a las funcionalidades que le corresponden.

1. Inicialización y Carga:

- Variables y elementos de la interfaz: `var loader = $("#loader"); var content = $("#content");`
- Mostrar indicador de carga: `loader.show();`
- Ocultar contenido principal: `content.hide();`

2. Obtención de la Cuenta de MetaMask:

- Obtener la cuenta: `web3.eth.getCoinbase(function (err, cuenta) { ... });`
- Almacenar y mostrar la cuenta:
`App.cuenta = cuenta; $("#accountAddress").html("Tu cuenta: " + cuenta);`

3. Despliegue del Contrato y Verificación del Administrador:

- Desplegar el contrato:
`App.contracts.Elecciones.deployed().then(function(instance) { ... });`
- Verificar el administrador: `return eleccionesInstance.administrador();`
- Ocultar elementos no visibles para el administrador:
`if (administrador && administrador !== App.cuenta)
{ document.querySelector('.buy-tickets').style.display = 'none'; }`

4. Obtención y Visualización de Candidatos:

- Obtener el número de candidatos:
`return eleccionesInstance.cantidadCandidatos();`
- Limpiar elementos HTML:
`candidatesResults.empty(); candidatesSelect.empty();`
- Iterar y mostrar candidatos:
`for (var i = 1; i <= cantidadCandidatos; i++)
{ eleccionesInstance.candidatos(i).then(function(candidate) { ... });
}`

5. Verificación de Votación del Usuario:

- Verificar si el usuario ha votado:
`return eleccionesInstance.votantes(App.cuenta);`
- Ocultar formulario de votación y mostrar mensajes:
`if (haVotado) { $('form').hide(); ... }`
- Mostrar contenido principal: `loader.hide(); content.show();`

6. Obtención y Visualización de Usuarios Registrados:

- Obtener el número de usuarios: `return eleccionesInstance.cantidadUsuarios();`
 - Limpiar elementos HTML: `var voterz = $("#voterz"); voterz.empty();`
 - Iterar y mostrar usuarios: `for (var i = 1; i <= cantidadUsuarios; i++) { eleccionesInstance.usuarios(i).then(function (user) { ... }); }`
-

7. Estado de Finalización de la Elección:

- Verificar y manejar el estado de finalización:

```
if (localStorage.getItem("finalizarEleccion") === "1") { ... }
```

En la pantalla de inicio de la DApp, una vez registrados como administrador, aparece en la parte inferior un recuadro lila con la cuenta con la que hemos iniciado sesión. Como se puede observar, es la cuenta de administrador que aparece en Ganache en la figura 5.12 (0x9666a...401F1). Además, se presenta un panel de registro que es meramente didáctico y sirve como ejemplo para mostrar cómo se rellenan estos datos. Este panel interactúa con los contratos inteligentes, almacenando toda la información en la blockchain.

Es importante mencionar esto ya que, en realidad, estamos utilizando una herramienta Web3. Por lo tanto, nuestra información podría estar almacenada si interactuamos con protocolos de Identidad Descentralizada, lo cual es lo ideal para este tipo de DApps relacionadas con la ciudadanía. Estos protocolos permiten que la información personal sea gestionada de manera segura y privada, utilizando la blockchain para asegurar la integridad y autenticidad de los datos. Así, se garantiza que la información sensible no esté centralizada en una sola entidad, sino que esté distribuida y controlada por los propios usuarios, alineándose con los principios de descentralización y privacidad que promueven las tecnologías blockchain.

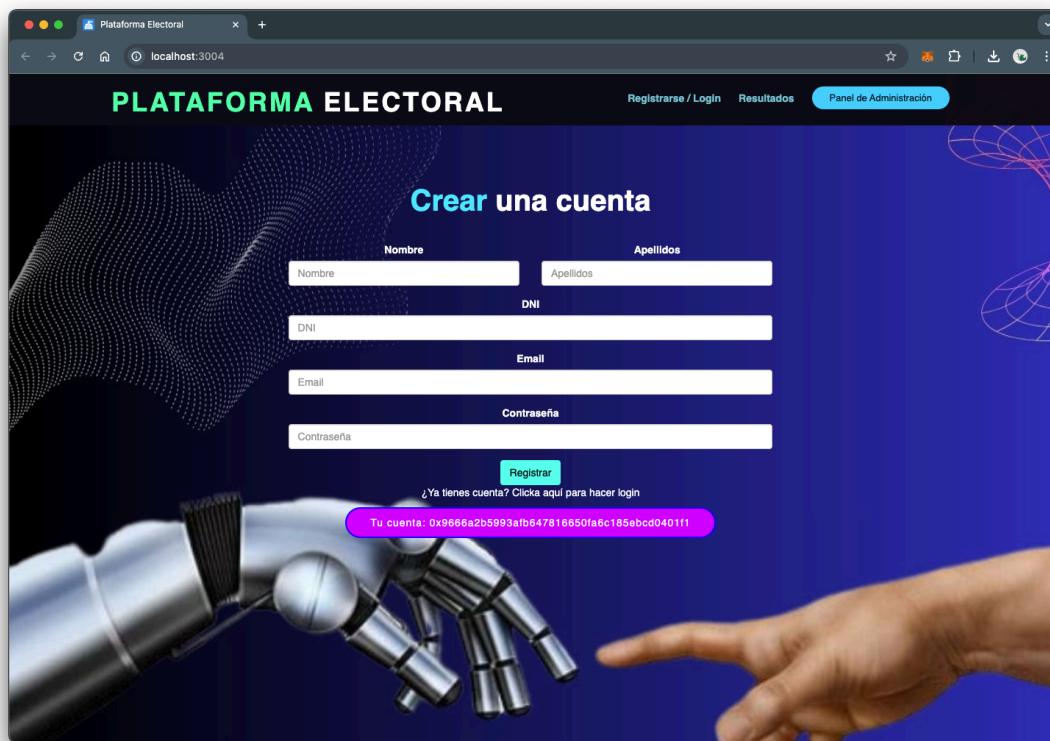


Figura 5.24: Inicio de sesión la Dapp como administrador.

Debemos cambiar a una cuenta de usuario para poder registrarnos. Al hacer esto, no se muestra el Panel de Administración que antes nos aparecía arriba a derecha.



Figura 5.25: Inicio de sesión la Dapp como usuario.

La información proporcionada en el formulario de inicio de sesión se almacena en el contrato inteligente cuando se llama a la función `agregarUsuario` en `app.js`. Esta función se ejecuta cuando se envía el formulario de registro, y los datos del formulario se pasan como argumentos a la función del contrato.

Código 5.6: Función para agregar usuarios

```

1
2 agregarUsuario: async function () {
3     try {
4         var firstName = $('#firstName').val();
5         var lastName = $('#lastName').val();
6         var idNumber = $('#idNumber').val();
7         var email = $('#email').val();
8         var password = $('#password').val();
9         var app = await App.contracts.Elecciones.deployed();
10
11     const gasAmount = 5000000;
12
13     await app.agregarUsuario(firstName, lastName, idNumber, email, password, {from: App.cuenta, gas: ←
14         ↪ gasAmount, gasPrice: '20000000000'});
15
16     console.log("Usuario agregado con éxito");
17     $("#content").hide();
18     $("#loader").show();
19     document.querySelector('.vot').style.display = 'block';
20     location.href = 'vote.html';
21 } catch (error) {
22     console.error("Error al agregar usuario:", error);
23     alert("Hubo un error al agregar el usuario. Por favor, revisa la consola para más detalles."←
24         ↪ );
25 }

```

La llamada a `app.agregarUsuario(...)` se traduce en una transacción enviada al contrato inteligente, ejecutando la función `agregarUsuario` del contrato en Solidity. Ahí, los datos del usuario se almacenan en el mapeo usuarios del contrato inteligente, utilizando `cantidadUsuarios` como clave cuyo concepto se explicará al detallar más adelante los contratos inteligentes utilizados.

Código 5.7: Función para agregar usuarios en el Smart Contract

```

1
2 function agregarUsuario(string memory _nombre, string memory _apellido, string memory ←
    ↪ _numeroIdentificacion, string memory _email, string memory _contrasena) public {
3     cantidadUsuarios++;
4     usuarios[cantidadUsuarios] = Usuario(_nombre, _apellido, _numeroIdentificacion, _email, _contrasena, ←
    ↪ msg.sender);
5 }
```

Al ejecutar `await app.agregarUsuario(...)`, MetaMask detecta la transacción y muestra un prompt al usuario para que firme la transacción con la dirección del contrato y el gas que costará. En la parte superior izquierda vemos que es una cuenta de Usuario.

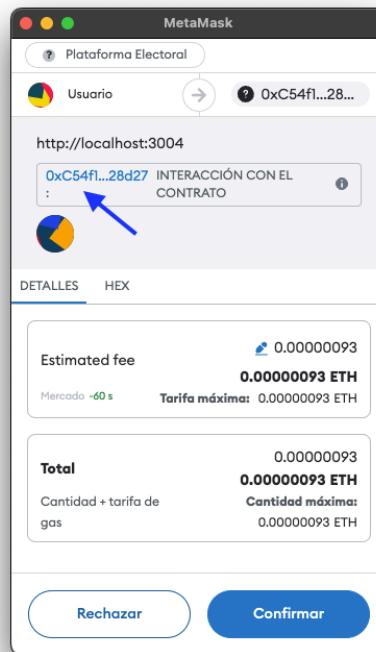


Figura 5.26: Prompt de firma del contrato en Metamask.

Una vez que el usuario firma la transacción, MetaMask la envía a la red Ganache para ser procesada. En la ventana "Transacciones" de Ganache, podemos ver toda la información detallada de la interacción con el contrato y, por ende, de una transacción realizada. Esta ventana proporciona una visión completa de los detalles de cada transacción, permitiendo un seguimiento exhaustivo de las operaciones.

- **Cuenta de Origen:** Muestra desde qué cuenta se ha realizado la transacción. En este caso, se trata de una cuenta de usuario.
- **Dirección del Contrato:** Indica la dirección del contrato inteligente con el cual se está interactuando.
- **Función del Contrato:** Especifica la función del contrato que ha sido llamada durante la transacción.
- **Inputs del Contrato:** Muestra los parámetros o datos de entrada que se han proporcionado a la función del contrato.

Esta información es crucial para entender cómo se están utilizando los contratos inteligentes y para verificar que las transacciones se están ejecutando correctamente. Proporciona transparencia y permite la auditoría de las operaciones realizadas en la blockchain, asegurando que todas las interacciones se registren de manera precisa y verificable.

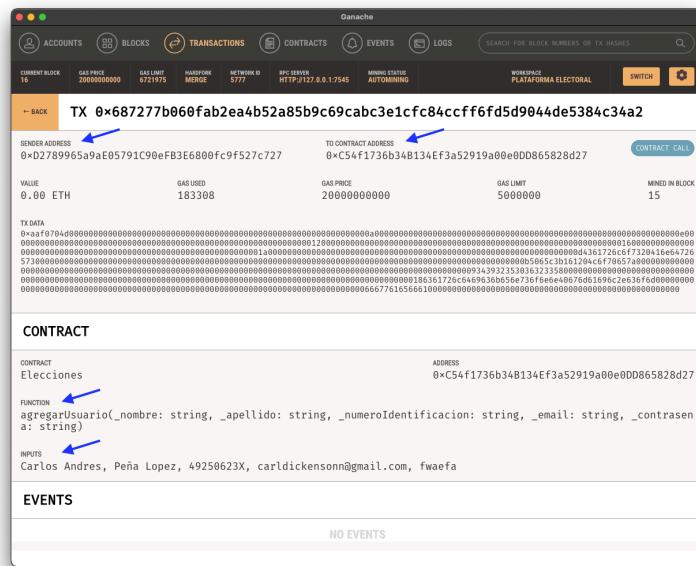


Figura 5.27: Información detallada de la interacción con el contrato inteligente en Ganache.

El panel de administración cuenta con dos recuadros donde se mostrarán los usuarios registrados y los candidatos, así como el número de votos y demás información relevante. En el recuadro de usuarios registrados, se listarán todos los usuarios que se han registrado en la Dapp, mostrando información como las direcciones de sus cuentas y su estado de registro.

El recuadro de candidatos mostrará los candidatos que participan en las elecciones, incluyendo el número de votos que cada candidato ha recibido y otros datos pertinentes. Existe un botón para añadir candidatos y más botones para finalizar y comenzar de nuevo la votación aunque este último no se ha logrado implementar al 100%.

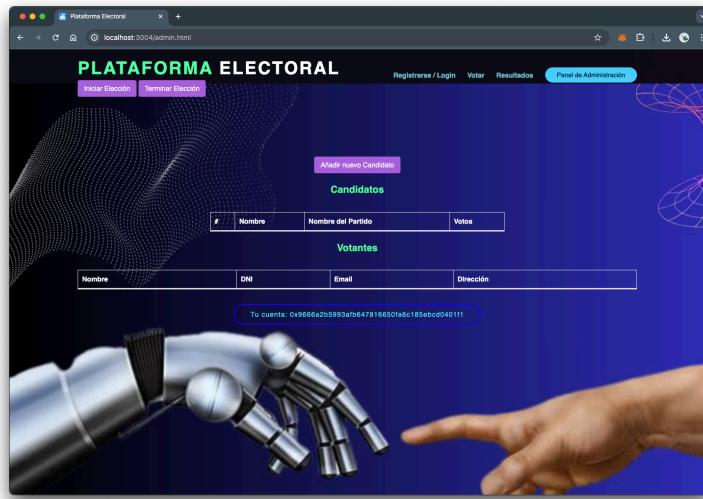


Figura 5.28: Panel de Administración.

Vemos de forma sencilla que solo hace falta proporcionar el nombre, apellido y nombre del partido para completar el registro de un candidato.



Figura 5.29: Añadir nuevo candidato.

Al pulsar en añadir candidato pasa exactamente lo mismo que al agregar usuario, se llama

a la funciones correspondientes y se guarda en la blockchain al firmar el prompt de Metamask.

Código 5.8: Función para agregar candidatos en app.js

```

1
2 agregarCandidato: async function () {
3     try {
4         var CffirstName = $('#CfirstName').val();
5         var ClastName = $('#ClastName').val();
6         var CidNumber = $('#CidNumber').val();
7         var app = await App.contracts.Elecciones.deployed();
8         await app.agregarCandidato(CfirstName, ClastName, CidNumber, {from: App.cuenta, gas: 5000000, ←
9             ↪ gasPrice: '20000000000'});
10        $("#content").hide();
11        $("#loader").show();
12        location.href = 'admin.html';
13    } catch (error) {
14        console.error("Error al agregar candidato:", error);
15        alert("Hubo un error al agregar el candidato. Por favor, revisa la consola para más detalles←
16            ↪ .");
17    }
18},
19

```

Código 5.9: Función para agregar candidatos en el Smart Contract

```

1
2 function agregarCandidato(string memory _nombre, string memory _apellido, string memory ←
3     ↪ _numeroIdentificacion) public soloAdministrador {
4     cantidadCandidatos++;
5     candidatos[cantidadCandidatos] = Candidato(cantidadCandidatos, _nombre, _apellido, ←
6         ↪ _numeroIdentificacion, 0);
7 }

```

Ahora tenemos candidatos añadidos tanto en el panel del usuario como en el panel de administración y los usuarios pueden votar.

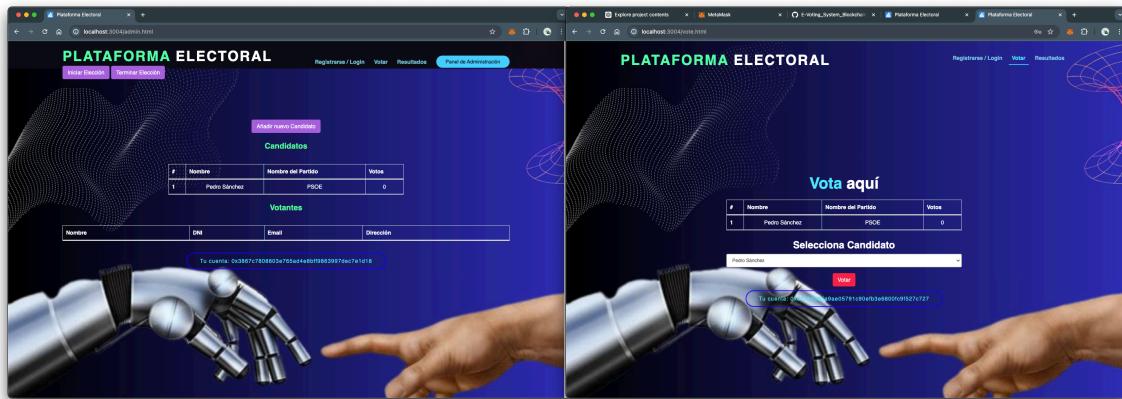


Figura 5.30: Vista del Panel de Administración con candidatos añadidos.

Figura 5.31: Vista del Panel de Usuarios con candidatos añadidos.

Al votar se despliega el prompt de Metamask de nuevo para firmar el contrato con nuestro voto.



Figura 5.32: Firma del contrato para votar.

Para almacenar la información proporcionada para votar en el contrato inteligente, se sigue un flujo similar al de añadir un candidato, `emitirVoto` recoge los datos del formulario (el candidato seleccionado) y llama a la función del contrato inteligente para emitir el voto. La función `votar` en el contrato inteligente almacena la información del voto, marcando al votante como que ya ha votado y aumentando el conteo de votos del candidato seleccionado.

Código 5.10: Función para votar candidatos

```

1
2emitirVoto: async function () {
3    try {
4        var candidatoid = $('#candidatesSelect').val();
5        var app = await App.contracts.Elecciones.deployed();
6        await app.votar(candidatoid, {from: App.cuenta, gas: 5000000, gasPrice: '20000000000'});
7        $('#content').hide();
8        $('#loader').show();
9        location.href = 'results.html';
10   } catch (error) {
11       console.error("Error al emitir voto:", error);
12       alert("Hubo un error al emitir el voto. Por favor, revisa la consola para más detalles.");
13   }
14},

```

Código 5.11: Función para votar candidatos en el Smart Contract

```

1
2function votar(uint _candidatoid) public {
3    require(!votantes[msg.sender]);
4    require(_candidatoid > 0 && _candidatoid <= cantidadCandidatos);
5
6    votantes[msg.sender] = true;
7    candidatos[_candidatoid].cantidadVotos++;
8
9    emit eventoVotacion(_candidatoid);
10}

```

En el Panel de Administración, nos aparecen los usuarios ya registrados, además del voto realizado, que en este caso es solo uno. En el código 5.5, vemos cómo la función render hace uso de las instancias para obtener la información necesaria de los contratos inteligentes y así poder mostrarla. Esta función se encarga de interactuar con los contratos desplegados, solicitando datos específicos y actualizando la interfaz de usuario con la información recuperada, asegurando que los usuarios vean datos actualizados y precisos en tiempo real.

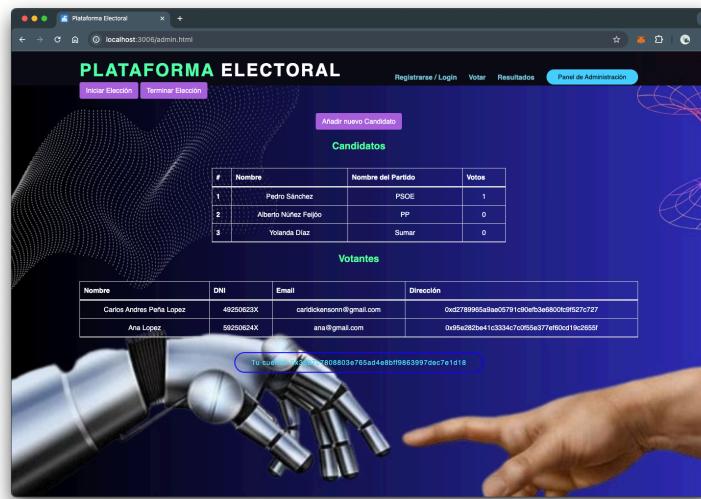


Figura 5.33: Firma del contrato para votar.

Vemos en Ganache ahora en la ventana eventos, el evento de la votación y su información detallada.

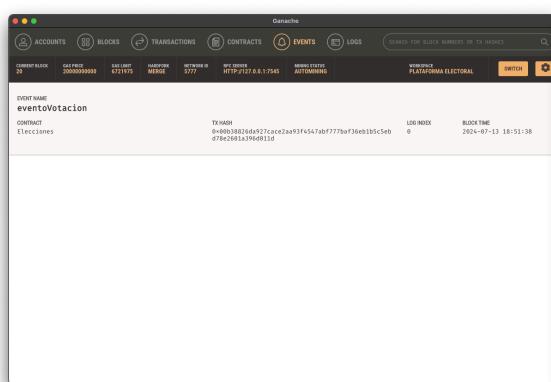


Figura 5.34: Vista del la ventana "Eventos".

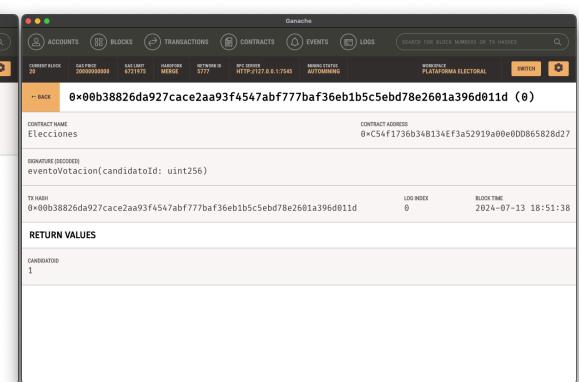


Figura 5.35: Vista detallada del evento.

En Ganache, al realizar todas estas transacciones vemos en la pestaña detalla del contrato Elecciones toda la información. 6 txs (3 candidatos, 2 usuarios) y un evento de votación.

Figura 5.36: Vista del la ventana "Eventos".

Figura 5.37: Vista detallada del evento.

Al pulsar el botón de "Finalizar Votación" en el Panel de Administración, la pestaña de resultados nos mostrará el resultado y la pestaña de votar nos avisará de que no hay elecciones pendientes.

Figura 5.38: Pestaña de resultados al cerrar las elecciones.

Figura 5.39: Pestaña para votar al cerrar las elecciones.

5.6.1 Smart Contracts

Código 5.12: Contrato de Elecciones

```

1 pragma solidity ^0.8.0;
2
3
4 contract Elecciones {
5     address public administrador;
6
7     struct Candidato {
8         uint id;
9         string nombre;
10        string apellido;
11        string numeroIdentificacion;
12        uint cantidadVotos;
13    }
14 }
15
16 mapping (address => bool) public votantes;
17
18 mapping (uint => Candidato) public candidatos;
19
20 uint public cantidadCandidatos;
21
22 event eventoVotacion (
23     uint indexed candidatoId
24 );
25
26 constructor() {
27     administrador = msg.sender;
28 }
29
30 function agregarCandidato(string memory _nombre, string memory _apellido, string memory _numeroIdentificacion) public soloAdministrador {
31     cantidadCandidatos++;
32     candidatos[cantidadCandidatos] = Candidato(cantidadCandidatos, _nombre, _apellido, _numeroIdentificacion, 0);
33 }
34
35 modifier soloAdministrador() {
36     require(msg.sender == administrador);
37     _;
38 }
39
40 function votar(uint _candidatoId) public {
41     require(!votantes[msg.sender]);
42     require(_candidatoId > 0 && _candidatoId <= cantidadCandidatos);
43
44     votantes[msg.sender] = true;
45     candidatos[_candidatoId].cantidadVotos++;
46
47     emit eventoVotacion(_candidatoId);
48 }
49
50 struct Usuario {
51     string nombre;
52     string apellido;
53     string numeroIdentificacion;
54     string email;
55     string contrasena;
56     address direccion;
57 }
58

```

```

59 mapping (uint => Usuario) public usuarios;
60
61 uint public cantidadUsuarios;
62
63 function agregarUsuario(string memory _nombre, string memory _apellido, string memory ↵
64     ↵ _numeroIdentificacion, string memory _email, string memory _contrasena) public {
65     cantidadUsuarios++;
66     usuarios[cantidadUsuarios] = Usuario(_nombre, _apellido, _numeroIdentificacion, _email, ↵
67         ↵ _contrasena, msg.sender);
68 }
69 }
```

El contrato Elecciones gestiona una elección, permitiendo la adición de candidatos, el registro de usuarios y la emisión de votos. A continuación se detalla cada parte del contrato y su funcionamiento:

- **address public administrador:** Dirección del administrador del contrato, que tiene privilegios especiales.
- **struct Candidato:** Estructura que define un candidato con campos como:
 - **uint id:** Identificador único del candidato.
 - **string nombre:** Nombre del candidato.
 - **string apellido:** Apellido del candidato.
 - **string numeroIdentificacion:** Número de identificación del candidato.
 - **uint cantidadVotos:** Contador de los votos recibidos por el candidato.
- **mapping (address => bool) public votantes:** Mapa que lleva un registro de las direcciones que han votado. Se utiliza para evitar que una misma dirección vote más de una vez.
- **mapping (uint => Candidato) public candidatos:** Mapa que almacena los candidatos usando un ID único. Permite acceder a los detalles de un candidato a través de su ID.
- **uint public cantidadCandidatos:** Contador de la cantidad de candidatos registrados en la elección.
- **struct Usuario:** Estructura que define un usuario con campos como:
 - **string nombre:** Nombre del usuario.
 - **string apellido:** Apellido del usuario.
 - **string numeroIdentificacion:** Número de identificación del usuario.
 - **string email:** Correo electrónico del usuario.
 - **string contrasena:** Contraseña del usuario.
 - **address direccion:** Dirección de la cuenta del usuario.
- **mapping (uint => Usuario) public usuarios:** Mapa que almacena los usuarios registrados usando un ID único.
- **uint public cantidadUsuarios:** Contador de la cantidad de usuarios registrados.

Eventos

eventoVotacion: Evento que se emite cuando se emite un voto. Incluye el ID del candidato votado.

Constructor

Inicializa el contrato estableciendo la dirección del administrador como la cuenta que despliega el contrato.

Funciones

agregarCandidato:

- Permite al administrador agregar nuevos candidatos.
- Incrementa el contador de candidatos.
- Añade el nuevo candidato al mapa candidatos.
- Restringida solo al administrador mediante el modificador soloAdministrador.

soloAdministrador:

- Modificador que restringe el acceso de ciertas funciones solo al administrador.
- Verifica que la dirección que llama a la función es la del administrador.

votar:

- Permite a los usuarios votar por un candidato.
- Verifica que el votante no haya votado antes y que el candidato exista.
- Marca la dirección del votante como que ha votado.
- Incrementa el contador de votos del candidato.
- Emite el evento eventoVotacion con el ID del candidato votado.

agregarUsuario:

- Permite a los usuarios registrarse en el sistema.
- Incrementa el contador de usuarios.
- Añade el nuevo usuario al mapa usuarios con sus datos (nombre, apellido, número de identificación, email, contraseña, y dirección de la cuenta).

Código 5.13: Contrato de Migración

```

1 pragma solidity ^0.8.0;
2
3 contract Migraciones{
4     address public propietario;
5     uint public ultimaMigracionCompletada;
6
7     modifier restringido() {
8         require(msg.sender == propietario);
9         --
10    }
11
12
13    constructor() {
14        propietario = msg.sender;
15    }
16
17    function establecerCompletado(uint completado) public restringido {
18        ultimaMigracionCompletada = completado;
19    }
20
21    function actualizar(address nuevaDireccion) public restringido {
22        Migraciones actualizado = Migraciones(nuevaDireccion);
23        actualizado.establecerCompletado(ultimaMigracionCompletada);
24    }
25}

```

A continuación se detalla cada parte del contrato y su funcionamiento:

- **address public propietario:** Dirección del propietario del contrato, que tiene privilegios especiales.
- **uint public ultimaMigracionCompletada:** Almacena el número de la última migración completada.

Constructor

Inicializa el contrato estableciendo la dirección del propietario como la cuenta que despliega el contrato.

Modificadores

restringido:

- Modificador que restringe el acceso de ciertas funciones solo al propietario.
- Verifica que la dirección que llama a la función es la del propietario.

Funciones

establecerCompletado:

- Permite al propietario actualizar el estado de la última migración completada.

- Recibe un número como parámetro y actualiza la variable ultimaMigracionCompletada.
- Restringida solo al propietario mediante el modificador restringido.

actualizar:

- Permite al propietario actualizar la dirección del contrato de migraciones.
- Recibe una nueva dirección como parámetro.
- Crea una instancia del nuevo contrato de migraciones y establece el estado de la última migración completada en el nuevo contrato.
- Restringida solo al propietario mediante el modificador restringido.

6 Propuestas de Mejora

En esta sección se propone algunas propuestas de mejora de la Dapp.

1. Mejora de la Interfaz de Usuario (UI)

- **Rediseño de la UI:** Modernizar la apariencia de la interfaz con un diseño más intuitivo y atractivo.
- **Responsive Design:** Asegurar que la aplicación sea completamente funcional y atractiva en dispositivos móviles y tabletas.
- **Mejora de la Usabilidad:** Simplificar los flujos de trabajo y formularios para mejorar la experiencia del usuario.

2. Seguridad

- **Autenticación de Dos Factores (2FA):** Implementar 2FA para aumentar la seguridad de las cuentas de usuario.
- **Encriptación de Datos:** Asegurar que todos los datos sensibles almacenados en el blockchain estén encriptados.
- **Auditoría y Monitoreo:** Implementar sistemas de auditoría y monitoreo para detectar y prevenir actividades sospechosas.

3. Optimización del Rendimiento

- **Optimización de Gas:** Revisar y optimizar el consumo de gas de los contratos inteligentes para reducir costos.
- **Escalabilidad:** Implementar soluciones de escalabilidad, como sidechains o rollups, para mejorar el rendimiento y reducir la congestión en la red principal.

4. Funcionalidades Adicionales

- **Historial de Votaciones:** Añadir una sección donde los usuarios puedan ver su historial de votaciones.
- **Notificaciones:** Implementar notificaciones para alertar a los usuarios sobre eventos importantes, como el inicio y fin de las elecciones.
- **Comentarios y Feedback:** Permitir a los usuarios dejar comentarios y feedback sobre los candidatos.

5. Integración con Otras Tecnologías

- **IPFS:** Integrar IPFS para el almacenamiento descentralizado de documentos y resultados de votaciones.

- **IA y Analítica:** Utilizar técnicas de inteligencia artificial para analizar los datos de las votaciones y proporcionar insights.

6. Accesibilidad

- **Compatibilidad con Lectores de Pantalla:** Asegurar que la aplicación sea accesible para usuarios con discapacidades visuales.
- **Soporte Multilingüe:** Añadir soporte para múltiples idiomas para que la aplicación sea accesible a una audiencia más amplia.

7. Mejora de la Documentación

- **Guías de Usuario:** Crear guías detalladas para ayudar a los usuarios a entender y utilizar la aplicación de manera efectiva.
- **Documentación Técnica:** Proveer documentación técnica detallada para desarrolladores, facilitando la colaboración y contribución al proyecto.

8. Pruebas y Validación

- **Pruebas Automatizadas:** Implementar un conjunto robusto de pruebas automatizadas para asegurar la calidad y estabilidad del código.
- **Programas de Bug Bounty:** Establecer programas de recompensas por la detección de bugs para incentivar a la comunidad a encontrar y reportar fallos de seguridad.

9. Implementación

a) Fase 1: Análisis y Planificación

- Realizar un análisis detallado de las necesidades de mejora.
- Establecer un plan de acción con metas y cronograma claro.

b) Fase 2: Desarrollo e Integración

- Iniciar el desarrollo de las mejoras propuestas.
- Integrar las nuevas funcionalidades y realizar pruebas unitarias.

c) Fase 3: Pruebas y Validación

- Implementar pruebas automatizadas y realizar pruebas manuales.
- Validar la seguridad y el rendimiento de la aplicación.

d) Fase 4: Despliegue y Monitoreo

- Desplegar la nueva versión de la aplicación.
- Monitorear el rendimiento y la seguridad post-despliegue.

e) Fase 5: Feedback y Mejora Continua

- Recoger feedback de los usuarios y realizar mejoras continuas basadas en sus comentarios.
-

7 Conclusiones

En el contexto actual, donde la tecnología está transformando de manera continua diversas facetas de la vida cotidiana, la adopción de soluciones digitales en procesos críticos como las elecciones se vuelve cada vez más relevante. Este proyecto se ha centrado en explorar y aplicar tecnologías emergentes como blockchain, Web3 y smart contracts para modernizar y proteger el sistema de votación, enfrentando desafíos contemporáneos como la desconfianza en los procesos electorales, el riesgo de fraudes y la necesidad de accesibilidad.

A través de una investigación exhaustiva, se ha comprendido cómo estas tecnologías pueden mejorar la seguridad, transparencia y accesibilidad del proceso electoral. La aplicación desarrollada demuestra que es posible emitir votos de manera segura y verificar resultados de forma transparente, promoviendo una mayor confianza en el proceso electoral.

El análisis de blockchain ha mostrado sus características únicas como la descentralización y la inmutabilidad, esenciales para asegurar la integridad de los datos en sistemas de votación. Web3, con su enfoque en la descentralización y la propiedad de datos por parte de los usuarios, ha demostrado su potencial para crear infraestructuras más seguras y eficientes. Los smart contracts, por su parte, han resaltado la importancia de la automatización y la garantía de cumplimiento de acuerdos de manera segura.

Este trabajo no solo contribuye al conocimiento académico en el campo de las tecnologías emergentes, sino que también demuestra cómo las competencias adquiridas en una formación en ingeniería pueden aplicarse para innovar en áreas emergentes. La implementación práctica de estas tecnologías subraya su viabilidad y su potencial para tener un impacto positivo significativo en la sociedad, mejorando la confianza pública en los procesos electorales y promoviendo una mayor participación ciudadana.

Bibliografía

- Baimamboukar. (n.d). *Github repository*. <https://github.com/baimamboukar>.
- Balastegui-García, G., Sabau, E. M., Payá, A. S., y Mora, H. (2023). Corporate digital identity based on blockchain. En A. Visvizi, O. Troisi, y M. Grimaldi (Eds.), *Research and innovation forum 2022* (pp. 645–655). Springer International Publishing.
- Barański, S., Szymański, J., Sobecki, A., Gil, D., y Mora, H. (2020). Practical i-voting on stellar blockchain. *Applied Sciences*, 10(21). doi: 10.3390/app10217606
- Blockchain Basics . (2016). *Hash functions*. <http://www.blockchain-basics.com/HashFunctions.html>.
- Casino, F., Dasaklis, T. K., y Patsakis, C. (2019). A systematic literature review of blockchain-based applications: Current status, classification and open issues. *Telematics and Informatics*, 36, 55-81. doi: <https://doi.org/10.1016/j.tele.2018.11.006>
- Drescher, D. (2017). *Blockchain basics: A non-technical introduction in 25 steps*. Frankfurt am Main, Germany: Apress. (Library of Congress Control Number: 2017936232) doi: 10.1007/978-1-4842-2604-9
- GitHub. (n.d). *Blockchain voting topics on github*. <https://github.com/topics/blockchain-voting>.
- Haber, S., y Stornetta, W. S. (1991). How to time-stamp a digital document. *Journal of Cryptology*, 3, 99-111. <https://api.semanticscholar.org/CorpusID:14363020>.
- IBM. (2023). *What is blockchain technology?* <https://www.ibm.com/es-es/topics/blockchain>.
- KashifCh-eth. (n.d). *Github repository*. <https://github.com/KashifCh-eth>.
- Kemmoe, V. Y., Stone, W., Kim, J., Kim, D., y Son, J. (2020). Recent advances in smart contracts: A technical overview and state of the art. *IEEE Access*, 8, 117782-117801. doi: 10.1109/ACCESS.2020.3005020
- Krish-Depani. (n.d). *Github repository*. <https://github.com/Krish-Depani>.
- Nakamoto, S. (2009). Bitcoin: A peer-to-peer electronic cash system. <http://www.bitcoin.org/bitcoin.pdf>.
- Namasudra, S., Chandra, D. G., Prashant, J., Mohammad, H., y H., G. A. (2021). The revolution of blockchain: State-of-the-art and research challenges. *Archives of Computational Methods in Engineering*, 28(3), 1497–1515. doi: 10.1007/s11831-020-09426-0

- Orange. (2022). *Blockchain: éste fue el principal motivo de su creación.* <https://www.orange.es/metaverso/noticias/curiosidades/blockchain-este-fue-el-principal-motivo-de-su-creacion>.
- Osuna, A. P., Alzibak, M., Bole, A. D., Payá, A. S., y Mora, H. (2023). Addressing the challenges of biological passport through blockchain technology. En A. Visvizi, O. Troisi, y M. Grimaldi (Eds.), *Research and innovation forum 2022* (pp. 133–143). Springer International Publishing.
- Preukschat, A., Kuchkovsky, C., Lardies, G. G., García, D. D., y Íñigo Molero. (2017). *Blockchain: la revolución industrial de internet.* Centro Libros PAPF, S.L.U. (Primera edición: mayo de 2017)
- Pujol, F., Mora, H., Ramírez, T., Rocamora, C., y Bedón, A. (2024). Blockchain-based framework for traffic event verification in smart vehicles. *IEEE Access*, 12, 9251-9266. doi: 10.1109/ACCESS.2024.3352738
- PwC. (2023). *Bitcoin, blockchain and cryptocurrency.* <https://www.pwc.com/us/en/industries/financial-services/fintech/bitcoin-blockchain-cryptocurrency.html>.
- RedesZone. (2024). *Tipos de redes p2p (peer-to-peer).* <https://www.redeszone.net/tutoriales/internet/tipos-redes-p2p-peer-to-peer/>.
- Roa, M. M. (2021). *Capitalización de mercado de las principales criptomonedas.* <https://es.statista.com/grafico/26156/capitalizacion-de-mercado-de-las-principales-criptomonedas/>.
- SamarthGhante. (n.d.). *Github repository.* <https://github.com/SamarthGhante>.
- Shifaj22. (n.d.). *Github repository.* <https://github.com/shifaj22>.
- Szabo, N. (1994). *Smart contracts.* <https://www.fon.hum.uva.nl/rob/Courses/InformationInSpeech/CDROM/Literature/LOTwinterschool2006/szabo.best.vwh.net/smарт.contracts.html>.
- Visvizi, A., Mora, H., y Varela-Guzman, E. G. (2023). The case of rwallet: A blockchain-based tool to navigate some challenges related to irregular migration. *Computers in Human Behavior*, 139, 107548. doi: <https://doi.org/10.1016/j.chb.2022.107548>
- Wikipedia. (2022). *Árbol de merkle.* https://es.wikipedia.org/wiki/%C3%81rbol_de_Merkle.
- Wikipedia contributors. (2023). *Archivo:arpanet 1972 map.* https://es.wikipedia.org/wiki/Archivo:Arpanet_1972_Map.png.
- Wikipedia contributors. (2024a). *Blockchain.* <https://en.wikipedia.org/w/index.php?title=Blockchain&oldid=1231285727>.
- Wikipedia contributors. (2024b). *Web3.* <https://en.wikipedia.org/w/index.php?title=Web3&oldid=1232037651>.

