



GRADO EN INGENIERÍA EN TECNOLOGÍAS DE TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

SISTEMA DE VOTACIÓN ELECTRÓNICO UTILIZANDO SMART CONTRACTS, TECNOLOGÍA BLOCKCHAIN

Autor: Carlota Ciruelos Marcilla

Director: Atilano Fernández-Pacheco Sánchez-Migallón

Madrid

Declaro, bajo mi responsabilidad, que el Proyecto presentado con el título
“Sistema de votación electrónico utilizando smart contracts, tecnología Blockchain”
en la ETS de Ingeniería - ICAI de la Universidad Pontificia Comillas en el
curso académico 2022/23 es de mi autoría, original e inédito y
no ha sido presentado con anterioridad a otros efectos.

El Proyecto no es plagio de otro, ni total ni parcialmente y la información que ha sido
tomada de otros documentos está debidamente referenciada.

Fdo.: Carlota Ciruelos Marcilla

Fecha: 04/07/2023

Autorizada la entrega del proyecto

EL DIRECTOR DEL PROYECTO

Fdo.: Atilano Fernández-Pacheco Sánchez-Migallón

Fecha: 04/07/2023



GRADO EN INGENIERÍA EN TECNOLOGÍAS DE TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

SISTEMA DE VOTACIÓN ELECTRÓNICO UTILIZANDO SMART CONTRACTS, TECNOLOGÍA BLOCKCHAIN

Autor: Carlota Ciruelos Marcilla

Director: Atilano Fernández-Pacheco Sánchez-Migallón

Madrid

Agradecimientos

Quiero agradecer a mis seres queridos, en especial a mis padres por estar siempre a mi lado en cada etapa de mi vida.

A mis amigos y compañeros de clase, que están siempre dispuestos a escucharme y ayudarme con lo que necesito.

Y a mi director Atilano por haberme acompañado y guiado en esta importante tarea, por su dedicación y paciencia.

SISTEMA DE VOTACIÓN ELECTRÓNICO UTILIZANDO SMART CONTRACTS, TECNOLOGÍA BLOCKCHAIN

Autor: Ciruelos Marcilla, Carlota

Director: Fernández-Pacheco Sánchez-Migallón, Atilano

Entidad Colaboradora: ICAI – Universidad Pontificia Comillas

RESUMEN DEL PROYECTO

En este Trabajo de Fin de Grado se ha creado un sistema que es una aplicación web descentralizada, conectada a un contrato inteligente. El sistema se ha diseñado e implementado desde cero. Este desarrollo tiene como objetivo emplear la tecnología blockchain para implementar un sistema de votaciones electrónico. La finalidad del proyecto es conseguir un sistema electoral transparente que garantice el voto secreto.

Palabras clave: Votaciones, Blockchain, contrato inteligente

1. Introducción

La tecnología Blockchain aparece por primera vez en 2008, y hoy en día, está revolucionando múltiples sectores de la industria, especialmente en la economía con las criptomonedas. Además de su función como medio de intercambio monetario, ofrece una gestión segura y transparente de la información en diversos escenarios. Destaca por la capacidad de crear contratos inteligentes y el almacenamiento descentralizado, que agiliza los procesos y garantiza la inmutabilidad y escalabilidad de la información [1].

En el ámbito electoral, Blockchain busca mejorar el recuento de votos mediante un sistema digital seguro y viable. Los sistemas electorales actuales presentan deficiencias, como ineficiencia, manipulación de votos y amenazas de seguridad. La aplicación de Blockchain pretende maximizar beneficios y superar desafíos, proporcionando transparencia y confiabilidad [2].

Los sistemas de votación tradicionales son ineficientes y generan problemas como largas colas, manipulación de votos y dificultades de acceso. Los intentos de implementar votación electrónica han fallado debido a la falta de transparencia, problemas de integridad y altos costos [3]. Blockchain se presenta como una solución para brindar transparencia, seguridad y reducir costos en los procesos electorales, permitiendo una gestión de información confiable y descentralizada.

2. Definición del Proyecto

Este proyecto consiste en el diseño e implementación de una aplicación descentralizada, para mejorar la comodidad, transparencia y seguridad del sistema electoral.

Para lograr este objetivo se desplegará en Ethereum un contrato inteligente que desarrollado en Solidity [4]. Este contrato, permitirá a los votantes emitir su voto de una forma inmutable, puesto que las transacciones en una cadena de blockchain no se pueden eliminar o modificar.

Para facilitar el uso del sistema, se desarrollará una interfaz gráfica con la que tanto el usuario como el administrador pueden interactuar con el contrato.

Para comenzar el desarrollo de la aplicación, se definirán unos requisitos e historias de usuario que determinarán el flujo de la funcionalidad a implementar. Lo más importante es garantizar que los usuarios se puedan autenticar y que cada usuario vote de una forma única y secreta.

3. Descripción del modelo/sistema/herramienta

En la figura se puede observar como el sistema desarrollado se divide en cuatro componentes principales. Estos son: la DApp, la red de blockchain, el proveedor de web3 y la base de datos SQL.

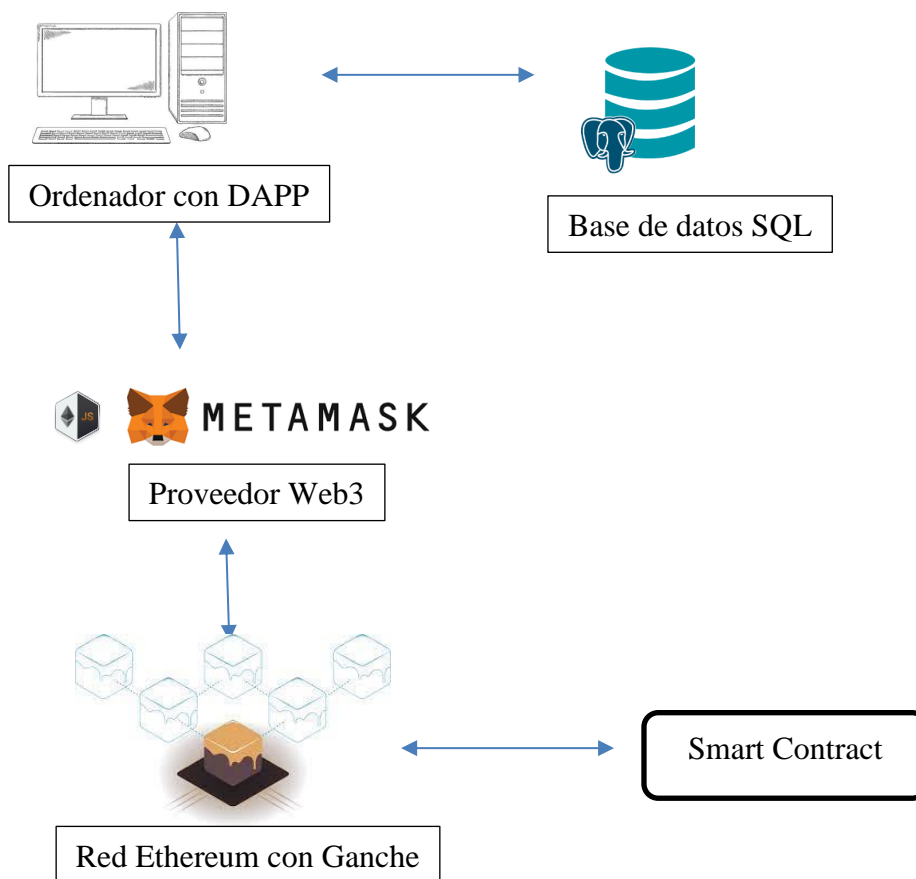


Figura 1: Arquitectura del sistema

El desarrollo de la DApp se ha realizado en la red Ethereum, conocida por su eficiencia en aplicaciones descentralizadas basadas en contratos inteligentes. Para probar y desplegar el contrato inteligente, se ha utilizado la herramienta Ganache.

Para conectarse a la red Ethereum y utilizar la DApp, se ha empleado MetaMask, un proveedor de Web3 que permite a los usuarios acceder a la blockchain desde sus propios dispositivos sin comprometer la privacidad de sus datos.

La base de datos relacional de solo lectura utiliza PostgreSQL como sistema de administración. Almacena información sobre los votantes y candidatos, evitando ralentizar la aplicación en la blockchain. Se utiliza una API para establecer la comunicación entre la aplicación y la base de datos centralizada, realizando llamadas a *endpoints* en la parte del servidor.

Por otro lado, la comunicación con el contrato inteligente se realiza a través de Web3. Se crea una instancia de Web3, se establece la cuenta y se instancia el contrato utilizando la dirección de contrato y el ABI. Esto permite interactuar directamente con las funciones definidas en el contrato inteligente, las cuales son tanto de lectura como de escritura. Cada vez que se realiza una acción de escritura se añade un bloque a la blockchain.[5]

4. Resultados

En este proyecto se ha conseguido crear una aplicación descentralizada que interactúa con un contrato inteligente para registrar el conteo de votos en un proceso electoral. Para conseguir que el usuario interactúe de una manera sencilla y cómoda con el sistema, se ha desarrollado una interfaz gráfica en forma de web. Desde la web se hacen peticiones tanto a la base de datos relacional como llamadas a los métodos del contrato.

En la *Figura 2* y en la *Figura 3* se muestran las vistas más representativas de nuestro sistema. La primera es la página web con la que se encuentra el votante una vez la jornada electoral ha dado comienzo y el usuario se ha registrado. La segunda es la interfaz con la que el administrador puede gestionar las elecciones.

Votaciones

Candidates

Vote

Select Candidate:

Figura 2: Página web votante

Votaciones

Administrador

Empezar la votación

Dirección del votante

Enter address

Añadir votante

Nombre del Candidato

Enter candidate

Añadir candidato

Terminar la votación

Resultados de la votación

Reiniciar la votación

Figura 3: Página web administrador

5. Conclusiones

Durante el desarrollo de este Trabajo de Fin de Grado, se ha demostrado cómo la tecnología blockchain es una herramienta que nos permite encontrar soluciones a problemas que antes parecían insolubles. En el ámbito de las votaciones, se ha logrado crear un sistema que, a pesar de ser transparente, garantiza el voto secreto, lo cual podría mejorar significativamente la democracia en muchos países.

Para implementar este sistema a nivel nacional, se deben realizar mejoras adicionales. Los siguientes pasos podrían ser: modificar el orden de los bloques en la cadena para evitar la asociación de registros con votantes, escalar el proyecto para incluir los datos de todos los ciudadanos y garantizar la concurrencia de múltiples hilos, y unificar los datos de diferentes circunscripciones para poder ajustar el número de escaños asignados.

6. Referencias

- [1] S. Nakamoto (2008). *Bitcoin: A peer-to-peer electronic cash system*.
<https://bitcoin.org/bitcoin.pdf>
- [2] Tien Tuan Anh Dinh, Rui Liu, Meihui Zhang, Member, IEEE, Gang Chen, Member, IEEE, Beng Chin Ooi, Fellow, IEEE, and Ji Wang (17 de Agosto de 2017). *Untangling Blockchain: A Data Processing View of Blockchain Systems*.
<https://arxiv.org/pdf/1708.05665v1.pdf>

- [3] Ministerio de Asuntos Exteriores y de Cooperación, Carlos Vegas González (2014) *Manual práctico para observadores electorales de corta duración, Capítulo XXI. Observación del voto electrónico.*
<https://www.exteriores.gob.es/es/ServiciosAlCiudadano/PublicacionesOficiales/Manual%20pr%C3%A1ctico%20para%20observadores%20electorales%20de%20corta%20duraci%C3%B3n.pdf>
- [4] Varios autores (junio de 2023). *Ethereum blockchain app platform.*
<https://www.ethereum.org/>
- [5] IBM (junio de 2023). *¿Qué son los contratos inteligentes en blockchain?*
<https://www.ibm.com/es-es/topics/smart-contracts>

ELECTRONIC VOTING SYSTEM USING SMART CONTRACTS, A BLOCKCHAIN TECHNOLOGY

Author: Ciruelos Marcilla, Carlota.

Supervisor: Fernandez-Pacheco Sanchez-Migallon, Atilano.

Collaborating Entity: ICAI – Universidad Pontificia Comillas

ABSTRACT

In this Final Degree Project, a decentralized web application connected to a smart contract has been created. The system has been designed and implemented from scratch. The aim of this development is to employ blockchain technology to implement an electronic voting system. The project's goal is to achieve a transparent electoral system that guarantees secret voting.

Keywords: Voting, Blockchain, smart contract

1. Introduction

Blockchain technology first emerged in 2008 and is now revolutionizing multiple industries, particularly the economy with the advent of cryptocurrencies. In addition to its role as a medium of monetary exchange, it offers secure and transparent information management in various scenarios. Notable features include the ability to create smart contracts and decentralized storage, which streamline processes and ensure the immutability and scalability of information.[1]

In the electoral domain, Blockchain aims to improve the vote counting process through a secure and viable digital system. Current electoral systems suffer from inefficiency, vote manipulation, and security threats. The application of Blockchain seeks to maximize benefits and overcome challenges by providing transparency and reliability [2].

Traditional voting systems are inefficient and give rise to issues such as long queues, vote manipulation, and accessibility difficulties [3]. Attempts to implement electronic voting have failed due to lack of transparency, integrity problems, and high costs. Blockchain emerges as a solution to deliver transparency, security, and cost reduction in electoral processes, enabling reliable and decentralized information management.

2. Project definition

This project involves the design and implementation of a decentralized application to enhance the convenience, transparency, and security of the electoral system. To achieve this objective, a smart contract will be deployed on Ethereum and developed using Solidity[4]. This contract will enable voters to cast their votes in an immutable manner, as transactions

on a blockchain cannot be deleted or modified. To facilitate the use of the system, a graphical interface will be developed, allowing both users and administrators to interact with the contract.

To initiate the development of the application, requirements and user stories will be defined to determine the flow of the functionality to be implemented. The most important aspect is to ensure that users can authenticate themselves and that each user can vote in a unique and confidential manner.

3. System description

In the figure, we can observe how the developed system is divided into four main components. These are: the DApp, the blockchain network, the web3 provider, and the SQL database.

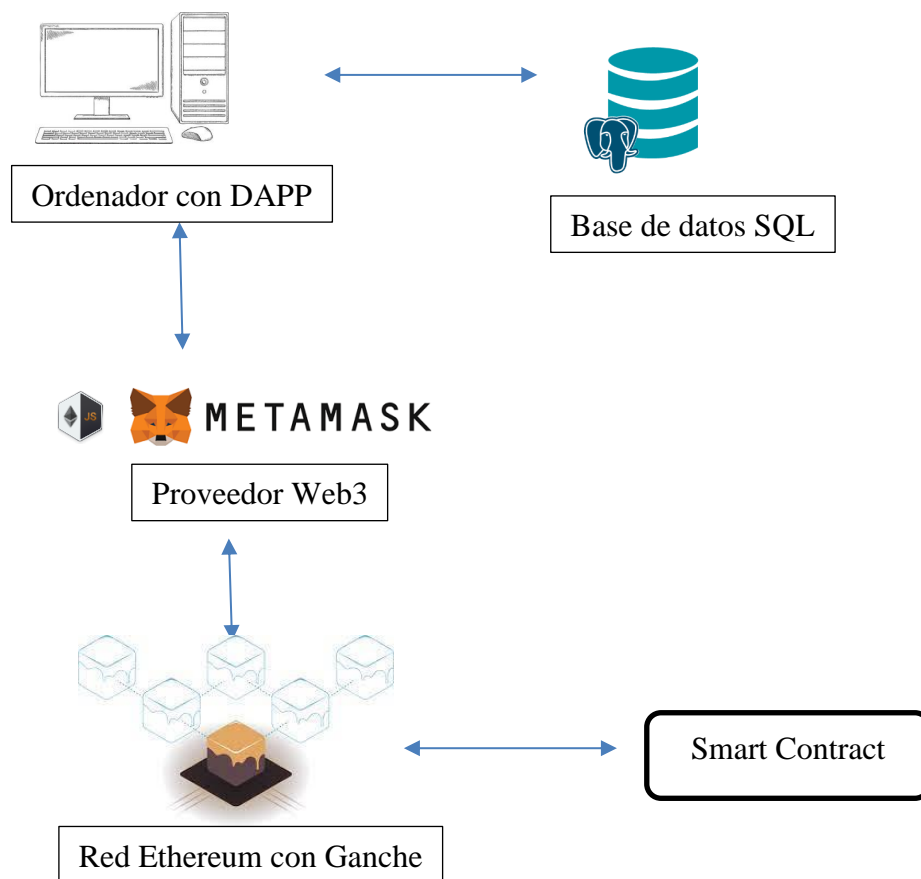


Figura 4: System architecture

The development of the DApp has been carried out on the Ethereum network, known for its efficiency in decentralized applications based on smart contracts. The Ganache tool has been used to test and deploy the smart contract.

To connect to the Ethereum network and use the DApp, MetaMask has been employed, which is a Web3 provider that allows users to access the blockchain from their own devices without compromising the privacy of their data.

The read-only relational database utilizes PostgreSQL as the management system. It stores information about voters and candidates, preventing the application from being slowed down on the blockchain. An API is used to establish communication between the application and the centralized database, making calls to endpoints on the server side.

On the other hand, communication with the smart contract is done through Web3. A Web3 instance is created, the account is set, and the contract is instantiated using the contract address and ABI. This enables direct interaction with the functions defined in the smart contract, which can be both read and write operations. Each time a write action is performed, a block is added to the blockchain [5].

4. Results

This project has successfully created a decentralized application that interacts with a smart contract to record the vote count in an electoral process. To ensure that users can interact with the system in a simple and convenient manner, a web-based graphical interface has been developed. From the web interface, requests are made to both the relational database and the contract's methods.

Figure 5 and *Figure 6* showcase the most representative views of our system. The first view is the web page that the voter encounters once the election day has started and the user has registered. The second view is the interface through which the administrator can manage the elections.

Votaciones

Candidates

Vote

Select Candidate:

Vote

Figura 5: voter website

Votaciones

Administrador



Empezar la votación

Dirección del votante

Enter address

Añadir votante

Nombre del Candidato

Enter candidate

Añadir candidato

Terminar la votación

Resultados de la votación

Reiniciar la votación

Figura 6: admin website

5. Conclusions

During the development of this project, it has been demonstrated how blockchain technology is a tool that allows us to find solutions to problems that previously seemed unsolvable. In the field of voting, a system has been created that, despite being transparent, guarantees secret voting, which could significantly improve democracy in many countries.

To implement this system at a national level, additional improvements must be made. The following steps could be taken: modifying the order of blocks in the chain to avoid associating records with voters, scaling the project to include the data of all citizens and ensuring the concurrency of multiple threads, and consolidating data from different constituencies to adjust the number of assigned seats.

6. References

- [1] S. Nakamoto (2008). *Bitcoin: A peer-to-peer electronic cash system*.
<https://bitcoin.org/bitcoin.pdf>
- [2] Tien Tuan Anh Dinh, Rui Liu, Meihui Zhang, Member, IEEE, Gang Chen, Member, IEEE, Beng Chin Ooi, Fellow, IEEE, and Ji Wang (17 of August 2017). *Untangling Blockchain: A Data Processing View of Blockchain Systems*.
<https://arxiv.org/pdf/1708.05665v1.pdf>

- [3] Ministerio de Asuntos Exteriores y de Cooperación, Carlos Vegas González (2014) *Manual práctico para observadores electorales de corta duración, Capítulo XXI. Observación del voto electrónico.*
<https://www.exteriores.gob.es/es/ServiciosAlCiudadano/PublicacionesOficiales/Manual%20pr%C3%A1ctico%20para%20observadores%20electorales%20de%20corta%20duraci%C3%B3n.pdf>
- [4] multiple authors (june 2023). *Ethereum blockchain app platform.*
<https://www.ethereum.org/>
- [5] IBM (junio de 2023). *¿Qué son los contratos inteligentes en blockchain?*
<https://www.ibm.com/es-es/topics/smart-contracts>

Índice de la memoria

Capítulo 1. Introducción	1
Capítulo 2. Descripción de las Tecnologías.....	4
2.1 Blockchain.....	4
2.1.1 Cadena de bloques	5
2.1.2 Blockchain ethereum	8
2.1.3 Smart Contracts.....	8
2.2 DApps.....	10
2.3 Aplicación	11
2.3.1 Base de datos.....	11
2.3.2 Back-end.....	12
2.3.3 Front-end.....	13
2.4 otras herramientas utilizadas	13
2.4.1 PGAdmin	14
2.4.2 Postman.....	14
2.4.3 Visual Studio Code	14
2.4.4 Ethereum Remix.....	14
2.4.5 Metamask.....	15
2.4.6 Ganache.....	15
Capítulo 3. Estado de la Cuestión	16
3.1 Ineficacia y problemas del sistema electoral actual.....	16
3.2 Votación en España	17
3.3 Votación electrónica.....	18
3.4 Blockchain en la industria 4.0	21
3.4.1 DApps	22
3.5 Blockchain en sistemas de votaciones.....	22
3.6 Riesgos y desafíos del sistema de votación basado en Blockchain	25
Capítulo 4. Definición del Trabajo	26
4.1 Motivación	26
4.2 Objetivos	27

4.3 Metodología.....	27
4.4 Planificación.....	28
4.5 Estimación Económica	28
4.5.1 Recursos	29
4.5.2 Escenarios	30
Capítulo 5. Sistema Desarrollado	35
5.1 Análisis del Sistema	36
5.1.1 Requisitos del modelo.....	36
5.1.2 Historias de usuario	39
5.1.3 Casos de uso	41
5.2 Diseño.....	43
5.2.1 Diseño de la arquitectura.....	43
5.2.2 Estructura de datos.....	46
5.2.3 Diagrama de secuencia	52
5.3 Implementación.....	56
5.3.1 Smart contract	57
5.3.2 Backend	58
5.3.3 Frontend	61
Capítulo 6. Análisis de Resultados.....	64
6.1 Inicio de sesión.....	64
6.2 Pantalla administrador.....	66
6.3 Pantalla Votante	68
6.4 Resultados Ganache	71
Capítulo 7. Conclusiones y Trabajos Futuros.....	73
Capítulo 8. Bibliografía.....	75
ANEXO I: ALINEACIÓN DEL PROYECTO CON LOS ODS	78
ANEXO II 80	
A. II 1 Smart Contract.....	80
A. II 2 Ganache	84
A. II 3 Remix.....	86

A. II 4 Metamask..... 92

Índice de figuras

Figura 1: Arquitectura del sistema	X
Figura 2: Página web votante	XI
Figura 3: Página web administrador	XII
Figura 4: System architecture	XV
Figura 5: voter website	XVI
Figura 6: admin website	XVII
Figura 7: Diagrama de cadena de bloques [4]	6
Figura 8: almacenamiento DApp [8]	11
Figura 9: Pantalla táctil de votación en Francia [15]	19
Figura 10: Máquinas de lectura óptica [15]	20
Figura 11: Casos de uso del Sistema	41
Figura 12: Arquitectura del sistema	44
Figura 13: pgAdmin	49
Figura 14: Diagrama de la clase smart contract	50
Figura 15: Diagrama de secuencia del administrador	53
Figura 16: Diagrama de Secuencia solución 1	54
Figura 17: Diagrama de secuencia escenario 2 votante	55
Figura 18: Estructura de las carpetas del proyecto	57
Figura 19: Diagrama de navegabilidad de la página web	62
Figura 20: página web de inicio de sesión	64
Figura 21: Escenario 1, error al iniciar sesión	65
Figura 22: Escenario 2, error al iniciar sesión	66
Figura 23: Pantalla del administrador	67
Figura 24: Web Resultados	68
Figura 25: web votaciones 1	68
Figura 26: web votaciones conexión con metamask	69

Figura 27: conexión con metamask	69
Figura 28: web votante error 1	70
Figura 29: web votante error 2	70
Figura 30: Transacciones Ganache.....	71
Figura 31: transacciones al votar Ganache	72
Figura 32:Objetivos de desarrollo sostenible de las Naciones Unidas [31]	78
Figura 33: configuración de Ganache parte 1.....	84
Figura 34: configuración de Ganache parte 2.....	84
Figura 35: configuración de Ganache parte 3.....	85
Figura 36: configuración de Ganache parte 4.....	85
Figura 37: Remix plugin.....	86
Figura 38: Desplegar contrato con Remix	86
Figura 39: Interfaz de Remix 1	87
Figura 40: Interfaz Remix 2.....	88
Figura 41: Interfaz Remix 3.....	89
Figura 42: extensión de metamask	92
Figura 43: Inicio de sesión metamask	93
Figura 44: keys de cuenta de Ganache	93
Figura 45: gestión de cuentas de metamask	94

Índice de tablas

Tabla 1. Diagrama de GANT	28
Tabla 2: Ingresos escenario optimista	31
Tabla 3: Costos escenario optimista	32
Tabla 4: Balance anual escenario optimista	32
Tabla 5: Ingresos escenario realista.....	32
Tabla 6: Costos escenario realista	33
Tabla 7: Balance anual escenario realista.....	33
Tabla 8: Ingresos escenario pesimista	33
Tabla 9: Costos escenario pesimista.....	34
Tabla 10: Balance anual escenario pesimista	34
Tabla 11: Diagrama clase Votante.....	46
Tabla 12: Diagrama de clase Candidato	47

Capítulo 1. INTRODUCCIÓN

La tecnología Blockchain está generando una auténtica revolución en diversos sectores de la industria, y su impacto en el ámbito económico es especialmente notable debido al surgimiento de las criptomonedas. Sin embargo, sus aplicaciones van mucho más allá de su función como medio de intercambio monetario. Tras varios años de desarrollo, se ha comprobado es una herramienta que ofrece una gestión segura y transparente de la información, lo que la convierte en un recurso muy valioso en numerosos escenarios.

Uno de los aspectos más destacados de la tecnología Blockchain es la capacidad de crear *smart contracts* (contratos inteligentes). Los cuales son acuerdos digitales que permiten a los usuarios realizar transacciones manteniendo el anonimato y eliminan a los intermediarios innecesarios.

Otra característica muy relevante que nos proporciona esta tecnología es el almacenamiento descentralizado. Este implica el almacenamiento de datos en varios ordenadores o nodos de la red, este sistema no solo agiliza los procesos, sino que también garantiza la inmutabilidad y escalabilidad de la información almacenada. Proporcionando una única fuente de información compartida entre los usuarios, un registro inalterable al que solo tienen acceso los miembros autorizados de una red.

El objetivo principal de este proyecto es investigar y explorar las aplicaciones de la tecnología Blockchain en los procesos electorales. Se busca evaluar la viabilidad de desarrollar un sistema de recuento de votos digital basado en esta tecnología. Para lograrlo, se llevará a cabo la implementación en una red local, lo que permitirá comprobar la seguridad y viabilidad del sistema frente a los posibles desafíos asociados a esta aplicación.

Mediante la aplicación de esta tecnología, se pretende maximizar los beneficios que puede ofrecer, con el fin de mejorar sustancialmente el recuento de votos actual. Los sistemas electorales actuales sufren numerosas deficiencias que resultan en un proceso enormemente

ineficiente. No solo se requieren una gran cantidad de recursos humanos y económicos, sino que también llevan años sufriendo numerosas amenazas de seguridad como la manipulación de votos y los ataques de *malware*.

Los sistemas de votación utilizados actualmente en la gran parte de los países son ineficientes y anticuados. En la mayoría de los casos, entre ellos en España, la votación se sigue realizando de manera presencial. Los ciudadanos rellenan las papeletas electorales y las introducen manualmente en las urnas. Este sistema no es solo enormemente propenso a errores, sino que también es criticado por: las largas colas durante las elecciones, la manipulación de votos, el alto gasto en recursos y la dificultad de votantes discapacitados para acceder a los colegios electorales. Esto ha causado una gran frustración y descontento en los votantes provocando un notable descenso en el número de votantes.

El objetivo principal de las elecciones se ha visto amenazado, y por esta razón se ha intentado sustituir el sistema electoral tradicional por otros como la votación electrónica. Más de treinta países en todo el mundo han utilizado este sistema en su proceso electoral, tanto de forma parcial como única.

Sin embargo, muchos de ellos han dejado de utilizarlo. Hay tres grandes inconvenientes que ha hecho que no tengan el éxito esperado. El primero: la falta de transparencia (la arquitectura de las máquinas es como una caja negra y el código fuente no es público). El segundo: los problemas de integridad, los DREs han sido múltiples veces vulnerables a ciber ataques, las empresas que suministran las máquinas son externas y el suministrador no siempre es integro en su implementación. Tercero: el problema de costes, cada máquina cuesta alrededor de 4.000€ y se necesita una por cada 180 votante; además del coste de programarlas y mantenerlas en cada elección y entre elecciones, sin contar con el hecho de que la vida útil de las maquinas es de entre 10 y 20 años, por lo que cada década habría que hacer una gran inversión en el proceso electoral.

Como se puede ver, la tecnología Blockchain representa una oportunidad única para revolucionar los procesos electorales, brindando una gestión de la información segura, transparente y descentralizada. Con este proyecto, se busca impulsar avances significativos

en la forma en que se llevan a cabo las elecciones, superando los desafíos actuales y estableciendo un sistema más confiable, resistente y confiable. Se busca de esta forma aportar accesibilidad y confianza a los votantes.

Capítulo 2. DESCRIPCIÓN DE LAS TECNOLOGÍAS

Para llevar a cabo este proyecto será necesario emplear una serie de herramientas y tecnologías que serán nombradas y explicadas en este capítulo. Las más relevantes son Blockchain, DApp¹ y red Ethereum.

2.1 *BLOCKCHAIN*

Tal y como hemos mencionado en la introducción, las tecnologías Blockchain están triunfando en el mundo actual, en gran parte debido al éxito del Bitcoin [1], pero a parte de ese primer uso, sus aplicaciones han incrementado enormemente durante los últimos años. En el año 2017 se publicó un artículo [2], en el cual se intenta esclarecer que es una Blockchain y porque tiene sentido emplearla en otros sectores.

Una Blockchain, también conocida como tecnología de contabilidad distribuida, es esencialmente una estructura de datos de solo apéndice, es decir, que se pueden agregar datos al almacenamiento, pero los datos ya existentes son inalterables. Esta estructura es mantenida por una serie de nodos que no confían plenamente los unos en los otros; los nodos de la cadena de bloques se ponen de acuerdo en un conjunto ordenado de bloques, para permitir que tenga lugar una transacción. Por tanto, la cadena de bloques puede verse como un registro de transacciones ordenadas.

La característica principal de esta tecnología es que asume que los nodos pueden actuar de una manera arbitraria y, por tanto, a diferencia de las bases de datos tradicionales, estas redes tienen tolerancia a fallos bizantinos.

En el diseño original, la blockchain de Bitcoin almacena las monedas como estipula el sistema. Para esta aplicación, cada nodo implementa una copia de la máquina de estados

¹ aplicación descentralizada

que mueve monedas de una dirección a otra. Desde entonces, la blockchain ha crecido más allá de las cripto monedas para satisfacer estados definidos por el usuario. Por ejemplo, la red Ethereum **¡Error! No se encuentra el origen de la referencia.** permite la ejecución de cualquier aplicación repetida y descentralizada, estas son conocidas como *smart contracts*. Asimismo, el interés de la industria por el desarrollo de nuevos tipos de blockchain ha llevado a la implementación de plataformas diseñadas para el uso privado con autenticación de los usuarios.

2.1.1 CADENA DE BLOQUES

Un bloque es un fragmento de información que se almacena en la red. Dependiendo del tipo de red, se guarda unos ciertos datos u otros. En una criptomoneda, la información que se recoge en un bloque es el de una operación de envío o recibo que ha tenido lugar en la red.

El proceso de creación de bloques en un blockchain se conoce como minería. La minería implica la recopilación, verificación y agrupación de transacciones en un bloque, que luego se agrega a la cadena de bloques existente.

El primer paso es la recopilación de transacciones. Los nodos de la red blockchain recopilan transacciones pendientes de una pool de transacciones. Estas transacciones son generadas por los usuarios de la red y se almacenan temporalmente hasta que sean incluidas en un bloque.

El segundo es verificar esas transacciones. Los nodos mineros verifican la validez de las transacciones. Esto implica comprobar si las transacciones tienen la firma correcta, si los fondos están disponibles y si cumplen con las reglas y protocolos establecidos por el blockchain en particular.

Una vez que las transacciones han sido verificadas, los mineros las agrupan en un bloque. Cada bloque tiene un límite de capacidad, es decir, puede contener un número determinado de transacciones. Además, el bloque también incluye un encabezado que contiene información como el hash del bloque anterior, un número de secuencia y un *nonce*

Antes de que un bloque pueda ser añadido a la cadena de bloques, los mineros deben resolver un problema criptográfico complejo. Esto implica encontrar un nonce que, al combinarse con otros datos del bloque, genere un hash que cumpla con ciertas condiciones preestablecidas. Este proceso se conoce como prueba de trabajo (Proof of Work) y requiere una gran cantidad de poder de computación.

Una vez que un minero encuentra una solución válida, propaga el nuevo bloque a la red. Los demás nodos de la red lo validan y verifican que cumple con las reglas del blockchain. Si la mayoría de los nodos están de acuerdo en que el bloque es válido, se alcanza un consenso y se agrega a la cadena de bloques existente.

Podemos ver reflejado el funcionamiento de la cadena de bloques en la *¡Error! No se encuentra el origen de la referencia.* donde podemos observar, en cada bloque la información de cada una de sus transacciones junto a un hash el cual, es una cadena única codificada que se convierte en el identificador del bloque, y sirve para su conocer su posición dentro de la cadena.

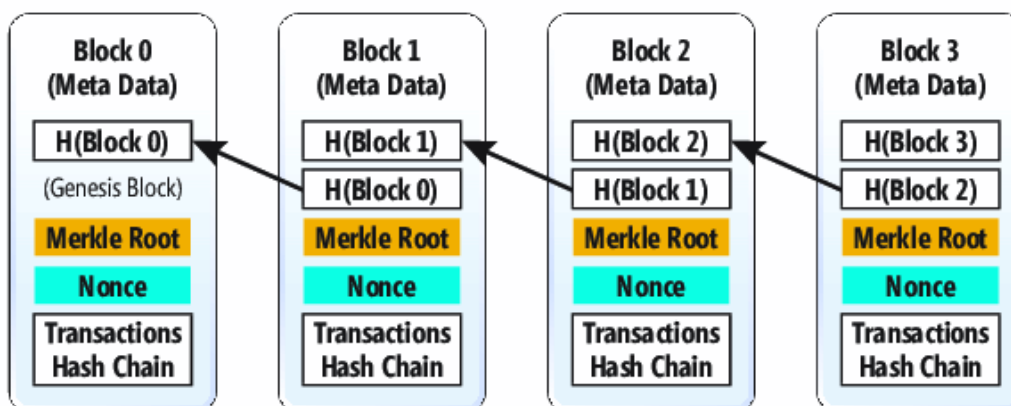


Figura 7: Diagrama de cadena de bloques [4]

El proceso de una transacción de cadena de bloques sería el siguiente. Primero el usuario 1 envía una transacción al usuario 2 para, por ejemplo, pagarle; esto representa un bloque. Después este bloque se envía a todos los nodos de la red para que lo verifiquen. Si el libro mayor de todos los miembros coincide, se aprueba la transacción. Por último, la transacción se considera completada y se añade el bloque como el último elemento de la cadena de bloques.

Además del hash, cada bloque contiene marcas de tiempo asociadas a las transacciones anteriores, junto al hash del bloque previo. De esta forma se crea un vínculo secuencial e inmutable entre los distintos bloques de la cadena, que evita la inserción de bloques nuevos, lo que impide la falsificación de la información de las transacciones.

2.1.1.1 PoW

El término Proof of Work (PoW) es un mecanismo de consenso, utilizado por la mayoría de las criptomonedas para validar las transacciones que tienen lugar en el Blockchain y ofrecer la recompensa correspondiente en el proceso de minería. Permite la transacción entre pares sin la necesidad de un tercero de confianza. **¡Error! No se encuentra el origen de la referencia.**

Su uso en redes de criptomonedas, como, por ejemplo, Bitcoin, se utiliza para verificar los recursos computacionales utilizados antes de cerrar el hash y añadirlo al bloque siguiente. La información de cada bloque se encripta mediante un algoritmo que genera un número hexadecimal determinado de 64 dígitos. Este proceso puede generarse en milisegundos, el hash incluye una serie de números generados para un solo uso (nonce).

Cuando el programa encargado de resolver el hash (el programa de minería de datos) genera un nuevo hash a partir de la información pública disponible. La dificultad del hash se define a partir de una fórmula matemática. Cuando este es menor que el objetivo de la red se ha resuelto el algoritmo, y se le asigna al programa que ha realizado el hallazgo un determinado valor.

Este proceso de minería es un proceso que necesita de amplísimos recursos computacionales, de tal forma que tienen mayor éxito los que utilizan mayores recursos. Exige a los nodos de la red presentar evidencia de la potencia de cálculo utilizada para alcanzar el consenso y de esta forma impedir la entrada de actores no autorizados.

2.1.1.2 Algoritmos de consenso

Las redes descentralizadas como blockchain, ofrecen el intercambio de información de manera segura, privada e inalterable sin la presencia de una autoridad central que valide o verifique las transacciones.

Esto se debe a un procedimiento llamado *Protocolo de Consenso* en el que todos los participantes en la red de Blockchain reconocen una serie de características de dichos bloques. Esencialmente, reconocen que cada bloque que se añade a la red es único e irrepetible. Este protocolo de consenso tiene unos objetivos específicos, tales como el alcanzar un acuerdo de verificación, colaboración y cooperación entre los nodos, igual y obligatoria participación de estos. Por lo tanto, los Algoritmos de Consenso pretenden un acuerdo común de principios computacionales que se aplican a toda la red.

2.1.2 BLOCKCHAIN ETHEREUM

Ethereum es una infraestructura informática descentralizada de código abierto que ejecuta contratos inteligentes. Utiliza una cadena de bloques para sincronizar los cambios de estado del sistema, junto con una criptomoneda Ether (ETH) para medir y limitar los costes de los recursos de ejecución. **¡Error! No se encuentra el origen de la referencia.**

La idea de Ethereum nace a finales de 2013 con Vitalik Buterin. El objetivo era crear una red de uso general, donde un desarrollador pudiese programar su aplicación particular sin tener que implementar los mecanismos subyacentes de las redes *peer-to-peer* y algoritmos de consenso entre otros. La plataforma Ethereum se diseñó para abstraer estos detalles, y ofrecer un entorno de programación determinista y seguro para aplicaciones de blockchain descentralizadas. Igualmente, proporciona una alta disponibilidad, auditabilidad, transparencia y neutralidad.

2.1.3 SMART CONTRACTS

Los Contratos Inteligentes son, *programas almacenados en una cadena de bloques que se ejecutan cuando se cumplen condiciones predeterminadas* [4]. Su principal función es la ejecución automática de un acuerdo, sin necesidad de una tercera persona. Los contratos siguen unas declaraciones lógicas escritas en código, si las condiciones estipuladas se cumplen, una red de nodos ejecuta las acciones y la cadena se actualiza al terminar la transacción.

En el contexto de Ethereum, se pueden definir como programas informáticos inmutables que se ejecutan de forma determinista en una máquina virtual de Ethereum como parte del protocolo de la red. Es decir, en el ordenador mundial descentralizado de Ethereum.

Hay varios lenguajes en los que podemos escribir el código de nuestro contrato. Para este proyecto en particular utilizaremos *Solidity*.

2.1.3.1 Solidity

Solidity es un lenguaje procedimental de alto nivel, orientado a objetos que se emplea para la implementación de contratos inteligentes.[6] Tiene una sintaxis similar a Java, C++ o JavaScript. Y es el lenguaje más popular y utilizado para los contratos inteligentes de Ethereum.

Un prototipo de un contrato en Solidity es el siguiente:

```
pragma solidity >=0.4.16 <0.9.0;

contract MyFirstContract {
    uint data;
    address owner;
    constructor() {
        owner = msg.sender;
    }
    modifier onlySmartContractOwner() {
        require(
            msg.sender == chairperson,
        );
        _;
    }
}
```

```
function set(uint x) public onlySmartContractOwner {
    data = x;
}

function get() public view returns (uint) {
    return data;
}
}
```

En la primera línea se especifica la versión en la que se compilará el programar, en este caso, especifica que la versión debe ser mayor o igual a 0.4.16 y menor que 0.9.0. El constructor, al igual que en otros lenguajes, se ejecuta solo una vez cuando se despliega el contrato en la cadena de bloques, y se asigna la dirección del usuario que ha ejecutado el contrato a la variable *owner*. El modificador, se utiliza para que solo se pueda ejecutar la función *set* cuando el usuario que está realizando una acción es el propietario del contrato, si la condición no se cumple, la ejecución se detiene y no se continua con la función. Por último, la función pública llamada *get*. Devuelve el valor actual de la variable *data*. La función está marcada como *view*, lo que significa que no modifica el estado del contrato.

2.2 DAPPS

Las DApps son aplicaciones que funcionan en una red descentraliza *peer-to-peer* y están basadas en la tecnología de cadena de bloques. A diferencia de las aplicaciones tradicionales, se ejecutan en una red distribuida y descentralizada, donde los nodos de la red participan en la ejecución y validación de las acciones. Gracias a esta característica, se pueden conseguir aplicaciones más transparentes, seguras y robustas a la censura. [7]

Las DApps se basan en contratos inteligentes, los cuales establecen las interacciones y transacciones entre los usuarios sin necesidad de intermediarios. Además, las DApps emplean criptomonedas nativas de la plataforma, en este caso ETH de Ethereum, para realizar transacciones.

Tal y como se observa en la siguiente figura, este tipo de aplicaciones guardan la información en los ordenadores de la red. Esto supone que, aunque se caiga algún nodo de red esta sigue funcionando.

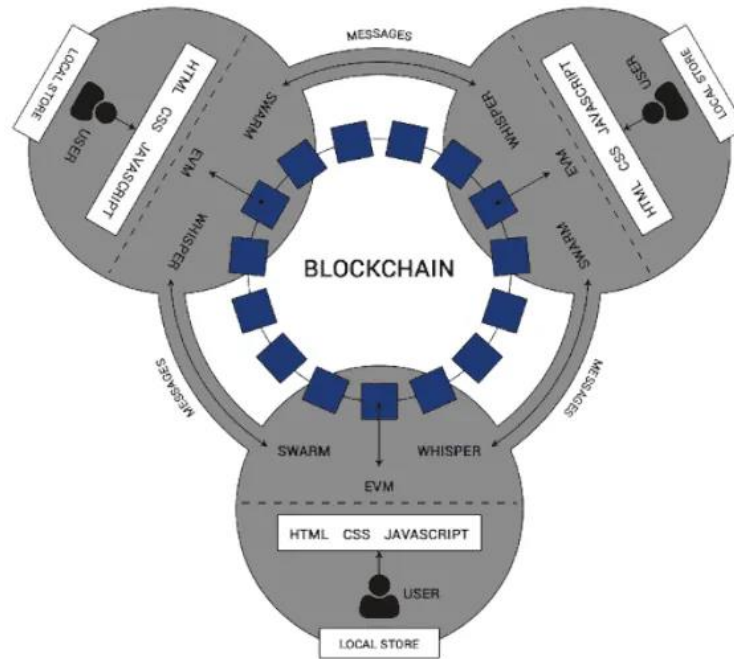


Figura 8: almacenamiento DApp [8]

2.3 APLICACIÓN

En esta sección explicaré las tecnologías empleadas para llevar a cabo el desarrollo de la aplicación. Voy a dividir este apartado en tres subsecciones: Base de datos, Back-end y Front-end.

2.3.1 BASE DE DATOS

Para poder desarrollar una aplicación es necesario almacenar los datos de esta en algún sistema de almacenamiento. Para este proyecto emplearemos una base de datos relacional, basada en tablas.

2.3.1.1 PostgresSQL

Es un sistema de gestión de bases de datos relacional que se puede instanciar en nuestro equipo de forma local, almacena por tanto la información en tablas de tamaño fijo, y permite lectura, escritura y actualización de filas durante la ejecución del programa.

Postgres funciona con SQL, un lenguaje que soporta consultas complejas para extraer información de la base de datos. Además, permite definir tipos de datos personalizados y crear índices en tablas para mejorar el rendimiento de las consultas.

2.3.2 BACK-END

Esta es la parte de nuestro programa que se une la base de datos con la interfaz gráfica, también es la encargada de la compilación y despliegue de nuestro contrato inteligente. Esta parte se ha desarrollado con java en un proyecto de Spring como lenguaje de programación e IntelliJ Idea como *framework*.

2.3.2.1 Spring

Spring Boot es un marco de desarrollo de aplicaciones de código abierto basado en Java. **¡Error! No se encuentra el origen de la referencia..** Proporciona un enfoque simplificado y eficiente para crear aplicaciones Java, reduciendo la complejidad y el tiempo de configuración, nos ayuda esencialmente a simplificar la creación de la estructura: *controller*, *service* y *respository*. Gracias a esta, podemos escribir *endpoints* de forma sencilla para acceder a la base de datos.

2.3.2.2 IntelliJ Idea

Es el entorno de desarrollo más usado para el desarrollo de Java con Spring gracias a su valioso conjunto de herramientas integradas para el desarrollador. **¡Error! No se encuentra el origen de la referencia..** Permite el autocompletado de código, detecta y sugiere soluciones a pequeños errores, y refactoriza de una manera fiable. Al mismo tiempo, permite una ejecución muy sencilla de la aplicación y los tests, así como proporciona técnicas de visualización avanzadas.

2.3.3 FRONT-END

Es la parte del código encargada de la interacción con el usuario. Es la interfaz gráfica, la parte visible de la aplicación, que se encarga de comunicarse de forma asíncrona mediante llamadas a *endpoints* con el *back-end* y también ejecuta el *smart contract*.

Esta parte se compone a su vez de tres partes: estructura, estilo y lógica. Para la estructura de la página web se ha utilizado HTML, para el estilo CSS y para implementar la lógica JavaScript. Todas estas partes se relacionan entre sí para

2.3.3.1 JavaScript y Web3.js

JavaScript es el lenguaje de alto nivel, interpretado y orientado a objetos. Es ampliamente utilizado para desarrollar aplicaciones web interactivas y dinámicas, tanto en el lado del cliente, como del servidor.

Tiene dos funciones esenciales: comunicarse con la base de datos y realizar acciones en el contrato inteligente.

Para la primera tarea, hacemos un *fetch* a los *endpoints* que hemos definido previamente en el lado del servidor, al crear las distintas capas del *back-end*. Para la segunda función, hacemos uso de una librería llamada *web3.js*. Es una biblioteca de JavaScript que permite interactuar con la red Ethereum. Permite enviar transacciones, escribir y leer datos en *smart contracts*.

2.4 OTRAS HERRAMIENTAS UTILIZADAS

Para este proyecto han sido necesario el uso de otras herramientas para llevar a cabo la realización del mismo.

2.4.1 PGADMIN

Es la herramienta elegida para la administración y desarrollo de bases de datos PostgreSQL; **¡Error! No se encuentra el origen de la referencia..** Proporciona una interfaz gráfica completa para visualizar nuestros esquemas y comprobar la correcta creación de las tablas y relaciones. También permite la interacción con la base de datos.

2.4.2 POSTMAN

Postman es una herramienta popular para probar, desarrollar y documentar APIs [11]. Tiene una interfaz gráfica que permite enviar solicitudes HTTP a una API, sin tener que desarrollar el *front-end*. Esta herramienta, es muy útil para hacer pruebas de la parte de servidor con la base de datos.

Algunas de sus funcionalidades más importantes son: envío de solicitudes HTTP, como GET, POST, PUT y DELETE; gestión de colecciones; configuración de entornos y variables; y automatización de pruebas.

2.4.3 VISUAL STUDIO CODE

VS Code es un entorno de desarrollo ligero creado por Microsoft **¡Error! No se encuentra el origen de la referencia..** Es usado muy comúnmente por desarrolladores de *software* para editar código fuente en diferentes lenguajes de programación. Y en este permite la instalación de plugins que son de gran utilidad para agilizar y facilitar el desarrollo y despliegue de código. Para este proyecto, haremos uso de Remix IDE para probar contratos inteligentes.

2.4.4 ETHEREUM REMIX

Remix IDE es un entorno de desarrollo y prueba en la plataforma Ethereum para contratos inteligentes. Proporciona una interfaz de usuario muy cómoda para editar, compilar, desplegar y probar *smart contracts*. Para utilizarlo, en este caso se ha optado por emplear Ethereum Remix, que es la extensión de VS Code.

2.4.5 METAMASK

Metamask es una cartera de criptomonedas, que permite a lo usuarios realizar transacciones en la red Ethereum y gestionar sus activos digitales **¡Error! No se encuentra el origen de la referencia..** Es una extensión del navegador y proporciona una manera sencilla y segura de acceder a las DApps basadas en Ethereum. Por lo funciona como una herramienta de prueba muy útil para el proyecto realizado.

2.4.6 GANACHE

Ganache es una herramienta de desarrollo *blockchain*. Proporciona un entorno de desarrollo local para poder probar aplicaciones descentralizas en la red de Ethereum. Crea una instancia de *blockchain* local sin la necesidad de conectarte a una red en vivo, y así no tener que utilizar fondos reales. Además, crea automáticamente una serie de cuentas preconfiguradas que serán nuestros usuarios de la aplicación para realizar transacciones en el entorno de prueba.

Capítulo 3. ESTADO DE LA CUESTIÓN

El objetivo de este capítulo es explicar las razones por las que se lleva a cabo el proyecto y se detallarán soluciones y trabajos similares existentes ya en el mercado.

3.1 INEFICACIA Y PROBLEMAS DEL SISTEMA ELECTORAL ACTUAL

El sistema electoral en España, al igual que en otros muchos países, se sigue realizando de manera presencial y a papel. Los ciudadanos eligen una papeleta electoral y tras identificarse, la introducen en la urna, al final de la jornada, cada mesa recuenta los votos de su urna. Lo que lo convierte en un sistema altamente ineficiente, propenso a errores y anticuado.

Asimismo, este sistema ha provocado un gran descontento entre los votantes debido a las grandes colas, la manipulación electoral y el alto gasto en recursos. Esto ha llevado a muchos ciudadanos a decidir no ejercer su derecho a voto. Otro problema que concierne el sistema de recuento electoral es el gran consumo de recursos humanos, materiales, de infraestructura y temporales.

El proceso electoral requiere la participación de un gran número de personas para asegurar el correcto desarrollo de las elecciones, realizar tareas como la verificación de la identidad, conteo de votos y la gestión de mesas electorales.

Además, el sistema se basa en el uso de papeletas y sobres para realizar el voto; lo que implica una gran cantidad de material impreso, y material adicional como urnas, cabinas de votación, sellos, bolígrafos, entre otros.

También, para llevar a cabo el proceso electoral necesitamos contar con una infraestructura física adecuada para los colegios electoral y Juntas Electorales; esto incluye el acondicionamiento de espacios, disponibilidad de mobiliario y suministro de energía.

Por último, el proceso electoral requiere un período de tiempo determinado tanto para la preparación de las elecciones, la jornada electoral y el posterior recuento de votos; de igual forma, la movilización de los votantes, que deben acudir físicamente a los colegios electorales en un horario establecido puede resultar problemático.

3.2 VOTACIÓN EN ESPAÑA

Para poder desarrollar el sistema es necesario conocer cómo se desarrollan las elecciones en España. Hay cuatro tipos de elecciones: generales, europeas, autonómicas y municipales. Cada una de estas elecciones eligen diferentes miembros para distintos parlamentos y organismos públicos. En nuestro sistema, vamos a enfocarnos en las elecciones generales, las cuales tienen ciertas características que determinarán el funcionamiento del recuento de votos.

El aspecto más relevante es el sistema D'Hont junto con la división por circunscripción [14]. En este sistema se supondrá que existe únicamente una circunscripción. Ya que el número de circunscripciones dependen del tipo de votación, y a cada una le corresponde un número distinto de diputados. Esto supone que el sistema se deberá implementar para cada circunscripción por separado.

A su vez, cada circunscripción divide escaños utilizando el sistema D'Hont, un método que reparte de forma aproximadamente proporcional a su porcentaje en votos. Es una aproximación, ya que no podemos asignar un número decimal de escaños.

El método D'Hont funciona de la siguiente manera. Primero se ordenan en orden decreciente los resultados obtenidos por candidatura y se calcula el porcentaje de votos de cada una teniendo en cuenta también los votos en blanco. Después, se eliminan las candidaturas que no lleguen al 3% en las generales y en las locales al 5% para evitar la excesiva fragmentación de la cámara. El siguiente paso es construir una tabla con tantas columnas como número de diputados a distribuir, completando cada columna con los votos de cada candidatura entre 1,2,3... hasta completarla.

Los diputados se asignan a los coeficientes más altos de la matriz que hemos construido. En caso de empate se asignará a la candidatura con mayor número de votos totales, y si este también coincide se haría por sorteo.

3.3 VOTACIÓN ELECTRÓNICA

El voto electrónico o *e-voto* se puede definir como el uso de las TICS² para las fases tanto de voto como de recuento. Muchos países ante los riesgos mencionados en el apartado anterior han optado por sustituir el sistema electoral tradicional por la votación electrónica. **¡Error! No se encuentra el origen de la referencia.**

Actualmente, existen múltiples sistemas de votación, los más utilizados son los DRE, las OMR y el voto por internet. En esta sección, se detallará en que consiste cada una junto a sus ventajas e inconvenientes.

El DRE o registro electrónico de voto es como una urna electrónica. Son máquinas que se instalan en los centros electorales, donde el votante elige una opción, la cual se registra electrónicamente. Al terminar la votación, los datos almacenados son transferidos al centro de conteo.

² Tecnologías de Información y Comunicación



Figura 9: Pantalla táctil de votación en Francia [14]

Las OMR o máquinas de lectura óptica, estas máquinas usan un sistema distinto. Primero el votante cumplimenta a mano una papeleta que a continuación se introduce en estos dispositivos.



Figura 10: Máquinas de lectura óptica [14]

El voto por Internet permite al elector realizar su elección desde la comodidad de su dispositivo electrónico personal, mediante la conexión a una página web. Países como Suiza, Estonia y Noruega lo han implantado en sus procesos.

Aunque estos sistemas, reducen algunos de los problemas a los que se enfrentaba el sistema tradicional, las organizaciones internacionales destacan que estos sistemas podrían acarrear una serie de obstáculos que no se deben ignorar. La Unión Europea considera que el *e-voto* solo puede triunfar en países con altos niveles de confianza en la integridad de la administración y proceso electoral.

Algunos de los riesgos potenciales que puede generar el voto electrónico son los siguientes. El primero, la disminución de transparencia, el *software* empleado en las máquinas no es público, y aunque lo fuera muchos de los electores no expertos tendrían dificultades de comprensión y no confiarían. El segundo, la violación del secreto de voto, ya que no se asegura en algunos de estos sistemas.

Otro gran riesgo, son los ataques externos de *malware*. Por último, la arquitectura de estos sistemas tiene un gran coste de fabricación, transporte, implementación y mantenimiento, sin contar con el hecho de que la vida útil de las máquinas es de veinte años a lo sumo.

3.4 BLOCKCHAIN EN LA INDUSTRIA 4.0

La Industria 4.0, es un término que se refiere a la transformación digital y tecnológica de los procesos industriales. Su objetivo es mejorar la productividad y flexibilidad de procesos mediante su automatización. Las aplicaciones de esta industria integran múltiples tecnologías, algunas de las más relevantes son las siguientes, el Internet de las cosas (IoT), la inteligencia artificial (IA), la robótica, la realidad virtual, el análisis de datos y la computación en la nube.

La tecnología Blockchain ha ganado un importante papel en la era de la industria 4.0. Gracias a sus excelentes resultados es ampliamente utilizado en cadenas de suministro, sistemas de salud, finanzas, pagos, economía, etc. Gracias a sus buenos resultados, el Blockchain, más concretamente los *smart contracts* son una buena solución para satisfacer las necesidades de un sistema electoral transparente e integro.

El objetivo fundamental es crear un sistema electoral justo que pueda ser usado desde cualquier parte del mundo. Sus beneficios más significativos son: la accesibilidad, la seguridad, la reducción de costes, la inmutabilidad, la rapidez y la transparencia. Asimismo, la blockchain proporciona un método descentralizado capaz de generar registros encriptados seguros, preservando el anonimato del participante mientras se mantiene abierto a una auditoría pública.

3.4.1 DAPPS

Tal y como se ha explicado en el apartado anterior, un DApp es una aplicación descentralizada, esto quiere decir que la información se guarda en todos los nodos de la red. Esta propiedad tiene unas ventajas enormes a la hora de crear una app de votaciones. Uno de los riesgos principales que consideramos a la hora de establecer un sistema electoral es la manipulación de los votos y los ataques malware, la finalidad de este apartado es intentar solventar estos problemas con el uso de la blockchain.

Esta tecnología basada en blockchain no están controlada por una sola entidad centralizada. Esto permite una mayor transparencia, seguridad y resistencia a la censura. Además, pueden aprovechar la criptografía avanzada y los registros inmutables para garantizar la seguridad de los datos y las transacciones. Esto reduce el riesgo de fraudes, manipulaciones y ciberataques.

Otra ventaja de este tipo de aplicaciones es que permiten la interacción directa entre los usuarios, eliminando la necesidad de intermediarios o intermediarios costosos. Esto puede reducir los costos y acelerar las transacciones, al tiempo que elimina las barreras y restricciones asociadas con terceros.

Asimismo, todas las transacciones realizadas en una DApp se registran en la cadena de bloques, lo que significa que son públicamente visibles y verificables. Esto aumenta la confianza de los usuarios al eliminar la necesidad de confiar en intermediarios o terceros

3.5 BLOCKCHAIN EN SISTEMAS DE VOTACIONES

Frente a la metodología de voto electrónico centralizado, en este proyecto se aborda el voto de una forma descentralizada mediante la tecnología blockchain y un mecanismo de votación que garantice la seguridad de identificación de votantes, la transmisión de la

elección y la verificación de datos [17]. Para ello, en esta sección se analizarán otras aproximaciones realizadas a el caso de uso.

En 2018 aparece BroncoVote, una forma de eliminar la problemática de los sistemas electorales convencionales mediante el uso de la tecnología blockchain [18]. Se desarrolla para preservar el anonimato de los votantes y mejorar la transparencia. BronoVote presenta un recuento de votos con contratos inteligentes en una red Ethereum para resultados electorales auditables en universidades. Las limitaciones de este sistema son que tiene un método de registro poco protegido y una autenticación débil.

Ese mismo año, Hjálmarsson [19] presenta un sistema de votaciones donde la blockchain es usada como un servicio para crear un sistema electrónico distribuido Este sistema tiene dos tipos de nodos: de distrito y de inicio. El nodo de distrito indica cada región de votación, y cada uno de esos nodos está equipado con un software que lo conecta con un nodo de inicio. Un nodo de inicio permite la identificación y conexión de un nodo de distrito. No obstante, este sistema no protege la privacidad de los votantes de una forma muy segura.

Jorge Lopes propone en 2019 un sistema de voto electrónico basado en blockchain, usando contratos inteligentes **¡Error! No se encuentra el origen de la referencia..** En su modelo hay tres categorías de personas que pueden interactuar con el programa. Estas personas son: el director, el desarrollador y el votante. Y hay tres contratos distintos: Registro, de Creador y Elección. Los contratos de Registro son los responsables de almacenar la información de registro del elector para su verificación en el paso de autenticación al iniciar la votación.

Después de la autenticación, la API envía una parte de los fondos al contrato de Creador, el cuál es responsable de establecer un nuevo contrato de Elección. El contrato de Elección es creado y envía su dirección al contrato de Creador para la emisión de voto. Antes de añadir la papeleta a la blockchain, se encripta con un tipo de encriptación simétrica conocida como cifrado homomórfico.

También en 2019 se introduce un algoritmo mejorado de *e-voto* usando tecnología blockchain **¡Error! No se encuentra el origen de la referencia..** Su algoritmo prueba la integridad del desarrollo de bloques, el bloqueo de bloques, la gestión de la información y diseño de una red blockchain; especialmente orientado para una red de máquinas de votación.

En 2020, se propone la idea de una arquitectura de votación digital que contiene un *smart contract* para hacerse cargo de las áreas que suponían un riesgo en los sistemas de votación electrónica [21]. Entre otros, se encarga de la autenticación, la transparencia, la precisión, la autonomía, la singularidad, la integridad y la movilidad, que pueden ocurrir durante el uso de la blockchain en una votación.

Basándose en la información aportado por el votante, un hash será creado y guardado en una cadena en su sistema. Debido a que la información se almacena como un hash en la blockchain, los votantes se beneficiarán de la escalabilidad y el anonimato. Los contratos inteligentes en la blockchain aseguran la seguridad y el anonimato. Cada bloque tiene su método único de contabilizar los votos³. Al finalizar la votación, el número total de votos del último bloque será el evaluado.

Al año siguiente se crean y construyen AMVchain, un sistema escalable y eficiente de votación que usa blockchain y contratos inteligentes para crear un sistema de votación transparente y descentralizado [22]. Después de examinar los problemas existentes en los sistemas de votación fundados en blockchain, se intenta dar una solución basada en la investigación. Siguiendo las especificaciones para un sistema electrónico fiable y eficiente, se usan firmas de anillos enlazables para romper la unión entre votos y votantes y garantizar el anonimato del voto.

En 2021 se presenta un sistema de votación de blockchain que usa encriptación del bloqueo temporal para proveer integridad, autenticación y confidencialidad **¡Error! No se**

³ Protocolos de consenso

encuentra el origen de la referencia.. La autenticación se consigue mediante el uso de una firma ciega. El bloqueo temporal se usa para garantizar la protección y la confidencialidad de la manipulación electoral. En este sistema, se logra inhabilitar la visualización de los resultados por parte de los partidos involucrados, antes de un tiempo predeterminado.

3.6 RIESGOS Y DESAFÍOS DEL SISTEMA DE VOTACIÓN BASADO EN BLOCKCHAIN

Después de analizar otros proyectos similares, podemos observar que la implementación de un sistema electoral eficaz y eficiente sigue suponiendo un reto hoy en día, aun así se puede intentar conseguir buenas soluciones.

Para que este sistema sea efectivo debemos asegurarnos de que cubrimos varios posibles riesgos basado en el apartado previo. Las propiedades que se deben garantizar para crear un sistema seguro son: anonimato, integridad, seguridad, privacidad, verificación personal y universal, movilidad e imparcialidad.

El principal es mantener el anonimato, uno de los grandes desafíos en este sistema. Garantizar la encriptación de la identidad digital es vital para este proceso. También debemos asegurarnos de que esta identidad no se manipula o suplanta por otros usuarios y que cada usuario solo puede añadir un bloque a la cadena.

Capítulo 4. DEFINICIÓN DEL TRABAJO

4.1 MOTIVACIÓN

La tecnología *Blockchain* es una de las más punteras de hoy en día en el mercado. Tal y como hemos visto en los capítulos previos, es un área que aún tiene muchos casos de uso por implementar, y tras una investigación se puede ver como muchas industrias han empezado a usarlo en sus diferentes productos. Esta herramienta permite una gestión más transparente, rápida y exacta de la información en cualquier negocio.

Asimismo, gracias a las propiedades de esta tecnología, existen numerosas aplicaciones que, desde un punto de vista social, pueden aportar comodidad y facilidades a la sociedad. En el caso de las votaciones puede ser muy beneficioso al aportar accesibilidad y confianza en los votantes.

En concreto, se ha observado que, en el campo de los procesos electorales, está siendo estudiada por numerosos investigadores para llegar a una solución que cubra todos los requisitos que un sistema de recuento de votos exige. Ya que gracias a los beneficios que nos proporciona la blockchain podemos hacer frente a muchos de los desafíos que supone la creación de un sistema electoral eficiente. Hoy en día, se siguen encontrando numerosos casos de elecciones que no han sido justas por la manipulación de los votos que nacen de sistemas tradicionales. Y en un mundo tan globalizado y digitalizado, no podemos permitir que ocurra este problema.

Por esos motivos quería realizar un proyecto en un área donde se pudiese implementar esta herramienta, y tuviese un efecto positivo respecto a los sistemas que se emplean en la actualidad.

4.2 OBJETIVOS

- Crear un sistema de votación que aumente el número de participantes en las votaciones, evitando las largas colas mediante un voto en remoto, y dando un acceso más cómodo al voto a personas con discapacidades
- Evitar el fraude electoral mediante un sistema electoral cuyos resultados son inmutables
- Crear un sistema que sea menos costoso, y más responsable con el medio ambiente eliminando todo el papeleo o la fabricación de maquinaria electoral
- Garantizar el voto secreto

4.3 METODOLOGÍA

Para el desarrollo del proyecto se combinará una metodología en cascada añadiendo ciertas ceremonias y técnicas agile. Es decir, el proyecto se organizará por etapas para un primer diseño, pero las mejoras o cambios se implementarán de forma agile. Las principales etapas son las siguientes:

1. Búsqueda de información: lo primero es ver cómo realizar el proyecto, aprender a usar la tecnología, y ver casos de uso parecidos.
2. Definir del problema y los requisitos: ver que requisitos debe cumplir nuestro programa para que sea efectivo y cumpla su objetivo.
3. Diseño: un diseño tanto de los casos de uso como de clases.
4. Desarrollo: programar el sistema
5. Pruebas: testeo del programa con test unitarios, de integración y end-to-end al terminar funcionalidades.
6. Documentación: redacción de la memoria del proyecto, la cual se irá elaborando en paralelo con el proyecto

Como herramientas de organización definirán historias de usuario para dividir el trabajo en pequeñas tareas que ir completando.

4.4 PLANIFICACIÓN

En la siguiente tabla, se puede observar la planificación que se ha seguido a lo largo del proyecto. Donde cada fase se divide a su vez en tareas más simples.

	10/22	11/22	12/22	01/23	02/23	03/23	04/23	05/23	06/23
Búsqueda de información									
Requisitos									
Diseño									
Desarrollo									
Pruebas									
Documentación									

Tabla 1. Diagrama de GANT

4.5 ESTIMACIÓN ECONÓMICA

Una vez estimada la duración del proyecto, se realizará un estudio de los costes de este trabajo. Se estimarán tanto los recursos materiales como humanos. Después, se presentarán tres posibles escenarios de beneficios económicos a la propuesta.

4.5.1 RECURSOS

4.5.1.1 Material

Para realizar este proyecto tan solo se ha necesitado únicamente un portátil. Las especificaciones de este se muestran en la siguiente tabla.

Portátil LG Gram

Procesador	Intel(R) Core(TM) i7-8565U CPU
Sistema operativo	Windows 10 Home Edition de 64 bits, procesador basado en x64
Tarjeta Gráfica	Intel® Iris Xe Graphics (Soporte 4K a 120Hz)
Pantalla	IPS LCD de 17" (43,6 cm), de brillo 350 nits y DCI-P3 99% antirreflejos, resolución WQXGA
Memoria RAM	16,0 GB de RAM (LPDDR5 Doble Canal 5200MHz), integrada en placa
Almacenamiento	1 TB SSD NVMe Gen. 4 (con Slot M.2 adicional para ampliación de memoria interna SSD hasta 4 TB)
Batería	Polímero de Litio de 80 Wh (10336 mAh)

4.5.1.2 Humanos

Para desarrollar este proyecto se necesita a un programador que se encargue del diseño y el despliegue del mismo. Un desarrollador de *software* tiene un salario de 29.721 € al año [24]. Es decir, que la estimación de la mano de obra de este trabajo su estimamos unos 9 meses de duración será:

$$\frac{29.721}{12} \times 9 = 22.290 \text{ €}$$

Ecuación 1: Cálculo salario programador

Asimismo, el soporte técnico será necesario para enseñar al administrador a usar la aplicación y a dar soporte en caso de cualquier posible problema. Por tanto, durante la jornada electoral se deberá dar soporte. Eso añade una jornada laboral de un día al programador, lo que supone alrededor de 50 €/h por un día completo, en el caso de contratarlo de forma puntual, lo que supone, en una jornada de 8 horas, un coste de 400 € por jornada electoral [25]. La otra opción es que directamente, el código sea mantenido por voluntarios, ya que es muy probable que en todos los partidos haya al menos un militante que conozca esta tecnología.

4.5.2 ESCENARIOS

En esta sección se presentarán tres posibles escenarios con los que nos podemos encontrar a la hora de llevar a cabo este proyecto, los denominaremos: optimista, pesimista y realista.

Para estimar el valor, debemos tener en cuenta que en las próximas elecciones del 23 de julio se prevé un presupuesto de 203 millones de euros **¡Error! No se encuentra el origen de la referencia.** De los cuales, un 46,55 % será destinado a Correos, un 6,54% destinado a la difusión del escrutinio y un 38,76% a Administraciones públicas, es decir, a fuerzas de seguridad que estén presentes en los colegios electorales. Es decir, 187,1 millones de euros que el gobierno podría no gastarse si se realizase de forma electrónica desde un dispositivo personal.

El resto de los gastos, que son 16,6 millones de euros destinados a la logística e imprevistos, podemos mantenerlos, puesto que seguimos necesitando los datos del censo electoral y la comunicación de cómo va a ser realizado el sistema de votaciones.

Teniendo en cuenta, que en España viven 36 millones de electores, se gasta 5,6 € por persona en unas elecciones. Teniendo en cuenta, que una parte de ese presupuesto se mantiene, podemos ofrecer un modelo más barato, donde el precio por votante sea de 2 € por ciudadano.

4.5.2.1 Escenario optimista

Si el gobierno está dispuesto a pagar 3 € por votante, y teniendo en cuenta que hay elecciones: generales, autonómicas, municipales y europeas. Cada una de ellas cada 4 años, menos las europeas que se celebran cada 5 años. Podemos estimar que todos los ciudadanos españoles votan una vez al año.

<i>Cuota de mercado</i>	<i>Número de ciudadanos</i>	<i>Ingresos anuales</i>
100 %	36.000.000	108.000.00 €

Tabla 2: Ingresos escenario optimista

Ahora bien, necesitamos al menos un desarrollador de apoyo por votación. Esto supone 400 € por cada votación distinta. Ahora bien, en las municipales, al ser un número mucho menor de personas por votación, podemos reducirlo a dos horas por votación, lo que supone 100 € por votación.

A la vista de los resultados de la tabla de costos en el escenario optimista. Se calcula un gasto de personal de 820.000 € cada cuatro años, lo que supone 205.000 € cada año.

<i>Tipo de votación</i>	<i>Número de programadores</i>	<i>Costo cada 4/5 años</i>
General	1	400 €
Autonómica	17	6800 €
Municipal	8124	812.400 €
Europea	1	400€

Tabla 3: Costos escenario optimista

Si tenemos todos estos gastos en cuenta, el beneficio anual que obtendremos, tal y como se recoge en el balance, será de 107,795 millones de euros anuales.

<i>Balance Anual</i>	
Ingresos	108.000.00 €
Gastos	205.000 €
Beneficios	107.795.000 €

Tabla 4: Balance anual escenario optimista

4.5.2.2 Escenario realista

Si el gobierno, ahora está dispuesto a pagar 2 € por votante, y teniendo en cuenta que solo se implanta en las elecciones generales. Las cuales ocurren cada 4 años. Podemos estimar que un 25 % todos los ciudadanos españoles votan una vez al año.

<i>Cuota de mercado</i>	<i>Número de ciudadanos</i>	<i>Ingresos anuales</i>
25 %	9.000.000	18.000.00 €

Tabla 5: Ingresos escenario realista

Ahora bien, necesitamos al menos un desarrollador de apoyo por votación, supongamos que son diez y están dos semanas trabajando con nosotros, al menos en la primera elección.

<i>Número de programadores</i>	<i>Costo cada 4 años</i>
2	40.000

Tabla 6: Costos escenario realista

En resumen, si tenemos en cuenta este escenario, tendremos un beneficio anual de 17,96 millones de euros.

<i>Balance Anual</i>	
Ingresos	18.000.00 €
Gastos	40.000 €
Beneficios	17.960.000 €

Tabla 7: Balance anual escenario realista

4.5.2.3 Escenario pesimista

Si el gobierno, ahora está dispuesto a pagar 0,5 € por votante, y teniendo en cuenta que solo se implanta en las elecciones municipales. Las cuales ocurren cada 4 años. Podemos estimar que un 25 % todos los ciudadanos españoles votan una vez al año.

<i>Cuota de mercado</i>	<i>Número de ciudadanos</i>	<i>Ingresos anuales</i>
25 %	9.000.000	4.500.00 €

Tabla 8: Ingresos escenario pesimista

Ahora bien, necesitamos al menos un desarrollador de apoyo por votación, y tenemos a un trabajador en cada provincia que está 10 días trabajando con nosotros, al menos en la primera elección. Y después un programador por municipio la semana de las elecciones. Lo que suma un total de 16.456.000 € cada cuatro años, es decir 4.114.000 euros anuales.

<i>Número de programadores</i>	<i>Días laborables</i>	<i>Costo cada 4 años</i>
52	10	208.000
8124	5	16.248.000

Tabla 9: Costos escenario pesimista

En resumen, si tenemos en cuenta este escenario, tendremos un beneficio anual de 386.000 euros.

<i>Balance Anual</i>	
Ingresos	4.500.00 €
Gastos	4.114.000 €
Beneficios	386.000 €

Tabla 10: Balance anual escenario pesimista

Capítulo 5. SISTEMA DESARROLLADO

En este capítulo se detallará el desarrollo del sistema en este Trabajo de Fin de Grado. El sistema consiste en una aplicación web descentralizada. Es un servicio API ⁴ REST, el cual se comunica con una base de datos SQL ⁵centralizada y, además utiliza un contrato inteligente. Ambas herramientas, se emplean para almacenar y leer información.

La base de datos centralizada tiene como objetivo principal proporcionar toda la información necesaria sobre los votantes, esta base solo tiene métodos de tipo GET. Y se rellena con los datos que se suministran del censo electoral desde la jefatura de gobierno.

El contrato inteligente, integra funciones más lógicas, no solo se emplea para leer información de él, sino que también se pueden crear y actualizar nuevos campos. Gracias al *smart contract*, se consigue un registro de cada transacción, creando un registro inalterable y accesible.

Para entender el desarrollo del sistema, se ha dividido en tres partes, que a su vez se dividirán en apartados más concreto. Estas secciones son: análisis del sistema, diseño e implementación. El primer apartado tiene como finalidad explicar toda la funcionalidad de nuestro desarrollo, y busca definir los requisitos principales que cubre nuestro sistema. En el diseño, se explicará la arquitectura de nuestro sistema, la estructura de las bases de datos, tanto la centralizada en SQL como la descentralizada con el contrato inteligente y, los diagramas de secuencia. Por último, se detallarán las características de los componentes más importantes: el contrato inteligente, las peticiones a la base de datos y la página web.

⁴ Interfaz de programación de aplicaciones

⁵ Lenguaje de Consulta Estructurado

5.1 ANÁLISIS DEL SISTEMA

El objetivo principal de este desarrollo es mejorar el sistema electoral mediante un sistema electrónico basado en la tecnología blockchain. Para ello, se van a estudiar en un primer lugar que escenarios se pueden encontrar para entender que requisitos se deben satisfacer.

1. La primera situación que debemos valorar es el tamaño de la votación, necesitamos crear una base de datos u otra en función de las personas con acceso a voto, y generar tantas cuentas de metamask como electores.
2. El segundo aspecto, es garantizar que todos los candidatos estén en las listas de la interfaz gráfica para poder ser elegidos. Al igual que en paso interior, la junta electoral debe suministrar los datos de los partidos inscritos **¡Error! No se encuentra el origen de la referencia..** Los cuales deben hacerlo mediante la página web del ministerio del interior.
3. La tercera situación que nos encontramos es que el votante este registrado, y vote solo una vez desde su cuenta de metamask que se le ha asignado. Para ello debemos garantizar que se suministran antes de iniciar la votación por un organismo público.
4. Por último, la distribución de resultados. Hoy en día, se gastan millones de euros en comunicar los resultados electorales. Al ser un sistema electrónico, este recuento se puede realizar de forma automática, en cuestión de segundos. Y todos los usuarios tienen acceso a los resultados, una vez la jornada electoral se ha terminado.

5.1.1 REQUISITOS DEL MODELO

Para el correcto funcionamiento de la aplicación web, debemos garantizar que el sistema sea capaz de cumplir unos requisitos tanto funcionales como no funcionales. Y en este apartado, vamos a estudiar que requerimientos cumple nuestro modelo y cuales debería cumplir en caso de que se escalase a nivel nacional.

5.1.1.1 Requisitos no funcionales

Estos requisitos son aquellos que el usuario no observa en una primera instancia, pero que son vitales para el funcionamiento de la aplicación descentralizada. Representan las características y restricciones generales del sistema que se está desarrollando. Estos requisitos se clasifican en cuatro áreas usabilidad, eficiencia, dependibilidad y seguridad. Los principales de nuestro desarrollo *software* son:

- El sistema debe ser capaz de procesar el voto de todos los españoles, suponiendo que la jornada electoral tiene una duración de 10 horas, y que la frecuencia de votaciones sigue una distribución normal. Sabiendo que hay 36 millones de votantes. Esto supone que se gestionen 2000 peticiones por segundo. Por lo que habría que valorar hacer contratos por comunidad o área
- El sistema debe ser capaz de operar adecuadamente con hasta 60.000 usuarios con sesiones concurrentes, suponiendo que tardamos unos minutos en votar.
- El sistema debe estar totalmente funcional durante la jornada electoral y el día siguiente para ver los resultados de la votación.
- Solo debe haber un administrador por contrato.
- El sistema debe registrar cada transacción en un bloque.
- No se deben poder alterar ninguno de los bloques de la cadena del contrato inteligente.
- El usuario no debe estar más de cinco minutos en la web desde que se autentifica hasta que vota.
- El sistema debe contar con un manual de usuario para explicar la conexión con la cuenta de *metamask*.
- Un usuario no debe poder votar más de una vez
- El sistema debe proporcionar mensajes de error explicativos y orientados al usuario final en varios escenarios:
 - Si el usuario y/o contraseña no son correctos
 - Si la votación no ha comenzado.
 - Si no se ha conectado la cuenta correcta de *metamask*.

- Si ya ha votado ese usuario.
- El sistema debe tener una disponibilidad total durante la jornada electoral o los horarios establecidos para la votación.

5.1.1.2 Requisitos funcionales

En este caso, los requisitos funcionales tienen como fin describir las características que el usuario detecta. Deben ser objetivos SMART⁶, esto quiere decir que deben ser específicos sobre cómo debe funcionar el programa, medibles para analizar como lo está haciendo, alcanzable en el periodo establecido, deben aportar un valor al usuario y tener un límite temporal. Los más importantes para nuestro sistema son los definidos a continuación.

- El usuario debe poder iniciar sesión con su documento nacional de identidad y contraseña.
- El sistema debe reconocer que tipo de usuario es para redirigirlo a una página u a otra.
- El administrador debe ser único.
- El administrador es el único que puede acceder a la página web de administrador.
- El administrador debe poder gestionar la votación.
- El votante solo puede acceder a las votaciones una vez estas hayan comenzado.
- El votante solo puede votar una vez.
- El votante solo puede votar desde su cuenta asociada de *Ethereum*.
- Los candidatos y votantes admitidos no pueden ser modificados durante la votación.
- Ningún votante debe poder votar antes de que el administrador haya dado comienzo a las mismas, ni después de que el periodo de votación haya finalizado.

⁶ específicos, medibles, alcanzables, relevantes y limitados en el tiempo

- Todos los usuarios deben poder ver los resultados de las votaciones, si y solo si, estas han sido finalizadas.
- Nadie puede ver el resultado parcial de las votaciones.
- La cadena de bloques debe registrar las direcciones que ya han votado
- La cadena de bloques debe almacenar cada una de las transacciones y votos de los electores.

5.1.2 HISTORIAS DE USUARIO

Para entender el funcionamiento del desarrollo software, es importante conocer a los diferentes actores que interactúan con el sistema y cuál es el rol de cada uno.

1. **Administrador:** Es la persona responsable de llevar a cabo la gestión de las elecciones. Y de todas las acciones que involucran la organización global del proceso de la votación.
2. **Votante:** son todas las personas que están autorizadas a votar. En el caso de España hay 36 millones de votantes.
3. **Junta electoral:** es el organismo encargado de administrar la base de datos de todos los votantes, así como la de los candidatos
4. **Candidatos:** son aquellos partidos políticos que se presentan a las elecciones

Ahora que sabemos quiénes son los actores principales de nuestro sistema podemos definir las acciones que realizan cada uno de ellos, en lo que se conoce como historia de usuario. Una historia de usuario, al contrario que un requisito, se centran en el valor que aporta el sistema, en vez de una explicación detalla de lo que el sistema debe hacer; describen lo que el usuario es capaz de hacer. Y los más importantes son:

- Como administrador debo poder acceder a mi página de gestión siempre que quiera.
- Como administrador debo poder dar comienzo al proceso electoral.
- Como administrador debo poder dar acceso a los usuarios.
- Como administrador debo poder añadir candidatos al proceso electoral.

- Como administrador debo poder finalizar la jornada electoral.
- Como administrador debo poder reiniciar las votaciones.
- Como administrador debo poder ver los resultados de las votaciones.
- Como votante debo poder iniciar sesión.
- Como votante debo poder conectarme a mi cuenta asociada de *metamask*.
- Como votante debo poder votar.
- Como votante debo poder ver los resultados de las elecciones.
- Como votante puedo requerir que me envíen los resultados de las elecciones por correo.

5.1.3 CASOS DE USO

Una vez se han explicado los requisitos y las historias de usuario, debemos entender como se relacionan los distintos actores entre sí y con el sistema. Para ello usamos el diagrama de casos usos.

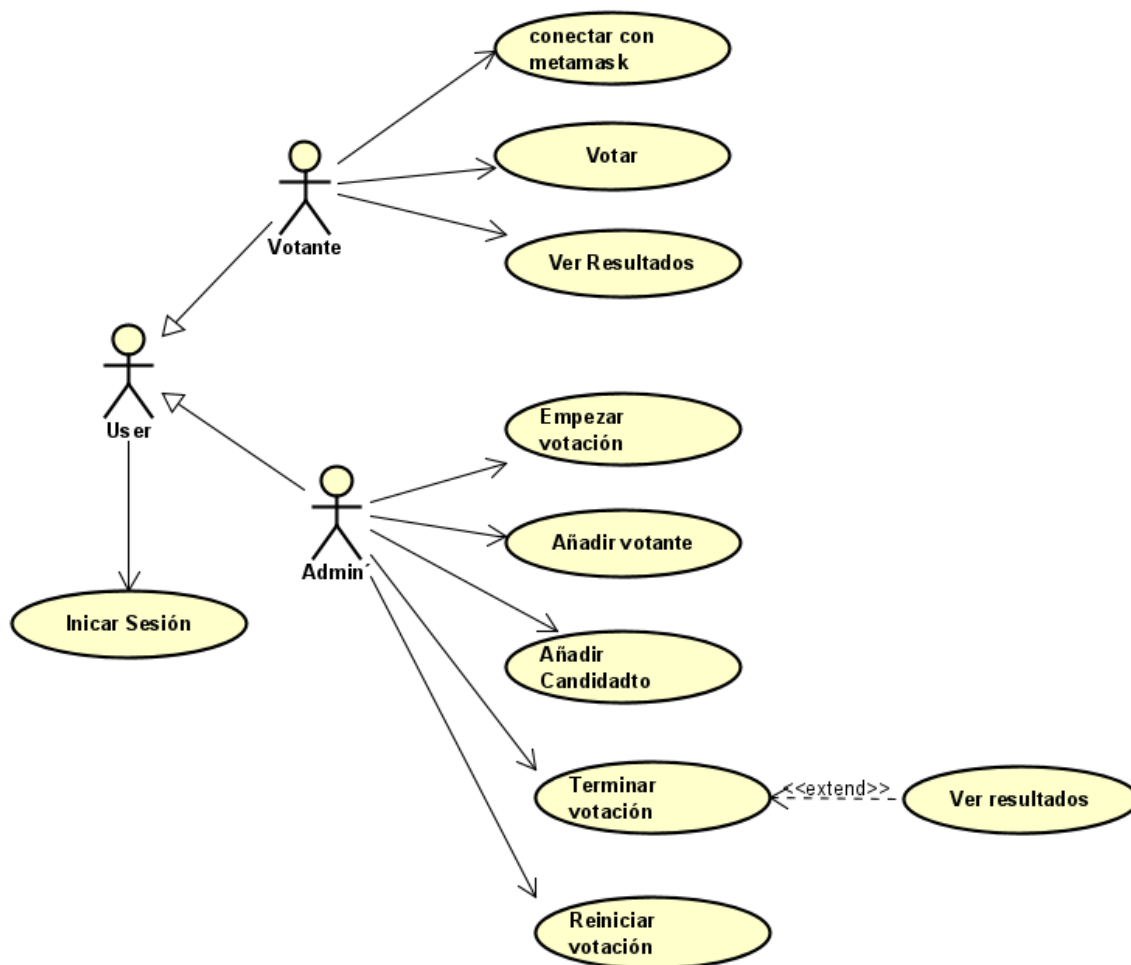


Figura 11: Casos de uso del Sistema

Como se ha definido con anterioridad, se puede observar que los actores más relevantes son el votante y el administrador, los cuales nacen de la clase usuario. Y son los únicos que pueden interactuar con el sistema y las bases de datos. Ahora bien, la jefatura de

gobierno no se ha incluido en el esquema al ser un paso previo al sistema y sale del alcance del proyecto.

Veamos un poco más detalladamente como tienen lugar estas interacciones.

5.1.3.1 Interacciones del votante

- Iniciar sesión: el usuario, intenta iniciar sesión en la página de log in, y si introduce correctamente sus credenciales y la votación ha empezado podrá pasar a la siguiente página.
- Votar: esta es la funcionalidad principal del actor, el primer paso es conectarse a metamask y si la cuenta es la misma que la registrada en la base de datos asociada a ese usuario, se le permite votar. Al votar interactúa con una transacción en la red blockchain.
- Ver resultados: esta opción solo dará un resultado si las elecciones han finalizado, abriendo un enlace a otra web con los resultados de las votaciones.

5.1.3.2 Interacciones del administrador

- Añadir candidatos: antes de iniciar las votaciones, el administrador puede modificar la lista de candidatos que aparecerán a los votantes. Es una opción muy segura, ya que añadiéndolos de esta forma se puede garantizar que todos los candidatos estaban registrados a la vez y que ninguno tiene ningún voto asociado a él.
- Añadir votante: el administrador puede añadir direcciones de nodos a la red antes del comienzo de la jornada electoral. Se pueden añadir todos los usuarios a mano o automatizarlo, de esta forma queda registrado en la blockchain que se ha asignado un voto por ciudadano.
- Empezar votación: habilita a los votantes a poder hacer su elección, puesto que antes no podían realizar el voto.
- Finalizar votación: finaliza el proceso electoral, y por tanto no se admiten más transacciones de tipo voto en la red.

- Reiniciar votación: esta es una funcionalidad que surge de la necesidad de probar nuestro contrato sin tener que desplegar uno nuevo cada vez que se cambia un detalle ajeno a él. Permite realizar dos votaciones independientes con el mismo contrato, sin necesidad de desplegar uno nuevo.

5.2 DISEÑO

Tras haber analizado el sistema y sus requerimientos, junto con las necesidades de los usuarios, se puede realizar el diseño del sistema que se desarrollará en este proyecto. El primer paso, es tener una visión global de toda la arquitectura, y para eso se emplea el diagrama de componentes. A continuación, para poder comprender la arquitectura, se usarán diagramas de clases para explicar las bases de datos y diagramas de secuencia para entender el flujo entre los distintos componentes. El último aspecto que se debe diseñar es la interfaz de usuario, es decir la página web con la que interactúan los actores de nuestro sistema.

5.2.1 DISEÑO DE LA ARQUITECTURA

En la figura se puede observar como la arquitectura se divide en cuatro componentes principales. Estos son: la DApp, la red de blockchain, el proveedor de web3 y la base de datos SQL.

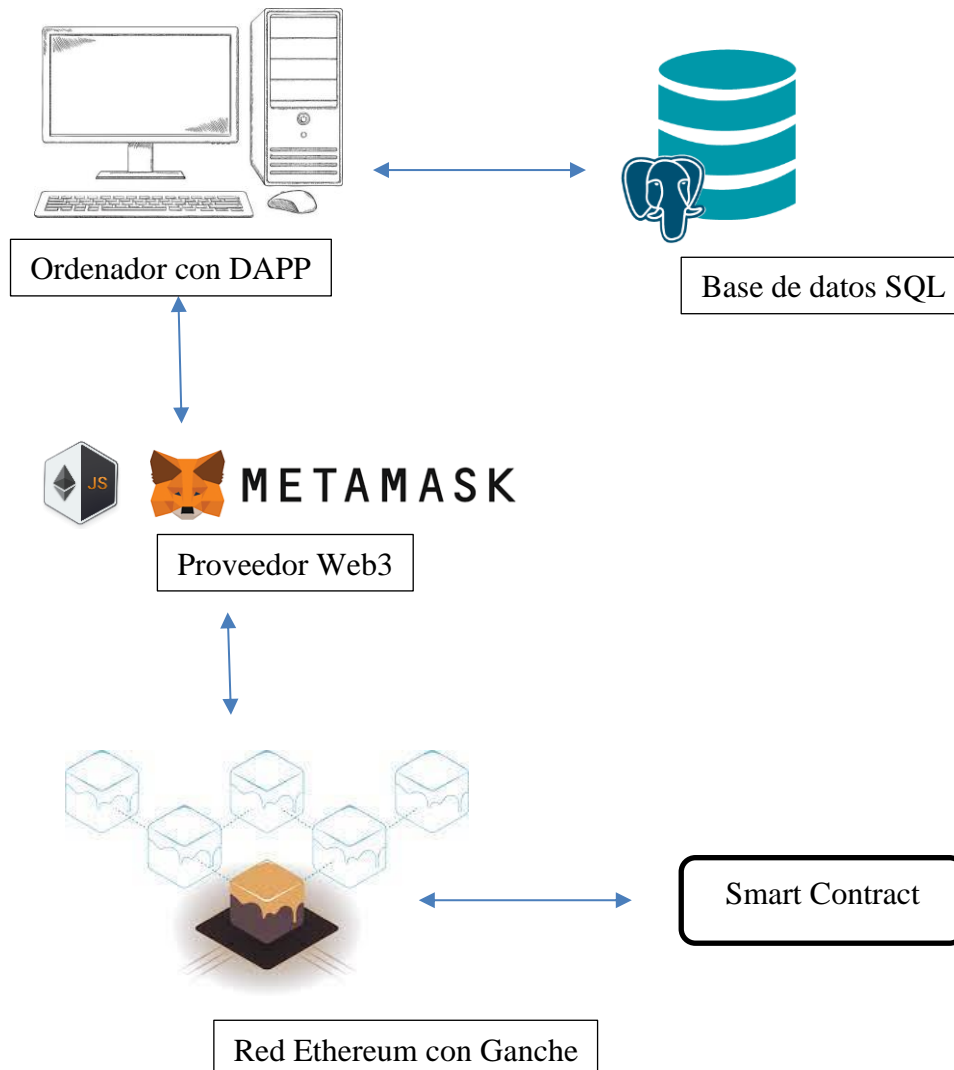


Figura 12: Arquitectura del sistema

La primera parte es la Dapp, tal y como se hemos explicado con anterioridad, a diferencia de en una base de datos relacional, una DApp permite que los datos queden almacenados en la red de blockchain **¡Error! No se encuentra el origen de la referencia..** De esta forma se garantiza que nunca se pierdan los datos del usuario. Otra razón por la que se decidió hacer uso de una aplicación centralizada fue porque están basadas en *software* libre; esto no solo permite que se alcance un significativo nivel de seguridad y transparencia,

sino que también que exista una gran comunidad de desarrolladores detrás para dar soporte a la red.

La DApp se desarrolla en una red Ethereum, ya que es una de las redes más utilizadas y que mejor funcionan con aplicaciones descentralizadas basadas en contratos inteligentes. Para desarrollar la aplicación, se ha utilizado la herramienta Ganache, que permite crear una red Ethereum personal donde se ha podido probar e implementar el contrato inteligente. Gracias a esta herramienta, se suministran 10 cuentas desde las cuales se puede probar nuestro sistema de votaciones.

Para utilizar la DApp debemos poder conectarnos a la red Ethereum donde esta desplegado el smart contract. Para realizar este paso, se ha empleado *metamask*, un proveedor de web3, que permite a cualquier usuario conectarse a la blockchain desde su ordenador personal sin necesidad de monetizar sus datos.

El último componente principal de nuestra arquitectura es la base de datos relacional. Esta base de datos se ha desarrollado con postgresSQL, un sistema de administración de bases de datos relacional orientada a objetos. Se ha empleado esta herramienta además del Smart contract para almacenar la información sobre los votantes y candidatos con la que se inicializa nuestro Smart contract. También se emplea para el primer paso de autenticación, previo al acceso a la blockchain. Esto se implementa porque si esta información estuviera en la red de blockchain, se ralentizaría la aplicación, y los tiempos de respuestas serían muy grandes. ´

Para establecer la comunicación de la aplicación con la base de datos centralizada, se usan una API, es decir, se hace una llamada a un *endpoint* definido anteriormente en el *backend*. Cuando el usuario realiza ciertas acciones en la interfaz web, se hace una llamada a distintas URIs, para obtener datos o actualizar campos del servidor.

Por otro lado, la comunicación con el contrato inteligente difiere un poco, se crea una instancia de web3, se establece la cuenta desde la que se quiere interactuar y luego se instancia el contrato desplegado utilizando la dirección y el ABI proporcionados. Esto

permite interactuar con el contrato utilizando métodos y eventos definidos en el ABI. Llamando directamente a las funciones definidas en el *smart contract*.

5.2.2 ESTRUCTURA DE DATOS

Las estructuras de datos son formas de organizar grandes cantidades de información de manera eficiente. En nuestro sistema tenemos dos herramientas para manejar los datos, la base de datos relacional y el contrato inteligente.

5.2.2.1 Base de datos relacional

La base de datos relacional es la encargada de almacenar los datos de los candidatos y de los votantes. Se emplea esta base de datos para llenar la blockchain y cargar con mayor rapidez las respuestas. También permite acceder a datos de los usuarios que no queremos que sean conocidos por toda la red de blockchain. Para hacer una representación visual de los objetos del sistema, usamos un diagrama de clases, el cual se puede ver en la siguiente figura.

VOTANTE	
PK	id: INTEGER
	dni: VARCHAR(10) NOT NULL
	password: VARCHAR(255) NOT NULL
	address: VARCHAR(255) NOT NULL
	user_type: INTEGER
	email: VARCHAR(255)

Tabla 11: Diagrama clase Votante

CANDIDATO	
PK	id: INTEGER
	name: VARCHAR(255) NOT NULL
	votos: INTEGER

Tabla 12: Diagrama de clase Candidato

Para entender mejor cada tabla se explicarán individualmente, ya que no tienen una relación directa entre ellas.

5.2.2.1.1 Tabla votantes

Almacena la información de todos los ciudadanos que pueden votar en esas elecciones. Sus atributos son:

- *dni*: sirve para la autenticación de los votantes
- *password*: sirve para que solo el votante asociado a ese dni pueda iniciar sesión.
- *address*: es la dirección única de metamask a la cual está asociada
- *user_type*: esta variable tiene como finalidad identificar si el usuario es de tipo votante o administrador del sistema. El de tipo administrador, se identifica con un 0 y el de tipo votante con un 1.
- *email*: es el mail de los votantes en caso de que se les quiera notificar de la cuenta de *metamask* o del fin de las elecciones.

5.2.2.1.2 Tabla candidatos

Almacena los datos de los candidatos a las elecciones, y se emplea para cargar los candidatos a la web y para garantizar que el conteo de votos esta a cero. Sus atributos son los siguientes:

- *name*: es el nombre del candidato o partido político
- *votos*: es el número de votos por urna, se inicializa a cero.

5.2.2.1.3 Inicialización de la base de datos

Para inicializar la base de datos desde la DApp se emplean tres archivos: *schema.sql*, *data.sql* y *applications.properties*.

El primero define las entidades y tiene la misma estructura que las tablas definidas en el apartado anterior:

```
DROP TABLE IF EXISTS "USER";

create table "USER" (
  "ID" integer primary key,
  "DNI" VARCHAR(10) NOT NULL,
  "PASSWORD" VARCHAR(255) NOT NULL,
  "ADDRESS" VARCHAR(255) NOT NULL,
  "USER_TYPE" integer,
  "EMAIL" VARCHAR(255)
);

DROP TABLE IF EXISTS "CANDIDATE";

create table "CANDIDATE" (
  "ID" integer primary key,
  "NAME" VARCHAR(255) NOT NULL,
  "VOTOS" integer
);
```

El segundo se emplea para definir los datos que queremos introducir en nuestra tabla, aunque también se puede hacer importando un CSV. ´

```
INSERT INTO "USER" ("ID", "DNI", "PASSWORD", "ADDRESS", "USER_TYPE", "EMAIL")
VALUES (1, '0a', 'aa', '0xe87068450527017f5c233aAA83cb9f5bA21bb6C0', 0,
'201900561@alu.comillas.edu');

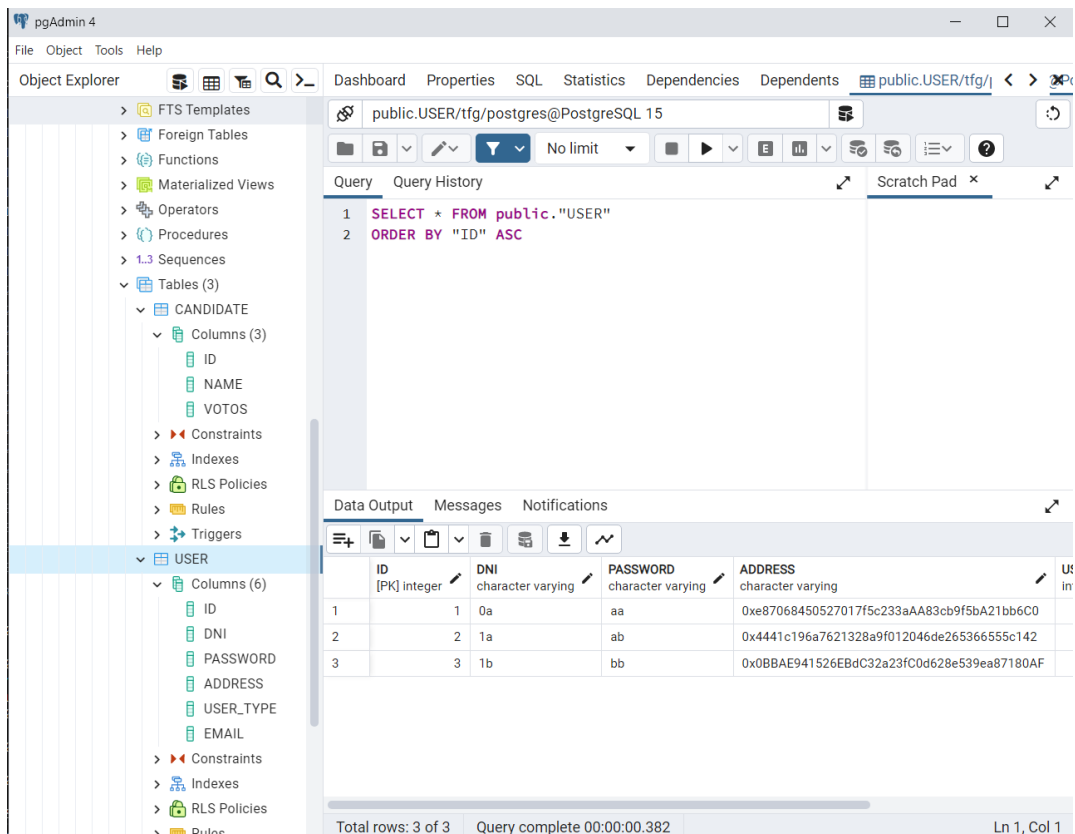
INSERT INTO "CANDIDATE" ("ID", "NAME", "VOTOS")
VALUES (3, 'PSOE', 0);
```

El ultimo es el que especifica a la aplicación que debe conectarse con la base de datos centralizada sql en postgres e inicializarla con los valores definidos en los dos archivos previos.


```
spring.datasource.url= jdbc:postgresql://localhost:5432/tfg
spring.datasource.username= postgres
spring.datasource.password= postgres
spring.datasource.initialization-mode=always

spring.datasource.sql-script-encoding=utf-8
#spring.datasource.initialization-mode=embedded
#spring.datasource.platform=h2
spring.datasource.schema=classpath:schema.sql
spring.datasource.data=classpath:my_data.sql
```

Esta base de datos se al ser creada con *postgresSQL*. Tiene una interfaz de visualización asociada donde se pueden ver las tablas creadas. Esta herramienta se llama pgAdmin y podemos observar como muestra las tablas en la siguiente imagen. En esta captura en concreto, se observan todas las entradas de la tabla votantes.



The screenshot shows the pgAdmin 4 interface. The left sidebar displays the 'Object Explorer' with the 'USER' table selected under 'Tables (3)'. The main window shows a query: `SELECT * FROM public."USER" ORDER BY "ID" ASC`. The 'Data Output' pane displays the following table:

ID	DNI	PASSWORD	ADDRESS	USER_TYPE
1	0a	aa	0xe87068450527017f5c233aAA83cb9f5bA21bb6C0	
2	1a	ab	0x4441c196a7621328a9f012046de265366555c142	
3	1b	bb	0x0BBAE941526EBdC32a23fC0d628e539ea87180AF	

Total rows: 3 of 3 Query complete 00:00:00.382 Ln 1, Col 1

Figura 13: pgAdmin

5.2.2.2 Smart Contract

Para este proyecto tan solo se necesita un contrato, a este contrato lo denominamos *Election*. Esta contrato contiene dos estructuras, cuatro atributos, una enumeración, cuatro modificadores y nueve métodos. En la figura siguiente se puede ver la clase *Election*, con sus métodos, atributos y estructuras en un diagrama de clases.

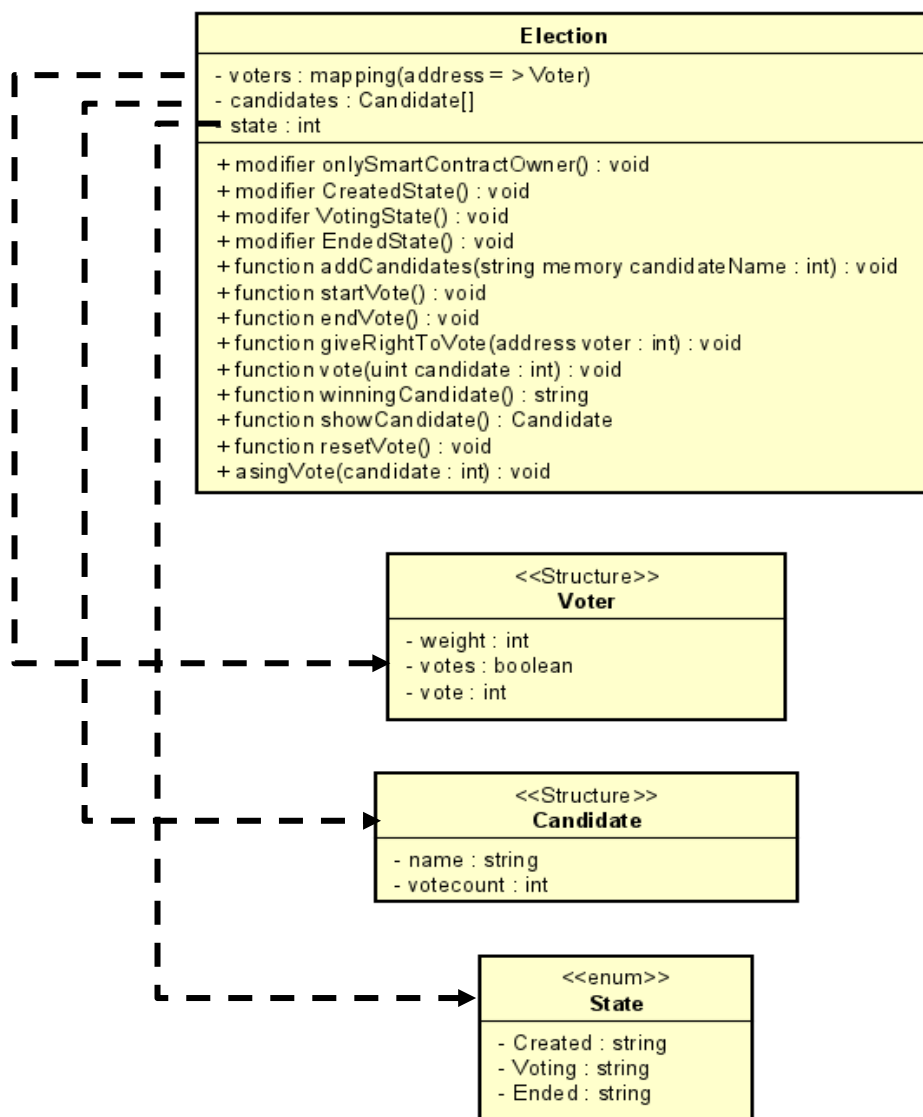


Figura 14: Diagrama de la clase smart contract

La estructura *Election* contiene tres atributos: *voters*, *candidates* y *state*. El primero es una variable mapeada, lo cual es equivalente a un diccionario en Python, con una clave y valor, que asocia una dirección de una cuenta de la red de blockchain con un votante. El segundo es una lista de candidatos, y el último es una variable de tipo *State*, una enumeración que indica en que paso de la elección está el sistema.

Además, en este contrato podemos encontrar todos los datos relevantes para el conteo de los votos. La idea es simular a una urna electoral. Es decir, se registran todos los usuarios para comprobar que cada usuario solo vota con una cuenta de *metamask*, también se comprueba que un votante no vote dos veces con la misma cuenta. Y en cada instancia de candidatos se va sumando la opción del votante, ahora bien, ese registro se hace desde la cuenta del administrador, simulando que se lee una papeleta de la urna.

La estructura *Voter*, tiene dos atributos relevantes: *weight* y *voted*. El primero, se emplea para comprobar que ese usuario puede votar en estas elecciones, es decir, que el administrador le ha registrado en el periodo electoral. Y la segunda variable, indica si el votante ha votado ya o no.

La estructura *Candidate* tiene como finalidad almacenar el recuento de votos, se almacena el nombre del candidato o partido político junto con el número de votos asociados a él.

La última estructura de datos relevante en nuestra implementación es la enumeración *State*. Esta enumeración contiene tres estados: *created*, *voting* y *ended*. Dependiendo del estado en el que nos encontramos, algunas funciones u otras estarán disponibles para ejecutar en el contrato inteligente. Esto lo definimos con los modificadores.

En la documentación de Solidity se explican los *modifiers* de la siguiente forma: “*Los modificadores de función se usan para enmendar de un modo declarativo la semántica de las funciones*”[7]. Es decir, se usan para cambiar el comportamiento de un método de forma ágil. Por ejemplo, comprueban automáticamente una condición antes de correr la función. En este contrato hay cuatro modificadores:

- *onlySmartContractOwner()*: Indica si la dirección que va a ejecutar ese método es la misma que la dirección que ha creado el contrato.
- *CreatedState()*: Indica si la variable *state* está en el estado de creación.
- *VotingState()*: Indica si la variable *state* está en el estado de votación.
- *EndedState()*: Indica si la variable *state* está en el estado de fin.

Los métodos sirven para implementar a lógica entre las estructuras que se acaban de definir, y por tanto se explicarán en el apartado destinado a la implementación del sistema. Como idea general, las funciones definen el comienzo y final de las votaciones, añaden votantes y candidatos a las votaciones, hacen la acción de votar y emiten los resultados.

5.2.3 DIAGRAMA DE SECUENCIA

Para entender y completar el diseño del sistema a desarrollar en este proyecto, se han realizado tres diagramas de secuencia en los que se puede observar el flujo de los usuarios al realizar una acción en el sistema.

En todos los diagramas hay un primer paso de autenticación del usuario, si ese usuario es de tipo votante se abrirá el html de electores y si no se abre el de administrador, cada uno de ellos con su archivo JavaScript asociado.

Empezamos por el diagrama del administrador, el primer paso, es la identificación. Una vez hemos comprobado que es de tipo administrador, podemos acceder a todas sus funcionalidades. En este caso, se añade un votante al contrato y se empieza la votación.

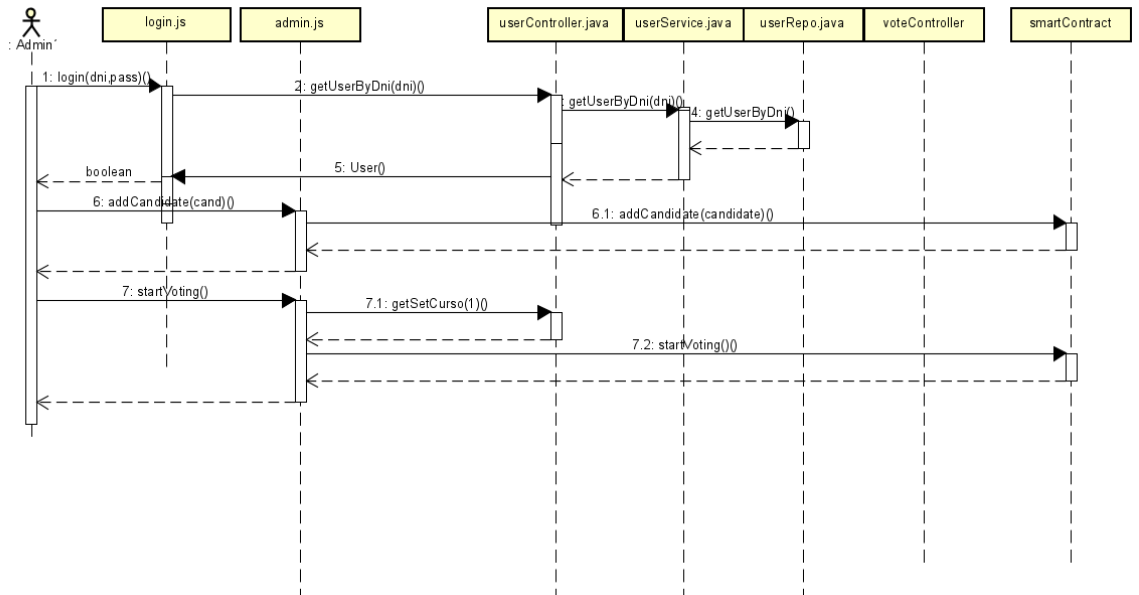


Figura 15: Diagrama de secuencia del administrador

El flujo para el resto de las acciones que puede ejecutar el administrador es muy similar, siempre hay una autenticación y una interacción con el contrato. Siempre empleamos la misma cuenta de *metamask* por defecto.

5.2.3.1 Anonimato

Uno de los retos más grandes de este sistema es garantizar el anonimato del voto. En la ley Orgánica 5/1985, sección 13 Artículo 86 establece: “*El voto es secreto*” **¡Error! No se encuentra el origen de la referencia.** El voto o sufragio secreto es una garantía del sistema electoral que garantiza que el voto de una persona pueda ser conocido o influenciado por otra persona.

Esto no supone que lo votado sea secreto, sino que no pueda asociarse a una persona en concreto. Buscando de esta forma un voto incondicional y libre. Para resolver este problema se proponen dos soluciones.

5.2.3.1.1 Solución 1: Simulación de Urna

La primera idea, que se puede ver reflejada en el siguiente diagrama de secuencia, es almacenar los votos en una base de datos centralizada después de la autenticación del votante. Y al llegar a un número preestablecido de votos, el número de votos por candidatos se envía a la blockchain como una única transacción.

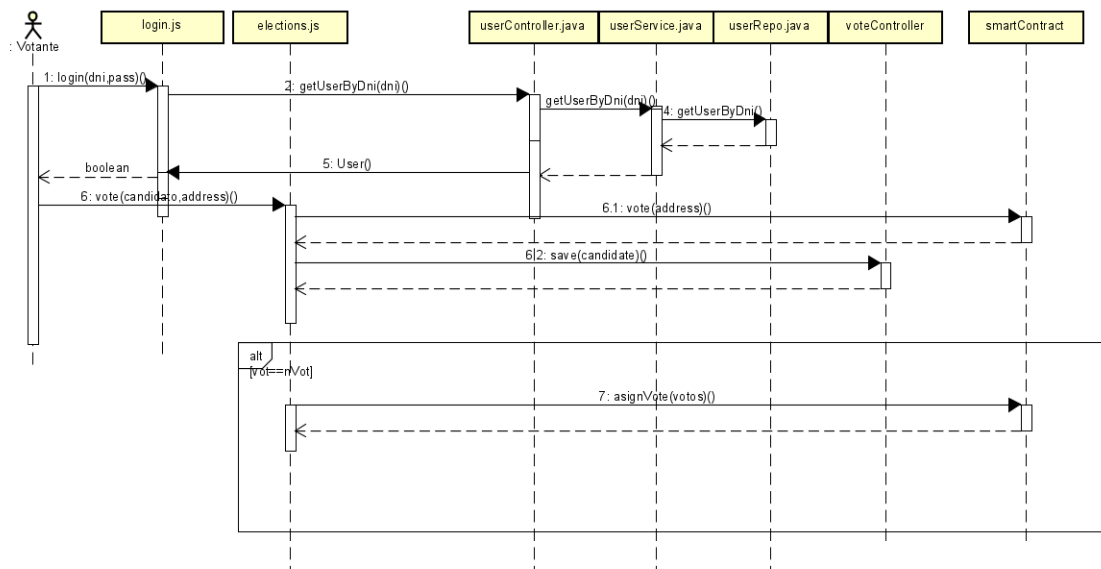


Figura 16: Diagrama de Secuencia solución 1

Ahora bien, esta solución tiene una gran desventaja, al almacenar los datos en la base centralizada antes de guardarlos a la blockchain tiene grandes riesgos de ciberseguridad. Una base de datos centralizada puede convertirse en un objetivo atractivo para los ciberdelincuentes. Si logran acceder de manera exitosa, podrían robar o manipular los datos almacenados. Esto puede resultar en la pérdida o corrupción de la información.

Además, si la base de datos centralizada experimenta un fallo o un error técnico, existe el riesgo de perder los datos almacenados en ella. Esto podría ser especialmente problemático si no se realizan copias de seguridad regulares o si las medidas de recuperación de datos no son efectivas.

Por último y como aspecto más crítico, con este sistema se busca la inmutabilidad del voto. Y en una base de datos centralizada, existe el riesgo de que los datos sean censurados o manipulados por la entidad que controla la base de datos.

Por estos motivos se propone la solución dos. Donde toda la información queda almacenada en la blockchain. La información almacenada en una blockchain se distribuye en múltiples nodos de la red, y se asegura mediante criptografía lo que brinda mayor confianza y resistencia a ataques cibernéticos.

5.2.3.1.2 Solución 2: registro como administrador

En esta solución, se identifica al votante, pero el voto en la blockchain queda registrado por el administrador automáticamente. Es decir, después de asegurarnos que el votante tiene permiso para votar, el administrador emite el voto a la blockchain con su cuenta, de esta forma el voto no queda asociado al votante.

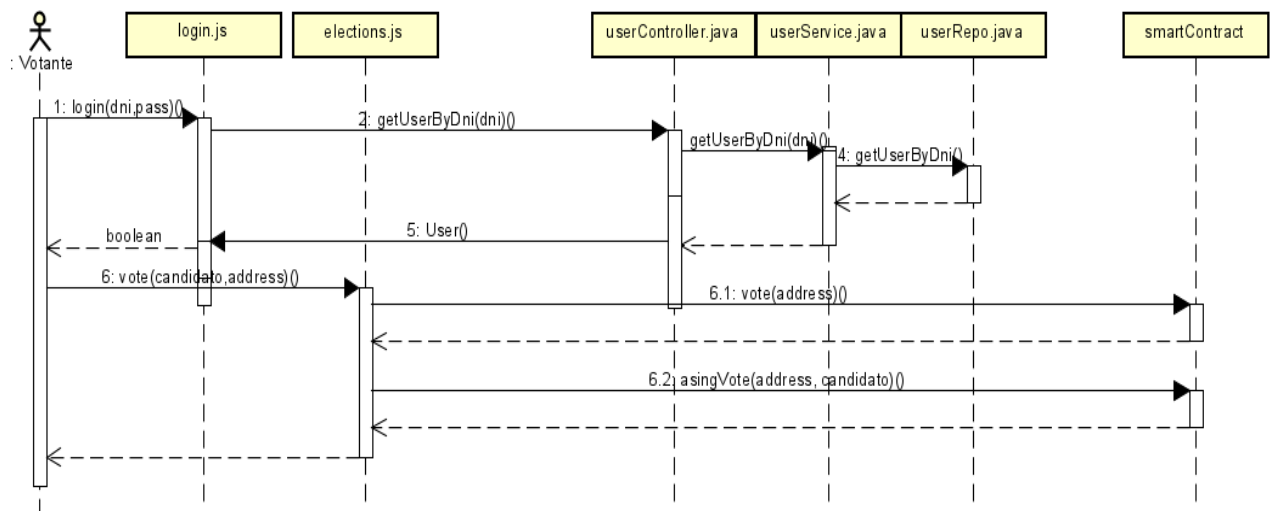


Figura 17: Diagrama de secuencia escenario 2 votante

Esta solución tiene un posible fallo que es asociar al votante al voto *timestamps* u orden de los bloques, este problema se puede solucionar con la ZKP⁷, la prueba de cero conocimientos. Para nuestro sistema, contamos con que miles de personas interactúan a la vez, por lo que el primer paso de autenticación se hace de manera casi concurrente y a continuación el paso de asignar el voto. Por tanto, al no ser el conjunto de los dos pasos atómicos, sino que tan solo son secuenciales, va a haber varias autenticaciones antes de las asignaciones o una asignación previa antes del del voto actúa, y de esa forma, se pierde a quien le corresponde cada voto.

5.3 IMPLEMENTACIÓN

Para entender la estructura del proyecto debemos centrarnos en tres partes principales: *frontend*, *backend* y contrato. Ambos tres dentro del proyecto la aplicación. La parte visual y la parte del servidor están dentro del main y a su vez, el *backend* en la parte de java y el *frontend* en recursos. Lo podemos ver de una forma más clara en la siguiente ilustración. En este apartado, se explicará como implementamos cada parte en nuestro sistema.

⁷ Zero-knowledge proof

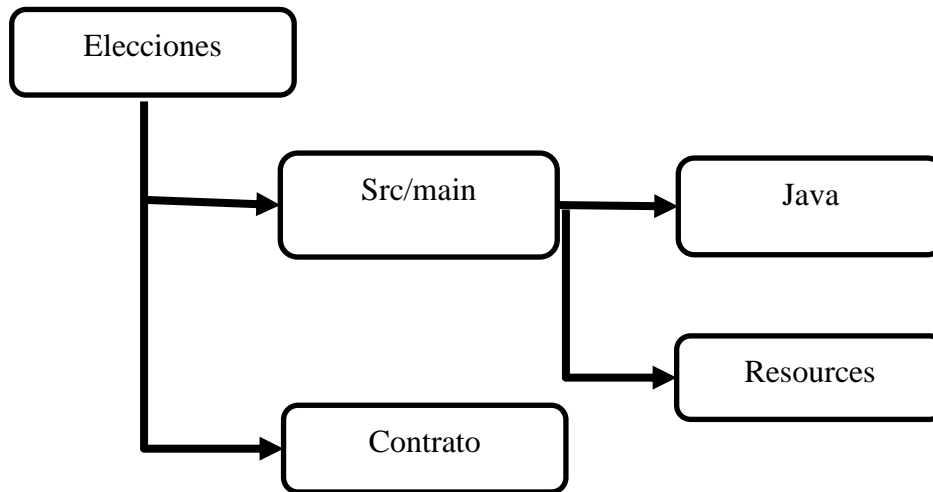


Figura 18: Estructura de las carpetas del proyecto

5.3.1 SMART CONTRACT

El Smart Contract que hemos definido anteriormente tiene ocho métodos que los usuarios pueden ejecutar, algunos han sido creados para pruebas, por lo que vamos a centrarnos en los más relevantes de nuestro sistema. El código del contrato se encuentra en el anexo II.

- function startVote() public onlySmartContractOwner CreatedState: pone la variable state en el estado de votación.
- function giveRightToVote(address voter) public onlySmartContractOwner: Este método sirve para añadir todas las cuentas de los ciudadanos que pueden votar en estas elecciones y el modificador indica que solo puede ser ejecutado por el dueño del contrato
- function vote() public VotingState: en este método los usuarios quedan registrados como que van a emitir un voto, solo se les añade en una lista. Es lo mismo que si subrayarán tu nombre en la lista del censo antes de votar.

- function winningCandidate() public EndedState: sirve para ver los resultados una vez la votación haya terminado.
- function endVote() public onlySmartContractOwner VotingState: pone la variable state en el estado de terminado.
- function addCandidates(string memory candidateName) public CreatedState: se emplea para añadir a los candidatos antes de comenzar la votación
- function asignVoto(uint candidate) public onlySmartContractOwner VotingState: sirve para asignar el voto al candidato desde la cuenta de administrador

5.3.2 BACKEND

Esta es la parte del servidor. Si entramos en las carpetas de nuestro proyecto en visual code, podemos observar la estructura de los paquetes. Estos se dividen en entidades, controlador, servicio y repositorio.

Primero definimos las clases de java User y Candidate, ambas clases tienen como fin representar a las tablas a las que están asociadas. Para asociarlas a una tabla en la base de datos usamos la notación de *spring* @Table y para indicar que atributo está asociada a cada atributo utilizamos el decorador @Column. Gracias a *spring* con la dependencia de Lombok podemos insertar un constructor por defecto con @AllArgsConstructor y todos los *getters* y *setters* con @Data. El código de la clase User queda de la siguiente forma:

```
@AllArgsConstructor
@NoArgsConstructor
@Table("USER")
public class User {
    private @Column("ID") @Id Long id;

    private @Column("DNI") String dni;

    private @Column("PASSWORD") String password;

    private @Column("EMAIL")
    @Pattern(regexp = "^[a-zA-Z0-9_\\.+-]+@[a-zA-Z0-9-]+.[a-zA-Z0-9-\\.]+$")
    String email;

    private @Column("USER_TYPE") Integer userType;

    private @Column("ADDRESS") String address;
```

El siguiente paso es definir el repositorio asociado a estas dos clases, uno por cada una de las tablas, con la herencia de *CrudRepository* vienen bastantes *queries* que necesita nuestro sistema por defecto. Para indicar que es el repositorio, usamos la notación `@Repository`

```
@Repository
public interface UserRepository extends CrudRepository<User, Long> {
    List<User> findByDni (String dni);
}
```

El repositorio es llamado por el *service*. En este paso se transforman los datos que devuelve la llamada a la base de datos en objetos de tipo *User* o *Candidate*. El servicio se define con `@Service` y usa la inyección de dependencias con la anotación `@Autowired` para inyectar al repositorio. En el siguiente código se expone un ejemplo de un servicio con un método que devuelve todas las filas de la base de datos de usuarios transformados en una lista de objetos de tipo *User*.

```
@Service
public class UserServiceImpl implements UserService {

    @Autowired
    UserRepository userRepository;

    public List<User> getAllUsers () {
        return StreamSupport.stream(userRepository.findAll().spliterator(),
false)
                .map(obj -> new User(
                    obj.getId(),
                    obj.getDni(),
                    obj.getPassword(),
                    obj.getEmail(),
                    obj.getUserType(),
                    obj.getAddress())) .toList();
    }
}
```

Por último, para conectar esta parte con el *frontend*, usamos la clase controlador. En esta clase se definen los endpoints a los que llamamos desde el JavaScript para comunicarnos con la base de datos relacional. Para indicar que es el controlador añadimos la anotación `@RestController`. También especificamos la ruta a los endpoints de ese controlador con `@RequestMapping`. En el siguiente código se observan las peticiones de tipo GET para la tabla User:

```
@RestController
@RequestMapping("/api/users")

public class UserController {

    @Autowired
    UserService userService;

    @GetMapping("/allUsers")
    public ResponseEntity<List<User>> getAllUsers () {
        LOGGER.info("entra");
        return ResponseEntity.ok().body(userService.getAllUsers());
    }

    @GetMapping("dni/{dni}")
    public ResponseEntity<User> getUserbyDNI (@PathVariable("dni") String dni) {
        List<User> userList = userService.getUserByDni(dni);

        if(userList.isEmpty()){
            return ResponseEntity.noContent().build();
        }else{
            User u = userList.get(0);
            return ResponseEntity.ok().body(u);
        }
    }

    @GetMapping("en_curso")
    public ResponseEntity<Integer> getCurso () {
        if(en_curso==0) {
            return ResponseEntity.ok().body(0);
        }else{
            return ResponseEntity.ok().body(1);
        }
    }
}
```

El último paso que debemos configurar en la parte del servidor de la aplicación son las propiedades de la aplicación y las dependencias. Las dependencias se añaden al pom.xml al crear el proyecto con *spring initializer*. Y las propiedades se definen en application.properties. Donde además de conectarnos con la base de datos, indicamos a que puerto debemos conectarnos para abrir la aplicación, en este caso 8080.

```
server.port=8080
spring.output.ansi.enabled=always
logging.level.web=INFO
#logging.level.org.springframework.jdbc.core=DEBUG
logging.level.org.springframework.jdbc.core=TRACE
```

5.3.3 FRONTEND

Dentro de la carpeta *resources* encontramos todos los HTMLS junto con sus archivos de *javascript*. Tenemos un *.js* por cada *html*. Y tenemos cuatro: *index.html*, *admin.html*, *elecciones.html* y *resultados.html*.

Al llamar a la *url* de localhost:8080 se abre por defecto el *html* denominado *index.html*, esta es la página para iniciar sesión. Una vez introducimos el usuario y contraseña, se abre una página u otra dependiendo del tipo de usuario que sea. Si es un votante se abre *elecciones.html* si ha comenzado el periodo de votación, sino devuelve un error. Si se inicia sesión con un usuario de administrador se abre automáticamente la página de *admin.html* donde puede ejecutar las diferentes acciones. Por último, se abre la página de *resultados* al concluir la jornada electoral. En la figura siguiente podemos observar todos los escenarios posibles:

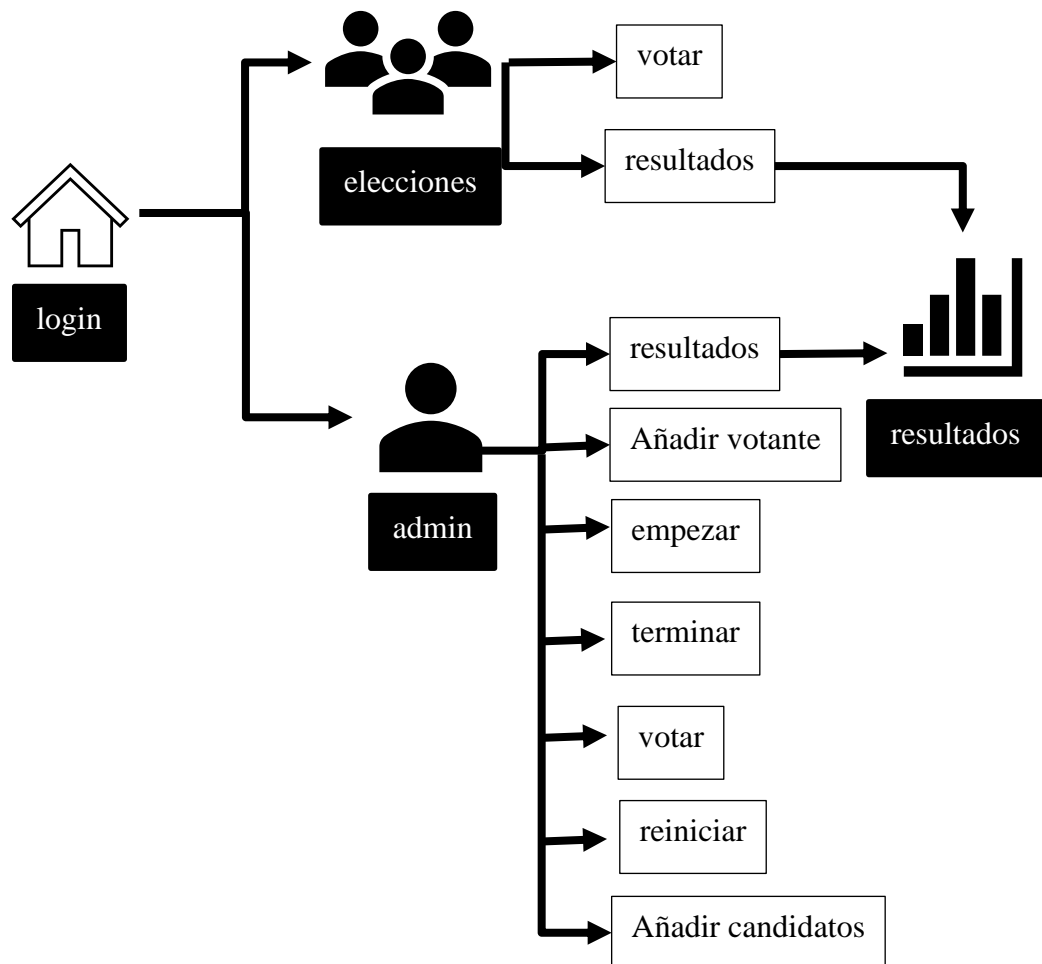


Figura 19: Diagrama de navegabilidad de la página web

El último elemento que debemos incluir en la lógica de nuestra interfaz gráfica es la conexión con la base de datos relacional y con el contrato.

5.3.3.1 Conexión con el contrato

Para poder interactuar con el contrato, es necesario establecer una conexión con el proveedor de web3, y crear una instancia del tipo *Contract* con la dirección del contrato y la variable ABI. Por último, debemos conocer la dirección desde la que se va a hacer la llamada al contrato, e instanciar el método. En el siguiente código, se está finalizando la votación.

```
var web3 = new Web3(new Web3.providers.HttpProvider("http://localhost:8545"));

//Se instancia el contrato desplegado
const deployedContractAddress = "0x8E46556364e34a2c3B86fcFa31562bEeF0b7dA70";

var contratoElecciones = new web3.eth.Contract(ABI, deployedContractAddress);
const accounts = await window.ethereum.request({ method: 'eth_requestAccounts'
});
account = accounts[0];

contratoElecciones.methods.endVote().send({from: account }, (error,result) => {
console.log(error);
```

5.3.3.2 Conexión con la base de datos centralizada

En este caso hay que realizar un fetch a la url deseada. Como parámetro, hay que poner el *endpoint* definido en el controlador en la parte del servidor.

```
return fetch(`api/users/en_curso`).then(response => response.json())
.then(data => en_curso = data)
```

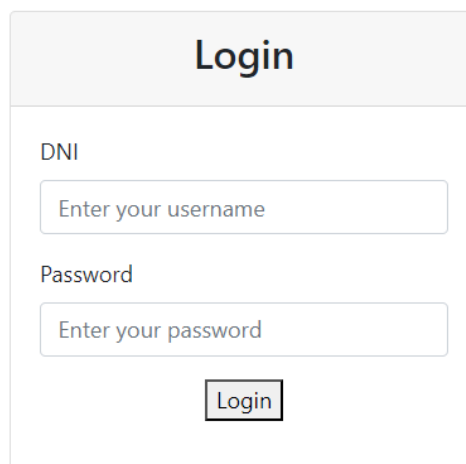
Capítulo 6. ANÁLISIS DE RESULTADOS

En este capítulo se mostrarán los resultados de la implementación detallada en el capítulo previo, y se demostrará como se han logrado los requisitos descritos.

El objetivo del este Trabajo de Fin de Grado es crear una aplicación web desde la que cualquier ciudadano pueda emitir un voto, y la jefatura de gobierno pueda gestionar. Por tanto, se analizará el producto final de la interfaz digital, en el cual se hacen todas las llamadas a la base de datos y al contrato. Se explicarán todos los posibles escenarios que podemos encontrar.

6.1 INICIO DE SESIÓN

Lo primero es la página de inicio de sesión, que se despliega por defecto y es la que se observa en la siguiente imagen. En ella se introducirá el usuario y la contraseña. Y podemos encontrar con dos escenarios de error. Si se accede correctamente se abrirá la correspondiente página de votante o administrador.

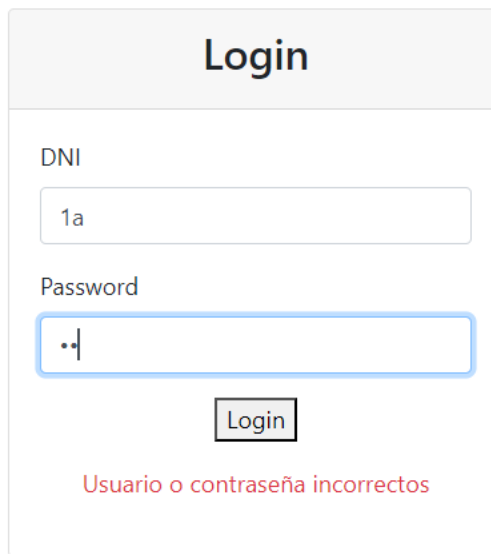


The image shows a web form titled "Login". It contains two input fields: one for "DNI" with the placeholder text "Enter your username", and one for "Password" with the placeholder text "Enter your password". Below the fields is a "Login" button.

Figura 20: página web de inicio de sesión

6.1.1.1 Inicio sesión con credenciales incorrectos

La primera comprobación es la correcta combinación de usuario y contraseña. El JavaScript comprueba que el usuario exista en nuestra base de datos y que la contraseña sea la misma que la asignada. Sino muestra un mensaje de error, tal y como se observa en la siguiente figura.



Login

DNI
1a

Password
·|

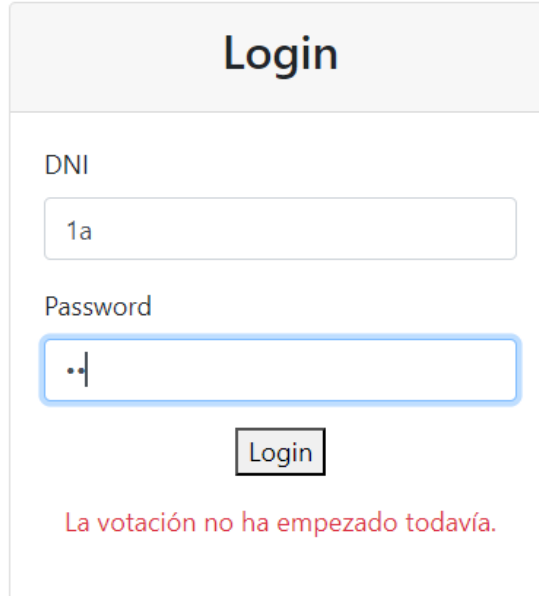
Login

Usuario o contraseña incorrectos

Figura 21: Escenario 1, error al iniciar sesión

6.1.1.2 Inicio de sesión antes del inicio de la votación

Para que un votante pueda iniciar sesión y emitir su voto, el administrador debe haber dado comienzo a la votación. Esto quiere decir, que el administrador debe haber iniciado sesión y haber pinchado en iniciar votación. Sino aparecerá el mensaje de error de la siguiente figura.



Login

DNI
1a

Password
••|

Login

La votación no ha empezado todavía.

Figura 22: Escenario 2, error al iniciar sesión

6.2 PANTALLA ADMINISTRADOR

Si iniciamos sesión con el usuario de tipo administrador, aparece la pantalla de la siguiente figura. En esta pantalla podemos introducir a los candidatos, y votantes manualmente si queremos. No obstante, al pinchar en iniciar sesión, se cargan unas cuentas de votantes por defecto.

Para que los votantes puedan iniciar sesión, el administrador debe pinchar en empezar la votación. Una vez hayamos pinchado, es importante recalcar, que no se podrán añadir más candidatos a la votación. También, podemos terminar la votación, reiniciar la votación y ver los resultados.

Votaciones

Administrador



The interface consists of several elements: a top button labeled 'Empezar la votación', a text input field for 'Dirección del votante' with the placeholder 'Enter address', a button labeled 'Añadir votante', a text input field for 'Nombre del Candidato' with the placeholder 'Enter candidate', a button labeled 'Añadir candidato', and three bottom buttons labeled 'Terminar la votación', 'Resultados de la votación', and 'Reiniciar la votación'.

Figura 23: Pantalla del administrador

Si hacemos *click* en resultados de la votación se abre otra pantalla, la pantalla de resultados. Esta pantalla calcula en número de escaños, a partir de los votos, siguiendo la ley D'Hondt. Para que tuviese sentido los resultados, y como estamos limitados a 10 cuentas gratuitas con Ganache, se han moqueado los resultados de la votación. Para los siguientes datos: el PP con 4200 votos, el PSOE con 36000 votos y Vox con 11000 votos. El resultado obtenido se muestra en la siguiente imagen.

Para poder ver los escaños asignados a cada partido, debemos pinchar el botón de mostrar los resultados.

Mostrar resultados

Resultados

Partidos	Escaños
PP	4
PSOE	3
VOX	1

Figura 24: Web Resultados

6.3 PANTALLA VOTANTE

La última pantalla que nos queda por analizar en el sistema es la del votante. El votante, únicamente puede votar. Pero para votar debe estar conectado a una cuenta correcta de metamask. Al pinchar en *vote* en la siguiente imagen, se abre la extensión para conectarnos con *metamask*.

Votaciones

Candidates

Vote

Select Candidate:

Vote

Figura 25: web votaciones 1

Una vez nos aparezca la extensión como vemos en la imagen, debemos iniciar sesión y seleccionar una cuenta de blockchain.

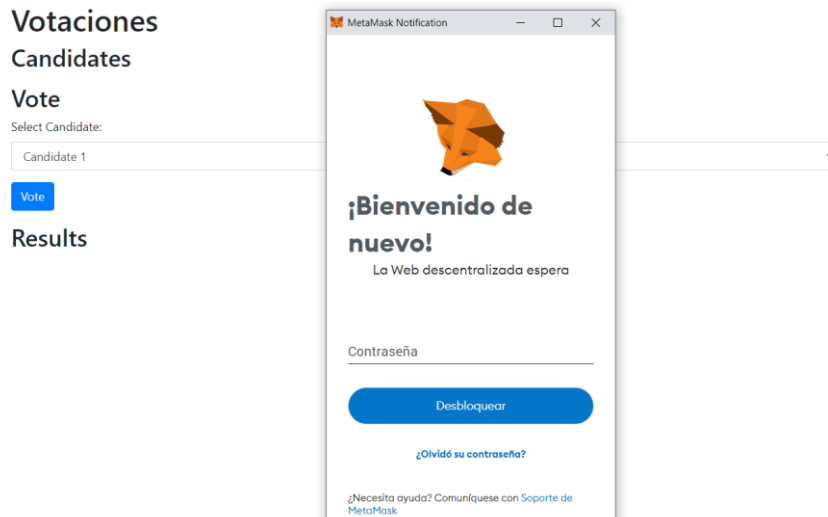


Figura 26: web votaciones conexión con metamask

La conexión con *metamask* se explica en el anexo, lo más importante, es asegurarnos que nos encontramos conectados a la red de la blockchain de las elecciones con la cuenta asignada a nuestro DNI.

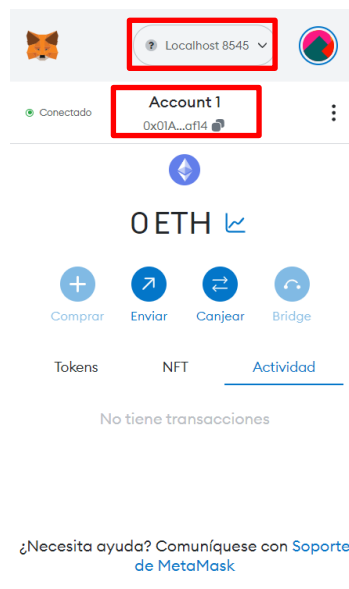


Figura 27: conexión con metamask

Si la conexión no se ha realizado de forma correcta, se pueden encontrar dos tipos de errores diferentes:

El primero, la cuenta no es la asociada a ese usuario, o no es reconocida por el sistema, y se anuncia con el mensaje de error que vemos en la imagen.

Votaciones

Candidates

Vote

Select Candidate:

Error: Cuenta de blockchain incorrecta

Figura 28: web votante error 1

El segundo, esa dirección ya ha votado y por tanto, no puede emitir otro voto.

Votaciones

Candidates

Vote

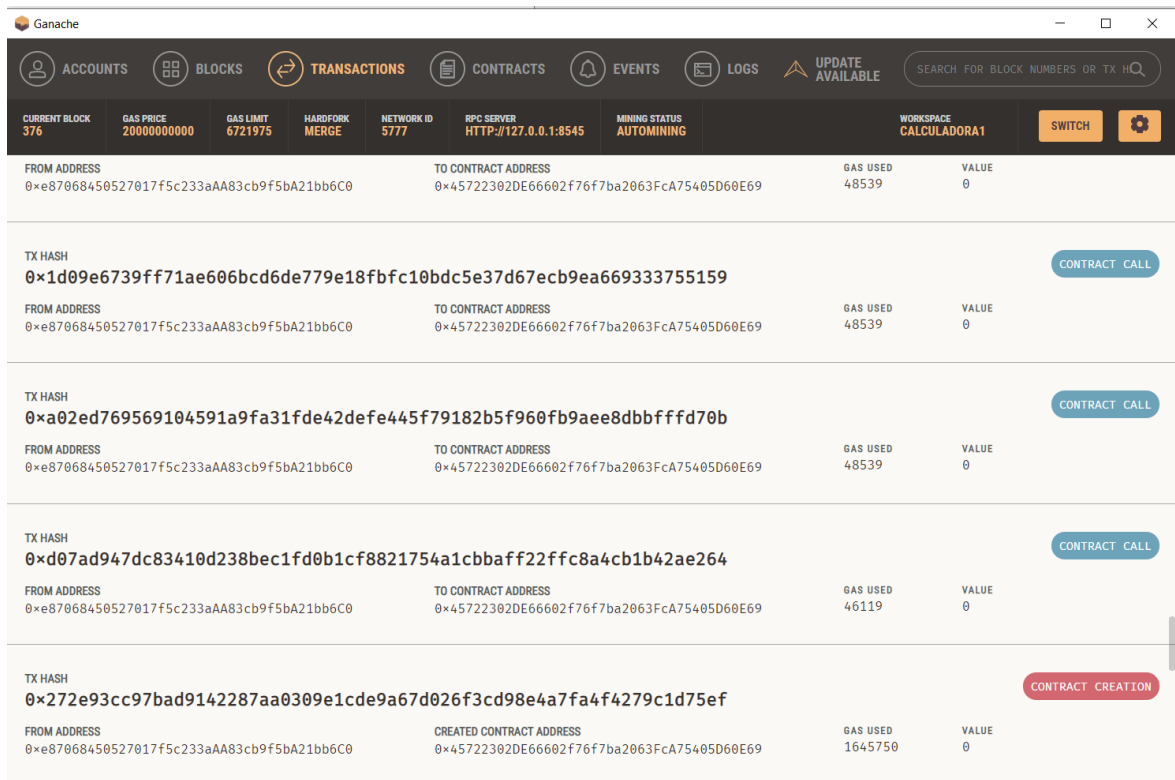
Select Candidate:

Error: Returned error: VM Exception while processing transaction: revert Already voted.

Figura 29: web votante error 2

6.4 RESULTADOS GANACHE

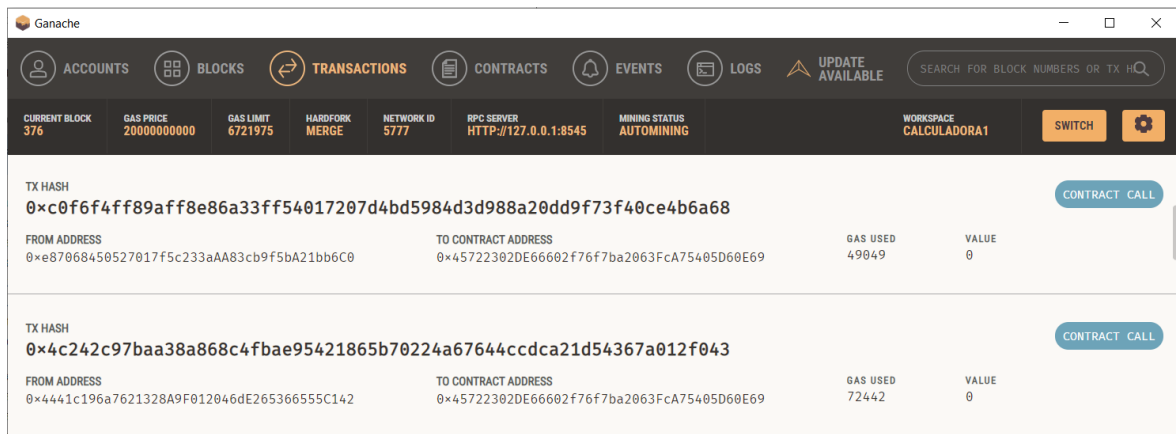
Si entramos en las transacciones de Ganache, se observa cómo se ha creado el contrato y desplegado, así como todas los bloques de la cadena que se han ido creando. Las operaciones de lectura, no se consideran una transacción, y por tanto esas no quedan registradas en la blockchain. Aquí se observan las operaciones que han alterado alguna variable en la red.



TX HASH	FROM ADDRESS	TO CONTRACT ADDRESS	GAS USED	VALUE	Operation
0x1d09e6739ff71ae606bcd6de779e18fbfc10bdc5e37d67ecb9ea669333755159	0xe87068450527017f5c233aAA83cb9f5ba21bb6C0	0x45722302DE66602f76f7ba2063FcA75405D60E69	48539	0	CONTRACT CALL
0xa02ed769569104591a9fa31fde42defe445f79182b5f960fb9aee8dbbfffdd70b	0xe87068450527017f5c233aAA83cb9f5ba21bb6C0	0x45722302DE66602f76f7ba2063FcA75405D60E69	48539	0	CONTRACT CALL
0xd07ad947dc83410d238bec1fd0b1cf8821754a1cbbaff22ffc8a4cb1b42ae264	0xe87068450527017f5c233aAA83cb9f5ba21bb6C0	0x45722302DE66602f76f7ba2063FcA75405D60E69	46119	0	CONTRACT CALL
0x272e93cc97bad9142287aa0309e1cde9a67d026f3cd98e4a7fa4f4279c1d75ef	0xe87068450527017f5c233aAA83cb9f5ba21bb6C0	0x45722302DE66602f76f7ba2063FcA75405D60E69	1645750	0	CONTRACT CREATION

Figura 30: Transacciones Ganache

Además de la vista general, si miramos en detalle, podemos ver como al votar, la dirección es la que hemos introducido en los votantes, y que justo a continuación, se crea otro bloque, que es el registro de esa votación con la cuenta del administrador. Se observa en la figura siguiente.



The screenshot shows the Ganache application interface. At the top, there is a navigation bar with icons for ACCOUNTS, BLOCKS, TRANSACTIONS (highlighted), CONTRACTS, EVENTS, and LOGS. A search bar is on the right. Below the navigation bar, a status bar displays various network metrics: CURRENT BLOCK (376), GAS PRICE (2000000000), GAS LIMIT (6721975), HARDFORK (MERGE), NETWORK ID (5777), RPC SERVER (HTTP://127.0.0.1:8545), MINING STATUS (AUTOMINING), and WORKSPACE (CALCULADORAT). There are buttons for SWITCH and a settings icon.

Two transaction entries are visible:

TX HASH	FROM ADDRESS	TO CONTRACT ADDRESS	GAS USED	VALUE
0xc0f6f4ff89aff8e86a33ff54017207d4bd5984d3d988a20dd9f73f40ce4b6a68	0xe87068450527017f5c233aAA83cb9f5bA21bb6C0	0x45722302DE66602f76f7ba2063Fca75405D60E69	49049	0
0x4c242c97baa38a868c4fbae95421865b70224a67644ccdca21d54367a012f043	0x4441c196a7621328A9F012046dE265366555C142	0x45722302DE66602f76f7ba2063Fca75405D60E69	72442	0

Figura 31: transacciones al votar Ganache

Capítulo 7. CONCLUSIONES Y TRABAJOS FUTUROS

En este capítulo se analizará si se hay cumplido los objetivos descritos en el Capítulo 4 para este Trabajo de Fin de Grado. Y se explicarán futuras mejoras a esta implementación.

- El primer objetivo es crear un sistema de votación que aumente el número de participantes en las votaciones evitando las largas colas mediante un voto en remoto y dando un acceso más cómodo al voto a personas con discapacidades. Esto se ha logrado ya que la aplicación permite el acceso desde cualquier dispositivo personal, sin necesidad de ir a un colegio electoral.
- El segundo es evitar el fraude electoral mediante un sistema electoral cuyos resultados son inmutables. Todas las transacciones de escritura en el contrato quedan registradas en un bloque de la cadena. Todos los nodos de la red tienen acceso a la cadena de bloques, y por tanto nadie puede alterarlos, y todo el mundo puede verlos.
- El tercero es crear un sistema que sea menos costoso y más responsable con el medio ambiente, eliminando todo el papeleo o la fabricación de maquinaria electoral. Esto se cumple, ya que los recursos que empleamos son dispositivos que ya existían, y el almacenamiento de los datos e información está en los dispositivos de la red.
- El cuarto es garantizar el voto secreto, este objetivo es el más complicado de todos. De cara al resto de votantes, el voto de cada votante es secreto, puesto que los otros votantes no conocen tu dirección de usuario de la blockchain. No obstante, al tener que hacer una autenticación, el administrador si lo conoce, para que este administrador, no pueda identificar de quien es el voto, se registran todos los votos de la red bajo la misma dirección. No obstante, esta implementación puede encontrarse con un fallo, ese fallo es poder identificar al votante por el orden de los bloques. Hay algunas técnicas y pruebas matemáticas que permiten garantizar este anonimato, pero se escapan de los límites de este proyecto.

Durante el desarrollo de este Trabajo de Fin de Grado se puede apreciar como la tecnología blockchain, es una herramienta que nos permite encontrar soluciones a problemas que hace unos años parecían que no tenían una solución.

En el ámbito de las votaciones, se ha conseguido crear un sistema que aún suendo transparente, permite el voto secreto. Una solución que podría mejorar significativamente la democracia en muchos países.

No obstante, este proyecto cuenta con limitaciones de tiempo y presupuesto. Pero para que este sistema se pueda implementar a escala nacional, se deben hacer algunas mejoras. Los siguientes pasos serían:

- Modificar el orden en el que se ordenan los bloques de la cadena para no poder identificar cual era la transacción anterior y así no asociar el registro con el votante.
- Escalar el proyecto a los datos de todos los ciudadanos y garantizar la concurrencia de múltiples hilos
- Juntar los datos de distintas circunscripciones y poder modificar el número de diputados a los que debemos asignar escaños.

Capítulo 8. BIBLIOGRAFÍA

- [1] S. Nakamoto (2008). *Bitcoin: A peer-to-peer electronic cash system*.
<https://bitcoin.org/bitcoin.pdf>
- [2] Tien Tuan Anh Dinh, Rui Liu, Meihui Zhang, Member, IEEE, Gang Chen, Member, IEEE, Beng Chin Ooi, Fellow, IEEE, and Ji Wang (17 de Agosto de 2017). *Untangling Blockchain: A Data Processing View of Blockchain Systems*.
<https://arxiv.org/pdf/1708.05665v1.pdf>
- [3] Varios autores (junio de 2023). *Ethereum blockchain app platform*.
<https://www.ethereum.org/>
- [4] Jonathan Waldman (marzo de 2018). *Cadena de bloques: aspectos básicos de la cadena de bloques*. <https://learn.microsoft.com/es-es/archive/msdn-magazine/2018/march/blockchain-blockchain-fundamentals>
- [5] Jake Frankenfield (27 de mayo de 2023). *What Is Proof of Work (PoW) in Blockchain?*
<https://www.investopedia.com/terms/p/proof-work.asp>
- [6] IBM (junio de 2023). *¿Qué son los contratos inteligentes en blockchain?*
<https://www.ibm.com/es-es/topics/smart-contracts>
- [7] Solidity (junio de 2023) *Solidity*. <https://docs.soliditylang.org/en/latest/>
- [8] Kaidong Wu, Yun Ma, Gang Huang, Xuanzhe Liu (3 de septiembre de 2019). *A First Look at Blockchain-based Decentralized Applications*. <https://arxiv.org/pdf/1909.00939.pdf>
- [9] VMware Tanzu (junio de 2023). *Spring Boot*. <https://spring.io/projects/spring-boot>
- [10] JetBrains (junio de 2023). *IntelliJ IDEA*. <https://www.jetbrains.com/idea/>
- [11] pgAdmin (junio de 2023) *PostgreSQL Tools* <https://www.pgadmin.org/>
- [12] Postman (junio de 2023) *What is postman?* <https://www.postman.com/product/what-is-postman/>
- [13] Visual Studio Code (junio de 2023). *Visual Studio Code*. <https://code.visualstudio.com/>
- [14] Metamask (junio de 2023) *Metamask*. <https://metamask.io/>
- [15] Ministerio del Interior (junio de 2023). *Método D'Hont*.
<https://infoelectoral.interior.gob.es/opencms/es/proceso-electoral/visitas-virtuales/metodo-dhont/>

- [16] Ministerio de Asuntos Exteriores y de Cooperación, Carlos Vegas González (2014) *Manual práctico para observadores electorales de corta duración, Capítulo XXI. Observación del voto electrónico*.
<https://www.exteriores.gob.es/es/ServiciosAlCiudadano/PublicacionesOficiales/Manual%20pr%C3%A1ctico%20para%20observadores%20electorales%20de%20corta%20duraci%C3%B3n.pdf>
- [17] Syada Tasmia Alvi , Mohammed Nasir Uddin , Linta Islam , Sajib Ahamed (October 2022) *A blockchain-based decentralized mechanism to ensure the security of digital voting system voting system*.
<https://www.sciencedirect.com/science/article/pii/S1319157822002221>
- [18] Gaby Dagher, Praneeth Marella, Matea Milojkovic, and Jordan Mohler. (enero de 2018) *Broncovote: Secure voting system using ethereum's blockchain. páginas 96–107*.
- [19] F. p. Hjálmarsson, G.K. Hreiðarsson, M. Hamdaqa, and G. Hjálmtýsson. Blockchainbased e-voting system. In 2018 IEEE 11th International Conference on Cloud Computing (CLOUD), pages 983–986, 2018.
- [20] José Luís Pereira Jorge Lopes. Blockchain based e-voting system: A proposal. In Twenty-fifth Americas Conference on Information Systems, Cancun, 2019, 2019
- [21] B. Shahzad and J. Crowcroft, "Trustworthy Electronic Voting Using Adjusted Blockchain Technology," in IEEE Access, vol. 7, pp. 24477-24488, 2019, doi: 10.1109/ACCESS.2019.2895670.
- [22] S. T. Alvi, M. N. Uddin and L. Islam, "Digital Voting: A Blockchain-based E-Voting System using Biohash and Smart Contract," 2020 Third International Conference on Smart Systems and Inventive Technology (ICSSIT), Tirunelveli, India, 2020, pp. 228-233, doi: 10.1109/ICSSIT48917.2020.9214250.
- [23] Mohammed Nasir Uddin, Sadman Ahmmed, Imtiaz Ahmed Riton, and Linta Islam. An blockchain-based e-voting system applying time lock encryption. In 2021 International Conference on Intelligent Technologies (CONIT), pages 1–6, 2021.
- [24] E. Zaghoul, T. Li and J. Ren, "d-BAME: Distributed Blockchain-Based Anonymous Mobile Electronic Voting," in IEEE Internet of Things Journal, vol. 8, no. 22, pp. 16585-16597, 15 Nov.15, 2021, doi: 10.1109/JIOT.2021.3074877.
- [25] Indeed (23 de junio de 2023). *¿Cuánto se gana como uno Desarrollador/a de software en España?* <https://es.indeed.com/career/desarrollador-de-software/salaries>

- [26] Zask (junio 2023), *¿Cuánto cuesta un servicio de programador?*, <https://www.zaask.es/cuanto-cuesta/programador-php>
- [27] Cristian Gallegos (28 de mayo de 2023). *¿Cuánto les cuesta a los españoles celebrar unas elecciones autonómicas y municipales?* <https://www.eleconomista.es/actualidad/noticias/12294868/05/23/cuanto-les-cuesta-a-los-espanoles-celebrar-unas-elecciones-autonomicas-y-municipales-.html>
- [28] Ministerio del Interior (30 de junio de 2023) *Registro de partidos políticos* <https://infoelectoral.interior.gob.es/opencms/es/partidos-politicos/registro-de-partidos-politicos/registro-de-partidos-politicos/>
- [29] Javier Sáez Hurtado (25 de febrero de 2022). *Que son las DApps o Aplicaciones Descentralizadas y varios ejemplos.* <https://www.iebschool.com/blog/dapps-o-aplicaciones-descentralizadas-que-son-y-como-funcionan-finanzas/>
- [30] Junta electoral Central (junio de 2023). *Ley Orgánica 5/1985, de 19 de junio, del Régimen Electoral General.* http://www.juntaelectoralcentral.es/cs/jec/loreg/contenido?idContenido=547335&idLeyJunta=1&idLeyModificacion=546161&p=1379061423059&paux=1379061423059&template=Loreg/JEC_Contentido
- [31] Naciones Unidas (junio de 2023) *Objetivos de desarrollo sostenible.* <https://www.un.org/sustainabledevelopment/es/objetivos-de-desarrollo-sostenible/>

ANEXO I: ALINEACIÓN DEL PROYECTO CON LOS ODS

Los ODS⁸ son unos objetivos mundiales, aprobados en 2015 por las naciones Unidas, con el objetivo de terminar con la pobreza, garantizar la paz y proteger el planeta en 2030. Los objetivos son todos los que se pueden observar en la siguiente figura[31].



Producido en colaboración con TROLLBACK COMPANY | TheGlobalGoals@trollback.com | +1.212.529.1010
Para cualquier duda sobre la utilización, por favor consulte con: dpcampa@un.org

Figura 32: Objetivos de desarrollo sostenible de las Naciones Unidas [31]

⁸ Objetivos de Desarrollo Sostenible

En este proyecto se cumplen tres de ellos:

- **El número 3: SALUD Y BIENESTAR:** Desde el inicio de la pandemia son muchos los que aún no se atreven a estar entre grandes multitudes o en espacios públicos donde el tránsito de personas es elevado. Las elecciones realizadas de la forma tradicional pueden poner en riesgo la salud de los grupos más vulnerables. Por eso, creo que el bienestar de las personas se mejora con esta medida al no tener que exponerse a los riesgos de estar en una sala llena de personas. Se reduce el riesgo de contagio.
- **El número 9: INDUSTRIA, INNOVACIÓN E INFRAESTRUCTURAS:** Debido al COVID-19 y la actual guerra de Ucrania, se han provocado alteraciones en el suministro de productos a nivel mundial. Esto ha llevado a la necesidad de descubrir soluciones duraderas para los desafíos económicos y medioambientales. Gracias a este proyecto conseguimos un uso más eficiente de los recursos, utilizando los dispositivos de los usuarios.
- **El número 12 Y/O 15: PRODUCCIÓN Y CONSUMO RESPONSABLES-VIDA DE ECOSISTEMAS TERRESTRES:** llevamos más de un siglo con un progreso económico que ha causado una degradación del medio ambiente, por ello se necesitan actividades que reviertan las tendencias actuales. Por ello este proyecto es un claro ejemplo de cómo hacer más y mejor con menos. Con esta tecnología reducimos el gasto de papel y en consecuencia disminuyen las emisiones de carbono. Asimismo, es una forma de detener la degradación del ecosistema forestal.
- **El número 16: PAZ, JUSTICIA E INSTITUCIONES SÓLIDAS.** El objetivo es alcanzar sociedades justas e inclusivas, con instituciones transparentes y fiables. La tecnología Blockchain ayuda a acabar con el fraude electoral criticado en los sistemas actuales.

ANEXO II

En esta sección se explicará como instalar he instanciar nuestro desarrollo, además de adjuntar el *smart contract*. Las herramientas que se deben emplear para que todas las partes funcionen son: ganache, remix y metamask.

A. II 1 SMART CONTRACT

```
pragma solidity >=0.7.0 <0.9.0;
pragma abicoder v2;

/**
 * @title Ballot
 * @dev Implements voting process along with vote delegation
 */
contract Ballot {

    struct Voter {
        uint weight; // weight is accumulated by delegation
        bool voted; // if true, that person already voted
        uint vote; // index of the voted proposal
    }

    struct Candidate {
        string name; // candidate name
        uint voteCount; // number of accumulated votes
    }

    address public chairperson;

    mapping(address => Voter) public voters;

    Candidate[] public candidates;

    enum State { Created, Voting, Ended } // State of voting period

    State public state;

    constructor() {
        chairperson = msg.sender;
        voters[chairperson].weight = 1;
        state = State.Created;
    }
}
```



```
// MODIFIERS
modifier onlySmartContractOwner() {
    require(
        msg.sender == chairperson,
        "Only chairperson can start and end the voting"
    );
    _;
}

modifier CreatedState() {
    require(state == State.Created, "it must be in Started");
    _;
}

modifier VotingState() {
    require(state == State.Voting, "it must be in Voting Period");
    _;
}

modifier EndedState() {
    require(state == State.Ended, "it must be in Ended Period");
    _;
}

function addCandidates(string memory candidateName)
    public
    CreatedState
{
    state = State.Created;
    candidates.push(Candidate({
        name: candidateName,
        voteCount: 0
    }));
}

// to start the voting period
// onlySmartContractOwner
function startVote()
    public
    CreatedState
{
    state = State.Voting;
    voters[chairperson].weight = 0;
}

/*
 * to end the voting period
 * can only end if the state in Voting period
 */
function endVote()
    public
    onlySmartContractOwner
    VotingState
```

```

{
    state = State.Ended;
}

/**
 * @dev Give 'voter' the right to vote on this ballot. May only be called by
 'chairperson'.
 * @param voter address of voter
 */
function giveRightToVote(address voter) public {
    require(
        msg.sender == chairperson,
        "Only chairperson can give right to vote."
    );
    require(
        !voters[voter].voted,
        "The voter already voted."
    );
    voters[voter].weight = 1;
}

/**
 * @dev Give your vote (including votes delegated to you) to candidate
 'candidates[candidate].name'.
 */
function vote()
    public
    VotingState
{
    Voter storage sender = voters[msg.sender];
    require(sender.weight == 1, "Has no right to vote");
    require(!sender.voted, "Already voted.");
    sender.voted = true;

    voters[chairperson].weight = 1;
}

/**
 * @param candidate index of candidate in the candidates array
 */

function assignVote(uint candidate)
    public
    onlySmartContractOwner
{
    candidates[candidate].voteCount += voters[chairperson].weight;
    voters[chairperson].weight = 0;
}

function winningCandidate()
    public

```

```
    EndedState
    view
    returns (string memory winnerName_)
{
    uint winningVoteCount = 0;

    for (uint p = 0; p < candidates.length; p++) {
        if (candidates[p].voteCount > winningVoteCount) {
            winningVoteCount = candidates[p].voteCount;
            winnerName_ = candidates[p].name;
        }
    }
}

function showCandidate()
    public
    view
    returns (Candidate [] memory candidates_)
{
    candidates_ = candidates;
}

function resetVote()
    public
    onlySmartContractOwner
    EndedState
{
    state = State.Created;

    uint length = candidates.length;
    for (uint i = 0; i < length; i++) {
        candidates.pop();
    }
}
}
```

A. II 2 GANACHE

El primer paso es instalar la herramienta de Ganache en nuestro sistema. Una vez descargada, la abrimos y creamos un nuevo espacio de trabajo, tal y como se observa en la siguiente imagen.

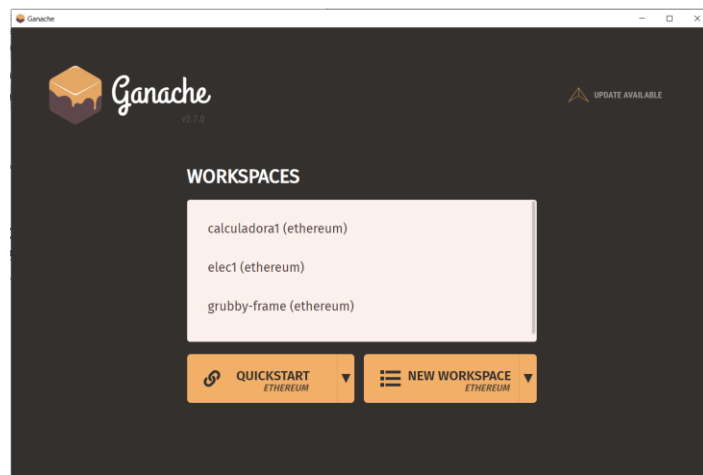


Figura 33: configuración de Ganache parte 1

A continuación, en *workspace* definimos un nombre y en *server* configuramos un *hostname*, en este caso es la dirección 127.0.0.1 al ejecutarse en local, y un puerto. El puerto que he empleado es el 8454, pero se pueden usar otros si ese lo tenemos ya ocupado.

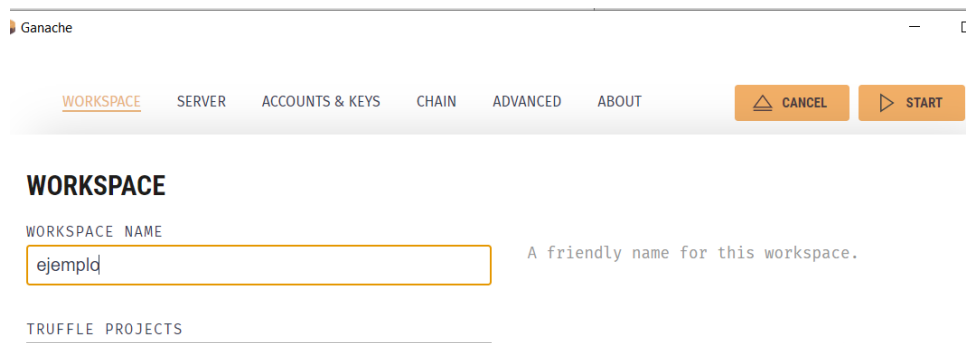


Figura 34: configuración de Ganache parte 2

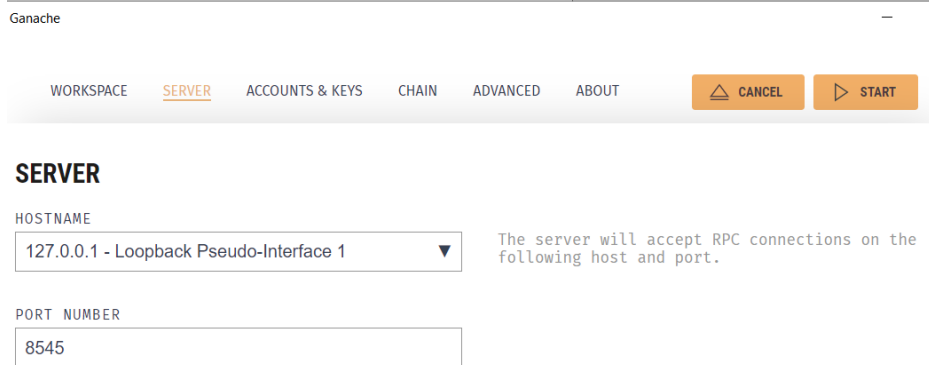


Figura 35: configuración de Ganache parte 3

Una vez establecidos estos parámetros, le damos al botón de *start* que se encuentra arriba a la derecha. Y deberíamos observar la siguiente pantalla, desde la cual podremos ver todas las cuentas y su balance.

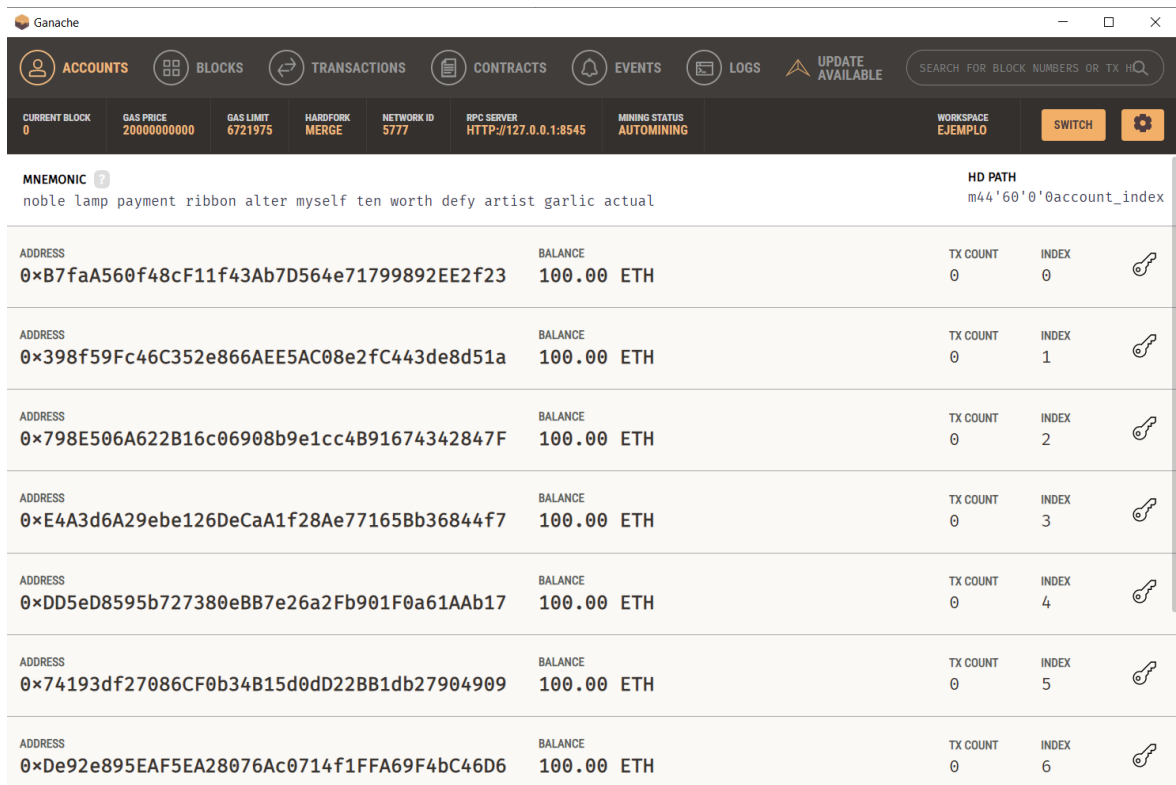


Figura 36: configuración de Ganache parte 4

A. II 3 REMIX

El primer paso es descargar la extensión de Remix para Visual Code, aunque también hay una opción web si se prefiere o no se puede instalar el *plugin*. Podemos comprobar que lo tenemos si nos sale el siguiente signo en el menú lateral izquierdo.

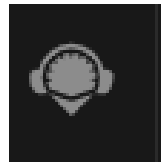


Figura 37: Remix plugin

Para poder desplegar un contrato debemos tenerlo abierto en una ventana de Visual Code, pinchar en *Run & Deploy* y seleccionar *activate*. Y se nos debería abrir una ventana con una interfaz gráfica para interactuar con el contrato.

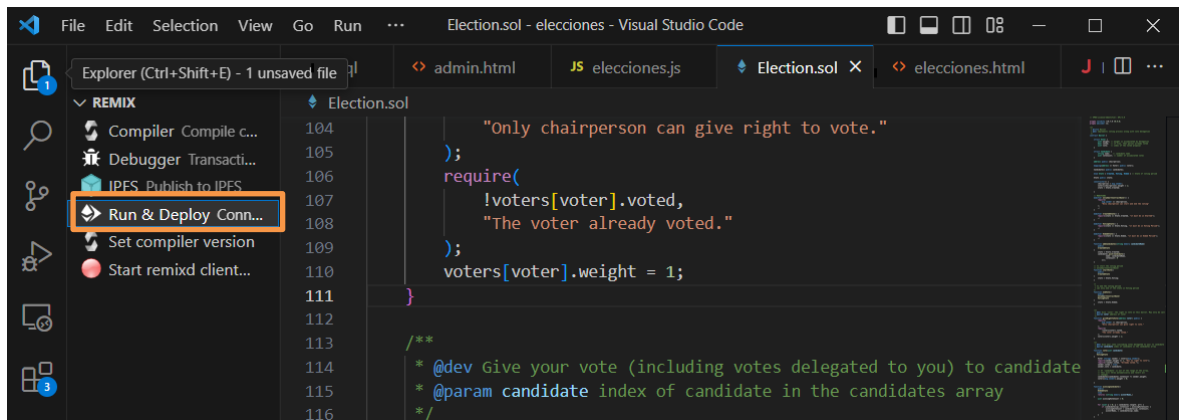


Figura 38: Desplegar contrato con Remix

Para poder interactuar, debemos conectarnos a una red de blockchain, que en nuestro sistema es la red de Ganache. Para hacer la conexión, la aplicación debe estar abierta con la sesión que se ha definido, y se debería observar la pantalla de la última figura del apéndice de Ganache. El siguiente paso es poner la URL del equipo y puerto que se ha definido en nuestra red de Ganache, y pinchar en conectar.

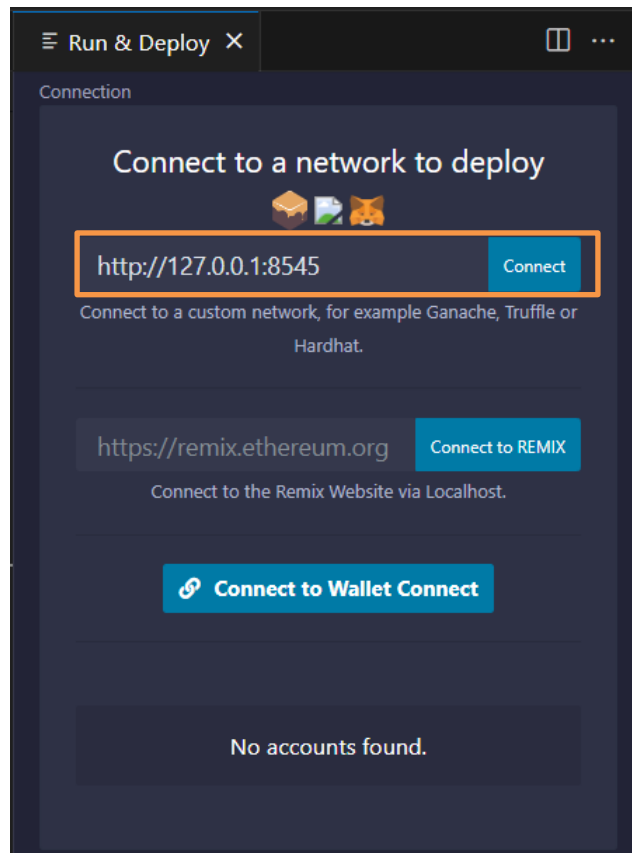


Figura 39: Interfaz de Remix 1

El siguiente paso consiste en compilar el contrato y desplegarlo. Primero pinchamos en *compile*, y luego en *Deploy*. Al pinchar en *compile* obtenemos el ABI, y al pinchar en *Deploy*, obtenemos la dirección del contrato desplegado. Ambas variables son las que debemos incorporar en JavaScript para interactuar con el contrato.

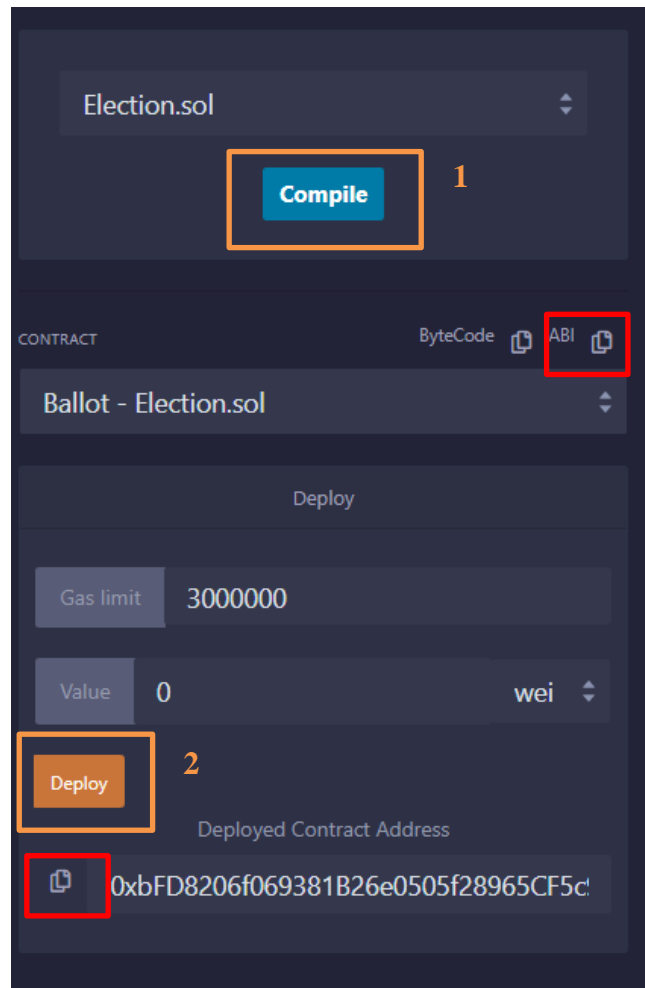


Figura 40: Interfaz Remix 2

Por último, tenemos la opción de ejecutar las funciones del contrato desde esta pantalla, lo cual es muy útil para hacer pruebas. Encontramos todos los métodos que hemos definido en nuestro *Smart contract*. Y emite mensajes de error en caso de que los valores introducidos sean incorrectos o la condición del *modifier* no se cumpla. Asimismo, permite pasarles valores a los métodos si estos los requieren.

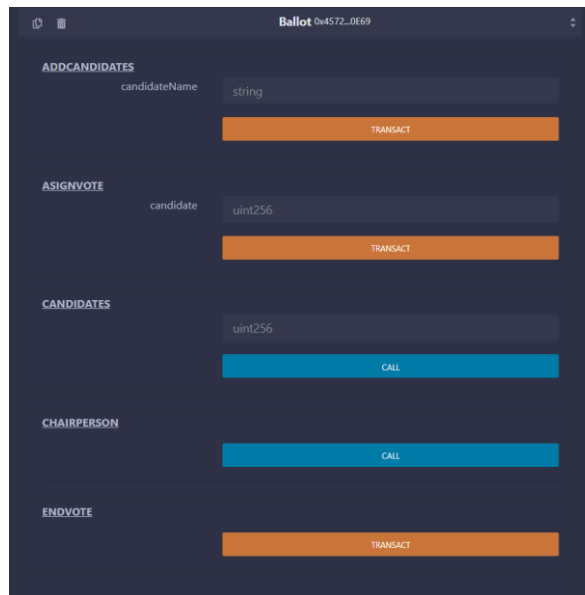


Figura 41: Interfaz Remix 3

Se puede comprobar que los métodos han sido ejecutados en el output de la consola de Remix IDE. Miremos más en detalle el siguiente *log*. Primero añade un candidato, desde la cuenta del administrador a la dirección del contrato, y podemos ver el gas utilizado en esta transacción, su hash y la dirección de envío y la del contrato.

```
[10:18:20 AM]: Send data to method 'addCandidates' with ["\pp\"] from
0xe87068450527017f5c233aAA83cb9f5bA21bb6C0 at contract address
0x45722302DE66602f76f7ba2063FcA75405D60E69
[10:18:20 AM]: TRANSACTIONHASH :
[10:18:20 AM]:
"0x80e2b34a543c88d7090f8cb7dbc37ea40e7f33a182f2f674a961bf9f4bf9462f"
[10:18:20 AM]: BLOCKHASH :
[10:18:20 AM]:
"0x51f7a1828a22c4b20bc032993645a0abb2f524c2ca6728489046ff1a8f7644db"
[10:18:20 AM]: FROM :
[10:18:20 AM]: "0xe87068450527017f5c233aaa83cb9f5ba21bb6c0"
[10:18:20 AM]: TO :
[10:18:20 AM]: "0x45722302de66602f76f7ba2063fca75405d60e69"
[10:18:20 AM]: CUMULATIVEGASUSED :
[10:18:20 AM]: 72106
[10:18:20 AM]: GASUSED :
```

```
[10:18:20 AM]: 72106
[10:18:20 AM]: EFFECTIVEGASPRICE :
[10:18:20 AM]: "0x4a817c800"
```

El siguiente paso es empezar la votación, en la que se asigna el valor 1 o *voting* a la variable *state*

```
[10:18:49 AM]: Network is a local or custom network!
[10:18:49 AM]: Send data to method 'startVote' with [] from
0xe87068450527017f5c233aAA83cb9f5bA21bb6C0 at contract address
0x45722302DE66602f76f7ba2063FcA75405D60E69
[10:18:49 AM]: TRANSACTIONHASH :
[10:18:49 AM]:
"0x201781efe6908c9d8a072b5fc5474f59b7bf0ed0cd2b7e91b5c369cff77c0223"
[10:18:49 AM]: BLOCKHASH :
[10:18:49 AM]:
"0x7159a88309b2ac3b24d521a607783f422ceed913274ea0b2bb27f1bd57e12e22"
[10:18:49 AM]: FROM :
[10:18:49 AM]: "0xe87068450527017f5c233aaa83cb9f5ba21bb6c0"
[10:18:49 AM]: TO :
[10:18:49 AM]: "0x45722302de66602f76f7ba2063fca75405d60e69"
[10:18:49 AM]: CUMULATIVEGASUSED :
[10:18:49 AM]: 48119
[10:18:49 AM]: GASUSED :
[10:18:49 AM]: 48119
[10:18:49 AM]: EFFECTIVEGASPRICE :
[10:18:49 AM]: "0x4a817c800"
[10:18:53 AM]: Network is a local or custom network!
```

Ahora queremos conocer el valor de la variable *state*, y como es un método de solo lectura, no añade un bloque a la cadena, ni gasta gas. Lo mismo ocurre con *chairperson* y con *showCandidate*, que solo realizan una lectura de variables del contrato.

```
[10:18:53 AM]: Calling method 'state' with [] from
0xe87068450527017f5c233aAA83cb9f5bA21bb6C0 at contract address
0x45722302DE66602f76f7ba2063FcA75405D60E69
[10:18:53 AM]: "1"
```

```
[10:18:59 AM]: Network is a local or custom network!  
[10:18:59 AM]: Calling method 'chairperson' with [] from  
0xe87068450527017f5c233aAA83cb9f5bA21bb6C0 at contract address  
0x45722302DE66602f76f7ba2063FcA75405D60E69  
[10:18:59 AM]: "0xe87068450527017f5c233aAA83cb9f5bA21bb6C0"  
[10:19:02 AM]: Network is a local or custom network!  
[10:19:02 AM]: Calling method 'showCandidate' with [] from  
0xe87068450527017f5c233aAA83cb9f5bA21bb6C0 at contract address  
0x45722302DE66602f76f7ba2063FcA75405D60E69  
[10:19:02 AM]: [{"\pp\"", "0"}]  
[10:19:35 AM]: Network is a local or custom network!
```

Por ultimo, llamamos a la función que finaliza la votación, y comprobamos que se haya realizado correctamente, llamando a *state* y comprobando que el valor es 2.

```
[10:19:35 AM]: Send data to method 'endVote' with [] from  
0xe87068450527017f5c233aAA83cb9f5bA21bb6C0 at contract address  
0x45722302DE66602f76f7ba2063FcA75405D60E69  
[10:19:35 AM]: TRANSACTIONHASH :  
[10:19:35 AM]:  
"0x6973d0fe24b63e9037a090193ded4e3dda66bc8fddb02f0f9cb70ec9cc1d70ad"  
[10:19:35 AM]: BLOCKHASH :  
[10:19:35 AM]:  
"0x2cb6f77e27add039c28f4ae3879ed9071b594daafb69ef228acb51a5112be776"  
[10:19:35 AM]: FROM :  
[10:19:35 AM]: "0xe87068450527017f5c233aaa83cb9f5ba21bb6c0"  
[10:19:35 AM]: TO :  
[10:19:35 AM]: "0x45722302de66602f76f7ba2063fca75405d60e69"  
[10:19:35 AM]: CUMULATIVEGASUSED :  
[10:19:35 AM]: 28740  
[10:19:35 AM]: GASUSED :  
[10:19:35 AM]: 28740  
[10:19:35 AM]: EFFECTIVEGASPRICE :  
[10:19:35 AM]: "0x4a817c800"  
[10:19:39 AM]: Network is a local or custom network!  
  
[10:19:39 AM]: Calling method 'state' with [] from  
0xe87068450527017f5c233aAA83cb9f5bA21bb6C0 at contract address  
0x45722302DE66602f76f7ba2063FcA75405D60E69  
[10:19:39 AM]: "2"
```

A. II 4 METAMASK

Cada usuario debe tener instalado en su dispositivo la extensión de *metamask* en su navegador, es una extensión muy flexible, que contiene hasta una opción para móviles. Metamask es una cartera de activos, en este caso, se emplea para conectarse a la red de blockchain con una cuenta, la cual se asocia introduciendo una clave privada.



Figura 42: extensión de metamask

Se nos debería abrir la siguiente pestaña al intentar conectarnos a una cuenta de blockchain con metamask en el javascript. En la siguiente sesión, iniciamos sesión, con nuestro usuario de metamask.

MetaMask Notification



¡Bienvenido de nuevo!

La Web descentralizada espera

Contraseña

Desbloquear

[¿Olvidó su contraseña?](#)

[¿Necesita ayuda? Comuníquese con Soporte de MetaMask](#)

Figura 43: Inicio de sesión metamask

El siguiente y último paso es conectar la cuenta de la blockchain. Para ello debemos conocer la clave privada. En este proyecto, se han empleado las proporcionadas por Ganche.

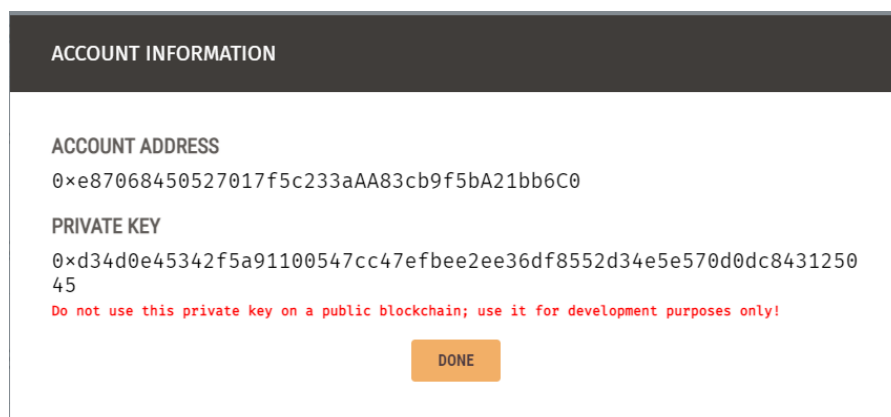


Figura 44: keys de cuenta de Ganache

Ahora la pinchamos arriba a la derecha, ahí podemos seleccionar la cuenta o añadir una. Para añadir una cuenta, pinchamos en importar cuenta y seleccionamos la opción de clave privada en la siguiente pestaña. Arriba a la derecha podemos ver las cuentas importadas.

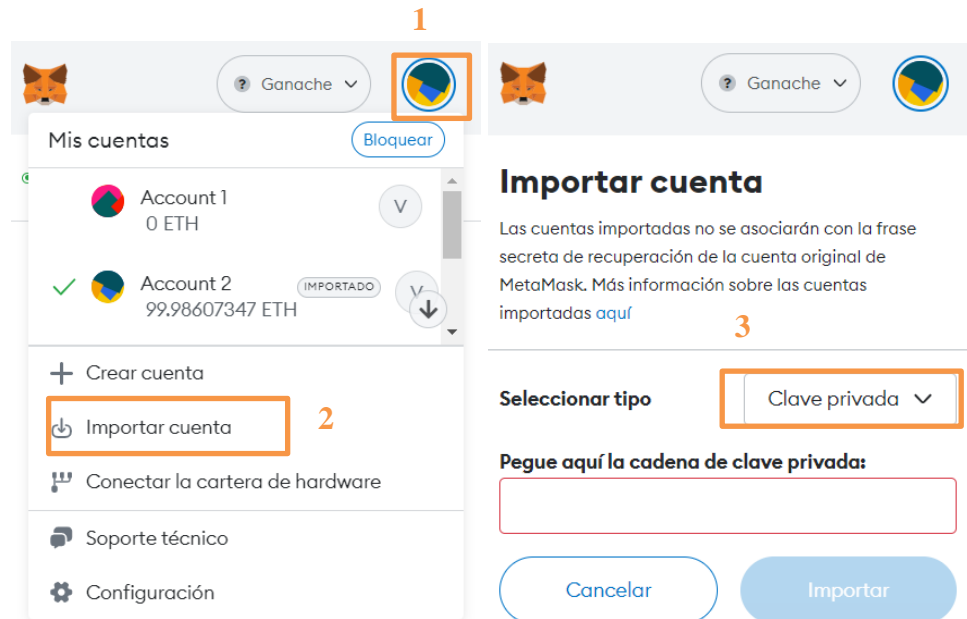


Figura 45: gestión de cuentas de metamask