Desarrollo de un sistema de votación electrónico para las elecciones universitarias
Development of an electronic voting system for university elections

Trabajo de Fin de Grado
Curso 2024–2025

Autores
CARLOS VIVERO BARRERA. Nota 7.5
DAVID DEL CAMPO PEÑUELA. Nota 7.5
DAVID HERRANZ TORIBIO. Nota 7.5

Directores
MARÍA ELENA GÓMEZ MARTÍNEZ
JOSÉ IGNACIO REQUENO JARABO

Grado en Ingeniería Informática
Facultad de Informática
Universidad Complutense de Madrid

Desarrollo de un sistema de votación electrónico para las elecciones universitarias
Development of an electronic voting system for university elections

Trabajo de Fin de Grado en Ingeniería Informática

Autores
CARLOS VIVERO BARRERA. Nota 7.5
DAVID DEL CAMPO PEÑUELA. Nota 7.5
DAVID HERRANZ TORIBIO. Nota 7.5

Directores
MARÍA ELENA GÓMEZ MARTÍNEZ
JOSÉ IGNACIO REQUENO JARABO

Convocatoria: *Junio 2025*

Grado en Ingeniería Informática
Facultad de Informática
Universidad Complutense de Madrid

16 de junio de 2025

Dedicatoria

Dedicado a toda aquella persona que no haya tenido acceso a la educación. Que este trabajo sirva como un pequeño homenaje a su dignidad, su capacidad y su valor.

El conocimiento debe ser un derecho, no un privilegio.

Agradecimientos

Primero de todo, agradecer y felicitar a nuestros tutores, Elena y José Ignacio, por la dedicación, interés y ayuda para el desarrollo de este Trabajo de Fin de Grado.

En segundo lugar, a nuestros compañeros de universidad, amigos y familiares que nos han apoyado durante estos meses. Gracias por su apoyo constante, sus palabras de aliento, su interés en el buen devenir del trabajo y por estar presentes en los momentos difíciles y también en los logros. Con ese granito de arena aportado por cada uno, el camino ha sido mucho más llevadero.

Asimismo, agradecer a los tres integrantes del grupo por su dedicación, esfuerzo y trabajo para que el trabajo haya seguido su planificación y se hayan conseguido los objetivos establecidos a principio de curso.

En definitiva, muchas gracias a todas las personas que han participado en este proyecto, sin todos ellos este proyecto no se hubiese completado.

Resumen

Desarrollo de un sistema de votación electrónico para las elecciones universitarias

El principal objetivo del trabajo ha sido establecer un sistema que permita realizar elecciones de delegados de una manera eficiente, segura y transparente. Como resultado de este trabajo, se ha obtenido una aplicación web que permite realizar las funcionalidades esenciales para ello, pero que permite añadir futuras mejoras.

En primer lugar, se definieron los objetivos y el alcance del sistema, investigando soluciones existentes y las mejores opciones tecnológicas, las cuales fueron Vue.js, Node.js y MariaDB. Tras ello, se diseñó el modelo general y se han aplicado los conceptos y metodologías de desarrollo software aprendidos durante la carrera, en particular, las metodologías ágiles. La aplicación permite gestionar Usuarios, Candidaturas y Elecciones.

Finalmente, con las tareas y planificación, se dividió el desarrollo en distintas partes para desarrollar el sistema. Las fases del proyecto fueron análisis, diseño, implementación y pruebas unitarias.

Palabras clave

Aplicación Web, Autenticación de Usuarios, Docker (1), Encriptación de Datos, Gestión de Proyecto, JavaScript (2), Transparencia electoral, Votación electrónica.

Abstract

Development of an electronic voting system for university elections

The main objective of the work has been to establish a system that allows delegate elections to be carried out in an efficient, secure and transparent manner. As a result of this work, a web application has been obtained that allows the essential functionalities for this to be carried out, but allows for future improvements to be added.

Firstly, the objectives and scope of the system were defined, investigating existing solutions and the best technological options, which were Vue.js, Node.js and MariaDB. After that, the general model was designed and the software development concepts and methodologies learnt during the degree were applied, in particular agile methodologies. The application allows to manage Users, Candidatures and Elections.

Finally, with the tasks and planning, the development was divided into different parts to develop the system. The phases of the project were analysis, design, implementation and unit testing.

Keywords

Web Application, User Authentication, Docker (1), Data Encryption, Project Management, JavaScript (2), Electoral Transparency, Electronic Voting.

Índice

Índice de figuras

Capítulo 1

Introducción

1.1. Motivación del Trabajo

Tal y como evidencia el crecimiento de la incertidumbre en el mundo en el que vivimos. Donde podemos cerciorarnos del considerable aumento de las dependencias entre los intereses económicos, sociales y políticos, a nivel mundial. Si añadimos a estos fenómenos, el aumento de la crispación mundial y la creciente desconfianza en los procesos tradicionales. Podemos llegar a la conclusión de que es cada vez más urgente la necesidad de establecer procesos democráticos que sean realmente **fiables** y operen de manera **segura**.

Con todo este contexto, nace la idea de nuestro proyecto. Este tiene como objetivo el desarrollo de un **sistema de votación electrónica** para las elecciones universitarias, específicamente para la **Universidad Complutense de Madrid**. Con esto conseguimos que no esté sujeto a interferencias de una fuente externa ajena a la universidad y que el sistema no tenga ningún tipo de interés derivado salvo el correcto funcionamiento de las votaciones.

Por tanto, este sistema pretende ser un modelo de **transparencia**, **seguridad** y **autonomía**, con el fin de garantizar que el proceso electoral se lleve a cabo de forma eficiente y sin alteraciones externas, asegurando que el voto de los estudiantes es autentico y los resultados son realmente los que se deciden democráticamente por una mayoría.

1.2. Propósitos del Trabajo de Fin de Grado

1.2.1. Propósito General

El objetivo general de este trabajo es el desarrollo de un sistema de votación electrónica para la celebración de elecciones universitarias de forma segura y trans-

parente.

1.2.2. Propósitos Específicos

A continuación se detallan otros propósitos importantes del proyecto:

- Asegurar la identidad de los votantes mediante autenticación confiable, permitiendo únicamente emitir su voto, a los estudiantes registrados y autorizados.
- Seleccionar el conjunto de tecnologías que se utilizarán para desarrollar el sistema web a través del estudio de las principales opciones existentes.
- Definir un sistema web de votaciones electrónicas que se corresponda a las necesidades mediante la realización de análisis a sistemas similares, la especificación de requisitos y la representación en maquetas de la aplicación.
- Desarrollar una interfaz intuitiva y amigable para que cualquier usuario no tenga ningún tipo de problema en votar de manera correcta.
- Elaborar un Manual de Instalación que permita a cualquier usuario que disponga del código fuente ejecutar el sistema bajo las requisitos especificados.
- Elaborar un Manual de Usuario como soporte para resolver dudas o inconvenientes de los profesores o alumnos.

1.3. Esquema y Planificación del Trabajo de Fin de Grado

1.3.1. Metodología de Desarrollo

En este proyecto, hemos adoptado la metodología ágil **Sprint** para gestionar el desarrollo de la votación electrónica. Las reuniones con nuestros tutores se realizaron con una frecuencia media de **dos a tres semanas**, siguiendo sus directrices para establecer un ritmo de tareas eficiente y asegurar el cumplimiento de plazos. En cada reunión, se establecían los objetivos y metas del siguiente ciclo de trabajo (sprint), lo que nos permitió ajustar el enfoque de manera constante y asegurarnos para que el desarrollo del proyecto avanzara sin contratiempos.

Cada sprint tuvo una duración de **dos semanas**, y al final de cada uno, se realizaba una reunión de **revisión** para evaluar los avances y recibir feedback en partes importantes del sistema, como la **autenticación de usuarios** o la **integración de la funcionalidad de votación**.

Al finalizar estas reuniones de planificación, los tres miembros del equipo concordábamos la organización de tareas de manera equitativa, de acuerdo con las prioridades definidas en cada sprint. Cada miembro asumía responsabilidades específicas, como el desarrollo de la **base de datos**, la **diseño de la interfaz de usuario**, o la implementación de la **seguridad del sistema**, lo que permitía avanzar de forma paralela en distintas áreas del proyecto.

Además, establecíamos **reuniones internas semanales**, para sincronizar el trabajo individual de cada uno, resolver problemas puntuales y mantenernos alineados con los objetivos del sprint. Este enfoque fomentó la **colaboración constante** entre los miembros del equipo y permitió ajustar rápidamente las tareas si surgían obstáculos imprevistos.

De esta forma, se garantizó que al final de cada sprint se entregara un **producto funcional** que pudiera ser probado, validado y, si era necesario, corregido antes del siguiente ciclo de trabajo.

En el caso de tener problemas con el plazo de alguna tarea del sprint, tras evaluarlo con los tutores, se incluía al siguiente sprint con una **prioridad mayor** y solo se aceptaba como finalizado tras el consentimiento de los mismos.

1.3.2. Planificación

Para la realización y seguimiento de las tareas hemos utilizado la herramienta JIRA Software. Ésta nos permite, después de haber establecido todo el progreso del proyecto mediante *sprints*, obtener el diagrama de Gantt (3) para visualizar las tareas y su cronograma en función del tiempo.

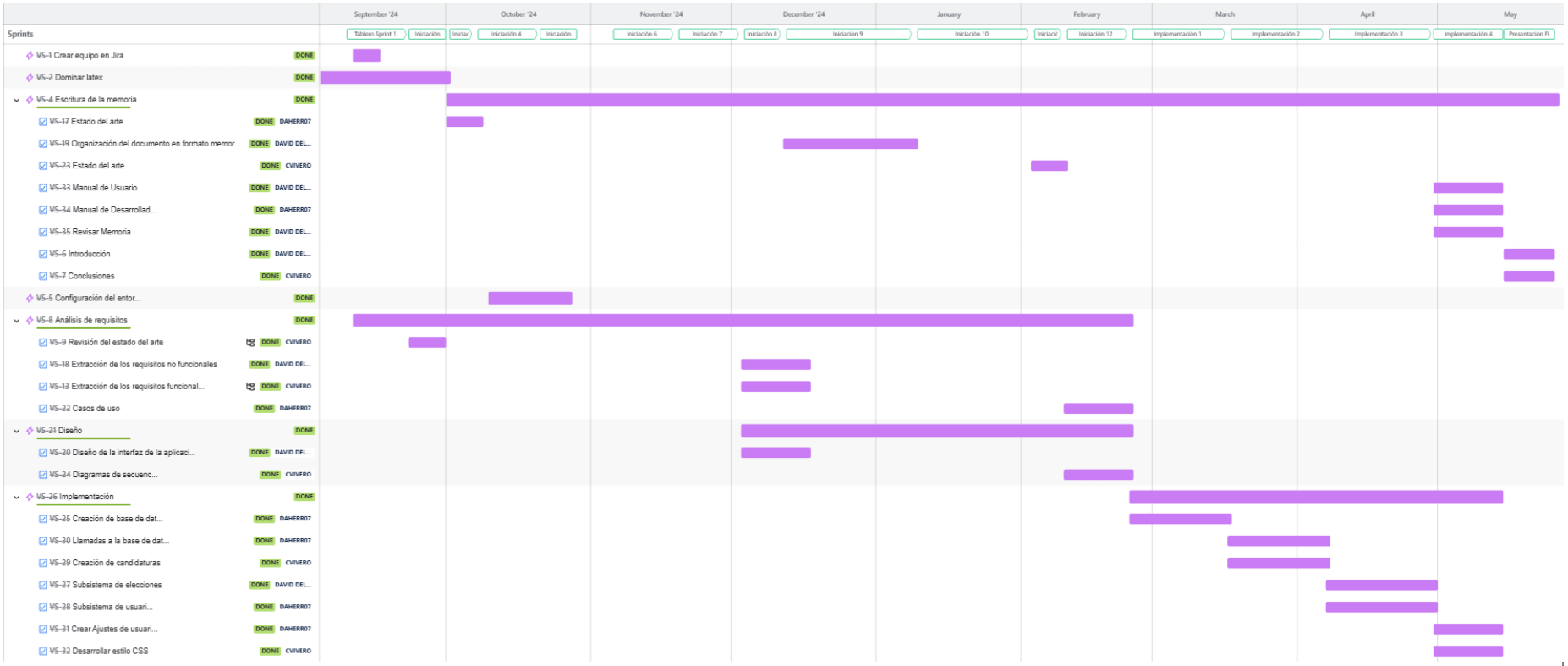


Figura 1.1: Diagrama de Gantt generado con JIRA Software.

1.4. Estructura del Trabajo

En este apartado se detalla la organización que hemos seguido durante el desarrollo del proyecto. Se incluyen la distribución de tareas y el diagrama de Gantt.

1.4.1. Organización de tareas

El sistema se compone como se analiza posteriormente en 3 subsistemas.

A pesar de haber trabajado conjuntamente como equipo en el desarrollo del proyecto e incluso apoyando en tareas de no asignadas. Respecto a la memoria los 3 hemos aportado en todos los apartados a excepción de los que se detallan, en la tarea "Escritura de la memoria de los apartados", a continuación. Las funciones principales por persona han sido las siguientes:

David Herranz Toribio:

- Diseño e implementación del Subsistema de Usuarios.
- Creación y mantenimiento de la Base de Datos.
- Documentación de Manual de Administrador, de Usuario y de instalación.
- Diagramas de Casos de Uso
- Escritura de la memoria de los apartados: Apéndices B y C.

Carlos Vivero Barrera:

- Diseño e implementación del Subsistema de Candidaturas.
- Creación y mantenimiento de repositorio de GitHub del proyecto.
- Diagramas de Clases.
- Escritura de la memoria de los apartados: 4 y 5.

David del Campo Peñuela:

- Diseño e implementación del Subsistema de Elecciones.
- Creación y organización del proyecto en JIRA.
- Diagramas de Secuencia.
- Diseño de Maquetas.
- Escritura de la memoria de los apartados: 1 y Apéndice A.

Chapter 2

Introduction

2.1. Work Motivation

As evidenced by the growing uncertainty in the world we live in. We can see the considerable increase in dependencies between economic, social and political interests at the global level. If we add to these phenomena the increase in global tensions and the growing distrust of traditional processes. We can conclude that there is an increasingly urgent need to establish democratic processes that are truly **reliable** and operate in a **secure** manner.

Against this background, the idea for our project was born. It aims to develop an **electronic voting system** for university elections, specifically for the **Complutense University of Madrid**. With this we achieve that it is not subject to interference from an external source outside the university and that the system does not have any type of derived interest except for the correct functioning of the voting.

Therefore, this system aims to be a model of **transparency**, **security** and **autonomy**, in order to guarantee that the electoral process is carried out efficiently and without external alterations, ensuring that the students' vote is authentic and that the results are really those decided democratically by a majority.

2.2. Purposes of the Bachelor's Thesis

2.2.1. General Purpose

The overall objective of this work is the development of an electronic voting system for holding university elections in a secure and transparent manner.

2.2.2. Specific Purposes

Other important purposes of the project are detailed below:

- Ensure the identity of voters through reliable authentication, allowing only registered and authorised students to cast their vote.
- Select the set of technologies to be used to develop the web-based system by studying the main existing options.
- Define a web-based e-voting system that corresponds to the needs by conducting analyses of similar systems, specifying requirements and representing them in mock-ups of the application.
- Develop an intuitive and user-friendly interface so that any user has no problem in voting correctly.
- To elaborate an Installation Manual that allows any user with the source code to execute the system under the specified requirements.
- To elaborate a User's Manual as a support to solve doubts or inconveniences of the teachers or students.

2.3. Outline and Planning of Bachelor's Thesis

2.3.1. Development Methodology

In this project, we adopted the agile **Sprint** methodology to manage the e-voting development. Meetings with our tutors were held with an average frequency of **two to three weeks**, following their guidelines to establish an efficient task rhythm and ensure deadlines were met. At each meeting, the objectives and goals of the next work cycle (sprint) were set, which allowed us to constantly adjust the focus and ensure that the project's development progressed smoothly. Each sprint lasted **two weeks**, and at the end of each sprint, a review meeting was held to evaluate progress and receive feedback on important parts of the system, such as **user authentication** or the **integration of voting functionality**.

At the end of these planning meetings, the three members of the team would concor- sequently organise the tasks in an equitable manner, according to the priorities defined in each sprint. Each member took on specific responsibilities, such as the development of the **database**, the **User interface (UI) design**, or the implementation of the **system's security**, which allowed us to advance in parallel in different areas of the project.

In addition, we set up **weekly internal meetings** to synchronise everyone's individual work, solve specific problems and keep us aligned with the sprint objec-

tives. This approach fostered **constant collaboration** among team members and allowed us to quickly adjust tasks if unforeseen obstacles arose.

This ensured that at the end of each sprint a **functional product** was delivered that could be tested, validated and, if necessary, corrected before the next work cycle. In the event of problems with the deadline of a sprint task, after assessed with the tutors, it was included in the next sprint with a **higher priority** and was only accepted as completed with their consent.

2.3.2. Tasks Organization

To develop and monitor the tasks we have used the JIRA Software tool. This allows us, after having established all the progress of the project by means of *sprints*, to obtain the Gantt diagram (3) to visualise the tasks and their chronogram in terms of time.

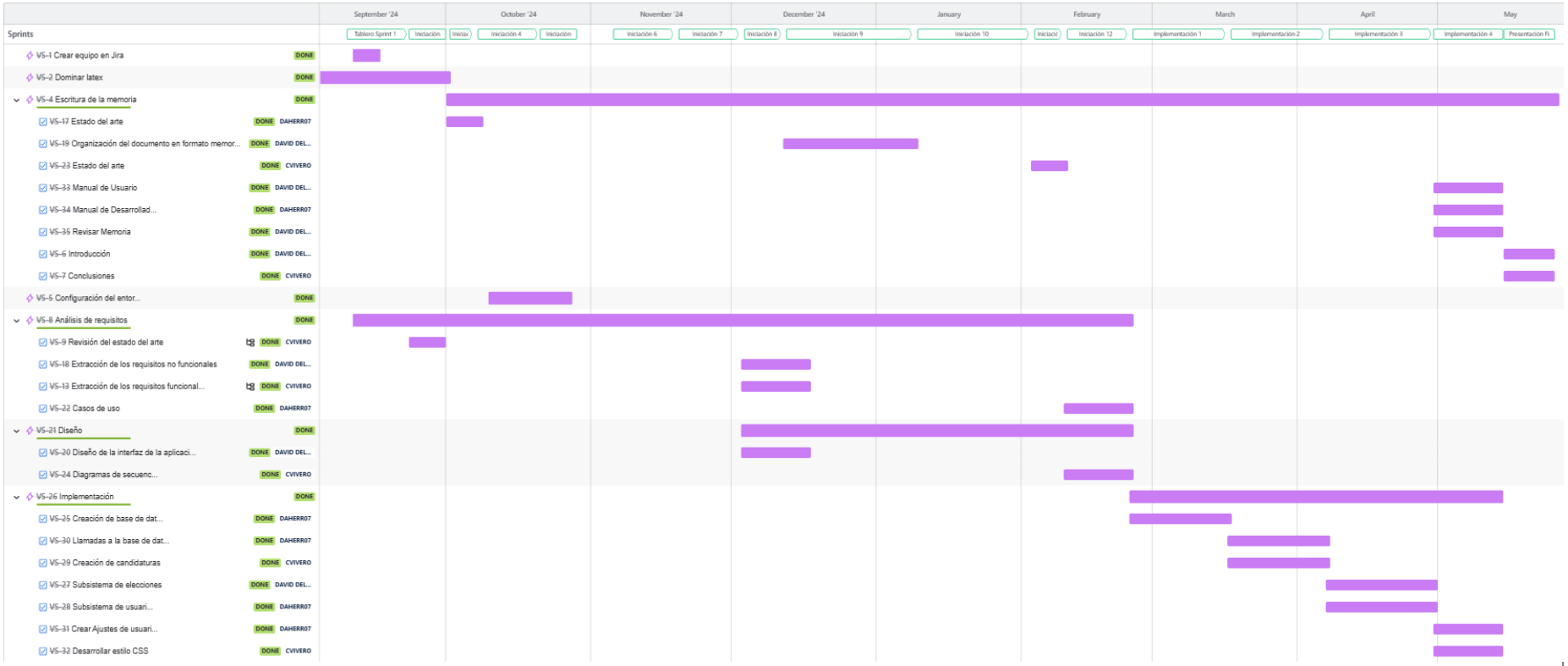


Figure 2.1: Gantt Diagram generated by JIRA Software.

2.4. Work Structure

The system is subdivided into 3 subsystems.

Despite having worked together as a team in the development of the project and even supporting in non-assigned tasks. Regarding the report, the 3 of us have contributed in all the sections except for those detailed in the task "Writing of memory sections", below. The main functions per person have been the following:

David Herranz Toribio:

- Design and implementation of the User Subsystem.
- Creation and maintenance of the Database.
- Documentation of Administrator, User and Installation Manuals.
- Use case diagrams.
- Writing of memory sections: Appendices B and C.

Carlos Vivero Barrera:

- Design and implementation of the Subsystem of Candidatures.
- Creation and maintenance of the project's GitHub repository.
- Class Diagrams.
- Writing of memory sections: 4 and 5.

David del Campo Peñuela:

- Design and implementation of the Elections Subsystem.
- Creation and organisation of the project in JIRA.
- Sequence Diagrams.
- Design of Mock-ups.
- Writing of memory sections: 1 and Appendix A.

Capítulo 3

Estado de la Cuestión

En este capítulo se explican las ideas principales extraídas antes de comenzar con el proyecto sobre los tipos de votaciones existentes y los ejemplos reales a día de hoy. Asimismo, se determinan las herramientas a utilizar dentro del proyecto en relación al análisis tecnológico, persistencia de datos y control de versiones.

3.1. Criptografía aplicada a Sistemas de Votación

Con el avance de las elecciones electrónicas, aumenta la probabilidad de que se produzcan ataques no deseados contra estos sistemas. Los votos pueden ser especialmente vulnerables a modificaciones maliciosas, ya sea directamente en el dispositivo del votante mediante *malware*, durante su transmisión por la red o a través de la manipulación de servidores. Por ello, es fundamental implementar mecanismos de verificación eficaces que hagan prácticamente inviable cualquier intento de alteración, garantizando así la celebración de elecciones seguras y libres de interferencias por parte de *hackers*.

En términos de votaciones, diferenciamos dos tipos:

- **Pública**: por ejemplo, una votación para la representación gubernamental del país. En este tipo, hay una alta probabilidad de amenazas de ataque por intereses de terceros.
- **Privada**: como puede ser la elección del delegado de clase en una universidad o el CEO (Chief Executive Officer) de una empresa. Hay menor riesgo de ataque pero es más probable que el ataque sea interno.

Ocultar o utilizar metodologías para ofuscar activamente el código no proporciona seguridad adicional y, de hecho, puede hacer que se pierdan los beneficios del código abierto. La metodología de criptografía de *security through obscurity*, no es

suficiente pues un algoritmo debe ser seguro aunque se sepa todo salvo su clave.

El blockhain produce más fiabilidad debido a que es inmutable. Pero el principal problema sigue siendo la parte del sistema antes de que los datos se registren en la blockchain, por lo que los sistemas deben diseñar contramedidas para evitar fallos. Es probable que en el futuro se desarrollen más ataques a sistemas de votación electrónicos por el aumento del presupuesto de ciberespionaje. Sobre todo por el desarrollo de "Quantum cryptography" que haría inútiles los actuales sistemas criptográficos.

Para nuestro proyecto hemos decidido usar cifrado simétrico (encriptación mediante una única clave privada) para mantener la seguridad. Esto nos aporta rapidez, eficiencia y simplicidad necesarias para los accesos a la base de datos y conexiones entre cliente y servidor.

3.2. Análisis Competitivo

Tras un proceso de investigación y selección de los distintos tipos de votación disponibles actualmente, se han evaluado las siguientes herramientas: Helios Voting (4), Scytl (5) y Simply Voting (6).

Helios Voting es un sistema de votación en línea de código abierto que se utiliza principalmente en elecciones no gubernamentales. Su transparencia y verificabilidad criptográfica permiten a los votantes confirmar sus votos sin comprometer el secreto, aunque presenta limitaciones en cuanto a escalabilidad y depende de la comunidad para soporte técnico.

Scytl, por su parte, es una plataforma integral utilizada en elecciones gubernamentales y comerciales, que destaca por su seguridad avanzada, cifrado extremo a extremo y escalabilidad, pero su costo elevado y su naturaleza propietaria limitan la posibilidad de auditorías externas.

Simply Voting es una plataforma en la nube fácil de usar, accesible para organizaciones de distintos tamaños, con un enfoque en la personalización de elecciones, aunque no cuenta con la verificabilidad criptográfica de Helios y puede resultar costosa a largo plazo para elecciones a gran escala.

Hay muchas características que no necesitamos en nuestro sistema de votacion, como la escalabilidad avanzada y el cifrado extremo a extremo de Scytl, dado que están pensados para contextos de gran escala y alto riesgo que no se ajustan a un proyecto de estas dimensiones. Además, el soporte para elecciones gubernamentales y cumplimiento de estándares legales de Scytl no sería relevante en un contexto más

pequeño y menos regulado.

Para nuestro proyecto, ya que buscamos un equilibrio entre seguridad, transparencia y facilidad de uso, implementaremos un sistema basado en Helios por su transparencia y verificabilidad, con elementos de Simply Voting para una interfaz intuitiva y personalización, aunque podríamos considerar la seguridad avanzada de Scytl si nuestro proyecto crece a mayor escala.

3.3. Análisis Tecnológico

En esta sección se hace un repaso de las tecnologías actuales, ventajas y desventajas de cada una de ellas y la elección de algunas para el desarrollo nuestra aplicación de voto seguro **DCD Vote**.

3.3.1. Frontend

Esta es la capa visual de la página web, encargada de renderizar los contenidos en las páginas web. Con **HTML** (HyperText Markup Language) se define el esqueleto de la página, las **CSS** (Cascading Style Sheets) dan estilo a la interfaz, y lenguajes como JavaScript añaden interactividad con la aplicación además de conectividad con el backend. Para su desarrollo se encuentran distintos frameworks y bibliotecas que hacen la construcción de la UI (Interfaz Gráfica) más eficiente, tal y como resume la Tabla 3.1.

Por un lado, **React.js** (7) sería ideal para la modularización reutilización de componentes, ya que nuestro proyecto tiene unos módulos muy definidos. Por otra parte, **Vue.js** (8) puede no ser más ideal en cuanto a la modularización, pero sí que tiene una curva de aprendizaje más suave que el resto. **Angular** (9) en cambio hemos considerado que no era ideal para nuestro proyecto dada nuestra inexperiencia, su curva de aprendizaje, y el hecho de estar más asociado a proyectos grandes, ya que consideramos que el nuestro no lo es. Finalmente nos hemos decantado por **Vue.js** principalmente por nuestra inexperiencia y su facilidad de aprendizaje junto con su gran documentación.

Para el desarrollo de las CSS está **Bootstrap** (10), el cual puede ser más amigable para usuarios sin experiencia ya que proporciona componentes pre-diseñados. Adicionalmente está **Tailwind CSS** (11), el cual no proporciona esos componentes pre-diseñados haciendo que cada página sea más única al no tener la misma base que otras. Para nuestro proyecto vamos a utilizar **Bootstrap** puesto que nos permitirá tener una base ya hecha para trabajar sobre ella, y dada nuestra inexperiencia con ambos consideramos que es lo mejor en nuestro caso.

Framework Biblioteca	Descripción	Ventajas	Desventajas
React.js	Biblioteca de JavaScript que permite crear interfaces de usuario interactivas de manera eficiente.	Permite crear componentes reutilizables y modulares, utiliza Virtual DOM para mejorar el rendimiento y tiene un gran ecosistema y comunidad.	Tiene una curva de aprendizaje moderada y depende de herramientas adicionales como Redux o React Router.
Vue.js	Framework progresivo para crear interfaces de usuario.	Es fácil de integrar con otros proyectos y tiene una documentación clara y sencilla, haciendo de él una curva de aprendizaje más suave.	Menor adopción en empresas grandes comparado con React.
Angular	Framework completo para construir aplicaciones web de gran escala.	Tiene un buen soporte para aplicaciones grandes y utiliza TypeScript.	Dispone de una mayor complejidad y curva de aprendizaje, además de ser excesivo para aplicaciones pequeñas.
Bootstrap	Framework CSS para desarrollo rápido de interfaces web.	Es una opción muy popular y ampliamente adoptada, además de proporcionar componentes pre-diseñados y listos para usar.	Puede hacer que la página se vea similar a otras si no se personaliza.
Tailwind CSS	Framework CSS basado en utilidades para crear diseños personalizados.	Dispone de un gran control sobre el diseño y estilo además de fomentar la creación de interfaces altamente personalizadas.	Requiere de un mayor trabajo para escribir y organizar las clases.

Tabla 3.1: Comparación de Frameworks y Bibliotecas

3.3.2. Backend

Esta es la capa interna que maneja la lógica del negocio, las bases de datos, la autenticación, y otros procesos que ocurren en el servidor. No es visible para los usuarios finales, pero es esencial para el funcionamiento de la aplicación, ya que gestiona todo lo que sucede detrás de la interfaz de usuario. Se hallan diversos frameworks y lenguajes para el desarrollo como se observa en la Tabla 3.2.

Dado el contexto de nuestra página, las operaciones que se vayan a realizar tienen una complejidad bastante baja, por lo que un menor rendimiento no es una gran desventaja. **Node.js** (12) nos permite utilizar JavaScript (2) tanto en el Backend como en el frontend, haciéndolo ideal al haber decidido el uso de Vue.js para el frontend. **Django** dispone de seguridad incorporada frente ataques críticos en nuestra página, pero esta se puede obtener igualmente con otros frameworks. **Spring Boot** (13) haría de nuestra sistema completamente dependiente de Java (14), lenguaje en el que los alumnos somos bastante experimentados, además de ser bastante modularizable, lo cual haría de este una opción muy ideal ya que separamos de forma muy clara los distintos módulos y funcionalidades de cada subsistema. **Laravel** (15) utiliza PHP (16), lenguaje en el que los alumnos tienen experiencia en el ámbito del desarrollo web, siendo este más fácil de usar que otros. Finalmente, pese a que Spring Boot podría ser el más interesante, o Laravel el más amigable para nosotros, nos hemos decantado por **Node.js** por su compatibilidad con Vue.js y su atractivo de ser en tiempo real.

Framework Tecnología	Descripción	Ventajas	Desventajas
Node.js	Entorno de ejecución para JavaScript diseñado para aplicaciones escalables y en tiempo real.	Tiene una velocidad y rendimiento mayor gracias a su arquitectura no bloqueante, haciéndolo ideal para aplicaciones en tiempo real. Además, permite utilizar JavaScript tanto en el frontend como en el backend.	No es recomendado para tareas intensivas de CPU, y la gestión de dependencias puede ser un problema.
Django	Framework de Python de alto nivel para desarrollo rápido y seguro de aplicaciones web.	Dispone de seguridad incorporada contra ataques de inyección SQL y XSS. Además, tiene una gran comunidad y una excelente documentación y es ideal para aplicaciones con datos complejos.	Puede ser pesado para aplicaciones pequeñas, tiene una curva de aprendizaje inicial moderada y un rendimiento menor comparado con otros frameworks más ligeros.
Spring Boot	Framework de Java diseñado para crear microservicios y aplicaciones con configuración mínima.	Tiene un excelente rendimiento para aplicaciones empresariales, una integración nativa con Spring y herramientas de Java, y es bastante escalable y modularizable.	Tiene una configuración inicial compleja, un mayor consumo de recursos y depende completamente del ecosistema de Java.
Laravel	Framework de PHP para construir aplicaciones web elegantes y eficientes.	Dispone de una sintaxis elegante y fácil de usar, una gran comunidad y ecosistema de herramientas como Forge y Nova, y es ideal para aplicaciones CRUD y proyectos rápidos.	Tiene un peor rendimiento y es menos idóneo para aplicaciones complejas y a gran escala.

Tabla 3.2: Comparación de frameworks y lenguajes

3.3.3. Persistencia de los datos

La base de datos es un conjunto de datos organizados almacenados y gestionados de una manera sencilla, permitiendo operaciones como leer, actualizar o eliminar datos. Existen las bases de datos (ver Tabla 3.3) no relacionales, que permiten una mayor flexibilidad en la estructura de los datos, y las relacionales, utilizan un esquema estructurado y tablas con columnas y filas. Nos hemos decantado por esta última dada su mayor simplicidad y la homogeneidad de nuestras estructuras de datos.

MySQL (17) es un sistema de gestión de bases de datos bastante usado, con la ventaja de estar muy probado. Su único problema reside en su mal rendimiento con datos complejos. Por otra parte, sistemas como **PostgreSQL** (18) son muy útiles para grandes bases de datos, pero su rendimiento es significativamente peor en proyectos de menor escala. **SQLite** (19) es muy simple de usar e ideal para sistemas pequeños, pero empeora bastante a mayor volumen de datos. Dado el contexto de nuestra página, el contenido de la base de datos no va a ser excesivamente grande, por lo que una mayor capacidad no es una ventaja, puede significar incluso una desventaja por el rendimiento. Tampoco consideramos que sea ideal una demasiado ligera, por lo que nos hemos decantado por sistemas tipo MySQL, ya que, además, contamos con experiencia previa en su uso.

Para mantener la limpieza y simplicidad en la base de datos, hemos optado por la utilización de un ORM (Object Relational Mapping) llamado **Sequelize** (20), que proporciona un mejor mantenimiento del código, mayor productividad y evita ciertos problemas de incompatibilidades. Adicionalmente, guardamos todos los datos y configuraciones necesarias en un Docker (1) mediante el sistema de gestión de bases de datos **MariaDB**, derivado de MySQL.

Base de datos	Descripción	Ventajas	Desventajas
MySQL	Sistema de gestión de bases de datos relacional ampliamente utilizado en aplicaciones web.	Dispone de un buen rendimiento, una gran comunidad de soporte y documentación además de una amplia compatibilidad con los lenguajes de programación.	Es limitado en operaciones avanzadas ya que su rendimiento decrece con datos complejos.
PostgreSQL	Sistema de gestión de bases de datos relacional, enfocado en extensibilidad y estándares.	Tiene un soporte avanzado para datos JSON y operaciones complejas, es muy robusto y confiable para grandes volúmenes de datos y es altamente personalizable y extendible.	Es más lento para operaciones simples y tiene una curva de aprendizaje más pronunciada.
SQLite	Base de datos ligera, integrada y autocontenida que no requiere un servidor para funcionar.	No requiere configuración ni instalación, es ideal para aplicaciones pequeñas o embebidas y es excelente para desarrollo rápido y pruebas.	No es adecuada para aplicaciones de gran escala o multiusuario, además de carecer de características avanzadas de bases de datos más grandes.

Tabla 3.3: Comparación de Bases de Datos

3.3.4. Entorno de desarrollo

Otorgan a los programadores un espacio para escribir, ejecutar y guardar el código. Suelen incluir herramientas avanzadas como depuradores, resaltado de sintaxis y autocompletado, lo que facilita el desarrollo de software. Los IDEs (Entornos Integrados de Desarrollo) más destacados se pueden apreciar en la Tabla 3.4.

Visual Studio Code (21) es muy usado, ligero, y adaptable a los proyectos gracias a sus extensiones. **WebStorm** (22) es ideal para el desarrollo de JavaScript aunque es más pesado y requiere de licencia. **IntelliJ IDEA** (23) es otro entorno muy bueno para el desarrollo en Java, pero consume muchos recursos y requiere de licencia. Otra opción es**PyCharm** (24), que es también muy pesado y se vende bajo

una licencia, aunque es muy útil para el desarrollo en Python (25). En nuestro caso vamos a programar en JavaScript, por lo que preferiríamos WebStorm por encima de los demás, pero, dado que requiere licencia y tenemos una gran experiencia con Visual Studio Code, preferimos usar este último por comodidad y su integración con **Git** (26).

Entorno	Descripción	Ventajas	Desventajas
Visual Studio Code	Editor de código fuente ligero y extensible. Compatible con múltiples lenguajes y herramientas de desarrollo.	Dispone de numerosas extensiones para mejorar el soporte de lenguajes y frameworks, es ligero y rápido, y tiene buena integración con Git.	Requiere de configuración para aprovechar al máximo sus capacidades y es dependiente de sus extensiones.
WebStorm	IDE diseñado específicamente para desarrollo en JavaScript, TypeScript y tecnologías web.	Tiene herramientas avanzadas para depuración y desarrollo de JavaScript, tiene integración nativa con frameworks como React y Vue.js y una refactorización inteligente y autocompletado robusto.	Requiere de licencia y consume más recursos en comparación a otros editores más ligeros.
IntelliJ IDEA	IDE profesional diseñado para el desarrollo en Java, ideal para aplicaciones con frameworks como Spring Boot.	Dispone de soporte nativo para Spring Boot y configuraciones automatizada, es excelente para desarrollo backend gracias a sus herramientas y tiene una refactorización avanzada y análisis de código.	Tiene una curva de aprendizaje elevada, consume muchos recursos y su versión completa requiere de una licencia.
PyCharm	IDE especializado en Python adecuado para desarrollo en Django, Flask, y análisis de datos.	Dispone de una integración nativa con Django, Flask y otros frameworks de Python, un depurador potente y herramientas para gestión de bases de datos, y un autocompletado y análisis de código avanzados.	Requiere una licencia y es más pesado que otros editores.

Tabla 3.4: Comparación de entornos de desarrollo

3.3.5. Control de versiones

Estas tecnologías facilitan el seguimiento y la gestión del código fuente a lo largo del desarrollo, asegurando un control eficiente de versiones y cambios. Las más destacadas se pueden apreciar en la Tabla 3.5.

Git (26) es muy popular y tiene una gran integración en entornos como Visual Studio Code, lo que permite aprovechar la plataforma **GitHub** (27). **SVN (Subversion)** (28) es más antiguo lo cual hace que sea más simple, pero al gestionar las versiones de forma centralizada, dificulta el trabajo sin conexión. **Mercurial** es muy bueno gracias a su simplicidad, consistencia y rendimiento, aunque tiene una comunidad más pequeña. **Perforce (Helix Core)** (29) es muy ideal para grandes proyectos en entornos empresariales debido a su complejidad y coste. Dado el tamaño de nuestro proyecto consideramos excesivo una tecnología como Perforce. Valoramos Mercurial por sus caraterísticas positivas, pero finalmente nos hemos decantado por **Git** dada la experiencia previa de los programadores con él y su buena integración con Visual Studio Code.

Entorno	Descripción	Ventajas	Desventajas
Git	Sistema de control de versiones distribuido utilizado para rastrear cambios en el código fuente y colaborar en proyectos.	Funciona de manera distribuida, permitiendo trabajar sin conexión, tiene ramas ligeras y eficientes y una gran comunidad y documentación, junto con ecosistemas de herramientas como GitHub.	Tiene una curva de aprendizaje inicial moderada, especialmente para principiantes, por ejemplo sus fusiones complejas pueden generar conflictos difíciles de resolver.
SVN (Subversion)	Sistema de control de versiones centralizado utilizado para mantener un historial único y centralizado de los cambios en un proyecto.	Este es sencillo para principiantes gracias a su modelo centralizado, y dispone de una gestión robusta de permisos y accesos en un entorno controlado.	Depende completamente de un servidor central, lo que dificulta el trabajo sin conexión, sus operaciones de branching y merging son más lentas y menos intuitivas, y tiene un menor soporte comunitario que otras.
Mercurial	Sistema de control de versiones distribuido, diseñado para ser eficiente y fácil de usar.	Tiene una interfaz más sencilla y consistente que otros, maneja bien proyectos grandes, y su rendimiento es sólido incluso con historiales extensos.	Menor popularidad y soporte comunitario.
Perforce (Helix Core)	Sistema de control de versiones centralizado y distribuido, utilizado en entornos empresariales y de desarrollo a gran escala.	Ofrece alto rendimiento con archivos grandes, control granular de permisos y una excelente gestión de ramas para grandes equipos.	Puede ser más complejo y costoso en comparación con alternativas, y su curva de aprendizaje es elevada.

Tabla 3.5: Comparación de tecnologías de control de versiones

Capítulo 4

Análisis del Sistema

En este capítulo se explica el objetivo de la página, sus funcionalidades, alcance, división en subsistemas y sus requisitos funcionales y no funcionales.

4.1. Características generales

A continuación se hace una breve introducción a la página web, desde su funcionalidad hasta la división en subsistemas.

4.1.1. Descripción del proyecto

DCD Vote es un sistema de votación seguro y fiable cuyo objetivo principal es habilitar elecciones pequeñas como la elección de delegado en las clases. Mediante una página web, permite a los profesores crear las elecciones y a los alumnos presentarse como candidatos y votar con el fin de realizar un posterior recuento de votos y mostrar los resultados.

La página principal del programa contiene un formulario de acceso o registro para el usuario. Una vez iniciado sesión con permisos de creación de elecciones (creador o administrador), se muestra la pestaña de creación de elecciones. Una vez aquí, se tendrá que elegir un título, los participantes convocados, la fecha de votación y opcionalmente una fotografía. El sistema generará un código identificador que pueda ser compartido con los alumnos para encontrarla fácilmente.

Los alumnos, habiendo iniciado sesión y buscado la elección, se pueden presentar como candidato, rellenando un formulario que incluye el eslogan de su candidatura, un texto de presentación y opcionalmente un vídeo. Además, también tienen la capacidad de consultar otras candidaturas y ejercer su voto de forma anónima. Una vez finalizada la votación, tanto ellos como el profesor podrán observar los dos ganadores mediante una lista ordenada de mayor a menor con los votos que ha

recibido cada candidato.

Se deben tener en cuenta algunas consideraciones: no se permitirá emitir más de un voto ni reemplazar uno ya realizado y la votación se llevará a cabo exclusivamente de manera en línea, sin posibilidad de participación presencial.

La vista general de la aplicación y su forma se detallan en las maquetas del apéndice de la memoria del proyecto, en el primer apéndice A.

4.1.2. Alcance del Proyecto

Hemos realizado la implementación de la totalidad del sistema descrito en este documento, desde la interfaz de usuario hasta la base de datos.
Se asume que los usuarios disponen de acceso a Internet y una cuenta de correo válida para la autenticación. No se integrará con plataformas externas como Moodle o Google Classroom y cada usuario tendrá una única cuenta para evitar fraudes. La plataforma será utilizada exclusivamente en un entorno académico, sin soporte para elecciones políticas o empresariales. El sistema garantizará la integridad de los datos mediante cifrado y medidas de seguridad y la votación se realizará de manera completamente digital, sin posibilidad de intervención manual o presencial.

4.1.3. División en Subsistemas

Hemos dividido **DCD Vote** en tres subsistemas:

1. **Subsistema de gestión de usuarios (SGU):** está a cargo de describir los permisos de los usuarios y como darse de alta, baja, consultar los datos y modificarlos. Este subsistema se encuentra detallado en la Sec. 4.2.1.1.

2. **Subsistema de gestión de elecciones (SGE):** este módulo permite el alta de una elección, su consulta, baja y modificación, además de trabajar conjuntamente con el subsistema de candidaturas para que este pueda realizar sus operaciones. También será el encargado de realizar los votos y su respectivo recuento. Se detalla en la Sec. 4.2.1.2.

3. **Subsistema de gestión de candidaturas (SGC):** desde una elección, el sistema le permite a un usuario perteneciente a ella dar de alta su candidatura, modificarla, consultar la suya propia y la de otros y darla de baja con ciertas restricciones detalladas en la Sec. 4.2.1.3.

4.2. Catálogo de requisitos

Los requisitos del sistema se dividen en dos categorías, los funcionales y los no funcionales. Los requisitos funcionales son aquellos que describen una funcionalidad de la página, mientras que los no funcionales son un conjunto de características, restricciones y atributos que mejoran la calidad del programa.

4.2.1. Requisitos funcionales

Los requisitos funcionales describen una funcionalidad específica que debe cumplir el sistema para su correcto funcionamiento. Se definen tres subsistemas de gestión, todos ellos encargados de una parte vital de la página.

4.2.1.1. Subsistema de Gestión de Usuarios

A continuación se detalla el subsistema de gestión de usuarios, desde sus permisos e ingreso a la página, hasta las funcionalidades de alta, baja, modificación, búsqueda y consulta de los mismos.

SGU-RF(1) El sistema soporta tres tipos de usuarios:

- **SGU-RF(1.1)** Usuario: un usuario puede consultar su perfil, modificarlo y eliminarlo, consultar elecciones en las que haya sido convocado, consultar las respectivas candidaturas y votar a un candidato en el periodo establecido.
- **SGU-RF(1.2)** Creador: tiene los permisos de un alumno además de dar de alta elecciones, modificar y dar de baja elecciones creadas por él mismo, dar de baja candidaturas si no ha empezado el proceso de votación y generar el recuento de votos.
- **SGU-RF(1.3)** Administrador: tiene incorporados los permisos de alumno y profesor, además de poder buscar, consultar, cambiar permisos y dar de baja a otros usuarios, y buscar elecciones aún si no está en el censo ni son suyas para modificarlas como un profesor.

  Dado que el administrador es el usuario privilegiado, no hace falta que disponga de varias cuentas para realizar las acciones de profesor o alumno. Éste puede directamente crear elecciones o votar en otras cuyo censo le incluya.

SGU-RF(2) Una persona puede darse de alta en la página web pulsando la opción "Registrarse", teniendo que rellenar un formulario.

- **SGU-RF(2.1)** Formulario con los datos del usuario a rellenar;

- Nombre del usuario. Cadena de 30 caracteres como máximo.
- Apellidos del usuario. Cadena de 60 caracteres como máximo.
- Correo electrónico. Cadena de 30 caracteres como máximo.
- Contraseña. Mínimo Cadena de 30 caracteres como máximo.

SGU-RF(2.2) La cuenta se da de alta con permisos de alumno, puesto que no se pueden elegir el tipo de usuario en el proceso de creación de cuenta.

SGU-RF(2.3) El sistema le asignará un identificador generado de manera automática.

SGU-RF(3) Un usuario registrado puede iniciar sesión en la página.

SGU-RF(3.1) Formulario con los datos del usuario a rellenar.

- Correo electrónico.
- Contraseña.

SGU-RF(4) En el apartado de configuración, un usuario puede consultar y modificar todos sus datos, cambiarse la foto de perfil y borrar su cuenta.

SGU-RF(5) Cualquier usuario puede modificar sus datos, mostrando un formulario parecido al de registro (**SGU-RF(2.1)**) con sus datos actuales predispuestos.

SGU-RF(6) Un usuario administrador tiene la opción de búsqueda de usuarios:

SGU-RF(2.1) La búsqueda tendrá los siguientes campos para su filtrado:

- Nombre del usuario. Cadena de 30 caracteres como máximo.
- Apellidos del usuario. Cadena de 60 caracteres como máximo.
- Correo electrónico. Cadena de 30 caracteres como máximo.

SGU-RF(2.2) El sistema le muestra una lista con todas las coincidencias, pudiendo consultar los perfiles, modificarlos, y darlos de baja.

4.2.1.2. Subsistema de Gestión de Elecciones

A continuación se detalla el subsistema encargado de las elecciones. Esto inlcuye darlas de alta, incluir el censo, consultarlas, darlas de baja y modificarlas, como se puede votar si puedes, o los permisos que tiene el administrador en este área.

SGE-RF(1) El sistema proporciona el siguiente tipo de votación para el creador.

- Sistema de representación mayoritaria uninominal: el delegado es el candidato con más votos, y el subdelegado el segundo candidato con más votos. Internamente el sistema guarda en una lista ordenada de mayor a menor todos los candidatos según el número de votos que han recibido, de tal manera que si un usuario por causas mayores no puede ser delegado o subdelegado, se sigue teniendo guardado el tercer candidato más exitoso.

SGE-RF(2) El sistema permite a un usuario con permisos de profesor dar de alta 0..* elecciones.

SGE-RF(2.1) El formulario con los datos a rellenar contendrá la siguiente información.

- Un título. Cadena de 30 caracteres como máximo (pe: Elección de delegado 1ºA).
- Lista de correos electrónicos correspondientes a los votantes de la elección. Tiene el tamaño del campo de cantidad de participantes y cada correo son máximo 30 caracteres acabados en "@ucm.es".
- Foto. En formatos .jpg, .jpeg, o .png, 5 MB máximo (opcional).
- Fecha de inicio y fin de la votación. Formato HH:mm - DD/MM/YYYY.

SGE-RF(2.2) Se genera automáticamente un código identificador para compartir con los votantes.

SGE-RF(2.3) Cada usuario incluido en el censo recibirá un correo electrónico avisándole de que ha sido convocado para unas elecciones, mostrándose además el link de acceso a la misma.

SGE-RF(3) Si está iniciada la sesión, la página le muestra al usuario todas las elecciones activas en las que sea partícipe de ella, además de la opción de búsqueda de elecciones (**SGE-RF(4)**).

SGE-RF(4) La página habilita a un usuario cualquiera la búsqueda de una elección.

SGU-RF(4.1) La búsqueda tiene los siguientes campos para su filtrado:

- Identificador de la elección. Código de 8 dígitos.
- Un título. Cadena de 30 caracteres como máximo (pe: Elección de delegado 1ºA).

SGU-RF(4.2) Se muestra una lista con todas las coincidencias en las que cuando la elección siga activa, además, si eres alumno debes estar incluido en el censo, y si eres profesor tienes que ser el creador.

SGU-RF(4.3) Desde aquí el página permite elegir la elección deseada para consultarla (**SGE-RF(5)**), modificarla (**SGE-RF(6)** para profesores y **SGE-RF(8)** para administradores), y darla de baja (**SGE-RF(7)**).

SGE-RF(5) El sistema otorga la capacidad de consultar una elección activa o inactiva en la que o estás incluido en el censo, o eres el creador de la misma. Para ello se muestra el título, imagen, tipo de votación, candidaturas, las fechas de inicio y fin de la votación, y adicionalmente los resultados si ya ha finalizado.

SGE-RF(6) El sistema le permite a un profesor modificar cualquier dato de la elección creada por él mismo, a excepción de las candidaturas, que en sí solo podrá darlas de baja. Para ello no podrá haber iniciado el periodo de votación, en cuyo caso solo puede alterar el título o la imagen. Una vez haya acabado la votación, no podrá modificarse nada. En cualquier caso, se muestra un formulario como el de alta (**SGE-RF(2.1)**) con los datos que se puedan modificar y teniendo predispuesta la información actual.

SGE-RF(7) En cualquier momento de una elección el sistema le permite al profesor creador de la misma darla de baja. Si la votación ha finalizado y ya se ha generado un resultado, se mantendrán durante un año el registro de votos y todos los datos excepto la fotografía.

SGE-RF(8) La página otorga al administrador la capacidad de modificar todos los datos de cualquier elección, a excepción del recuento de votos. Además, también las puede dar de baja (**SGE-RF(7)**).

SGE-RF(9) Una vez iniciado el periodo de votación el sistema notificará de ello a todos los usuarios pertenecientes a ella, mostrando además un enlace para que puedan acceder a ella.

SGE-RF(10) El sistema permite a un alumno votar 0..1 veces en una elección cuyo periodo de votación está activo. Para ello, al consultar la elección se revela, además de la lista de candidaturas, un botón de votar, al que al presionarlo aparece una pantalla acorde al tipo de elección **SGE-RF(1)**. Si solo requiere escoger un candidato, tendrá que seleccionar el que más le guste. Si requiere elegir más de uno, deberá que ordenar como prefiera la lista. Un usuario no podrá votar en nombre de otro.

SGE-RF(11) El alumno además, una vez efectuado su voto, verá un botón para descargar un comprobante de su voto.

SGE-RF(12) Una vez finalizado el periodo de voto desaparecerá la opción de votar y el creador de la elección tendrá que pulsar el botón de recuento de votos, generando y mostrando los elegidos a delegado y subdelegado. En todo caso se notificará a los votantes del resultado. Finalmente,

se puede descargar la lista con los votos efectuados a cada candidatura (de manera anónima). De ninguna manera podrá un usuario de cualquier rango alterar los cálculos.

4.2.1.3. Subsistema de Gestión de Candidaturas

A continuación, se detalla el subsistema encargado de la gestión de candidaturas, sus restricciones, el alta de candidaturas, su consulta, y qué puede y qué no puede hacer el profesor creador de la elección o un administrador.

SGC-RF(1) Las candidaturas están asociadas al código de la elección a la que pertenecen, por lo que si no tienes acceso a ella el sistema no te permitirá dar de alta una candidatura, ni consultarlas o modificarlas (a no ser que seas administrador, **SGC-RF(5)**, **SGC-RF(6)**).

SGC-RF(2) El sistema permite a un usuario con permisos de alumno dar de alta 0..1 candidaturas en 0..* elecciones distintas a las que pertenezca.

SGC-RF(2.1) Formulario de alta de candidatura:

- Eslogan de la candidatura. Cadena de 50 caracteres como máximo.
- Texto en el que se incluyan objetivos, motivaciones, etcétera. Cadena de 250 caracteres como máximo.
- Vídeo de presentación. En formato .mp4, 50 MB máximo (opcional).

SGC-RF(2.2) La página mostrará un aviso para confirmar su candidatura, ya que no se puede dar de baja ni modificar una vez comenzado el periodo de votación. Además, el profesor, que deberá dar el visto bueno para su candidatura, recibirá una notificación de ello.

SGC-RF(2.3) Una vez el profesor haya aceptado la candidatura, el sistema le notificará al alumno de ello.

SGC-RF(3) Se habilita un botón de consulta de una candidatura propia, además de las opciones de baja y modificación (ver requisito **SGC-RF(2.1)** con los datos predispuestos) si no ha iniciado el periodo de votación.

SGC-RF(4) El sistema muestra la lista de candidaturas, en la que se incluye par cada una el nombre del participante, su eslogan, y un botón para consultar más en detalle. Al presionar este botón se revela la fotografía del candidato, su nombre completo, eslogan, texto de introducción y opcionalmente el vídeo de presentación si el candidato lo ha incluido.

SGC-RF(5) El sistema otorga al administrador la capacidad de modificar cualquier dato de una candidatura, mostrando el formulario **SGC-RF(2.1)** con los datos predispuestos.

SGC-RF(6) El sistema permite al profesor creador de la elección, o a un administrador acceder a la lista de candidaturas, entrar en ellas, y darlas de baja siempre y cuando no haya comenzado el proceso de votación.

4.2.2. Requisitos no Funcionales

4.2.2.1. Seguridad

- **RNF-1** La Base de Datos del sistema debe ser segura. Solamente personal autorizado como administrador o usuarios especiales pueden acceder a ella, con permisos de lectura y escritura.
- **RNF-2** El sistema es capaz de responder a fallos sin perder el correcto funcionamiento gracias a un segundo servidor clonado con posibilidad de acceder a él en caso de fallo en cualquier momento.
- **RNF-3** El sistema confirma la identidad de la persona con Autenticidad de Datos. Esto se realiza mediante el doble factor de autenticación (se le mandará un código al correo) y el establecimiento de una clave simétrica con algoritmos híbridos. Estos son el algoritmo (AES-256), para la clave simétrica y el algoritmo (RSA-3072) para la clave asimétrica.
- **RNF-4** El sistema cumple tanto la privacidad de los datos de los usuarios como de sus votos. Con tal fin, se asigna a cada usuario un identificador con una función resumen (SHA-256) con la cual puede acceder a su voto, pero una persona externa no puede saber de quién es.
- **RNF-5** Se limita el número de consultas por dirección IP a 10, para evitar la denegación de servicio.
- **RNF-6** Se hace uso de DNSSEC para garantizar la autenticidad del servidor.
- **RNF-7** El sistema es robusto ante ataques como SQL injection, XSS y CSRF. Además, es capaz de validar y sanear toda entrada de datos del usuario.
- **RNF-8** Se utiliza el algoritmo criptográfico AES-256 para:
  - Asegurar la identidad de los usuarios (Datos personales).
  - Mantener la integridad de los votos.
  - Restringir el acceso a la información de usuarios y votos que se encuentra contenida en la base de datos y es sensible para el sistema.
  - Descifrar el contenido de los votos para contabilizarlos.
- **RNF-9** El sistema utiliza el algoritmo TLS 1.3 (Transport Layer Security) (30) para cifrar todas las comunicaciones en tránsito entre el cliente y el servidor.

4.2.2.2. Requisitos de Interfaz

- **RNF-10** El sistema tiene una interfaz gráfica de usuario que facilita el correcto funcionamiento de las elecciones.
- **RNF-11** El sistema tiene una interfaz Web.
- **RNF-12** El color de la página web es fondo blanco (hex: ffffff) con los detalles principales como títulos o logotipos en naranja salmón (hex: f2cd8c) y con el color de la letra únicamente en negro (hex: 000000) o blanco (hex: ffffff).
- **RNF-13** El sistema tiene los botones más relevante o con referencias a enlaces se establecen sobre un color azul (hex: 3e9bff).
- **RNF-14** La página dispone de una interfaz que cuenta con un manual de usuario para que sea fácil para el usuario realizar su voto.
- **RNF-15** El sistema es "Web Responsive"(Diseño web adaptable) con un reajuste automático para conseguir la mejor experiencia del usuario en función de su dispositivo.

4.2.2.3. Disponibilidad

- **RNF-16** El sistema está disponible para los usuarios en todo momento para realizar las votaciones de forma ininterrumpida desde las 6:00 a 23:59, permitiendo establecer el inicio y fin de las votaciones en dicho intervalo. A excepción de los festivos nacionales en España.
- **RNF-17** Para el uso de la página web del sistema el usuario necesita una conexión a Internet.

4.2.2.4. Mantenimiento

- **RNF-18** El sistema tiene actividades de mantenimiento programadas de manera semanal, los jueves de 3:00 a 5:00, para el correcto funcionamiento de la página.

4.2.2.5. Portabilidad

- **RNF-19** El sistema debe poder utilizarse en Navegadores web como Google Chrome (versión Chrome 91 o posterior) o Mozilla Firefox (versión Firefox 95 o posterior).

4.2.2.6. Documentación

- **RNF-20** El sistema está documentado y permite futuras actualizaciones gracias a un código legible y ordenado.

4.2.2.7. Usabilidad

- **RNF-21** El léxico del sistema es utilizado de manera de fácil lectura y en ningún caso es elaborado para evitar el uso de términos complejos que entorpezcan el entendimiento de la lógica de la página.

4.2.2.8. Rendimiento

- **RNF-22** Todas las operaciones del sistema tienen un tiempo máximo de respuesta de 10 segundos.
- **RNF-23** El sistema realiza el recuento de votos de manera automática, asegurando la precisión y la integridad de los resultados sin intervención manual, con un tiempo de respuesta no superior a 60 segundos.

4.2.2.9. Legislativos y Regulatorios

- **RNF-24** El sistema garantiza el cumplimiento de todas aquellas normativas que sean de aplicación a nivel estatal y en el ámbito de la Unión Europea.
- **RNF-25** La página cumple la ley de Reglamento (UE) 2016/679 del Parlamento Europeo y del Consejo, de 27 de abril de 2016 (31), relativo a la protección de las personas físicas en lo que respecta al tratamiento de datos personales y a la libre circulación de estos datos y por el que se deroga la Directiva 95/46/CE (Reglamento general de protección de datos).

4.3. Diagramas de Casos de Uso

En los siguientes casos de usos se muestran las acciones que los distintos tipos de usuario puede realizar en nuestra aplicación. Cualquier usuario debe crearse una cuenta e iniciar sesión.

El usuario Alumno, una vez iniciado sesión, puede:

- Consultar votaciones activas.
- Votar en una elección/votación activa.
- Modificar sus ajustes de usuario.
- Postularse como candidato en una votación no acabada.

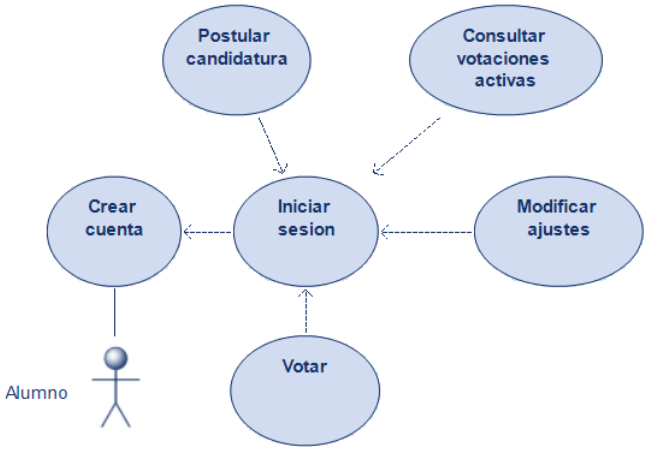


Figura 4.1: Diagrama de casos de uso para un usuario Alumno

El usuario Profesor (creador), una vez iniciado sesión, puede:

- Crear votaciones.
- Gestionar candidaturas presentadas en sus votaciones.
- Modificar sus votaciones no acabadas.

Además, el profesor puede hacer todo lo que hace el alumno excepto votar y presentar una candidatura

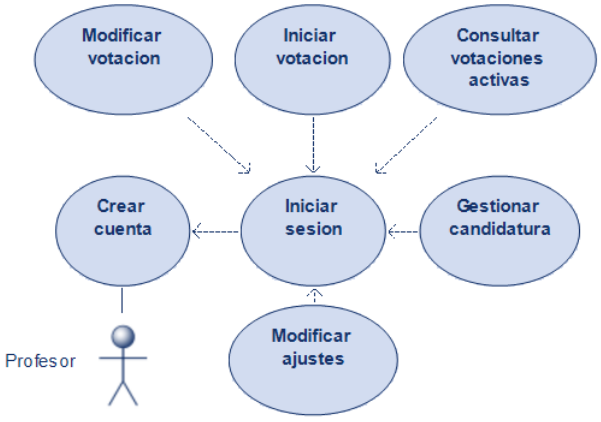


Figura 4.2: Diagrama de casos de uso para un usuario Profesor

El usuario Administrador, una vez iniciado sesión, puede:

- Modificar a todos los usuarios.
- Gestionar todas las candidaturas presentadas.
- Modificar todas las votaciones no acabadas.

A su vez, el administrador puede hacer todo lo que hace el alumno excepto votar y presentar una candidatura

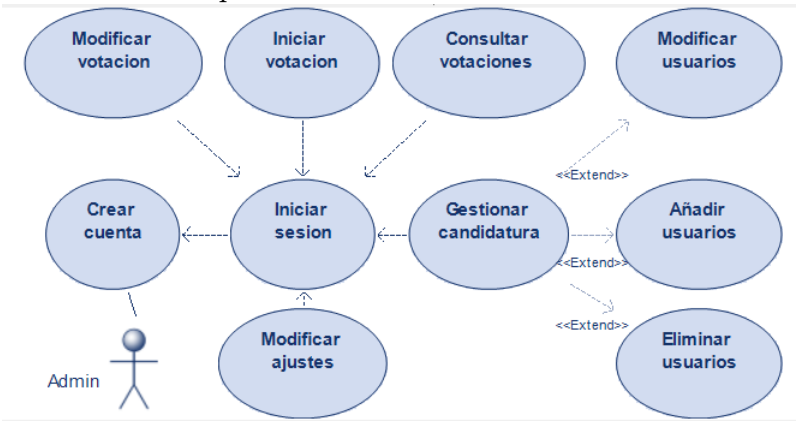


Figura 4.3: Diagrama de casos de uso para un usuario Administrador

Capítulo 5

Diseño del proyecto

En este capítulo se explica el diseño del proyecto con su correspondiente diagrama de clases.

5.1. Diseño del proyecto

Aquí detallamos la estructura y composición del proyecto a nivel de código y la arquitectura de alto nivel. Definimos los distintos componentes del programa, su funcionalidad y los diagramas de clases y secuencia para un mayor entendimiento.

5.1.1. Diagrama de clases

En esta sección mostramos los patrones que se han usado para dividir el código en distintos componentes para una mayor legibilidad, escalabilidad y facilidad a la hora de programar. Adjuntamos además una imagen que representa el diagrama de clases del proyecto.

Los patrones de programación son formas de gestionar el código para lograr un objetivo, ya sea dividir un programa en distintos componentes para separar las tareas de cada uno, comunicar distintos componentes entre sí, etcétera. En nuestro caso, distinguimos varias tareas que deben ser divididas para hacer la programación más fácil y el código más escalable. Primero, debemos separar el frontend (parte visual, ver 3.3.1) y el backend (contiene la lógica, ver 3.3.2). El backend se divide en varios componentes:

1. **Controlador:** se comunica con el frontend para recibir las peticiones, se las asigna al subsitema indicado (por ejemplo, registrar una candidatura será asignado al subsistema de gestión de candidaturas (SGE)), y finalmente se comunica con el servicio para transmitirle la petición.

2. **Servicio:** contiene toda la lógica de las solicitudes. Recibe las peticiones del controlador y realiza tareas como comprobar que los datos son correctos, verificar que los identificadores que recibe corresponden con un objeto real, etcétera. Finalmente se comunica con los repositorios para actualizar la información en la base de datos.

3. **Repositorio:** recibe las peticiones del servicio, ya sean de crear, borrar, modificar, buscar o derivadas y las realiza en las tablas de la base de datos correspondiente.

4. **Modelo:** define los objetos del programa en la base de datos y como se conectan, siendo estos las clases de usuario, elección, candidatura, y finalmente voto. Son utilizados por el resto de componentes para gestionar la información de forma fácil y efectiva.

La Figura 5.1 ilustra el flujo de la aplicación.
La Figura 5.2 muestra el diagrama de clases correspondiente al modelo. Cabe destacar que todos los atributos de las clases tienen sus correspondientes *getters* y *setters*.
Los principales componentes son:

- Clase **User**: diseñada para guardar la información de un usuario: nombre, apellidos, correo electrónico, etc. Los permisos se almacenan en forma del enumerado **Permissions** con tres posibles valores (usuario, creador y administrador).

- Clase **Candidacy**: planificada para representar una candidatura dentro de una elección. Los atributos de éstas son el usuario candidato, el id de la elección, datos generales de la candidatura, y si ha sido aprobada o no.

- Clase **Election**: clase diseñada para guardar la información de una elección. Almacena sus participantes y fecha de inicio y fin de votación y el título de la misma entre otros. Finalmente, guarda una lista de votos, estos representados por la clase **Vote** que contiene el id del votante y la candidatura a la que ha votado.

La Figura 5.3 muestra el diagrama de clases correspondiente al controlador, servicio y repositorio.
Las principales componentes son:

- Controladores: las clases **UserController**, **CandidacyController** y **ElectionController**, cada una para su respectivo subsistema. Los controladores tienen los métodos de alta, baja, modificación, búsqueda y consulta, además de métodos particulares para cada subsistema como es el caso del de elección, que contiene también métodos de voto, consultar voto, y añadir y borrar una candidatura. Como atributo tienen su respectivo servicio.

- Servicios: las clases de **UserService**, **CandidacyService** y **ElectionService**, cada una para su respectivo subsistema. Estas tienen los mismos métodos

que los controladores y cuentan con su repositorio como atributo, a excepción del de elección que cuenta también con el servicio de la candidatura para los métodos de añadir y borrar candidatura.

- Repositorios: la clase **Repository** contiene los métodos para la ejecución de una query a la base de datos de cualquier tipo. Las clases **UserRepository**, **CandidacyRepository**, **ElectionRepository** heredan estos métodos para realizar consultas particulares de cada subsistema.

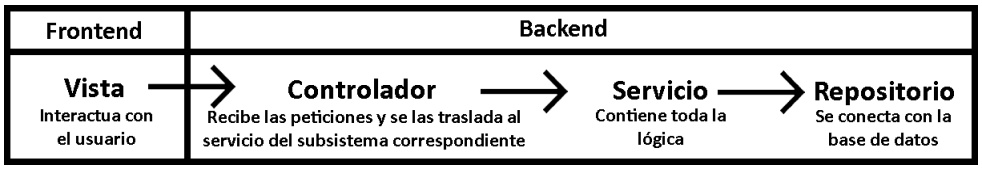


Figura 5.1: Flujo de la comunicación entre los distintos componentes

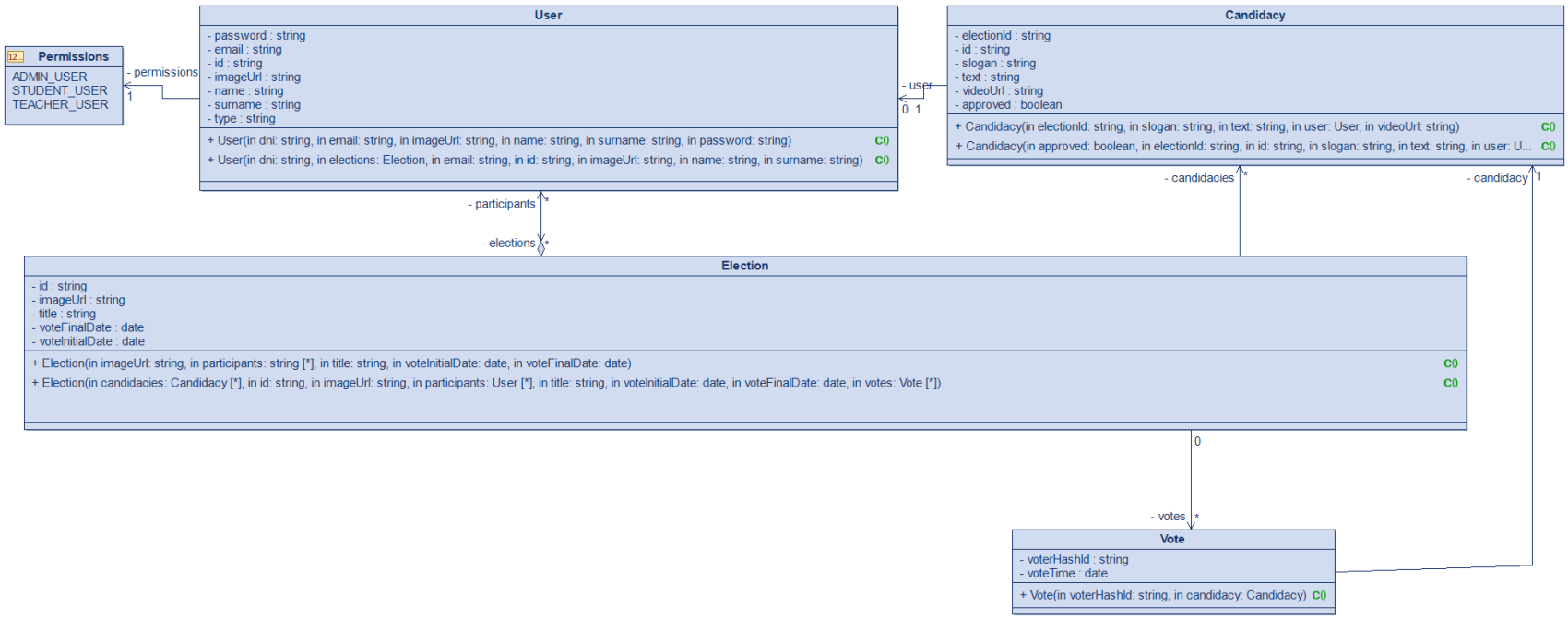


Figura 5.2: Diagrama de Clases del modelo

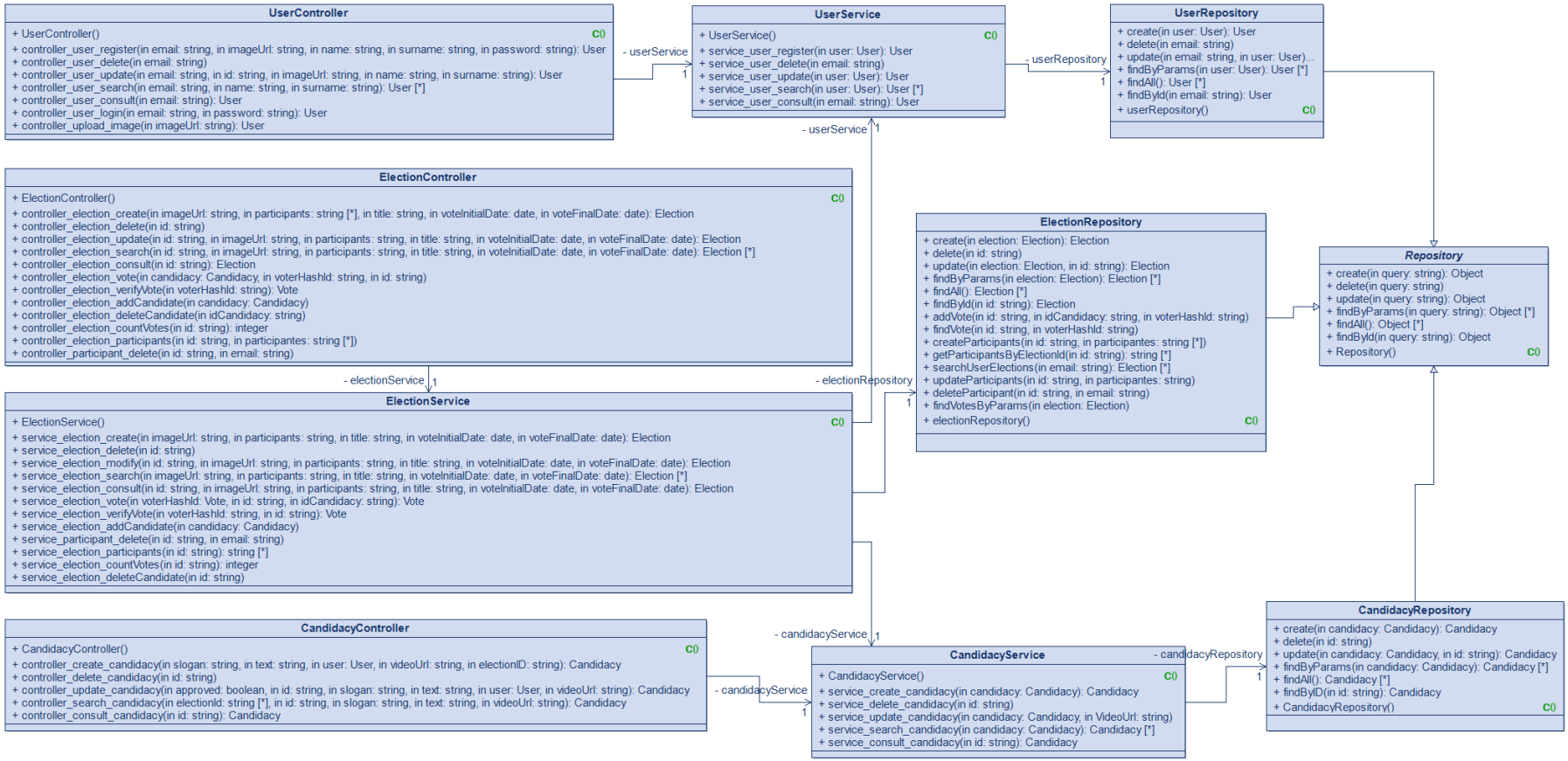


Figura 5.3: Arquitectura de Controlador, Servicio y Repositorio

5.1.2. Diagramas de secuencia

En esta nueva sección se describen los diagrama de secuencia más relevantes del sistema. Con el fin de evitar redundancias, hemos adjuntado los diagramas de secuencia más relevantes del sistema. Aunque existen otros diagramas relacionados con funcionalidades similares, entre Candidatura, Elección y Usuario, se comparte una lógica estructural prácticamente idéntica. Por ello, solo mostramos los que mejor enseñan cómo funciona el sistema en general, asumiendo que los demás siguen el mismo patrón de interacción y diseño.

Figura 5.4:
Un Usuario con rol de administrador o profesor puede crear una nueva elección. Los campos obligatorios son título, fecha de inicio, fecha de fin y lista de participantes. En el caso de que haya algún error con algún parámetro el sistema avisa de dicho inconveniente. La fecha de fin debe de ser posterior a la de inicio y la lista de participantes no puede estar vacía.

Para la creación se busca en el Repository por título si la elección ya existe. En el caso de que no, lanza la función de crear y el mensaje final "Elección creada con éxito."

Figura 5.5:
Para el voto de un usuario necesitamos el ID de la Elección, ID de la candidatura que se quiere votar y el valor del voto modificado con Hash. Se busca en el ElectionRepository que la Elección exista, si no muestra el error "La Elección asociada al candidato establecido no existe". Si existe, pasa de nuevo al ElectionService esta vez para registrar el voto y lo crea en el Repository.

En caso de que haya algún problema al crear el voto, como por ejemplo un fallo de conexión con la Base de Datos, muestra el mensaje "Error al registrar el voto.". En cambio, si el proceso es correcto "Voto registrado con éxito."

Figura 5.6:

Un Usuario puede actualizar su perfil en cualquier momento. Éste puede cambiar su email, contraseña, nombre, apellido e imagen. Al introducir los parámetros anteriores, el sistema busca en el Repository el ID del Usuario por medio del email. Cuando lo encuentra, modifica los nuevos parámetros introducidos y muestra el mensaje "Usuario modificado con éxito". Si no es así, ha ocurrido un problema al acceder al Usuario y muestra lo siguiente "El usuario no existe".

El diagrama de secuencia de modificar usuario sigue esencialmente el mismo proceso que el de crear y eliminar usuario. El diagrama refleja los casos de uso de un Usuario para cambiar sus ajustes como la contraseña o la foto de perfil.

Figura 5.7:
En el caso de consultar una Candidatura existente únicamente necesitamos el ID de la Candidatura. Por medio del CandidacyRepository accedemos a la Base de Datos

y se obtiene la Candidatura con éxito o se muestra el error “La candidatura no existe”. Una vez tenemos la candidatura consultada, comprobamos el Usuario asociado a la candidatura por medio del email. Volvemos a comprobar que el Usuario existe y en tal situación el mensaje mostrado es “Candidatura consultada con éxito”.

Figura 5.4: Crear Elección

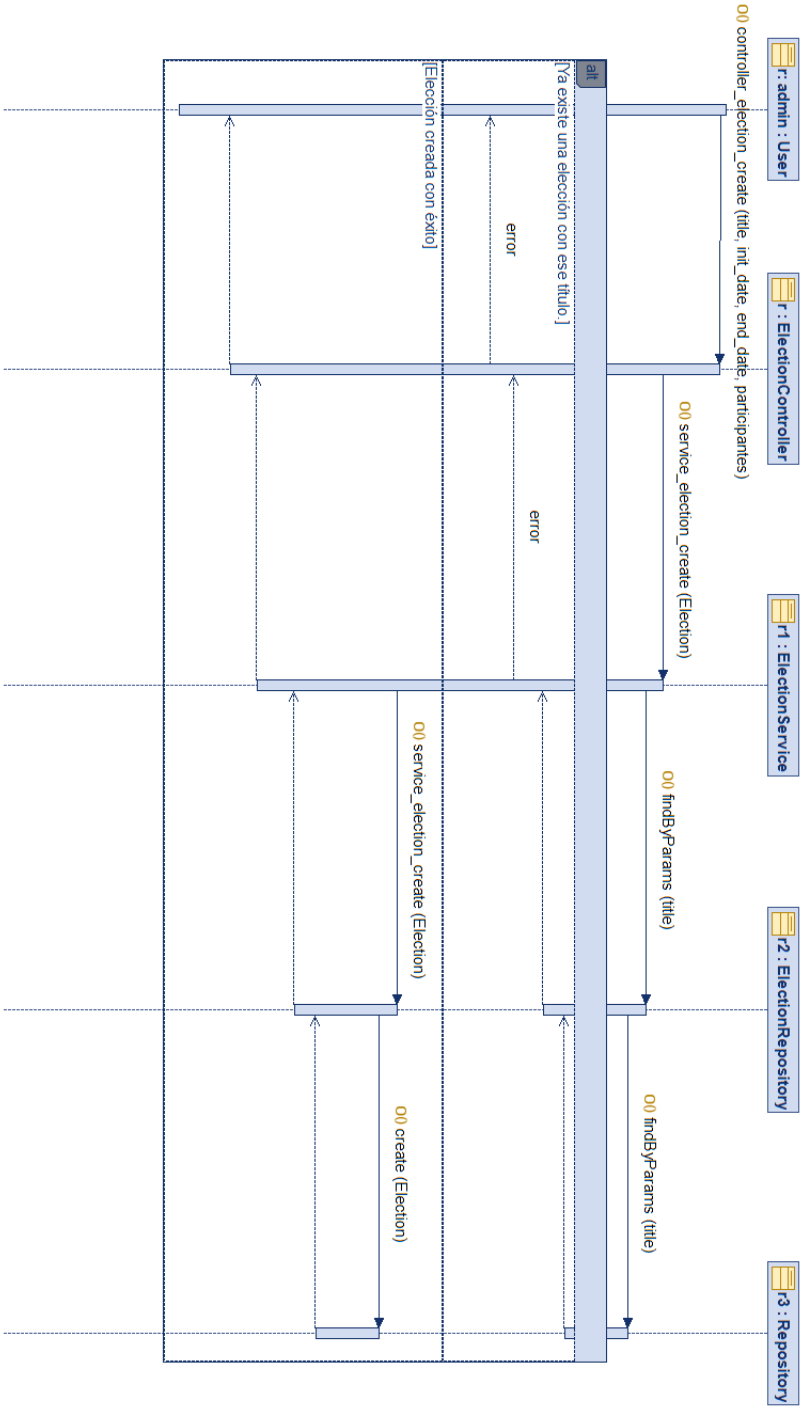


Figura 5.5: Votar

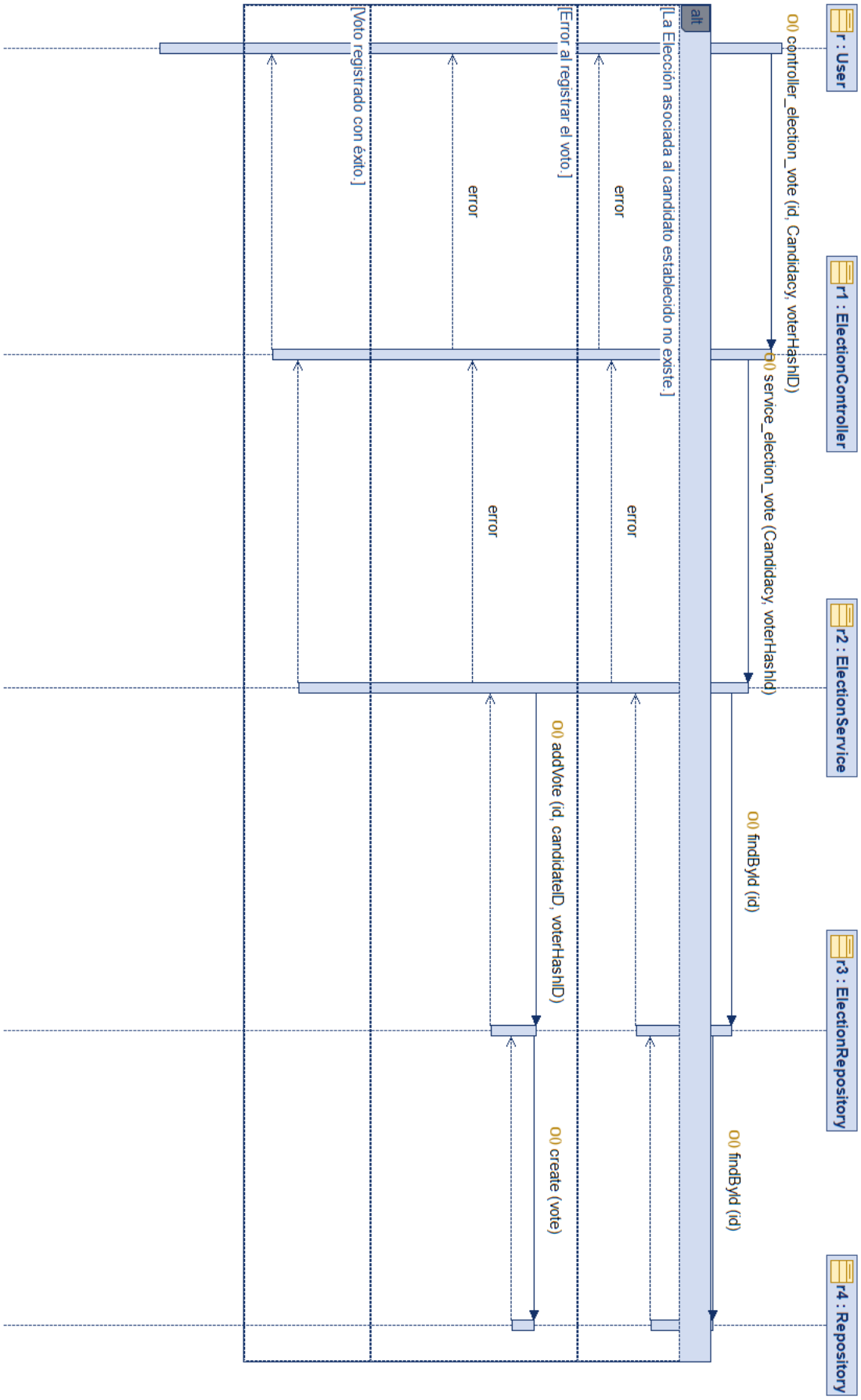


Figura 5.6: Modificar usuario

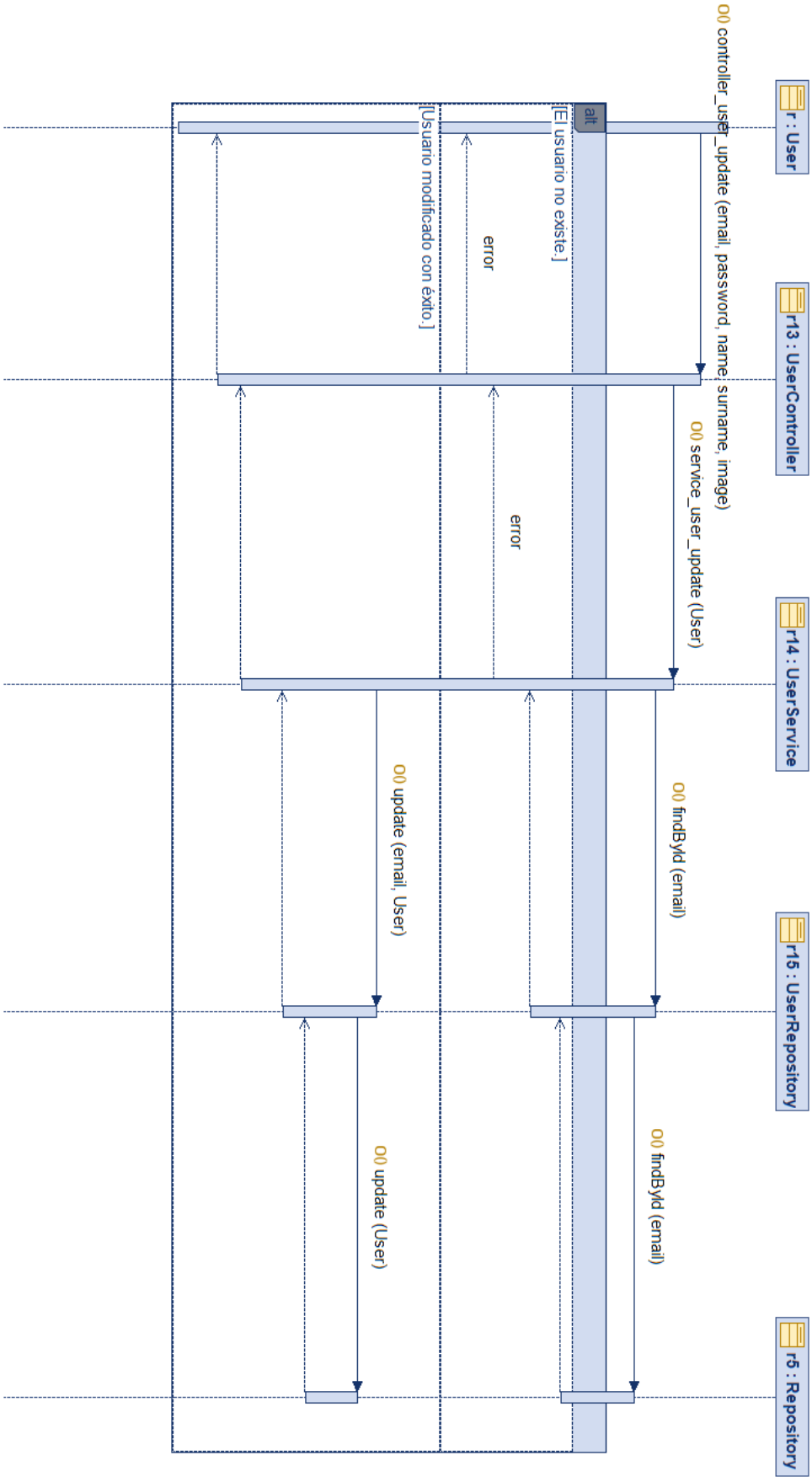
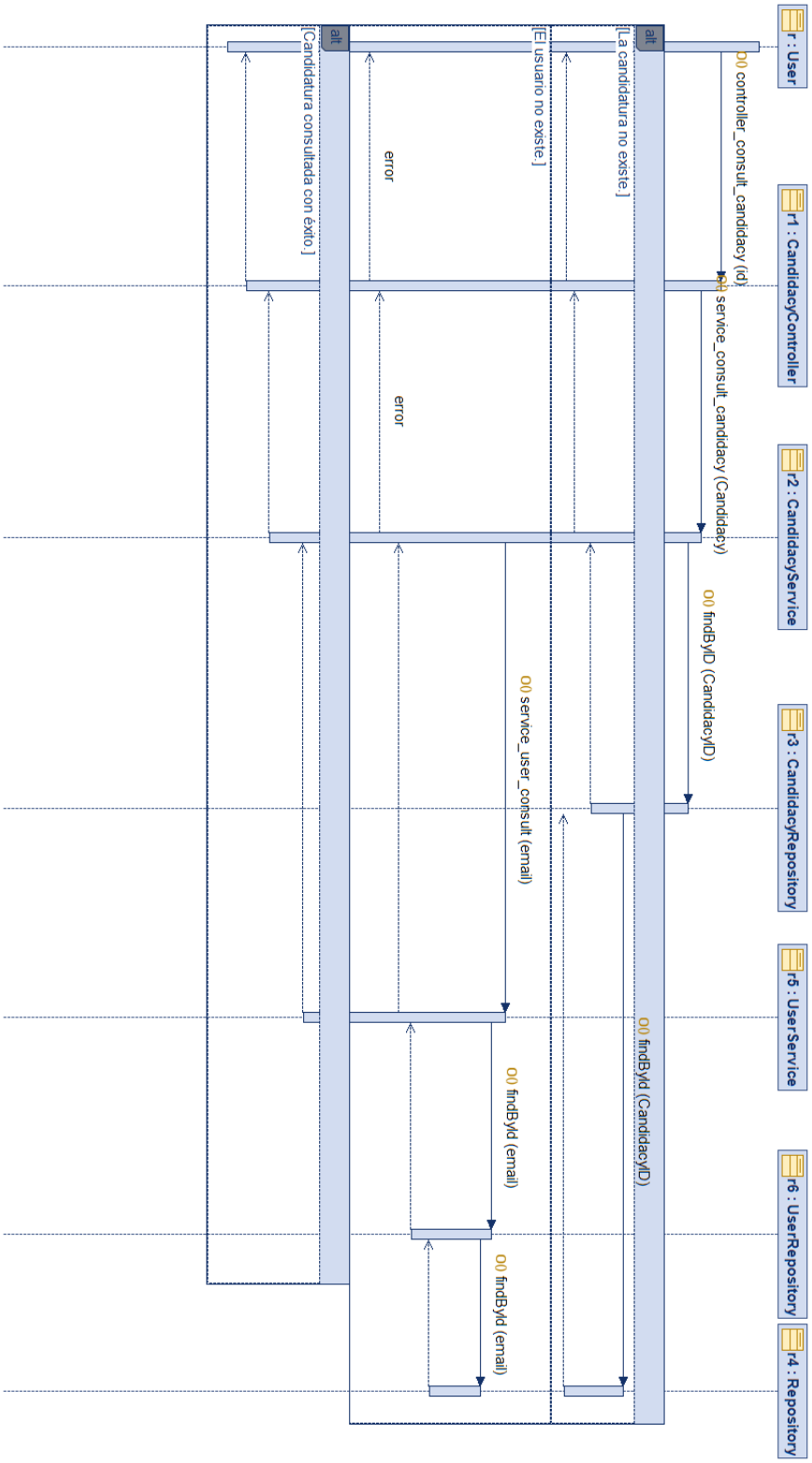


Figura 5.7: Consultar candidatura



Capítulo 6

Implementación

En este apartado se explica cómo hemos llevado a cabo la parte técnica del proyecto, qué tecnologías hemos utilizado, cómo está organizada la aplicación y algunas decisiones importantes que hemos tomado durante el desarrollo.

6.1. Tecnologías utilizadas

Para hacer este proyecto hemos usado varias tecnologías que se complementan bien. Para el diseño de la parte visual usamos **CSS junto con Bootstrap**, que nos ayuda a que la web tenga un diseño aceptable sin tener que hacerlo todo desde cero. Para la lógica del lado del cliente usamos **JavaScript con Vue.js**, y para el servidor usamos **Node.js**.

La base de datos es **MySQL** y la hemos montado con **Docker Desktop**, lo que facilita mucho el despliegue y que todo funcione igual en diferentes ordenadores. También usamos un archivo `docker-compose.yml` para configurar todo el sistema (base de datos, servidor, etc.) y que se conecte correctamente entre sí. Además, hacemos uso del ORM Sequelize para mantener limpieza y legibilidad en el código, corrección y previsión de errores en distintos entornos.

Para llevar el control de versiones y trabajar de forma organizada, usamos **Git** con **GitHub**, donde tenemos subido todo el código.

6.2. Arquitectura del sistema

La aplicación sigue un modelo cliente-servidor. El cliente (el navegador del usuario) hace peticiones al servidor usando **Axios** (32). Por ejemplo, cuando un usuario quiere registrarse o votar, el cliente manda una petición al servidor, que se encarga de procesarla.

En el servidor seguimos un flujo en tres capas:

- **Controlador**: recibe las peticiones del cliente.
- **Servicio**: contiene la lógica de negocio.
- **Repositorio**: se encarga de hablar con la base de datos.

Este modelo nos ha ayudado a tener el código más ordenado y fácil de mantener.

6.3. Subdivisión de la aplicación

La aplicación está dividida en tres partes principales, o subsistemas:

- **Usuarios**: incluye las cinco funciones básicas (crear, borrar, modificar, buscar y listar usuarios), además del **login** y **logout**.
- **Candidaturas**: también tiene las cinco funciones básicas.
- **Elecciones**: además de las cinco funciones básicas, incluye la parte de **gestión del voto**, que es la más importante del proyecto.

Cada una de estas partes tiene sus propias rutas y controladores, lo que hace que el proyecto sea modular y más fácil de entender.

6.4. Decisiones técnicas importantes

Durante el desarrollo hemos tenido que tomar algunas decisiones que creemos que son importantes comentar.

Para conectar la base de datos, usamos un contenedor de Docker que se levanta junto con el servidor, todo orquestado desde un archivo `docker-compose.yml`. Esto hace que la configuración sea más simple y reproducible. Además, hacemos uso de un ORM (Object-Relational Mapping) para javascript llamado Sequelize. Esto permite mantener las llamadas a la base de datos limpias y mejora la escalabilidad del código, eliminando la necesidad de escribir manualmente las *queries* y automatizando tareas.

Para gestionar la sesión del usuario, usamos **JSON Web Tokens** (33) que guardamos en el navegador del cliente (`localStorage`). Esto permite saber si un usuario ha iniciado sesión y con qué permisos.

En cuanto a la **seguridad**, para añadir una capa extra de anonimato al sistema de votación, decidimos **generar un hash** con varios datos del usuario. Esta función hash se asocia al voto, por lo que no se guarda directamente la identidad del votante. Esto también abre la puerta a implementar más adelante una funcionalidad que permita consultar un voto ya emitido, aunque todavía no está del todo implementado.

6.5. Pruebas Unitarias

Para garantizar fiabilidad y un correcto funcionamiento del sistema, se han implementado pruebas unitarias sobre los distintos componentes tanto en el backend como en el frontend.

El sistema permite algunos de los caracteres especiales, como el @, en los distintos campos de texto. A su vez, se ha confirmado su robustez frente a ataques maliciosos como SQL inyection y XSS (Cross Site Scripting).

Las fotos de perfil tienen una restricción sobre sus extensiones para evitar archivos maliciosos. Los videos opcionales en las candidaturas también sufren la misma restricción, además de una limitación del tamaño para evitar sobrecarga de espacio en el sistema.

Se han realizado pruebas para garantizar que los usuarios con rol de alumno no puedan llevar a cabo acciones que pueden hacer otros roles y viceversa.

Por otro lado, el sistema asegura que un mismo usuario pueda votar una única vez en cada elección mediante la lista de votantes, impidiendo que cualquier usuario no incluido en dicha lista pueda votar.

Por último, el sistema garantiza que cualquier intento de voto realizado una vez finalizado el periodo de votación sea rechazado.

6.6. Aplicación Final

En este apartado, se muestra la visualización de los principales casos de uso de la Aplicación Web. Además, en el primer anexo, (**Figura A.1**) se muestra el flujo completo del sistema en una panorámica. En esta, se encuentran todas las ventanas de la aplicación y la conveniente explicación de cada una de las pantallas con su maqueta.

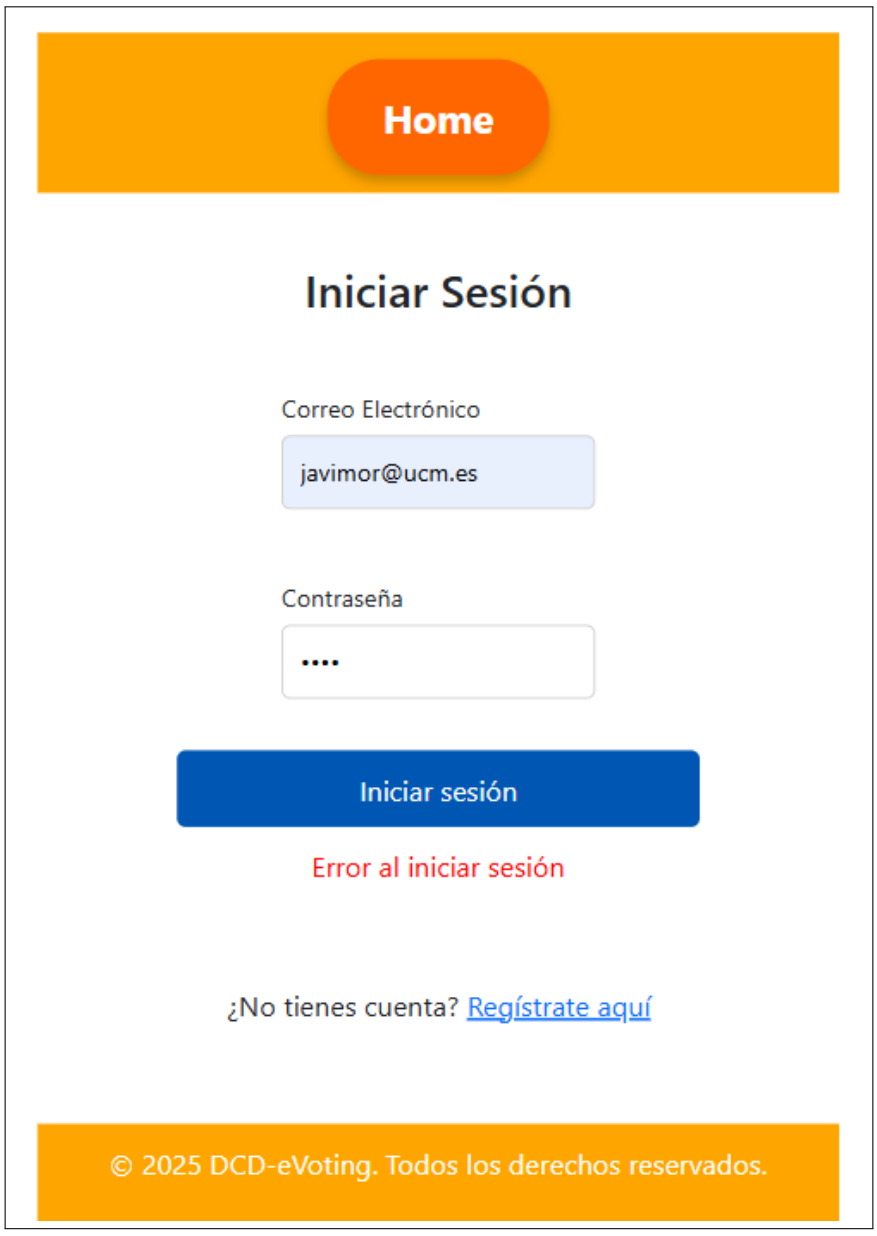


Figura 6.1: Iniciar Sesión

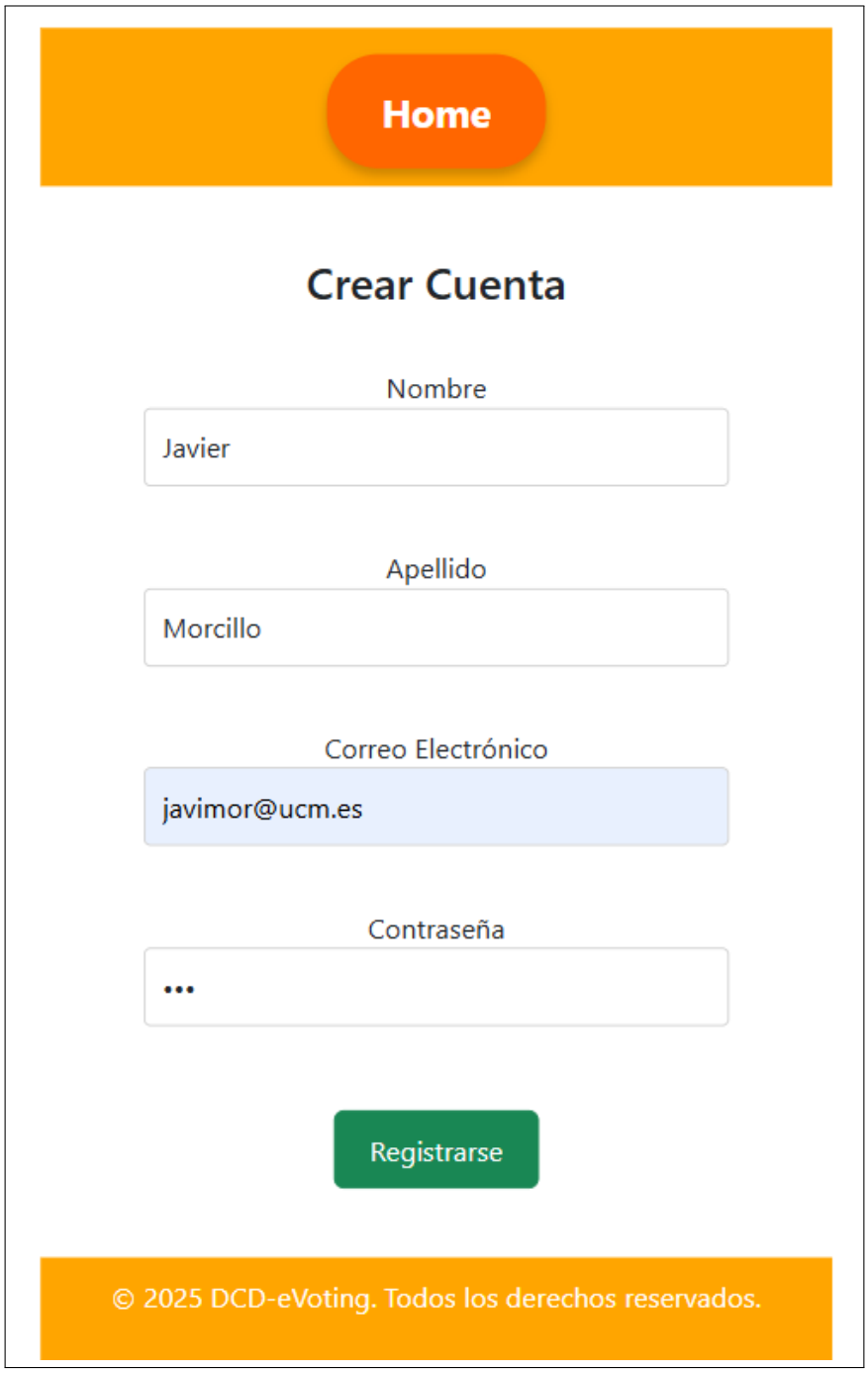


Figura 6.2: Registrar un nuevo Usuario

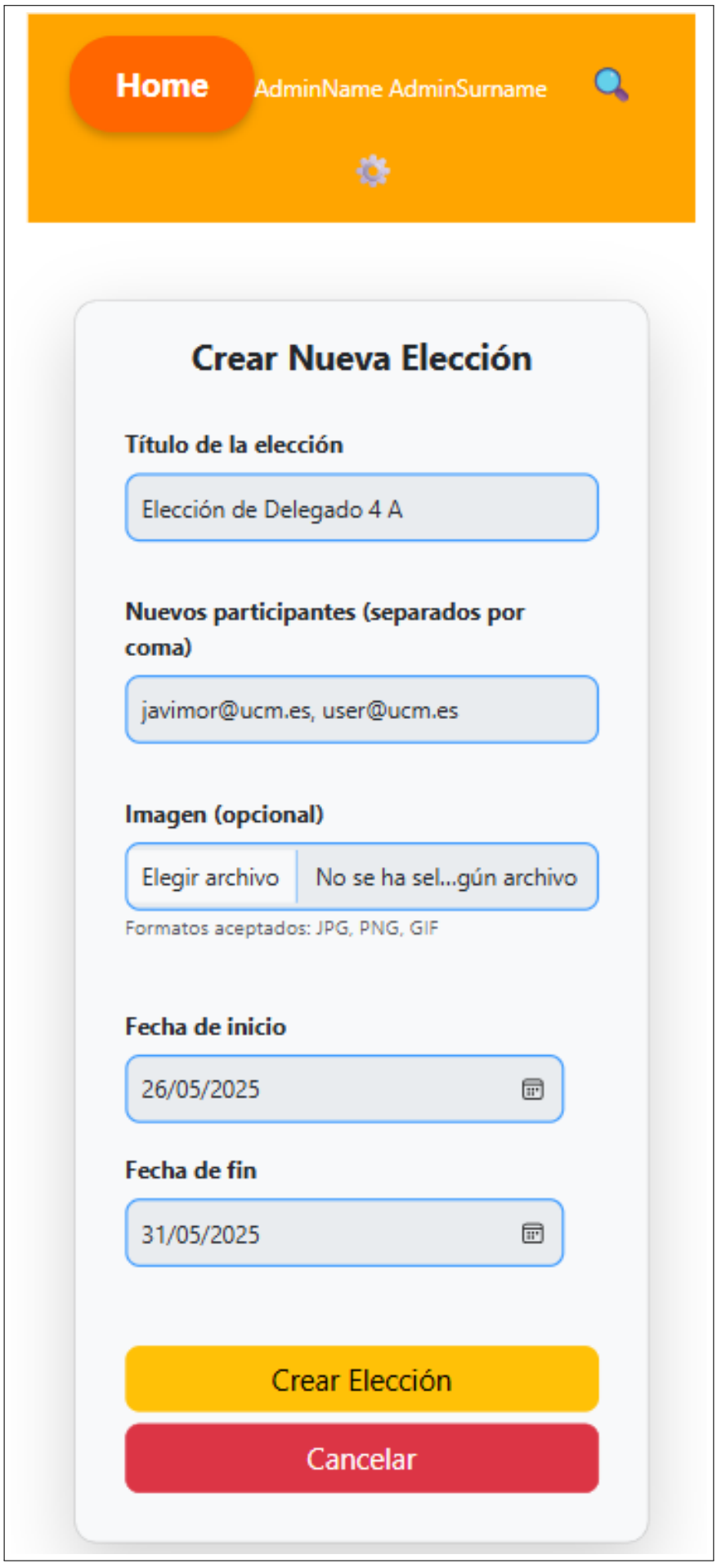


Figura 6.3: Crear Elección

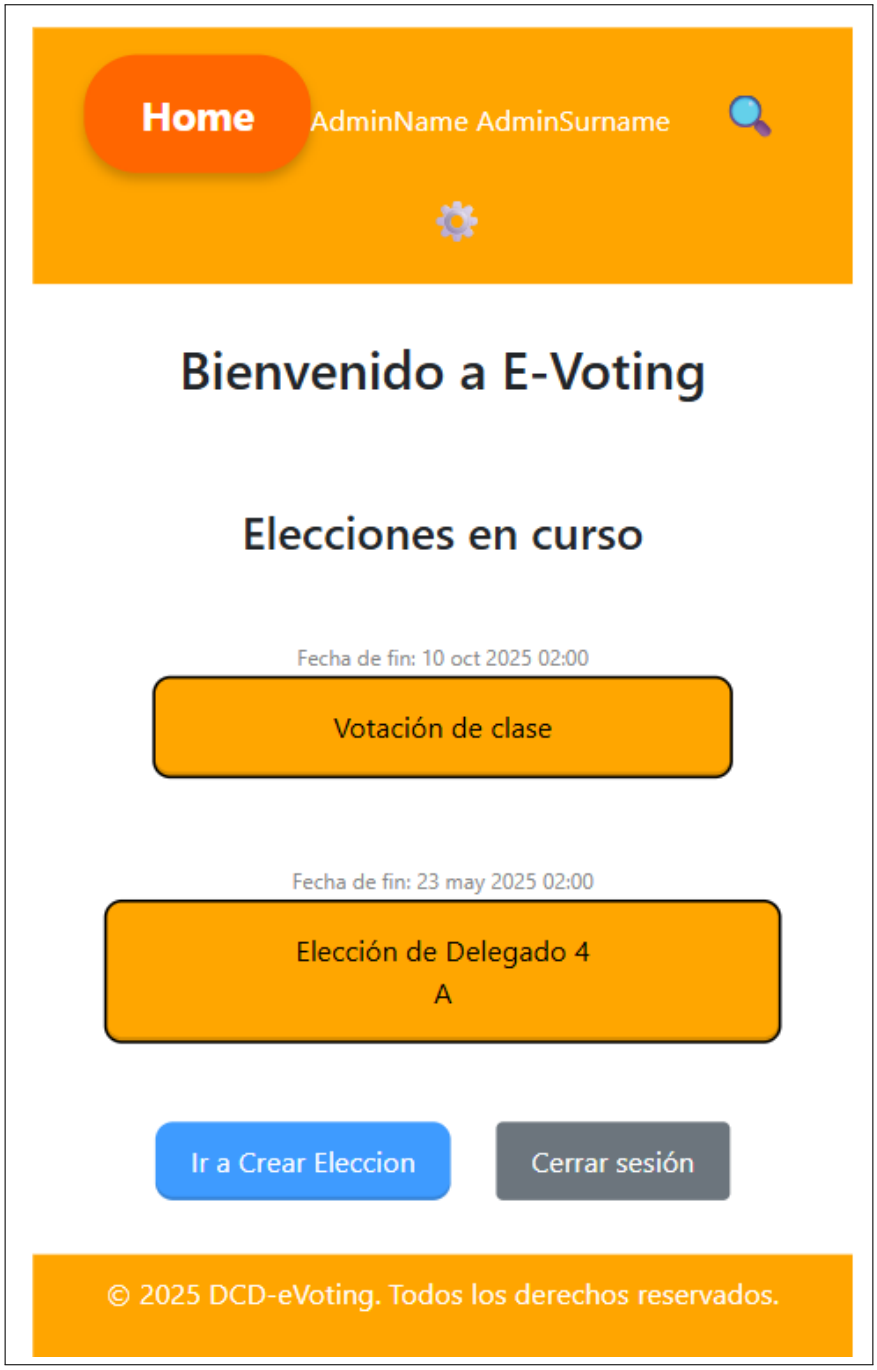


Figura 6.4: Página Principal

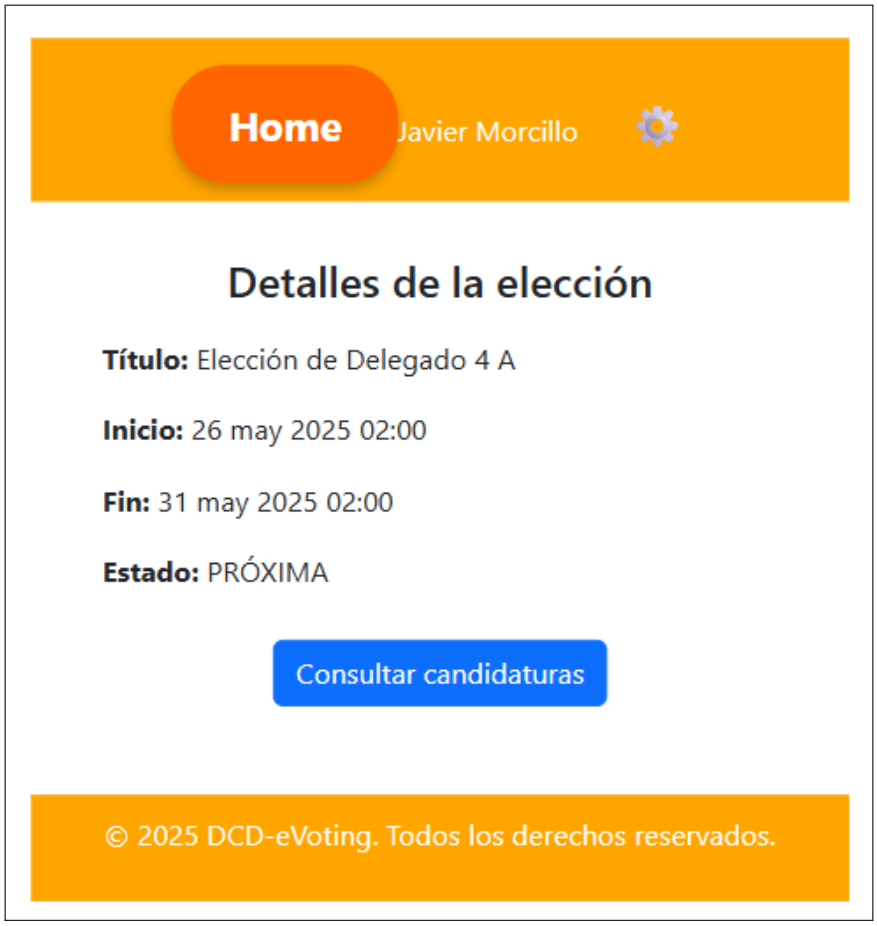


Figura 6.5: Consultar una Elección Inactiva

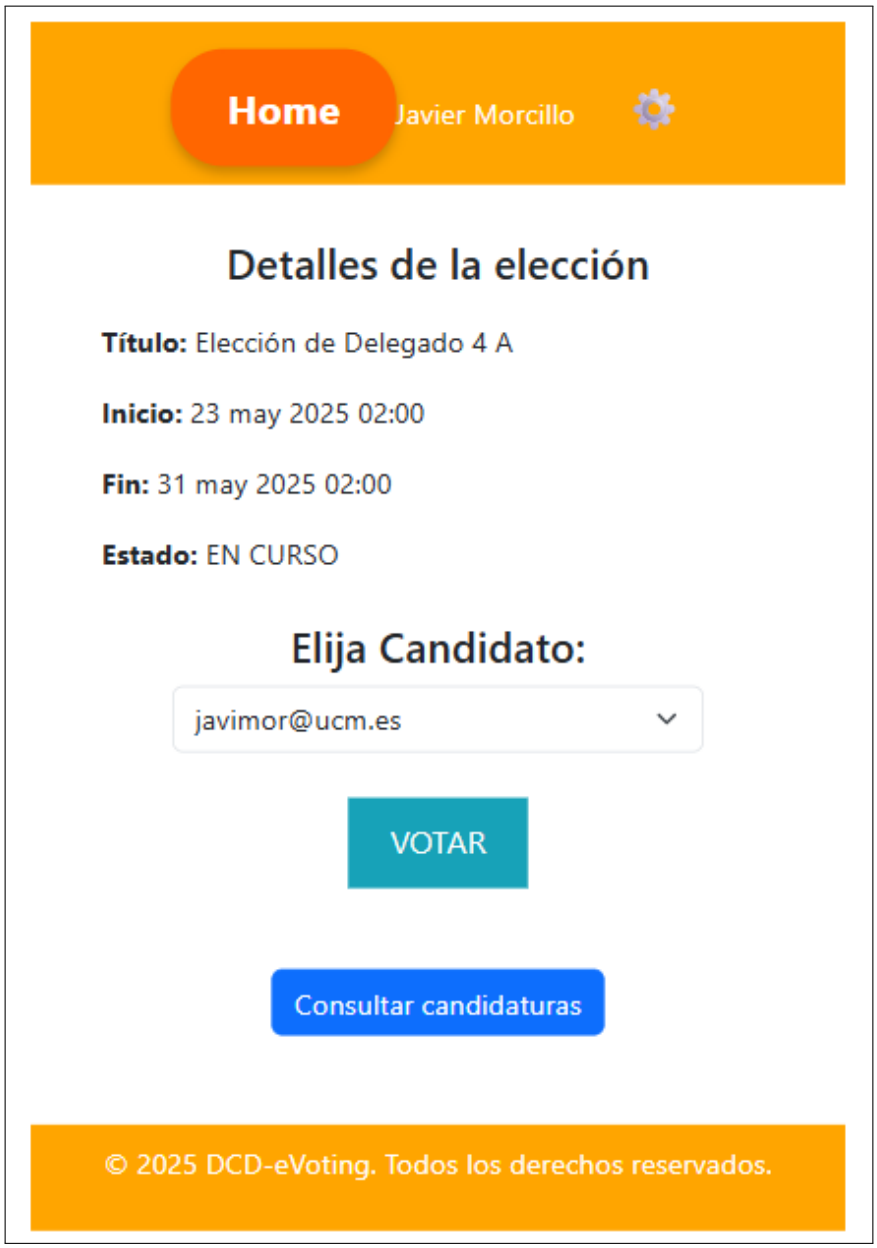


Figura 6.6: Consultar una Elección y Votar

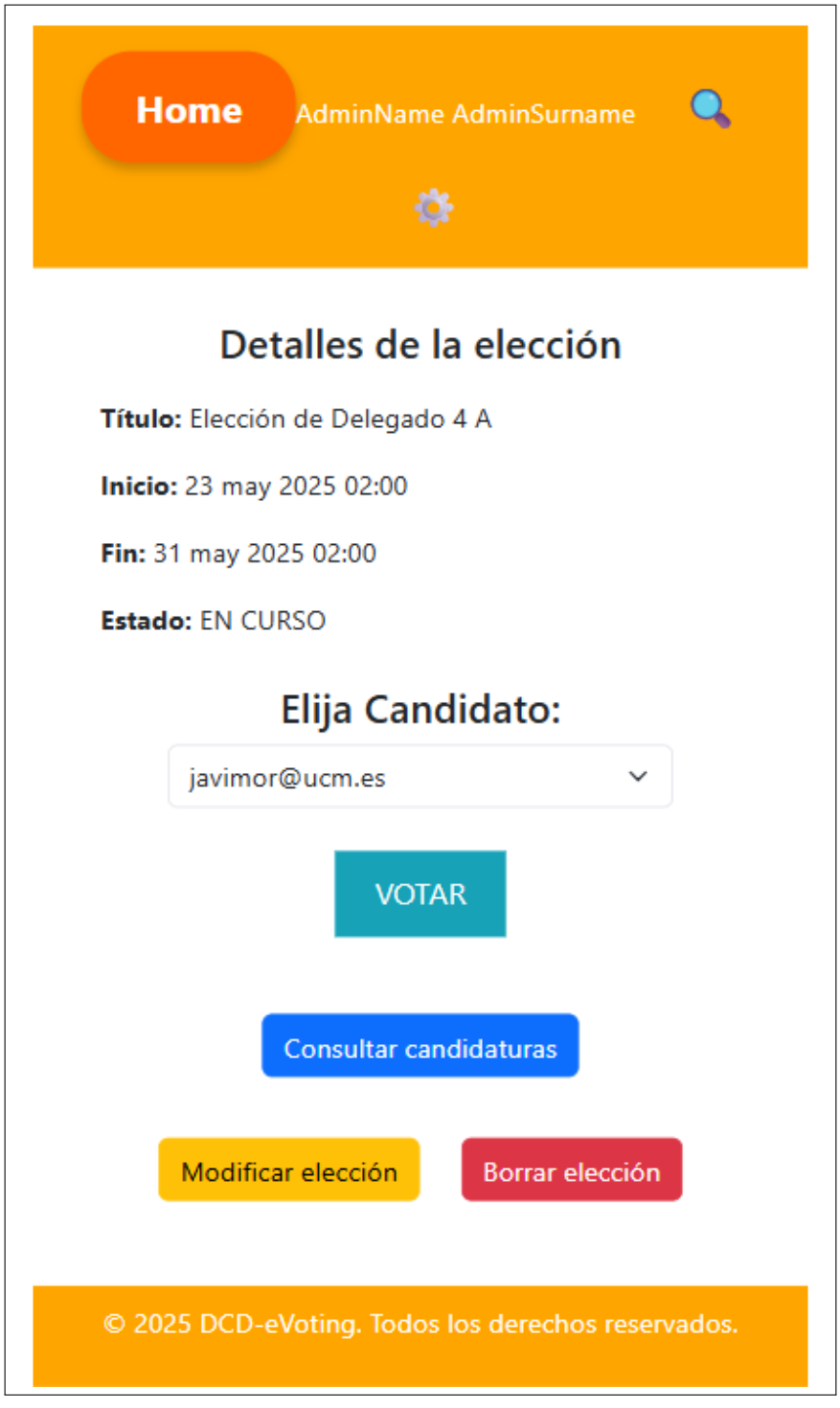


Figura 6.7: Consultar como Admin una Elección y Votar

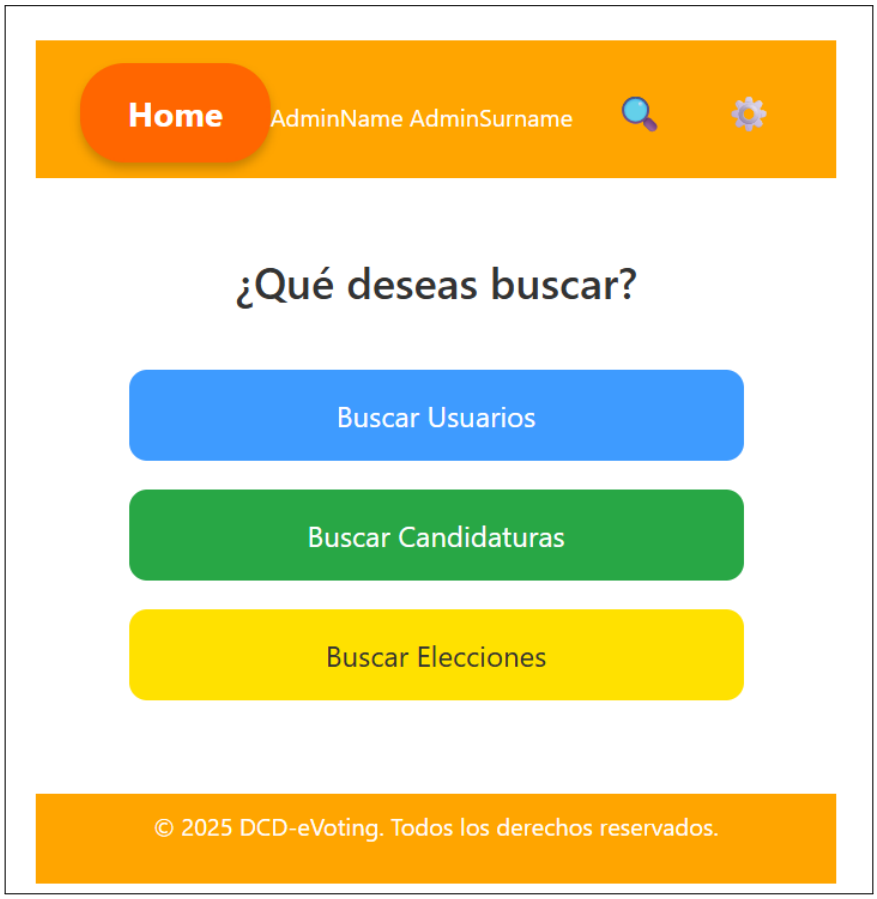


Figura 6.8: Buscar Usuarios, Candidaturas o Elecciones

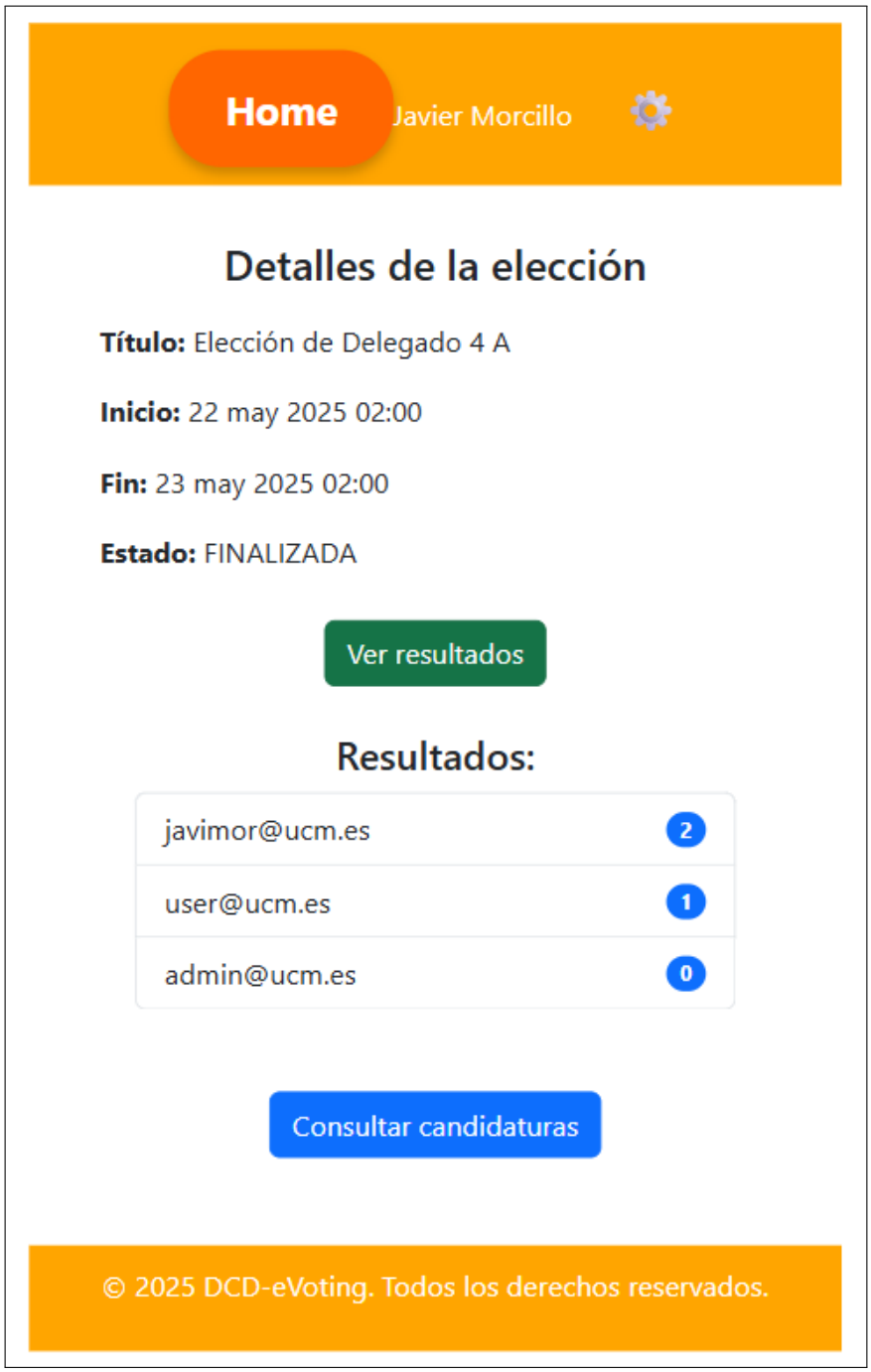


Figura 6.9: Resultados de una Elección Finalizada

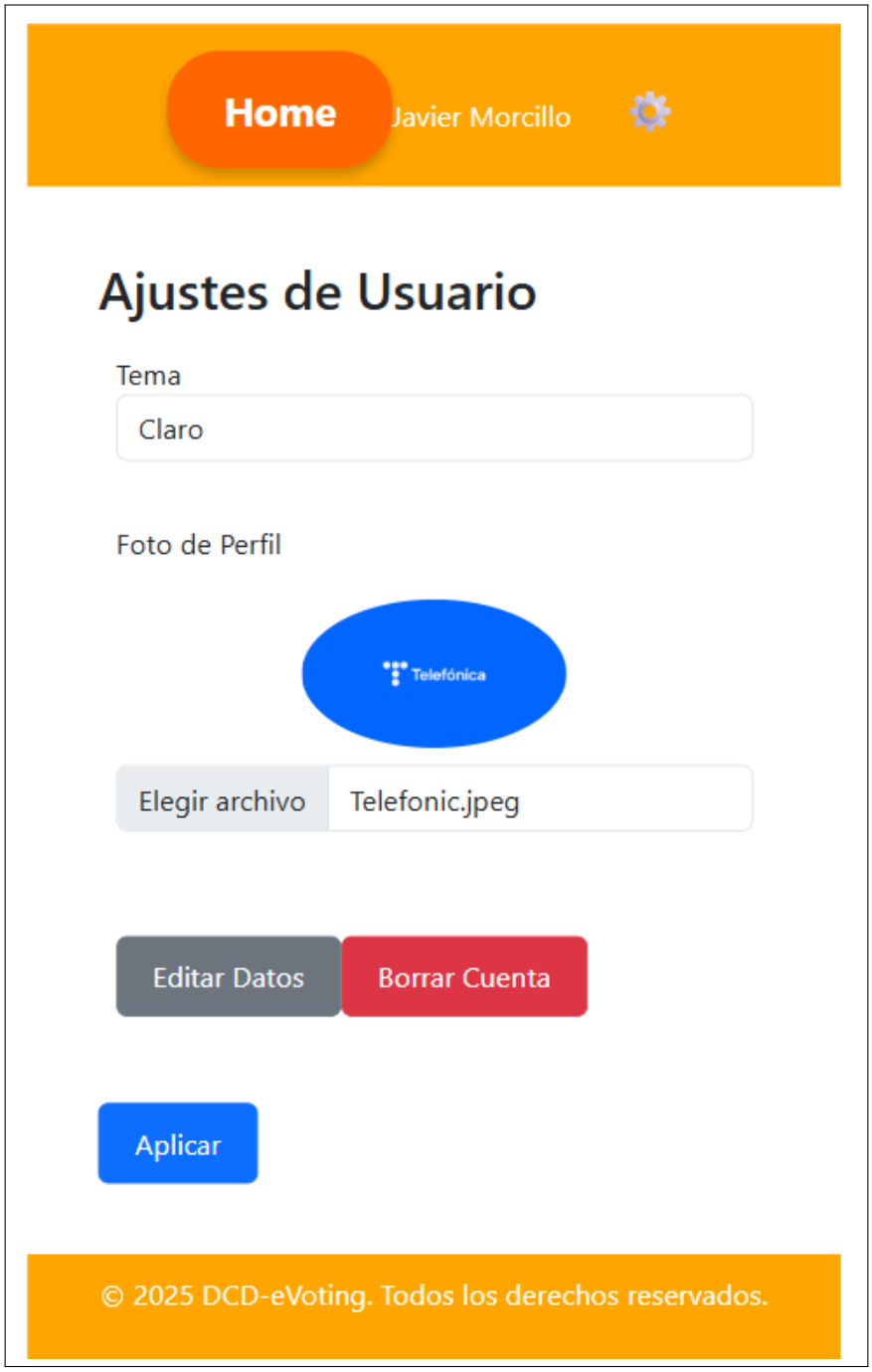


Figura 6.10: Ajustes de Usuario

Capítulo 7

Conclusiones y Trabajo Futuro

7.1. Conclusiones

Después de todo el proceso de desarrollo, podemos decir que hemos conseguido crear una plataforma completamente funcional que permite realizar elecciones universitarias con tres tipos de usuarios: el usuario base, el usuario con permisos para gestionar su propia elección y el administrador. El sistema funciona bien y está pensado para ser escalable en el futuro gracias a que lo hicimos de forma modular, separando claramente cada parte del proyecto.

Cuando empezamos, no teníamos experiencia previa con JavaScript (ni en el lado del cliente ni en el del servidor), y tampoco habíamos usado Vue.js ni Node.js. Bootstrap también era nuevo para nosotros. Solo conocíamos MySQL, pero tuvimos que aprender cómo usarlo dentro de un contenedor Docker y cómo conectarlo con el servidor.

Al ser un grupo de tres personas, nos organizamos para repartirnos el trabajo: cada uno se encargó de uno de los tres subsistemas principales (usuarios, candidaturas, elecciones) y además nos dividimos tareas comunes como montar la base de datos o hacer los estilos básicos con CSS. Esto nos obligó a planificarnos bien y a comunicarnos constantemente para que todo encajara correctamente al final.

Gracias a este proyecto hemos aprendido muchas cosas, desde cómo programar en JavaScript hasta cómo estructurar código para que sea más limpio y fácil de mantener. También hemos mejorado en el uso de bases de datos, documentación técnica y planificación de proyectos.

7.2. Limitaciones

En el desarrollo actual del sistema, algunos Requisitos No Funcionales no han podido implementarse debido a limitaciones de tiempo, recursos o cuestiones admi-

nistrativas. Estos requisitos se listan a continuación para su futura incorporación:

- **Seguridad** : 4.2.2.1, 4.2.2.1, 4.2.2.1, 4.2.2.1 y 4.2.2.1.
- **Mantenimiento** : 4.2.2.4.
- **Documentación** : 4.2.2.9 y 4.2.2.9.

7.3. Trabajo futuro

Aunque el sistema ya está en funcionamiento, las propuestas de mejora que tenemos pensadas para el futuro son las siguientes:

- Implementar un sistema de **doble factor de autenticación** para hacer el inicio de sesión más seguro.
- Mejorar el diseño visual con **CSS más personalizado** y no depender tanto de Bootstrap.
- Hacer más cómoda la interfaz para crear y modificar elecciones, especialmente la parte del **censo**, que ahora mismo no es muy intuitiva.
- Permitir a los usuarios **consultar su voto emitido** mediante un **comprobante o justificante**, sin comprometer el anonimato del voto.
- Incluir la opción de establecer la página en **otros idiomas** (**Español** e **Inglés**), uno por ser el idioma oficial del país y otro para estudiantes extranjeros.
- Establecer una medida del **progreso del usuario** para conocer el paso en el que se encuentra al realizar el **voto**.
- Incluir el funcionamiento de **añadir foto** a una Elección al crear o modificar.
- Inclusión de selección del **tipo de votación**, como el **Sistema a dos vueltas**. En este caso, si ningún candidato obtiene más del 50 % de los votos, los dos candidatos con más votos se enfrentan entre sí para decidir el delegado y subdelegado.

Chapter 8

Conclusions and Future Work

8.1. Conclusions

After completing the development process, we can say that we have successfully created a fully functional platform that allows university elections to be held with three types of users: the basic user, the user with permissions to manage their own election, and the administrator. The system works well and is designed to be scalable in the future, thanks to our modular approach, where each part of the project was clearly separated.

When we started, we had no prior experience with JavaScript (neither on the client side nor the server side), nor had we used Vue.js or Node.js before. Bootstrap was also new to us. We were only familiar with MySQL, but we had to learn how to use it within a Docker container and how to connect it with the server.

Since we were a group of three people, we organized ourselves to divide the work: each person took charge of one of the three main subsystems (users, candidacies, elections), and we also shared common tasks such as setting up the database and creating basic styles with CSS. This required us to plan carefully and communicate constantly to ensure everything fit together correctly in the end.

Thanks to this project, we have learned many things, from how to program in JavaScript to how to structure code to make it cleaner and easier to maintain. We have also improved our skills in database usage, technical documentation, and project planning.

8.2. Limitations

In the current development of the system, some Non-Functional Requirements could not be implemented due to time constraints, limited resources, or administrative issues. These requirements are listed below for future inclusion:

- **Security**: 4.2.2.1, 4.2.2.1, 4.2.2.1, 4.2.2.1, and 4.2.2.1.
- **Maintenance**: 4.2.2.4.
- **Documentation**: 4.2.2.9 and 4.2.2.9.

8.3. Future Work

Although the system is already functional, we have the following improvement proposals in mind for the future:

- Implement a **two-factor authentication system** to make the login process more secure.
- Improve the visual design with **more customized CSS** and reduce dependency on Bootstrap.
- Make the interface for creating and modifying elections more user-friendly, especially the **census** section, which is currently not very intuitive.
- Allow users to **verify their submitted vote** through a **receipt or confirmation**, without compromising vote anonymity.
- Include the option to set the platform in **multiple languages** (**Spanish** and **English**), one being the country's official language and the other for international students.
- Implement a measure of the **user's progress** to indicate which step they are at during the **voting process**.
- Add functionality to **attach a photo** to an Election when creating or modifying it.
- Include the ability to choose the **voting system**, such as the **Two-Round System**. In this case, if no candidate obtains more than 50% of the votes, the two candidates with the most votes compete against each other to decide the delegate and sub-delegate.

Bibliografía

[1] Docker Inc. (2025) Docker. [Online]. Available: https://docs.docker.com/

[2] Mozilla Contributors. (2025) Javascript — mdn web docs. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/JavaScript

[3] Wikipedia contributors. (2025) Diagrama de gantt — wikipedia, la enciclopedia libre. Última edición consultada el 18 de mayo de 2025. [Online]. Available: https://es.wikipedia.org/wiki/Diagrama_de_Gantt

[4] Helios Voting Project. (2025) Helios voting. [Online]. Available: https://heliosvoting.org

[5] Scytl. (2025) Scytl secure electronic voting solutions. [Online]. Available: https://www.scytl.com

[6] Simply Voting Inc. (2025) Simply voting: Online voting tool. [Online]. Available: https://www.simplyvoting.com/

[7] Meta Platforms, Inc. (2025) React – a javascript library for building user interfaces. [Online]. Available: https://reactjs.org/

[8] Evan You et al. (2025) Vue.js. [Online]. Available: https://vuejs.org

[9] Angular Team. (2025) Angular – one framework. mobile desktop. [Online]. Available: https://angular.io/

[10] The Bootstrap Authors. (2025) Bootstrap. [Online]. Available: https://getbootstrap.com

[11] Tailwind Labs. (2025) Tailwind css — a utility-first css framework. [Online]. Available: https://tailwindcss.com/

[12] Node.js Foundation. (2025) Node.js. [Online]. Available: https://nodejs.org

[13] VMware. (2025) Spring boot — create stand-alone, production-grade spring based applications. [Online]. Available: https://spring.io/projects/spring-boot

[14] Oracle Corporation. (2025) Java platform, standard edition documentation. [Online]. Available: https://docs.oracle.com/en/java/javase/

[15] Taylor Otwell. (2025) Laravel — the php framework for web artisans. [Online]. Available: https://laravel.com/

[16] The PHP Group. (2025) Php: Hypertext preprocessor. [Online]. Available: https://www.php.net/

[17] Oracle Corporation. (2025) Mysql :: The world's most popular open source database. [Online]. Available: https://www.mysql.com/

[18] PostgreSQL Global Development Group. (2025) Postgresql: The world's most advanced open source relational database. [Online]. Available: https://www.postgresql.org/

[19] SQLite Consortium. (2025) Sqlite home page. [Online]. Available: https://www.sqlite.org/

[20] Sequelize Contributors. (2025) Sequelize. [Online]. Available: https://sequelize.org

[21] Microsoft. (2025) Visual studio code. [Online]. Available: https://code.visualstudio.com

[22] JetBrains. (2025) Webstorm – the smartest javascript ide. [Online]. Available: https://www.jetbrains.com/webstorm/

[23] ——. (2025) Intellij idea – java ide for professional developers. [Online]. Available: https://www.jetbrains.com/idea/

[24] ——. (2025) Pycharm – python ide for professional developers. [Online]. Available: https://www.jetbrains.com/pycharm/

[25] Python Software Foundation. (2025) Python programming language. [Online]. Available: https://www.python.org/

[26] The Git Project. (2025) Git. [Online]. Available: https://git-scm.com

[27] GitHub, Inc. (2025) Github. [Online]. Available: https://github.com

[28] Apache Software Foundation. (2025) Apache subversion (svn) – version control system. [Online]. Available: https://subversion.apache.org/

[29] Perforce Software. (2025) Helix core — version control system. [Online]. Available: https://www.perforce.com/products/helix-core

[30] IETF. (2025) The transport layer security (tls) protocol version 1.3. [Online]. Available: https://datatracker.ietf.org/doc/html/rfc8446

[31] Parlamento Europeo y del Consejo de la Unión Europea. (2016) Reglamento (ue) 2016/679 del parlamento europeo y del consejo, de 27 de abril de 2016, relativo a la protección de las personas físicas en lo que respecta al tratamiento de datos personales y a la libre circulación de estos datos (reglamento general de protección de datos). [Online]. Available: https://eur-lex.europa.eu/legal-content/ES/TXT/?uri=CELEX%3A32016R0679

[32] Axios Authors. (2025) Axios: Promise based http client for the browser and node.js. [Online]. Available: https://axios-http.com/

[33] IETF. (2025) Json web token (jwt). [Online]. Available: https://datatracker.ietf.org/doc/html/rfc7519

Apéndice A

Flujo del Sistema y Maquetación

En esta sección se muestran todas las funcionalidades del usuario y su interacción por medio de las maquetas. El usuario siempre tendrá que iniciar sesión o, en caso de no tener una cuenta asignada, crear una nueva para poder acceder al resto de funcionalidades. Asimismo, puede crear una nueva contraseña.

Una vez iniciada la sesión del usuario, se redirigirá a la página principal del sistema donde puede:

- Acceder a una votación en curso.
- Acceder a una votación pasada.
- Acceder a una votación próxima.
- Ajustes.
- Redirigir a la página previa.

A.1. Flujo del Sistema

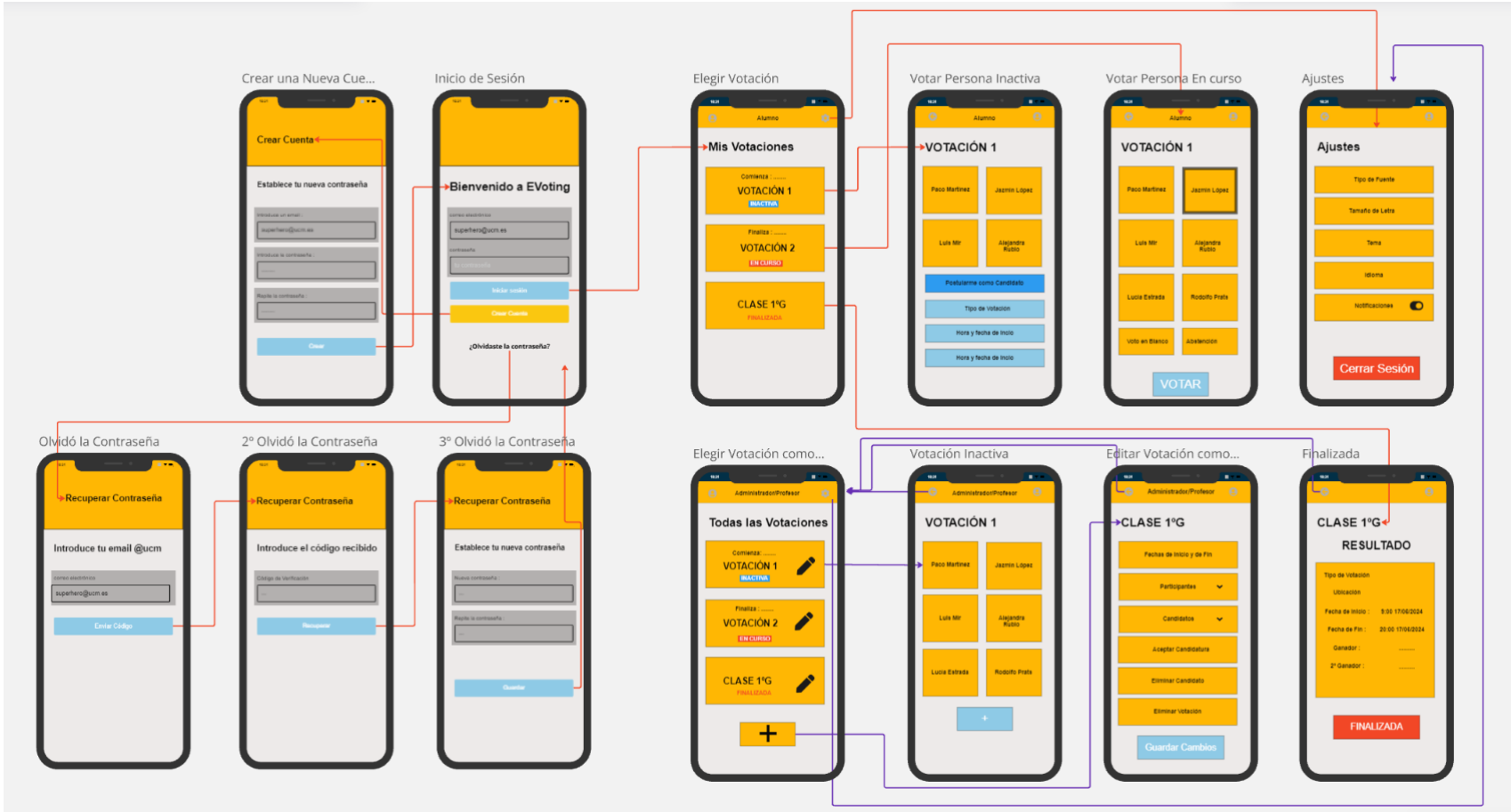


Figura A.1: Flujo entre los distintos tipos de usuarios del Sistema
Las flechas en rojo establecen el flujo de un alumno. Las flechas en morado establecen el flujo de un profesor/admin.
Para iniciar sesión, recuperar contraseña o ajustes, el tipo de usuario es irrelevante.

A.2. Maquetas

Figura A.2:

Esta maqueta representa la primera vista que tiene un usuario al abrir la aplicación, antes de cualquier otra gestión se debe autenticar. Si no tiene una cuenta asociada, debe realizar el formulario de creación de cuenta. Si ya tiene una cuenta, inicia sesión con su correo de la ucm y su contraseña.

Figura A.3:

Esta maqueta representa la creación de la cuenta. El correo electrónico debe tener el dominio @ucm.es y debe introducir la contraseña dos veces. Y las siguientes **Figura A.4**, **Figura A.5** y **Figura A.6** en realción a la secuencia establecida para el cambio de contraseña y recuperar la cuenta.

Figura A.7:

Una vez autenticado, un usuario normal accede a esta página principal. En esta tiene acceso a las votaciones en la que está registrado como participante, pudiendo ver la fecha en la que esta comienza, ha comenzado o comenzará. Al clicar en cualquiera de ellas, accede a los detalles de la votación sea cual sea su Estado. También, puede acceder a su perfil por medio del icono superior izquierdo o a los ajustes con el superior derecho.

Figura A.8: En el caso de clicar en una Votación Activa, el usuario tiene la opción de Votar. Por lo tanto, elige a su candidato y clica en el votón azul para registrar su voto.

Figura A.9: En el caso de clicar en una Votación Inactiva, el usuario puede ver detalles de la votación y los candidatos registrados hasta el momento pero no tiene la opción de votar hasta que no comience.

Figura A.10:

Opción para tanto Profesor como Usuario de personalizar la aplicación según quiera establecer los ajustes.

Figura A.11:

Esta es la vista de un Profesor al autenticarse. Es exactamente la misma que un usuario normal, salvo que él tiene la opción de modificar las Elecciones que él ha creado. Además, tiene acceso a su perfil y a los ajustes de la aplicación también.

Figura A.12:

Un profesor al clicar en una Elección Activa accede a los candidatos que ha aceptado y puede incluir más candidatos para que sean votados.

Figura A.13:

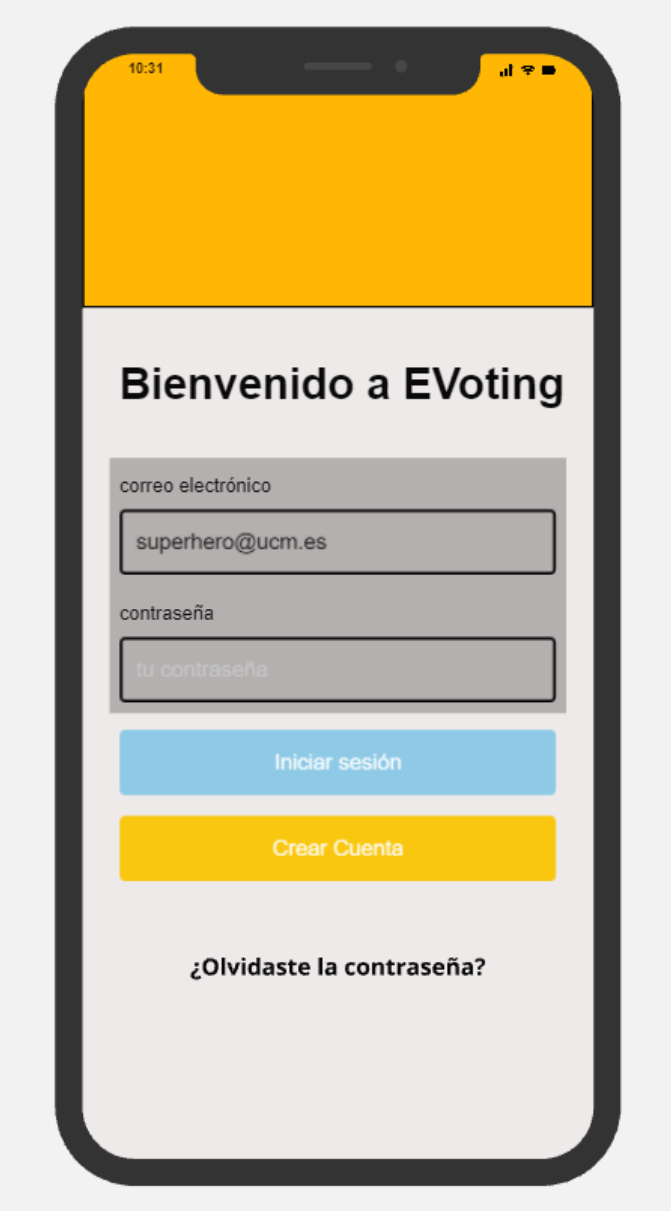
Un profesor al clicar en una Elección Inactiva accede a sus detalles y puede

modificar todos los datos de ella, incluso eliminarla.

Figura A.14:

Esta maqueta representa los resultados de una Elección finalizada, es la misma vista para un Profesor que para un Usuario. En ella se ven los resultados de la votación y principales datos de la misma.

Figura A.2: Iniciar Sesión

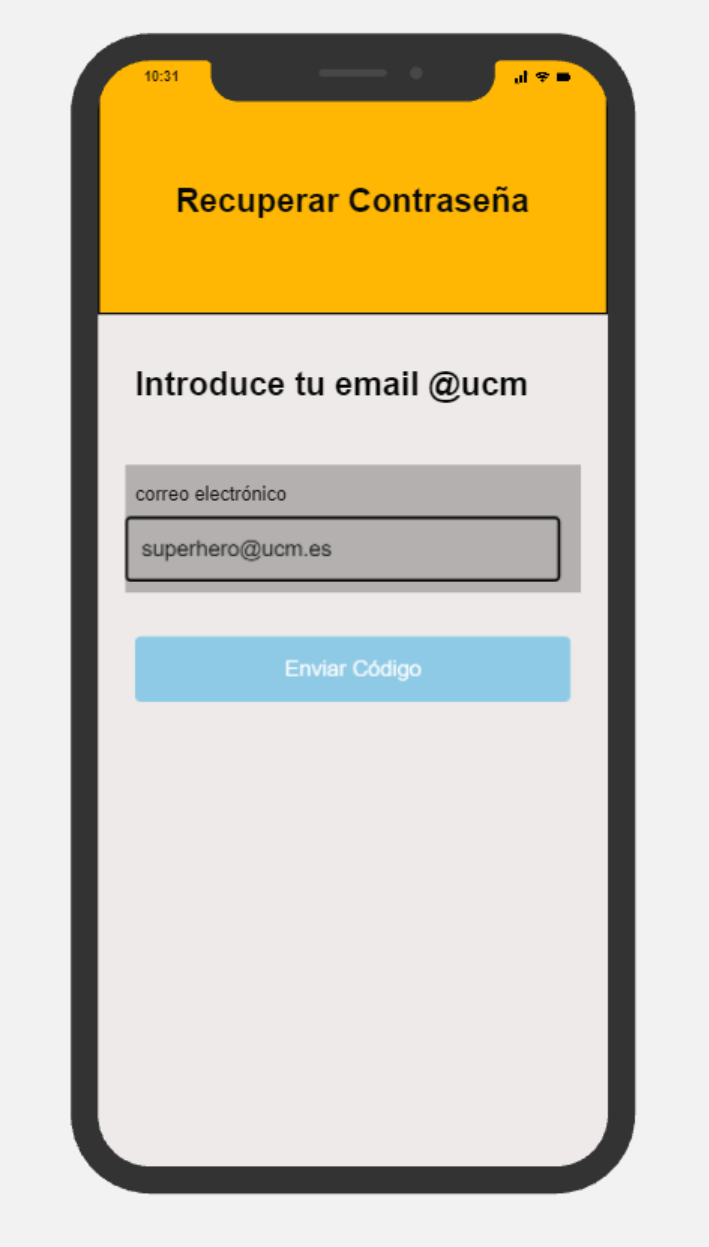


Pantalla para el inicio de sesión donde el usuario debe introducir su correo @ucm.es y su contraseña.

Figura A.3: Crear Cuenta

Pantalla para establecer la creación de una nueva cuenta.

Figura A.4: Cambiar Contraseña I

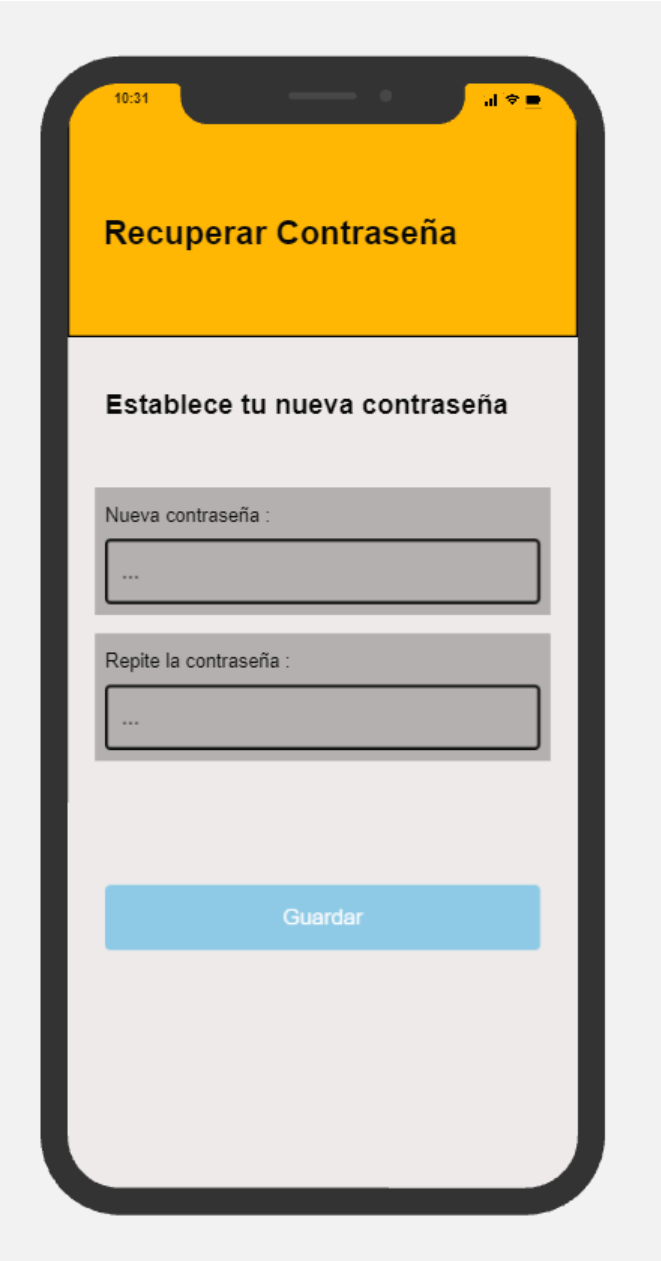


Pantalla relacionada con el primer paso para recuperar la contraseña.

Figura A.5: Cambiar Contraseña II

Pantalla para introducir el código recibido en el correo ucm.

Figura A.6: Cambiar Contraseña III

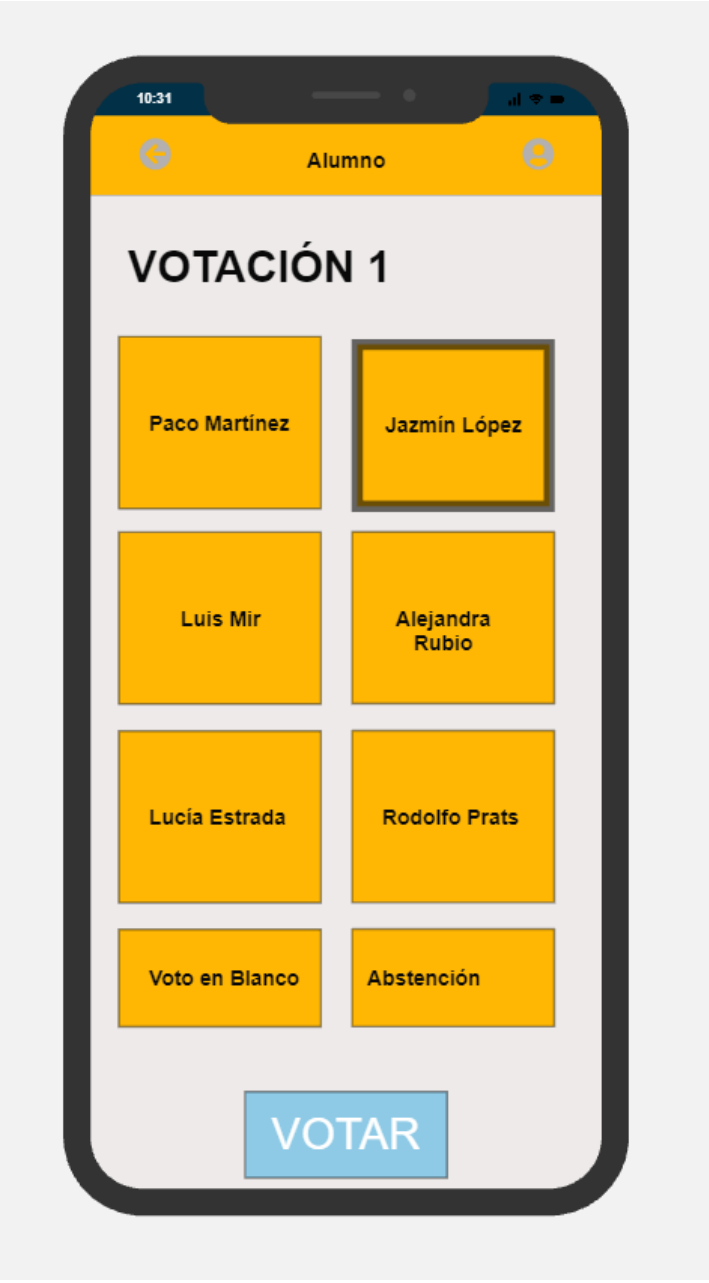


Pantalla con el último paso para cambiar la contraseña.

Figura A.7: Página Home

Pantalla principal donde se puede elegir la votación deseada.

Figura A.8: Votación Activa

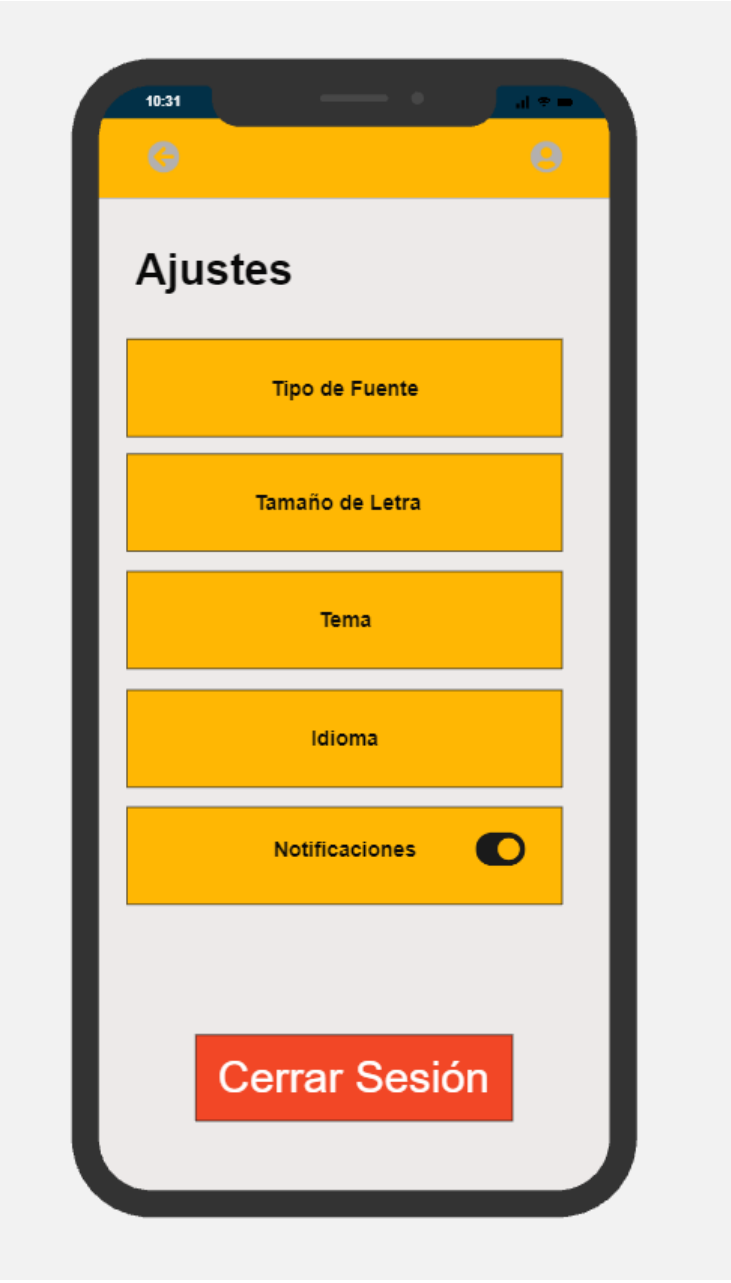


Pantalla donde podemos votar en una votación activa.

Figura A.9: Votación Inactiva

Pantalla donde podemos ver los datos relativos a una votación que aún no ha comenzado.

Figura A.10: Ajustes

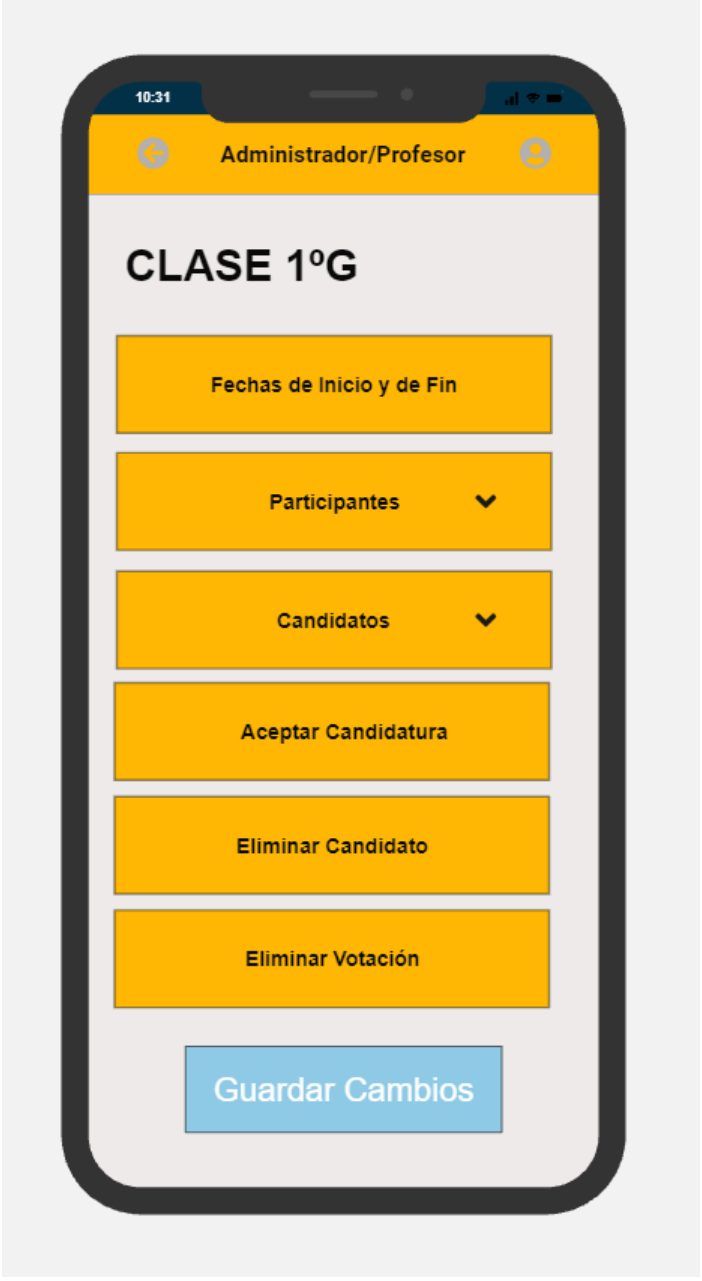


Pantalla para cambiar la configuración del sistema.

Figura A.11: Home Administrador

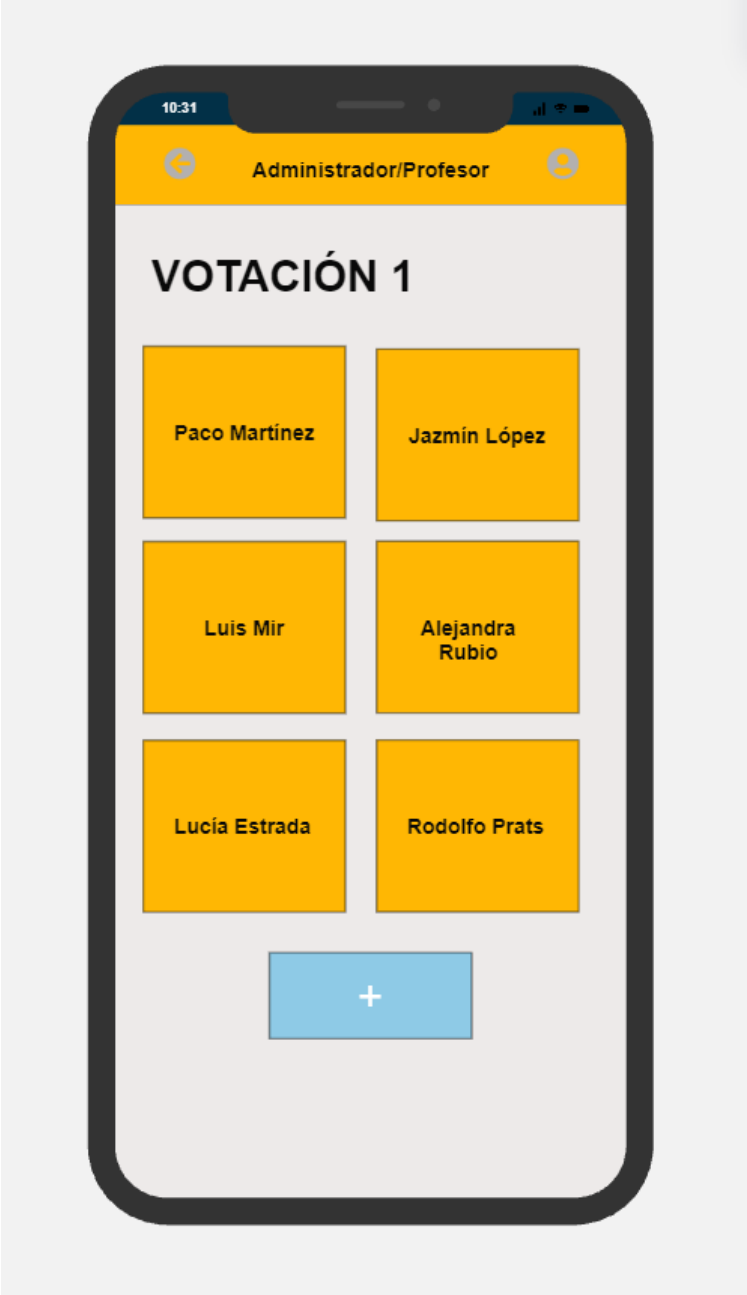
Pantalla de inicio para poder editar votaciones.

Figura A.12: Votación Activa para Administrador



Pantalla donde podemos editar una votación.

Figura A.13: Votación Inactiva para Administrador



Pantalla donde podemos editar los candidatos de una votación.

Figura A.14: Votación Finalizada

Datos e información de una votación finalizada.

Apéndice B

Manual de Usuario

La plataforma de votación online ha sido diseñada para ofrecer a los usuarios un entorno seguro donde puedan participar en elecciones, postularse como candidatos y gestionar sus cuentas de manera sencilla mediante una interfaz intuitiva. Desde la página principal, accesible a través del botón `Home`, el usuario podrá navegar por todas las secciones de la plataforma.

Para registrarse, es necesario pulsar el botón `Registrarse`, introducir el nombre, apellido, correo y contraseña. En caso de que el correo introducido ya esté registrado, será necesario utilizar una dirección de correo distinta.

El acceso a la plataforma se realiza pulsando el botón `Iniciar sesión` e introduciendo las credenciales correspondientes (correo y contraseña). Si el usuario aún no dispone de cuenta, puede registrarse directamente pulsando en `Regístrate aquí`. Las credenciales para acceder como usuario son user@ucm.es y contraseña 123

Una vez autenticado, el usuario podrá participar en las elecciones activas. Para ello, debe seleccionar la elección en la que desea emitir su voto, pulsar el botón `Votar`, marcar la opción deseada y confirmar el voto. Cabe destacar que solo se permite un voto por elección, y una vez emitido, no es posible modificarlo.

En caso de querer presentarse como candidato, el usuario deberá acceder a la sección de elecciones disponibles, seleccionar la elección correspondiente y pulsar el botón `Presentar candidatura`. A continuación, deberá completar los campos requeridos, como el nombre y el eslogan de la candidatura, y finalmente enviar la solicitud. La candidatura será visible en la elección únicamente si cumple con los requisitos establecidos por el administrador.

Por último, desde el apartado de configuración, accesible mediante el icono de engranaje, el usuario podrá personalizar su cuenta. Entre las opciones disponibles se encuentra el cambio de foto de perfil, que puede actualizarse subiendo un archivo desde el dispositivo, así como la posibilidad de alternar entre el tema claro y oscuro de la página. Si el usuario desea eliminar su cuenta, podrá hacerlo seleccionando la opción `Eliminar cuenta` y confirmando la acción cuando se le solicite.

Apéndice C

Manual de Desarrollador e Instalación

C.1. Guía de Instalación y Puesta en Marcha

1. Descarga el proyecto desde la página de **GitHub**.
2. Descarga e instala **Node.js**.
3. Descarga e instala **Docker Desktop** e inicia sesión. Si no dispones de cuenta, puedes utilizar las siguientes credenciales proporcionadas:
   - **Email:** dockerTFG@gmail.com
   - **Usuario:** TFGUsername
   - **Contraseña:** tfgpassword
4. Abre una consola de comandos y accede a la carpeta raíz del proyecto.
5. Ejecuta el siguiente comando en las carpetas `/votaciones/backend` y `/votaciones/frontend` para instalar las dependencias:

   ```
   npm i
   ```

6. Para iniciar la base de datos:
   - Accede a la carpeta `/votaciones/backend/DB` y ejecuta:

     ```
     docker-compose up -d
     ```

   - Después, entra en la carpeta `/DB/scripts` y ejecuta:

     ```
     node init_db.js
     ```

7. Para lanzar el **frontend**, ejecuta en la carpeta `/votaciones/frontend`:

   ```
   npm run dev
   ```

   Accede a la aplicación desde: `http://localhost:5173/`

8. Para lanzar el **backend**, ejecuta en la carpeta `/votaciones/backend`:

   ```
   node app.js
   ```

9. Recuerda que el cliente y el servidor deben ejecutarse en terminales separadas.

C.2. Manual de Administración del Sistema

Este manual está dirigido a los administradores de la plataforma de votación. Como administrador, se dispone de acceso a herramientas avanzadas de gestión de usuarios, elecciones y candidaturas, permitiendo así garantizar un proceso electoral limpio, transparente y organizado.

Para acceder como administrador, es necesario pulsar el botón `Iniciar sesión` e introducir las credenciales correspondientes que habilitan las funciones de administración. Las cedenciales para acceder como administrador son admin@ucm.es y contraseña 321

En cuanto a la gestión de usuarios, el administrador puede acceder a la pantalla de búsqueda pulsando el botón de la lupa. Desde esta sección, es posible localizar usuarios por nombre, apellido o correo electrónico. Una vez encontrados, se puede consultar información detallada, como el nombre, correo electrónico y actividad del usuario. Además, se ofrecen opciones para modificar sus datos, incluyendo nombre, apellido, correo electrónico y rol asignado, así como la posibilidad de eliminar usuarios de la plataforma.

Respecto a la gestión de elecciones, el procedimiento es similar. Pulsando en el icono de la lupa se accede a la búsqueda de elecciones, permitiendo localizar una elección concreta mediante su identificador o título. En caso de no introducir ningún criterio, se mostrará el listado completo de elecciones. Desde esta sección, el administrador puede consultar detalles como el título, fechas de inicio y fin, así como el estado actual de la elección. También es posible modificar estos datos, gestionar los participantes, cambiar la imagen asociada a la elección, eliminarla o consultar las candidaturas presentadas en la misma.

En lo que respecta a la gestión de candidaturas, la interfaz permite acceder a la pantalla de búsqueda mediante el icono de la lupa. Aquí se pueden localizar candidaturas por nombre, apellido, correo electrónico del candidato o eslogan de

la candidatura. Una vez encontrada la candidatura deseada, se pueden consultar sus detalles, modificar el título, los participantes, la imagen y las fechas, así como eliminarla o decidir si se acepta o rechaza. Además, es posible visualizar toda la información asociada al candidato.

Todas estas funciones han sido diseñadas para proporcionar al administrador un control total sobre el correcto funcionamiento del sistema de votación, asegurando en todo momento que el proceso electoral se desarrolle con las máximas garantías de integridad y transparencia.