



GRADO EN INGENIERÍA INFORMÁTICA

TRABAJO DE FINAL DE GRADO

**Desarrollo de una aplicación web para
realizar votaciones telemáticas seguras**

Realizado por:
Jordi MIRALLES COMINS

Supervisado por:
Vicente SANTOS MOCHOLÍ
Tutorizado por:
Manuel MOLLAR VILLANUEVA

Fecha de lectura: 14 de Julio de 2020
Curso académico 2019/2020

Resumen

El presente documento incluye la memoria del Trabajo de Final de Grado desarrollado durante la estancia en prácticas en la organización ERIA CONSULTING. Durante esta estancia, se ha llevado a cabo un proyecto cuyo objetivo consiste en desarrollar un sistema que permita realizar votaciones telemáticas de forma segura en grupos reducidos, a través del algoritmo de votación de Michael Merritt.

Este sistema se compone de diferentes módulos: una aplicación web progresiva, una base de datos para almacenar la información y una red privada virtual para controlar el acceso.

El proyecto se ha desarrollado siguiendo una metodología predictiva tradicional en la que las tareas son definidas y planificadas inicialmente. Además, se han identificado los requisitos que el sistema debe satisfacer, ayudando así a modelar y diseñar dicho sistema.

La implementación se ha realizado siguiendo técnicas y patrones de diseño de *software* que facilitan el desarrollo. Como resultado, se ha obtenido una aplicación web a través de la cual se pueden realizar votaciones en las que se garantiza la integridad y la confidencialidad de los votos.

Palabras clave

Votaciones telemáticas, ciberseguridad, algoritmo de Merritt, PWA, Angular, Node.js, RSA, VPN.

Keywords

Telematic voting, cybersecurity, Merritt's algorithm, PWA, Angular, Node.js, RSA, VPN.

Índice general

1. Introducción	15
1.1. Eria Consulting	15
1.2. Seguridad informática	16
1.3. Contexto y motivación del proyecto	17
1.4. Objetivos y alcance del proyecto	18
1.4.1. Objetivos generales del proyecto	18
1.4.2. Alcance funcional	18
1.4.3. Alcance organizativo	19
1.4.4. Alcance informático	19
1.5. Estructura de la memoria	20
2. Descripción del proyecto	21
2.1. Situación inicial	21
2.2. Descripción del sistema	22
2.2.1. Aplicación web progresiva	22
2.2.2. Protocolo de votación	23
2.2.3. Servidores	24
2.3. Tecnologías y herramientas utilizadas	25
2.3.1. Gestión y documentación	25
2.3.2. Estándares y algoritmos	26
2.3.3. Cliente web	29
2.3.4. Servidores	31
2.3.5. Herramientas de desarrollo	33

3. Planificación del proyecto	35
3.1. Metodología	35
3.2. Análisis y gestión de riesgos	36
3.3. Planificación temporal	37
3.3.1. Tareas y actividades	37
3.3.2. Estructura de Desglose del Trabajo	38
3.3.3. Distribución temporal	38
3.3.4. Diagrama de Gantt	40
3.4. Estimación de recursos y costes del proyecto	42
3.4.1. Costes <i>hardware</i>	42
3.4.2. Costes <i>software</i>	42
3.4.3. Costes humanos	43
3.5. Seguimiento del proyecto	44
4. Análisis del sistema	47
4.1. Actores	47
4.2. Diagrama de casos de uso	47
4.3. Requisitos funcionales	48
4.4. Requisitos de datos	55
4.5. Diagrama de clases	58
4.6. Diagramas de actividades	59
5. Diseño del sistema	61
5.1. Diseño de la arquitectura del sistema	61
5.2. Diseño de la base de datos	62
5.2.1. Diseño conceptual	62
5.2.2. Diseño lógico	62

5.2.3.	Diseño físico	64
5.3.	Diseño de la interfaz de usuario	66
5.3.1.	<i>Sitemap</i>	66
5.3.2.	Guía de estilo	66
5.3.3.	Prototipos	68
6.	Implementación del sistema	71
6.1.	Patrones diseño	71
6.2.	Implementación de servicios	73
6.2.1.	Red Privada Virtual	73
6.2.2.	Servidor de Nombres de Dominio	74
6.2.3.	Base de datos	75
6.3.	Implementación del <i>backend</i>	76
6.4.	Implementación del <i>frontend</i>	82
6.4.1.	Módulo de inicio de sesión	85
6.4.2.	Módulo de votaciones	85
6.4.3.	Módulo de usuarios	90
6.4.4.	Módulo para los procesos de votación	92
6.5.	Implementación del protocolo de votación	94
6.5.1.	Proceso de cifrado	94
6.5.2.	Proceso de envíos	96
7.	Verificación y validación del sistema	99
7.1.	Verificación	99
7.2.	Validación	102
7.2.1.	Validación de requisitos	102
7.2.2.	Validación de interfaces	103

7.2.3. Validación de la seguridad	105
8. Conclusiones y mejoras	107
8.1. Conclusiones	107
8.1.1. Conclusiones técnicas	107
8.1.2. Conclusiones personales	108
8.2. Posibles mejoras	108
Bibliografía	109

Índice de figuras

1.1. Logotipo de la entidad Eria Consulting [1].	15
1.2. Alcance informático del proyecto.	19
2.1. Esquema de la criptografía asimétrica o de llave pública.	23
2.2. Tablero del proyecto en Trello.	25
2.3. Control de versiones del proyecto en GitHub.	26
2.4. Ejemplos de mensajes en formato JSON usados en el sistema.	27
2.5. Ejemplo de <i>token</i> JWT generado por el sistema.	28
2.6. Esquema de la criptografía simétrica o de clave privada	29
2.7. Tecnologías utilizadas para el desarrollo del cliente web.	31
2.8. Tecnologías utilizadas para el desarrollo de los servidores.	33
2.9. Ejemplo de entradas de la base de datos en phpMyAdmin.	34
3.1. Esquema de la metodología en cascada.	36
3.2. EDT del proyecto.	38
3.3. Diagrama de Gantt del proyecto.	41
3.4. Diagrama de Gantt de seguimiento del proyecto.	46
4.1. Diagrama de casos de uso del sistema.	48
4.2. Diagrama de clases del sistema.	58
4.3. Diagrama de actividades - Acceso al sistema.	59
4.4. Diagrama de actividades - Proceso de votación.	60
4.5. Diagrama de actividades - Nueva votación/votante.	60
5.1. Diseño de la arquitectura del sistema.	61

5.2.	Diseño conceptual de la base de datos.	62
5.3.	Diseño lógico de la base de datos.	62
5.4.	<i>Sitemap</i> de la aplicación web.	66
5.5.	Guía de estilos - Paleta de colores.	67
5.6.	Guía de estilos - Fuente y tamaños.	67
5.7.	Guía de estilos - Iconos de Font Awesome.	68
5.8.	Guía de estilos - Logotipo de la aplicación.	68
5.9.	Guía de estilos - Elementos de Angular Material.	68
5.10.	Prototipos - Panel principal y Nuevo Votante.	69
5.11.	Prototipos - Listado usuarios y Perfil.	69
5.12.	Prototipos - Listado votaciones, Detalles votación, Votar y Resultados.	70
6.1.	Patrón MVC en el <i>backend</i>	72
6.2.	Patrón MVC en el <i>frontend</i>	72
6.3.	Estructura de los directorios del proyecto en el <i>backend</i>	76
6.4.	Esquema de los servidores web del proyecto.	77
6.5.	Estructura de los ficheros para la gestión de <i>sockets</i>	78
6.6.	Ejemplo de los correos enviados en el proyecto.	79
6.7.	Ejemplo de notificación recibida por un votante.	80
6.8.	Estructura de los directorios del proyecto en el <i>frontend</i>	82
6.9.	Estructura de los componentes en Angular.	83
6.10.	Estructura de los servicios en Angular.	84
6.11.	Interfaces - Inicio de sesión y panel principal.	85
6.12.	Interfaces - Creación de una nueva votación.	86
6.13.	Interfaces - Listado de votaciones.	87
6.14.	Interfaces - Detalles de una votación.	88

6.15. Interfaces - Modificar los datos de una votación.	89
6.16. Interfaces - Registrar un nuevo votante.	90
6.17. Interfaces - Gestionar los votantes del sistema.	91
6.18. Interfaces - Gestionar el perfil de un usuario.	91
6.19. Interfaces - Iniciar una votación.	92
6.20. Interfaces - Proceso de votación.	93
6.21. Interfaces - Resultados de una votación.	93
6.22. Esquema de la primera fase del proceso de envío de votos.	96
6.23. Esquema de la segunda fase del proceso de envío de votos.	97
6.24. Esquema de la tercera fase del proceso de envío de votos.	98
7.1. Ejemplo de uso de la consola de Google Developers.	100
7.2. Ejemplo de uso de la sección <i>Application</i> de Google Developers.	101
7.3. Resultado de la auditoría Lighthouse sobre la PWA implementada.	101
7.4. Interfaces - Validadores de campos.	104
7.5. Interfaces - Mensajes de error durante la votación.	106
7.6. Interfaces - Mensajes de error globales del sistema.	106

Índice de Tablas

1.1. Actividades de la entidad Eria Consulting [2].	16
2.1. Relación entre principios CRUD y métodos HTTP.	27
2.2. Bibliotecas utilizadas en Node.js	32
3.1. Distribución temporal de las tareas del proyecto.	39
3.2. Calendario del proyecto.	40
3.3. Distribución temporal de las tareas de documentación del proyecto.	40
3.4. Resumen de costes <i>hardware</i>	42
3.5. Resumen de costes <i>software</i>	43
3.6. Resumen de costes humanos	43
3.7. Resumen de costes adicionales	44
3.8. Resumen de costes del proyecto	44
3.9. Cambios en la distribución temporal del proyecto.	45
4.1. Actores principales del sistema.	47
4.2. Resumen de los casos de uso del sistema.	48
4.3. Especificación CU-01 - Crear o modificar una votación.	49
4.4. Especificación CU-02 - Elegir votantes.	50
4.5. Especificación CU-03 - Iniciar votación.	51
4.6. Especificación CU-04 - Registrar votante.	52
4.7. Especificación CU-05 - Votar.	53
4.8. Especificación CU-06 - Modificar votante.	54
4.9. Especificación CU-07 - Listar y visualizar votaciones.	55
4.10. Resumen de los requisitos de datos del sistema.	55

4.11. Especificación RD-01 - Votación.	56
4.12. Especificación RD-02 - Votante.	57
4.13. Especificación RD-03 - Administrador de votaciones.	57

Índice de Códigos Fuente

5.1. Diseño físico - Usuario	64
5.2. Diseño físico - Administrador	64
5.3. Diseño físico - Votante	64
5.4. Diseño físico - Votación	65
5.5. Diseño físico - Opción	65
5.6. Diseño físico - Participa	65
6.1. Configuración del servidor OpenVPN.	73
6.2. Configuración de las zonas DNS.	75
6.3. Esquema para la creación de <i>websockets</i>	78
6.4. Esquema para la creación de <i>tokens</i> JWT.	81
6.5. Esquema para la verificación de <i>tokens</i> JWT.	81
6.6. Implementación del proceso de cifrado.	95

Capítulo 1

Introducción

Contenidos

1.1. Eria Consulting	15
1.2. Seguridad informática	16
1.3. Contexto y motivación del proyecto	17
1.4. Objetivos y alcance del proyecto	18
1.4.1. Objetivos generales del proyecto	18
1.4.2. Alcance funcional	18
1.4.3. Alcance organizativo	19
1.4.4. Alcance informático	19
1.5. Estructura de la memoria	20

El presente capítulo tiene como objetivo situar en contexto el proyecto realizado como Trabajo de Final de Grado (TFG), así como la motivación que ha llevado a su desarrollo durante la estancia en prácticas. También se detallan los objetivos del proyecto y la estructura seguida en el presente documento.

1.1. Eria Consulting

Este proyecto se ha desarrollado en la empresa AWEN ERIA GROUP S.L., también conocida como ERIA CONSULTING, cuyo logo se puede ver en la figura 1.1. La estancia en prácticas se ha desarrollado en sus oficinas, situadas en el Parque Científico, Tecnológico y Empresarial de la Universitat Jaume I de Castellón de la Plana.



Figura 1.1: Logotipo de la entidad Eria Consulting [1].

Eria Consulting surgió en 2010 por la necesidad de aumentar la concienciación de los programadores en el ámbito de la ciberseguridad. Esta empresa está especializada en consultoría informática y de seguridad, el desarrollo de *software* y la gestión de proyectos. Sus actividades abarcan un gran abanico de soluciones como las que se observan en la tabla 1.1.







	Mantenimiento informático		Desarrollo de <i>software</i>
	Diseño, desarrollo y alojamiento de webs		Desarrollo de <i>hardware</i>
	Consultoría de seguridad y peritaje informático		Publicidad, marketing y organización de eventos

Tabla 1.1: Actividades de la entidad Eria Consulting [2].

1.2. Seguridad informática

La seguridad informática o ciberseguridad [3] es el área de la informática que se encarga de proteger los sistemas de información y la información que estos contienen.

En la actualidad, las organizaciones pueden sufrir incontables tipos de ataques y amenazas. Estos pueden afectar a múltiples ámbitos de los sistemas informáticos, desde el *software* y las aplicaciones al *hardware* y las comunicaciones. Por ello, las organizaciones deben realizar un esfuerzo por mejorar sus niveles de seguridad y aumentar así la protección de sus activos.

Para comprender el funcionamiento de la seguridad informática, es necesario conocer los cuatro pilares sobre los que esta se sustenta:

- **Confidencialidad:** Tal como es descrito por la ISO (Organización Internacional para la Estandarización) en el estándar ISO/IEC 27000 [4]: “La confidencialidad es la propiedad que garantiza que la información no se ponga a disposición o se revele a personas, entidades o procesos no autorizados”.
- **Integridad:** Tal como es descrito por la ISO en el estándar ISO/IEC 27000 [4]: “La integridad es la propiedad que garantiza el mantenimiento de la exactitud y completitud de la información”.
- **Autenticidad:** Tal como es descrito por la ISO en el estándar ISO/IEC 27000 [4]: “La autenticidad es la propiedad que garantiza que una entidad es lo que afirma ser”.
- **Disponibilidad:** Tal como es descrito por la ISO en el estándar ISO/IEC 27000 [4]: “La disponibilidad es la propiedad que garantiza que un recurso es accesible y utilizable a petición de una entidad autorizada”.

1.3. Contexto y motivación del proyecto

Inicialmente, este proyecto no estaba contemplado dentro de la entidad y fue propuesto por mi parte. El proyecto fue aceptado por el interés de Eria Consulting en ofrecer este tipo de producto a sus clientes. Con este producto, los clientes pueden mejorar sus procesos de votación internos y reducir el tiempo dedicado a la gestión y el control de estas.

Cabe destacar que este proyecto ha sido financiado por una beca¹ del Ayuntamiento de Castellón para el fomento del talento y la formación de los estudiantes. Además, la beca se ha desarrollado con la supervisión del grupo de investigación IRIS² de la Universitat Jaume I.

Para comprender correctamente el contexto y la motivación del proyecto, es necesario identificar y distinguir los diferentes sistemas de votación digital:

- **Sistemas de voto electrónico:** Tal como describe Carracedo J. [5], los sistemas de voto electrónico son aquellos que “basándose en la forma de operar del método convencional mediante papeletas, sustituyen alguno de sus elementos físicos y procedimientos manuales por algún tipo de sistema o de proceso electrónico”.
- **Sistemas de voto telemático:** Tal como describe Carracedo J. [5], los sistemas de voto telemático “hacen uso de las redes telemáticas y de agentes telemáticos a los que se accede de forma remota”

Por tanto, la principal motivación de este proyecto es el desarrollo de un sistema de votación telemático en el que se garanticen los pilares fundamentales de la seguridad informática:

- **Confidencialidad:** La confidencialidad garantiza que nadie puede conocer el voto del resto de votantes.
- **Integridad:** La integridad garantiza que ningún voto se ve modificado o alterado.
- **Autenticidad:** La autenticidad garantiza que el votante es quién dice ser y no alguien que usurpa su identidad. Además, también garantiza que solo los votantes autorizados pueden participar en la votación.
- **Disponibilidad:** La disponibilidad garantiza que los votantes autorizados tienen acceso y pueden participar en las votaciones.

Además, este proyecto incluye las siguientes características, necesarias en cualquier sistema de votación telemático, tal como describe Carracedo J. [5]:

- **Singularidad:** Cada participante solo puede realizar un voto.
- **Verificabilidad:** Cada participante debe poder comprobar que su voto se ha incluido.
- **Facilidad de uso y rapidez:** Cada participante puede utilizar el sistema de forma sencilla y rápida, sin la necesidad de conocimientos especiales.

¹ **Becas Talento Ciudad de Castellón**, www.castello.es/archivos/1432/Becas_Talento_CS_2019_Anuncio_cas.pdf

² **IRIS** (Integración y Re-Ingeniería de Sistemas), www.iris.uji.es/es1024/index.php

1.4. Objetivos y alcance del proyecto

1.4.1. Objetivos generales del proyecto

El principal objetivo de este proyecto es crear una aplicación móvil a través de la cual se pueda realizar una votación telemática en un grupo reducido, garantizando la seguridad del proceso.

El objetivo principal se puede desglosar en los siguientes subobjetivos:

- Implementar un algoritmo de votación suficientemente probado en la literatura de criptografía que garantice las propiedades deseadas de un sistema de votación.
- Acceder de forma controlada a la votación mediante una red privada virtual.
- Visualizar y participar en las votaciones a través de una aplicación móvil.
- Permitir a los administradores gestionar y crear nuevas votaciones a través de una aplicación móvil.
- Garantizar la seguridad de las votaciones a través de los módulos de cifrado y envío de datos.
- Almacenar las votaciones y los resultados en una base de datos.

En cuanto al alcance del sistema, este se puede definir desde tres puntos de vista, como se describe a continuación.

1.4.2. Alcance funcional

La aplicación incluye la siguiente funcionalidad:

- Permite a los administradores crear, modificar, gestionar e iniciar todas las votaciones.
- Permite a los administradores registrar, modificar, gestionar e incluir en votaciones a todos los votantes.
- Permite tanto a administradores como a votantes listar y visualizar todas aquellas votaciones a las cuales tengan permiso de acceso.
- Permite a los votantes participar en aquellas votaciones en las que estén incluidos, así como visualizar sus resultados.
- Permite realizar votaciones telemáticas garantizando la seguridad, la confidencialidad y la integridad de los votos y la autenticidad de los votantes.

1.4.3. Alcance organizativo

Al tratarse de una entidad de tamaño reducido, el proyecto no se realiza en ningún ámbito específico dentro de esta.

1.4.4. Alcance informático

El alcance informático de este sistema incluye el desarrollo de múltiples módulos o sistemas, tal como se observa en la figura 1.2:

- Desarrollo de la PWA (Aplicación Web Progresiva).
- Implementación de los algoritmos de cifrado y envío de datos para la votación.
- Desarrollo de una API (Interfaz de Programación de Aplicaciones) para alojar la aplicación web, resolver las peticiones a la base de datos, enviar notificaciones electrónicas y gestionar las conexiones mediante *sockets*.
- Diseño e implementación de una base de datos.
- Diseño e implementación de una VPN (Red Privada Virtual).
- Diseño e implementación de un DNS (Servidor de Nombres de Dominio).

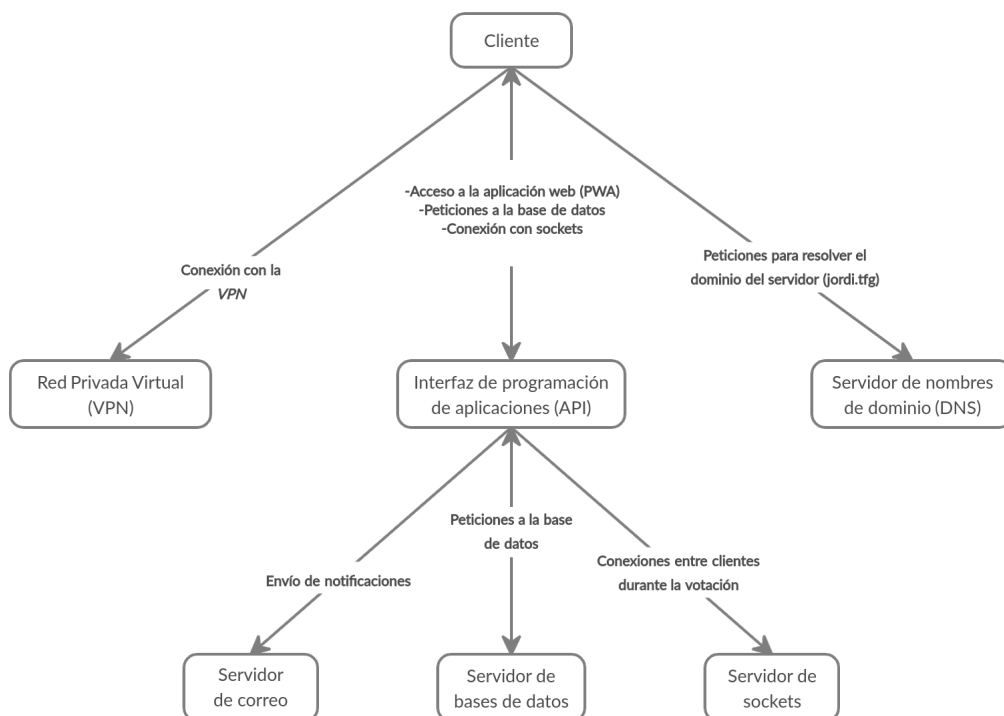


Figura 1.2: Alcance informático del proyecto.

1.5. Estructura de la memoria

La memoria de este proyecto consta de ocho capítulos, cada uno de los cuales describe un aspecto importante del proyecto.

El presente capítulo, que es la **Introducción**, muestra una visión global del proyecto, los objetivos y el entorno en el que se ha desarrollado.

El segundo capítulo contiene la **Descripción** del proyecto, incluyendo las características de este y las tecnologías utilizadas en el desarrollo.

Posteriormente, se incluye la **Planificación** del proyecto, donde se identifican las tareas a realizar y se realiza una estimación temporal y económica en función de la metodología empleada.

El cuarto apartado se centra en el **Análisis** del sistema desarrollado. En este, se identifican los requisitos más importantes que el sistema debe cubrir y se modelan las clases a implementar.

Tras modelar el sistema, se realiza el **Diseño** de este, tanto de la arquitectura como de la aplicación web.

A continuación, se describen las decisiones tomadas y los patrones utilizadas durante la **Implementación** y el desarrollo del sistema propuesto. Se muestran también los resultados finales de la aplicación.

Seguidamente, se indican las herramientas y técnicas utilizadas para la **Verificación y Validación** del correcto funcionamiento del sistema implementado.

Finalmente, se describen las **Conclusiones** y los resultados obtenidos, así como las posibles mejoras a implementar.

Capítulo 2

Descripción del proyecto

Contenidos

2.1. Situación inicial	21
2.2. Descripción del sistema	22
2.2.1. Aplicación web progresiva	22
2.2.2. Protocolo de votación	23
2.2.3. Servidores	24
2.3. Tecnologías y herramientas utilizadas	25
2.3.1. Gestión y documentación	25
2.3.2. Estándares y algoritmos	26
2.3.3. Cliente web	29
2.3.4. Servidores	31
2.3.5. Herramientas de desarrollo	33

El capítulo actual se centra en describir detalladamente la situación de partida del proyecto, las características de este y las tecnologías que se han utilizado para llevar a cabo su desarrollo.

2.1. Situación inicial

Anteriormente, la organización no disponía de ninguna herramienta de estas características para ofrecer a sus clientes en su catálogo de servicios.

Los clientes potenciales del producto desarrollado son aquellos que llevan a cabo sus votaciones internas de forma manual y física, mediante papeletas o “a mano alzada”. Este sistema de votación presenta una serie de carencias notables.

Por un lado, el sistema presenta una notable ausencia de confidencialidad, ya que cualquiera puede conocer el voto del resto. El uso de papeletas puede aumentar la confidencialidad, pero los votantes podrían conocer la caligrafía del resto. Además, se requiere la presencia física de los participantes.

Por otro lado, este sistema de votación requiere un esfuerzo mucho mayor por parte de los votantes y los administradores, tanto para la participación como para el recuento.

Además, existe una dificultad para mantener un histórico de votaciones, ya que no existe ningún tipo de base de datos donde almacenar los resultados. Para mantener estos, es necesario realizarlo manualmente mediante una hoja en papel o un programa de hojas de cálculo.

2.2. Descripción del sistema

En este proyecto se desarrolla un sistema que soluciona todas las carencias anteriores, incluyendo la funcionalidad especificada en el alcance. Este sistema se divide en diferentes módulos.

2.2.1. Aplicación web progresiva

Este proyecto consiste en el desarrollo de una PWA a través de la cual los miembros de una organización pueden realizar sus votaciones de forma segura.

En términos técnicos, una PWA [6] es una página web orientada a dispositivos móviles que cumple unos ciertos criterios que le permiten al usuario utilizarla como una aplicación nativa del dispositivo sin que esta esté instalada. Estos criterios son:

- Servir la aplicación a través de un servidor HTTPS¹ que garantice que las conexiones entre cliente y servidor se realizan a través de un canal seguro.
- Incluir *service workers*² que trabajen en segundo plano y proporcionen funcionalidades a la aplicación como el funcionamiento *offline*.
- Incluir un fichero de manifiesto que describa la información principal de la aplicación (*e.g.* nombre, iconos, etc).

Las aplicaciones que cumplen con los criterios de una PWA facilitan el acceso al usuario, ya que no es necesario que este acceda a la web a través del navegador.

La aplicación progresiva desarrollada se divide en dos módulos distintos:

- **Zona de votantes:** Este módulo permite a los votantes visualizar todo el listado de votaciones y participar en aquellas en las que tienen acceso. Además, también pueden visualizar los resultados de otras votaciones o modificar sus datos personales.
- **Zona de administradores:** A este módulo solo tienen acceso los usuarios con rol de “administrador”. Desde aquí, estos pueden crear o modificar votaciones, registrar nuevos votantes e iniciar las votaciones cuando todos los votantes estén preparados.

¹ **HTTPS** (Protocolo Seguro de Transferencia de HiperTexto) es una extensión de HTTP que proporciona seguridad a las comunicaciones. **HTTP** (Protocolo de Transferencia de HiperTexto) es un protocolo de comunicación a través del cual operan los servidores web en Internet.

² Un *service worker* es un *script* que el navegador ejecuta en segundo plano, el cual permite realizar acciones como gestionar la caché o recibir notificaciones.

2.2.2. Protocolo de votación

Con el fin de garantizar los cuatro pilares básicos de la seguridad informática descritos anteriormente, se implementa el algoritmo de votación de Michael Merritt, que utiliza criptografía asimétrica o de llave pública.

Los algoritmos de criptografía asimétrica [7] basan su funcionamiento en el uso de dos claves, como se puede ver representado en la figura 2.1:

- Una **clave pública**, accesible para cualquiera, que se utiliza normalmente para cifrar los datos.
- Una **clave privada**, solo conocida por el propietario, que suele ser utilizada para descifrar.

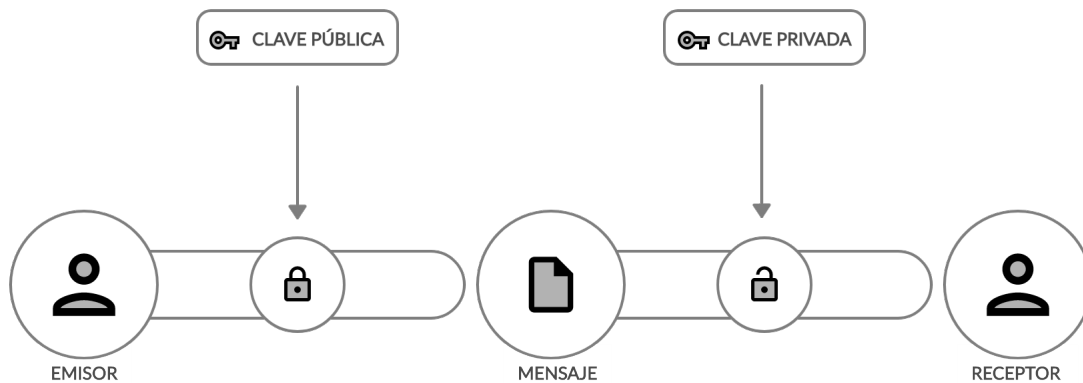


Figura 2.1: Esquema de la criptografía asimétrica o de llave pública.

Las claves son intercambiables, es decir, se puede cifrar con la clave pública y descifrar con la privada y viceversa. Esto permite que una persona envíe un mensaje de forma segura a otro a través de un canal inseguro, cifrando el mensaje con la clave pública del receptor.

El principal problema de estos sistemas de criptografía es el alto coste computacional de sus operaciones, motivo por el cual el sistema de votación implementado solo es viable para un grupo reducido de votantes (de 4 a 10).

El protocolo de Merritt utiliza RSA (Rivest, Shamir, Adleman), el algoritmo de criptografía asimétrica más extendido.

En este protocolo de votación [8], no es necesario que haya una entidad que controle y reciba los resultados de la votación. Son los propios participantes los que se encargan de realizar el recuento. Además, el propio funcionamiento del protocolo impide que la votación sea manipulada o alterada por cualquier votante o por un atacante externo.

2.2.3. Servidores

Para que el sistema funcione correctamente y cumpla con los objetivos, se requiere la configuración e implementación de una serie de servidores.

En primer lugar, es necesario desarrollar un servidor web que atienda, controle y resuelva tanto el acceso a la aplicación como las peticiones de los usuarios. Para ello, se ha desplegado un servicio o API³ REST⁴ que permite que los usuarios accedan a los recursos mediante una URI⁵ única. Este servidor también se encarga de la autorización de los usuarios en la aplicación y del envío de notificaciones.

Durante los procesos de votación, los votantes se comunican entre ellos mediante un canal de *websockets*⁶. Por tanto, es necesario también desarrollar los servidores web que permitan mantener las conexiones entre los votantes a través de estos *websockets*.

Por otro lado, el sistema incluye un servidor de base de datos que permite almacenar toda la información sobre las votaciones, los participantes y los resultados. Como la cantidad de datos a almacenar es relativamente pequeña, se ha elegido el Sistema de Gestión de Bases de Datos (SGBD) de MySQL, ya que es un sistema de código abierto bastante popular, extendido y de tamaño reducido.

Además, también se ha implementado un servidor de VPN. Una VPN [9] es una tecnología de red que permite extender una red privada a través de una red pública como por ejemplo Internet. De esta manera se pueden enviar y recibir datos de manera segura y confidencial, sin ningún tipo de preocupación. Esto se debe a que se crea una conexión virtual punto a punto entre el servidor de VPN y el usuario que se conecta, transfiriendo los datos a través de un túnel en el que los datos viajan cifrados.

Para este proyecto, se ha creado una VPN que permite a los usuarios autenticarse mediante el uso de certificados digitales. Al conectarse con el móvil a esta red privada virtual, los votantes pueden acceder a la aplicación y participar en las votaciones, mientras que la aplicación se encuentra inoperativa para los usuarios que no acceden a través de la VPN.

De este modo, se puede controlar qué usuarios emplean la aplicación para realizar las votaciones. Para implementar la VPN, se ha utilizado el *software* de OpenVPN, una herramienta de código abierto muy utilizada para la creación de redes privadas.

Finalmente, se ha desplegado un DNS⁷, que permite a los usuarios acceder a la aplicación web a través de su nombre simbólico.

³ Una **API** (Interfaz de Programación de Aplicaciones) se compone de una serie de procedimientos que permiten la comunicación entre el *software* del cliente y el del servidor.

⁴ **REST** (Transferencia de Estado Representacional) es un estándar de comunicación entre cliente y servidor a través del protocolo HTTP.

⁵ **URI** (Identificador de Recursos Uniforme) es un conjunto de caracteres que permite identificar un recurso a través de Internet.

⁶ **Websocket** es un tipo de tecnología que permite la conexión punto a punto entre dos clientes o entre un cliente y un servidor.

⁷ Un **DNS** (Servidor de Nombres de Dominio) proporciona un servicio que traduce direcciones URL (Localizador de Recursos Uniforme) de un dominio a direcciones IP físicas.

2.3. Tecnologías y herramientas utilizadas

En esta sección, se muestran las herramientas utilizadas para la gestión y el desarrollo del proyecto, así como las tecnologías utilizadas en el mismo.

2.3.1. Gestión y documentación

En el presente apartado, se describen los programas o aplicaciones web utilizadas para controlar, administrar y llevar un seguimiento del proyecto.

Trello

Trello [10] es un *software* que facilita la gestión y administración de proyectos. Esta herramienta está disponible como aplicación web y móvil tanto en Android como en iOS y permite algunas de las siguientes funcionalidades:

- Crear listas o *checklists* de tareas
- Asignar tareas a usuarios
- Asignar prioridad a las tareas mediante etiquetas
- Organizar temporalmente las tareas mediante fechas de vencimiento

En la figura 2.2, podemos ver un ejemplo del uso de esta herramienta.

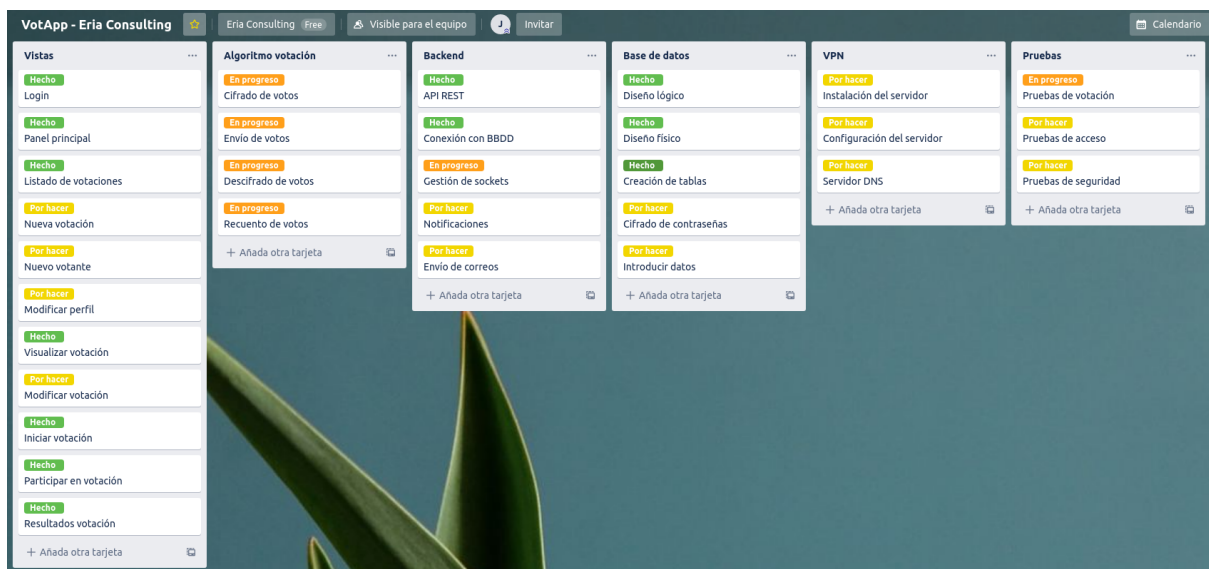


Figura 2.2: Tablero del proyecto en Trello.

GitHub

GitHub [11] es una plataforma de desarrollo colaborativo que permite compartir el código implementado con el resto del equipo. Esta herramienta hace uso del *software* Git⁸, que se encarga de realizar un control de versiones sobre el código desarrollado.

En la figura 2.3, podemos ver un ejemplo del control de versiones en GitHub.

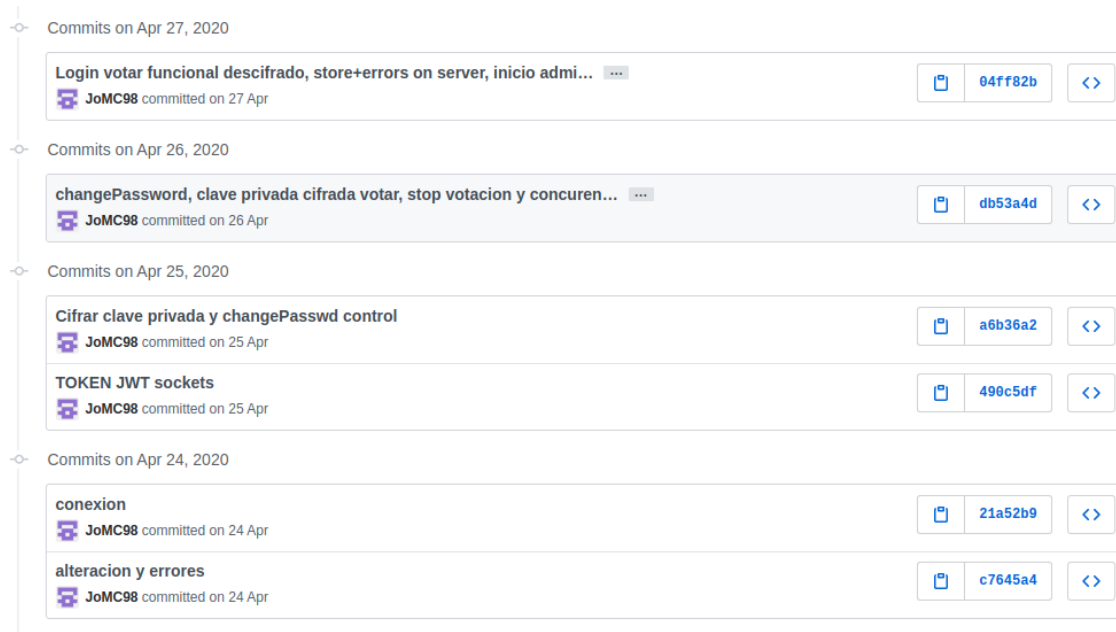


Figura 2.3: Control de versiones del proyecto en GitHub.

2.3.2. Estándares y algoritmos

En el apartado actual, se describen los estándares más importantes que se han seguido en la implementación del proyecto, así como los algoritmos de cifrado utilizados.

REST

Tal como se ha comentado anteriormente, en el sistema se ha implementado una API que sigue el estándar REST.

REST es un estándar que define los protocolos y la arquitectura de comunicación entre un cliente y un servidor siguiendo la filosofía CRUD⁹. Esta comunicación se realiza a través de peticiones HTTP que permiten acceder a los recursos del sistema.

⁸ **Git** es un *software* de control de versiones que permite gestionar los cambios realizados en el código fuente.

⁹ **CRUD** (Crear, Leer, Modificar, Borrar) es una filosofía de diseño que define los cuatro principios básicos de la capa de almacenamiento de un sistema.

En la tabla 2.1 se muestra la relación entre los métodos HTTP y los principios CRUD.

CRUD	Creación	Lectura	Modificación	Borrado
HTTP	POST	GET	PUT	DELETE

Tabla 2.1: Relación entre principios CRUD y métodos HTTP.

Con el siguiente ejemplo, se ilustra el funcionamiento de esta API REST, que se ha desarrollado con el *framework* `Express.js` de Node.js, el cual permite desplegar servidores web de forma muy sencilla:

1. El cliente solicita un recurso a la API a través de una URI única. Por ejemplo, solicita el listado de votaciones con la siguiente URI:

`https://jordi.tfg:4300/obtenerVotaciones`

2. Cuando la API recibe la petición, realiza las operaciones necesarias y devuelve el recurso correcto al cliente para que lo visualice.

JSON

JSON (*JavaScript Object Notation*) [12] es un formato de texto utilizado para el intercambio de datos. Se trata de un formato muy ligero, simple de leer tanto por humanos como por máquinas y compatible con múltiples lenguajes de programación.

Estas ventajas le convierten en un formato ideal para el intercambio de información. Por ello, toda la información que se intercambia entre los clientes de la aplicación y la API REST del sistema implementado se encuentra codificada en formato JSON.

En la figura 2.4 se observan dos ejemplos de mensajes en formato JSON utilizados en el sistema, uno con los datos de una votación y el otro con los datos de un usuario.

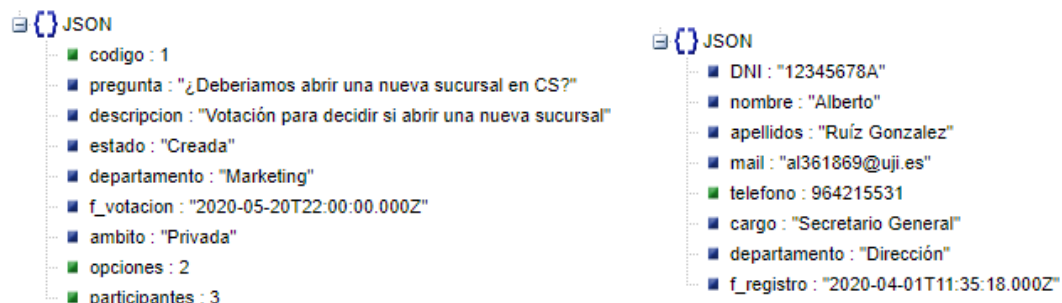


Figura 2.4: Ejemplos de mensajes en formato JSON usados en el sistema.

JWT

JWT (JSON Web *Tokens*) [13] es un estándar de seguridad que permite crear y propagar *tokens*¹⁰ de acceso de forma segura. Como su nombre indica, está basado en JSON. Los *tokens* permiten autenticar a un usuario en el sistema y de esta forma, permitir o denegar el acceso a un recurso.

Los *tokens* son generados y firmados mediante la clave del servidor y enviados posteriormente al cliente, de forma que ambos pueden confirmar su validez. Estos se componen de tres partes codificadas en base 64¹¹ y separadas por puntos:

- **Encabezado:** Contiene información del algoritmo y el *token*.
- **Carga útil:** Contiene toda la información útil del usuario (ID, permisos, etc) y la fecha de expiración del *token*.
- **Firma:** Contiene la firma del encabezado y la carga útil.

En la figura 2.5, se visualiza un ejemplo de un *token* generado en el sistema.

The image shows a web interface for decoding a JWT token. On the left, under the heading "Encoded" with a subtext "PASTE A TOKEN HERE", there is a text box containing a long string of base64 characters: `eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODBlIiwiaWF0IjoiYXNjaWkiLCJleHAiOiJ1OTA3ODY1MTJ9.wSCHgK59Nbox5yL92up6JKsmfOHhxejoTxxJ02OxVPZwDarC4FFPBC__NFpfqj5QI-fjxwOLqID5wtagGDuDGgqPEYNHzG7HH_4JnhiAfnakwKYNu4nIXBZr5gizBjeZR3DQJRkjjyEVw_EJLileGr44ZaH2_NtGFetKjQuySs`. On the right, under the heading "Decoded" with a subtext "EDIT THE PAYLOAD AND SECRET" and a "Signature Verified" status, there are two panels. The "HEADER: ALGORITHM & TOKEN TYPE" panel shows a JSON object: `{ "alg": "RS256", "typ": "JWT" }`. The "PAYLOAD: DATA" panel shows a JSON object: `{ "sub": "1234568A", "admin": true, "iat": 1590782912, "exp": 1590786512 }`.

Figura 2.5: Ejemplo de *token* JWT generado por el sistema.

Como se puede ver, el *token* está formado por una cabecera con el algoritmo; una carga útil que contiene datos como el DNI (Documento Nacional de Identidad) del usuario o su rol, además de las fechas de creación y expiración del propio *token*; y la firma, validada al introducir la clave pública correspondiente.

RSA y AES

Como se ha explicado anteriormente, el proyecto incorpora el sistema criptográfico asimétrico RSA para cifrar y descifrar los votos.

¹⁰ Un *token* es un conjunto de caracteres que carecen de significado pero representan e identifican un usuario o recurso dentro de un sistema informático.

¹¹ **Base 64** es un sistema numérico de codificación que representa los datos mediante 64 caracteres distintos.

Sin embargo, también se ha utilizado un algoritmo de cifrado simétrico o de clave privada, conocido como AES (*Advanced Encryption Standard*).

A diferencia de los algoritmos asimétricos, estos [7] utilizan la misma clave durante el cifrado y el descifrado. Por tanto, esta clave debe almacenarse de forma secreta y segura si se quiere mantener la confidencialidad.

En la figura 2.6 se observa una representación del funcionamiento de los sistemas criptográficos simétricos:

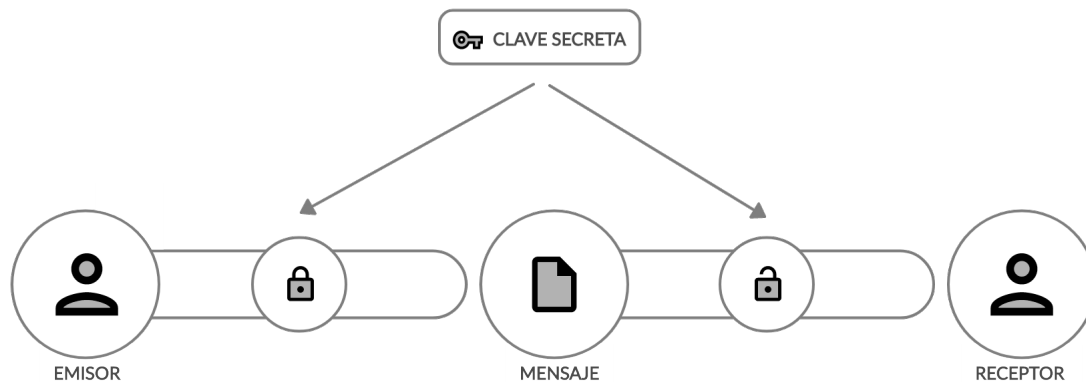


Figura 2.6: Esquema de la criptografía simétrica o de clave privada

El método AES ha sido utilizado para almacenar de forma segura las claves privadas de RSA que utilizan los votantes para participar en las votaciones. La clave secreta utilizada es la misma contraseña con la que los usuarios inician sesión en la aplicación, de forma que solo ellos pueden descifrar y utilizar su clave privada.

2.3.3. Cliente web

A continuación, se describen las tecnologías y los lenguajes de programación utilizados durante la implementación de la aplicación web progresiva.

HTML 5

HTML (*HyperText Markup Language*) [14] es un lenguaje de marcas que define el estándar para el desarrollo de páginas web. Este estándar para la creación de sitios web ha sido desarrollado por el W3C (*World Wide Web Consortium*) y actualmente se encuentra en su versión 5.

En el proyecto, se ha utilizado para la implementación de todas las vistas de la aplicación web progresiva.

CSS 3

CSS (*Cascading Style Sheets*) [14] es un lenguaje utilizado para definir el diseño y el estilo de una página web. CSS se utiliza de forma complementaria a HTML para definir el aspecto de las vistas desarrolladas.

JavaScript

JavaScript [15] es un lenguaje de programación interpretado¹², dinámico, orientado a objetos y sin tipado, procedente del estándar ECMAScript. En el desarrollo de una aplicación web, permite añadir la lógica necesaria para incrementar la funcionalidad de las vistas.

TypeScript

TypeScript [16] es un lenguaje de programación que extiende la sintaxis de JavaScript e incrementa su funcionalidad. El valor añadido que aporta este lenguaje sobre JavaScript es que facilita el trabajo de los programadores al añadir tipado a los datos y objetos basados en clases. De esta forma, se pueden detectar errores y problemas de código antes de su ejecución.

Angular 9

Angular 9 [17] es un *framework* de programación web desarrollado por Google y muy utilizado actualmente debido a sus múltiples ventajas. Este entorno de trabajo permite desarrollar aplicaciones web de forma sencilla, rápida y modular.

El desarrollo en Angular se realiza a través de diferentes módulos o componentes¹³ que facilitan notablemente la programación y la reutilización de código. Además de los componentes, también se pueden generar interfaces, servicios¹⁴ o directivas¹⁵, que aumentan la funcionalidad de la aplicación.

Estos bloques de código se pueden generar de forma sencilla e intuitiva a través del cliente de Angular (Angular CLI), mediante un conjunto de comandos.

¹² Un **lenguaje de programación interpretado** es aquel que puede ser ejecutado sin la necesidad de ser compilado a instrucciones máquina primero. Para ello, es necesaria la presencia de un intérprete que analice y ejecute el programa.

¹³ Un **componente** en Angular es un conjunto de bloques de código que incluyen la parte visual, el estilo y la lógica de una sección de la aplicación web.

¹⁴ Un **servicio** en Angular es un fragmento de código utilizado por los componentes para obtener información o realizar operaciones de la lógica de negocio.

¹⁵ Una **directiva** en Angular representa un atributo que se puede incorporar a una etiqueta HTML para incrementar su funcionalidad.

Angular permite integrar una gran cantidad de librerías que pueden aumentar la funcionalidad de una página web. A continuación, se muestran las más importantes que se han utilizado en este proyecto:

- **Angular PWA:** Está librería o paquete de *software* incluye todo lo necesario para convertir la aplicación en una PWA.
- **Angular Material:** Es una librería de estilos, basada en la normativa Material Design¹⁶, que permite integrar componentes con estilo pre configurado en el proyecto.
- **HTTP Client:** Es una librería integrada en Angular que proporciona una interfaz para realizar peticiones HTTP de forma sencilla.
- **CryptoJS y jsencrypt:** Librerías criptográficas utilizadas para los cifrados y descifrados en RSA y AES durante los procesos de votación.
- **WebSocket:** Librería de JavaScript utilizada para la comunicación entre los votantes mediante *websockets* durante los procesos de votación.

En la figura 2.7 se puede ver un resumen de las tecnologías utilizadas para el desarrollo de la aplicación web.



Figura 2.7: Tecnologías utilizadas para el desarrollo del cliente web.

2.3.4. Servidores

Tal como se indica en el apartado 2.2.3, se han implementado una serie de servidores para conseguir toda la funcionalidad deseada en el sistema. En esta sección, se describe con más detalle las tecnologías utilizadas para el despliegue de cada uno de estos servidores.

Node.js

Node [18] es un entorno de ejecución basado en JavaScript para la programación del lado del servidor. Su rapidez y simplicidad permite crear servidores web y APIs de forma muy sencilla.

¹⁶ **Material Design** es una normativa de diseño que define unos principios de estilo para aplicaciones Android.

En este proyecto, se ha utilizado Node.js para implementar el servidor web que atiende las peticiones a la página web y la API que resuelve las peticiones a la base de datos.

Además, Node permite incluir un gran número de bibliotecas para aumentar la funcionalidad. Por este motivo, sumado a su sencillez y eficiencia, se ha escogido esta tecnología frente a otras como Apache o Nginx.

En la tabla 2.2 se listan las bibliotecas más importantes que se han utilizado y su propósito en el proyecto.

Biblioteca	Propósito
<code>express.js</code>	<i>Framework</i> utilizado para desplegar la API.
<code>spdy</code>	Librería utilizada para crear el servidor web HTTPS.
<code>websocket</code>	Módulo que ha permitido desplegar servidores de <i>websocket</i> para los procesos de votación.
<code>nodemailer</code>	Biblioteca utilizada para el envío de notificaciones por correo electrónico.
<code>webpush</code>	Esta librería proporciona una herramienta para enviar notificaciones nativas a los usuarios.
<code>bcrypt</code>	Librería usada para el cifrado de contraseñas.
<code>jsonwebtoken</code>	Biblioteca utilizada para la generación y validación de <i>tokens</i> JWT.
<code>mysql</code>	Permite la conexión con la base de datos de MySQL.

Tabla 2.2: Bibliotecas utilizadas en Node.js

MySQL

MySQL es el SGBD elegido para la aplicación desarrollada. Se trata de un sistema de bases de datos relacionales¹⁷ desarrollado por Oracle que sigue el modelo cliente/servidor. Su popularidad se debe a que es un sistema sencillo y de código de abierto [19].

OpenVPN

El *software* de OpenVPN [20] permite implementar, configurar y desplegar una red privada virtual de forma muy sencilla. Mediante este programa, las comunicaciones entre el servidor y los clientes se realizan de forma segura a través del protocolo TLS¹⁸.

¹⁷ Una **base de datos relacional** es aquella en la cual sus entidades o tablas se encuentran relacionadas entre ellas.

¹⁸ **TLS** (Seguridad de la Capa de Transporte) es un estándar de cifrado de extremo a extremo que garantiza la confidencialidad de una comunicación.

Esta herramienta permite configurar muchas opciones dentro de la red privada. Algunas de las opciones que se han configurado en este proyecto son:

- No redirigir todo el tráfico de los clientes a través de la red privada virtual.
- Redirigir solo el tráfico que vaya destinado al servidor web o al servidor DNS que se encuentran dentro de la red privada.
- Indicar el servidor DNS que los clientes deben utilizar.
- Configurar IPs estáticas a los clientes.

BIND 9

BIND (*Berkeley Internet Name Domain*) es el servidor de nombres de dominio más extendido y utilizado en todo el mundo. Por ello, se ha utilizado esta librería de Unix para desarrollar el servidor DNS.

Para poder construir una PWA, uno de los requisitos principales es que el servidor web que aloja esta aplicación opere sobre el protocolo HTTPS. Por ello, es necesario que el servidor web posea un certificado válido sobre un dominio web.

Este es el principal motivo por el cual ha sido necesario el desarrollo de un servidor DNS, de forma que los clientes puedan acceder a la web a través de un nombre simbólico.

En la figura 2.8 se incluye un resumen de las tecnologías utilizadas para el desarrollo de los servidores del sistema.



Figura 2.8: Tecnologías utilizadas para el desarrollo de los servidores.

2.3.5. Herramientas de desarrollo

En este apartado se describe el *software* y las herramientas utilizadas durante todo el desarrollo del proyecto. Estos programas nos han permitido optimizar y realizar de forma cómoda la implementación del sistema.

Visual Studio Code

Visual Studio Code es un editor de código muy completo que facilita el trabajo de los programadores gracias a sus numerosas funcionalidades. Entre estas, destaca la integración de Git, lo que permite compartir los repositorios con GitHub de forma eficiente. Esta herramienta se ha utilizado para la implementación y las pruebas del proyecto, tanto en el desarrollo de la aplicación web como en el despliegue del servidor web y la API.

phpMyAdmin

Tal como se ha comentado anteriormente, el SGBD utilizado en el proyecto ha sido MySQL. Para implementar, configurar y mantener esta base de datos durante el desarrollo del proyecto, se ha utilizado phpMyAdmin.

PhpMyAdmin [21] es una herramienta escrita en PHP¹⁹ que permite gestionar y administrar bases de datos MySQL a través de una interfaz web. Entre las funcionalidades que aporta esta herramienta, se incluyen:

- Creación o modificación de tablas y relaciones
- Consulta, inserción, modificación o borrado de datos
- Exportación de la base de datos

En la figura 2.9 se muestra un ejemplo de esta herramienta, en el que se pueden visualizar algunas entradas de la tabla de usuarios de la base de datos del sistema.

DNI	nombre	apellidos	mail	telefono	departamento	cargo	VPN_IP	passwd	clavePublica	clavePrivada
12345678A	Alberto	Ruiz Gonzalez	al361869@uji.es	964215531	Dirección	Secretario General	192.168.210.250	\$2b\$10\$YPk4Sgxt	-----BEGIN PUBLIC KEY----- MIIBjANBgkqhkiG9w0BAQE...	{"salt":"73464a25b321
20904687X	Jordi	Miralles Comins	jordimc98@gmail.	143214124	Administración	CTO	192.168.210.246	\$2b\$10\$qCjvOLJC	-----BEGIN PUBLIC KEY----- MIIBjANBgkqhkiG9w0BAQE...	{"salt":"fd63fc9daa06
99999999A	José	Torres Villa	sarjor99@gmail.ct	12412412	Administración	COO	192.168.210.242	\$2b\$10\$6uDY4Zm	-----BEGIN PUBLIC KEY----- MIIBjANBgkqhkiG9w0BAQ...	{"salt":"25b912c6a53
11112222A	Manolo	Lopez Vega	sara.miralles.comi	658940696	Administración	CEO	192.168.210.238	\$2b\$10\$QAJQ8zW	-----BEGIN PUBLIC KEY----- MIIBjANBgkqhkiG9w0BAQ...	{"salt":"3143fe10c47f

Figura 2.9: Ejemplo de entradas de la base de datos en phpMyAdmin.

Postman

Postman es una herramienta utilizada para realizar pruebas sobre diferentes API. Este *software* permite realizar peticiones HTTP de cualquier tipo (*GET*, *POST*..) sobre un servidor web para comprobar el correcto funcionamiento de este.

Esta utilidad permite modificar el cuerpo y las cabeceras de las peticiones que se envían. En este proyecto, además de usar la herramienta para comprobar el funcionamiento de la API, ha sido de gran utilidad también para verificar el uso correcto de los *tokens* JWT para la autenticación y autorización de los usuarios.

¹⁹ **PHP** (*Hypertext Preprocessor*) es un lenguaje de programación utilizado para el desarrollo de código del lado del servidor.

Capítulo 3

Planificación del proyecto

Contenidos

3.1. Metodología	35
3.2. Análisis y gestión de riesgos	36
3.3. Planificación temporal	37
3.3.1. Tareas y actividades	37
3.3.2. Estructura de Desglose del Trabajo	38
3.3.3. Distribución temporal	38
3.3.4. Diagrama de Gantt	40
3.4. Estimación de recursos y costes del proyecto	42
3.4.1. Costes <i>hardware</i>	42
3.4.2. Costes <i>software</i>	42
3.4.3. Costes humanos	43
3.5. Seguimiento del proyecto	44

El capítulo actual tiene como objetivo mostrar la planificación que se ha seguido durante el proyecto, describiendo además la metodología utilizada y el seguimiento que se ha realizado. Para ello, se ha desglosado el proyecto en tareas y se ha estimado la duración de cada una de estas.

Por otro lado, se realiza un análisis de los posibles riesgos del proyecto y una estimación de los recursos y los costes necesarias para el desarrollo.

3.1. Metodología

La metodología que se ha seguido durante el desarrollo del proyecto ha sido una metodología predictiva. Se trata de una metodología tradicional en la que se definen una serie de etapas y cada una se realiza al finalizar la anterior, siguiendo un ciclo en cascada.

Para utilizar esta metodología correctamente, es necesario establecer los objetivos y definir los requisitos básicos del sistema.

Se ha escogido esta metodología debido a que el proyecto es desarrollado por una única persona, por lo que las tareas se deben hacer secuencialmente, una detrás de otra. Además, es la metodología que sigue la entidad en sus proyectos.

En la figura 3.1 se incluye un esquema con las fases básicas en las que se divide esta metodología.

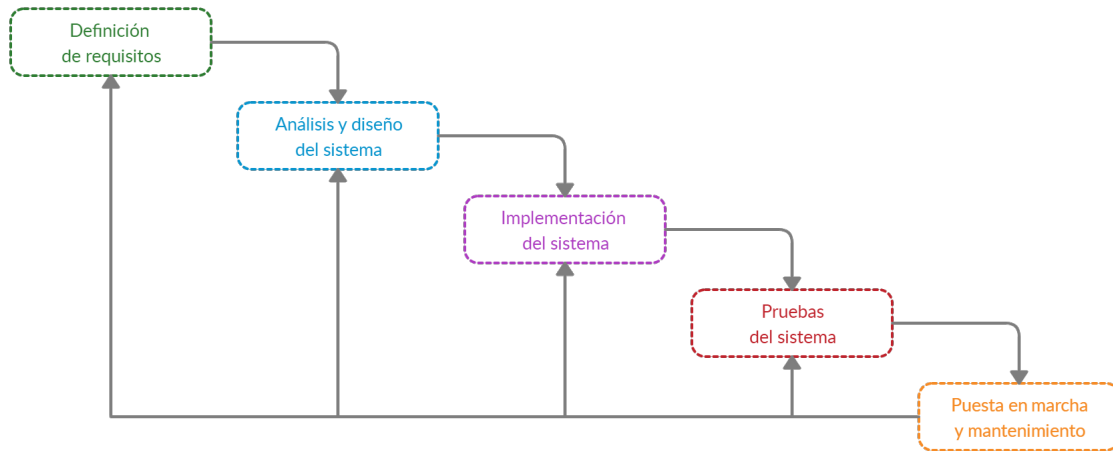


Figura 3.1: Esquema de la metodología en cascada.

3.2. Análisis y gestión de riesgos

Además de definir los objetivos y los requisitos, en la metodología predictiva también es necesario realizar un análisis de los posibles riesgos que el proyecto puede presentar. Esta tarea es esencial para asegurar el correcto funcionamiento del sistema a implementar. Estos riesgos pueden encontrarse en muchos ámbitos distintos: en el entorno, en el equipo de desarrollo, en los usuarios futuros, en las tecnologías a utilizar, etc.

En cuanto al **entorno** del propio proyecto, se identifican tres riesgos importantes:

- Restricciones temporales que impidan finalizar el proyecto a tiempo.
- Alteraciones en el orden de ejecución o la planificación temporal de las tareas.
- Errores en la definición de requisitos que alteren el desarrollo del proyecto.

Por ello, es necesario identificar bien las necesidades del proyecto y realizar una buena planificación de este.

Por otro lado, se puede encontrar un riesgo notable en la poca experiencia del **equipo de desarrollo** en el despliegue de proyectos web, sumado al desconocimiento de las **tecnologías** a emplear. Para mitigar este riesgo, es necesario formar al equipo en el *software* y los métodos de trabajo a utilizar.

Finalmente, también puede considerarse como un posible riesgo el conocimiento y la habilidad de los **usuarios** al utilizar el producto. Por ello, es necesario desarrollar una aplicación intuitiva y fácil de utilizar.

3.3. Planificación temporal

Tal como se ha explicado anteriormente, en este proyecto se ha seguido una metodología predictiva en la cual se han definido una serie de tareas y actividades. En esta sección se identifican estas tareas y se muestra la estimación temporal para cada una de ellas.

3.3.1. Tareas y actividades

Este proyecto se puede dividir en diferentes tareas o fases de primer nivel. Al utilizar una metodología predictiva, es necesario finalizar una tarea para poder abordar la siguiente. En general, se identifican las siguientes fases:

- **Inicio del proyecto:** En esta fase, se incluyen las actividades primordiales para poder iniciar el desarrollo del proyecto. Entre estas, destacan la descripción del proyecto y la identificación de los objetivos y el alcance de este.
- **Definición de requisitos:** Esta tarea consiste en identificar y definir los requisitos más importantes que el proyecto debe cumplir. Para ello se utilizan técnicas UML¹ como los diagramas de casos de uso.
- **Planificación:** Esta fase tiene como objetivo definir las tareas y actividades a realizar durante el proyecto. Además, se debe realizar una estimación de los recursos necesarios y los costes, tanto a nivel temporal como económico, así como los posibles riesgos durante el desarrollo.
- **Formación:** Durante esta etapa, el equipo de desarrollo debe estudiar y formarse en las tecnologías y herramientas a utilizar, así como en los métodos y estándares de trabajo de la entidad.
- **Análisis:** Durante el análisis, se realizan los diagramas de clases y de actividades UML que permiten identificar las clases y las relaciones más importantes del sistema.
- **Diseño:** Esta fase se centra en definir y representar la arquitectura del sistema a implementar, así como las interfaces de usuario que tendrá la aplicación final.
- **Implementación:** Esta etapa es la más importante del proyecto y la que presenta una mayor duración. Consiste en el desarrollo de todos los módulos necesarios para el funcionamiento del sistema. Esta fase se puede dividir en las siguientes actividades:
 - A. Implementación de la aplicación móvil para los votantes y el administrador.
 - B. Implementación de la API y la base de datos.
 - C. Implementación de los protocolos de cifrado y envío.
 - D. Implementación de la VPN y el servidor DNS.
- **Control de calidad:** En esta fase se valida y se verifica el correcto funcionamiento del sistema desarrollado.
- **Puesta en marcha:** El sistema implementado se despliega y se pone en funcionamiento dentro de la organización. Además, se realiza una formación a los futuros usuarios.

¹ El lenguaje UML (Lenguaje Unificado de Modelado) se usa para analizar y modelar productos de *software*.

3.3.2. Estructura de Desglose del Trabajo

En esta sección, se muestran las diferentes fases y tareas del proyecto mediante un diagrama EDT² (Esquema de Descomposición del Trabajo). Se puede observar este diagrama en la figura 3.2:

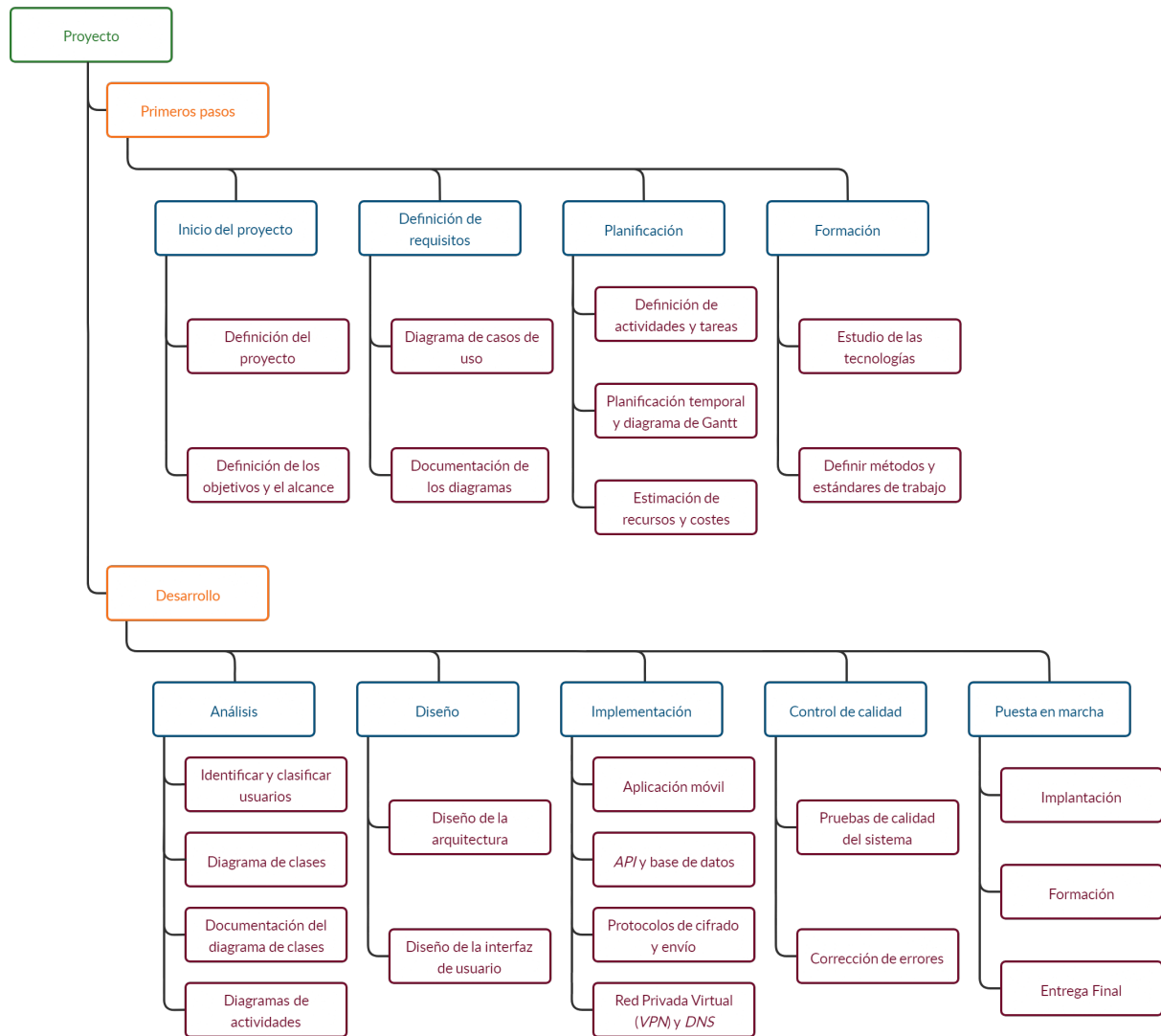


Figura 3.2: EDT del proyecto.

3.3.3. Distribución temporal

En esta sección se muestra de forma detallada la asignación de horas dedicadas a cada una de las fases o tareas descritas anteriormente. Además, se define el calendario que ha ocupado este proyecto.

En la tabla 3.1 se incluye el coste temporal de cada una de las fases del proyecto.

² Una EDT es un esquema o modelo que muestra de forma jerárquica las tareas y actividades de un proyecto.

Tarea	Tarea predecesora	Horas dedicadas
Realización del proyecto		300
1 Primeros pasos		48
1.1 Inicio del proyecto		7
1.1.1 Definir el proyecto		4
1.1.2 Identificar objetivos y alcance	1.1.1	3
1.2 Definición de requisitos		14
1.2.1 Diagramas de casos de uso	1.1	4
1.2.2 Documentación de los diagramas	1.2.1	10
1.3 Planificación del proyecto		13
1.3.1 Identificar actividades y tareas	1.2	5
1.3.2 Diagrama de Gantt	1.3.1	3
1.3.3 Estimación de recursos y costes	1.3.2	5
1.4 Formación		14
1.4.1 Estudio de las tecnologías a usar	1.3	12
1.4.2 Definir formatos y métodos de trabajo	1.3	2
2 Desarrollo del proyecto	1	252
2.1 Análisis		17
2.1.1 Identificar y clasificar los usuarios		2
2.1.2 Diagrama de clases	2.1.1	4
2.1.3 Documentación del diagrama de clases	2.1.2	6
2.1.4 Diagramas de actividades	2.1.1	5
2.2 Diseño		28
2.2.1 Diseño de la arquitectura del sistema	2.1	11
2.2.2 Diseño de la interfaz de usuario	2.1	17
2.3 Implementación		157
2.3.1 Implementar aplicación móvil	2.2	52
2.3.2 Implementar API y base de datos	2.3.1	43
2.3.3 Implementar los protocolos de cifrado	2.3.2	25
2.3.4 Implementar la VPN y el servidor DNS	2.3.3	37
2.4 Control de calidad		30
2.4.1 Pruebas de calidad del sistema	2.3	12
2.4.2 Corrección de errores	2.4.1	18
2.5 Puesta en marcha		20
2.5.1 Implantación	2.4	5
2.5.2 Formación	2.5.1	15
2.5.3 Entrega Final	2.5.2	0

Tabla 3.1: Distribución temporal de las tareas del proyecto.

El desarrollo del proyecto ha tenido una duración de 300 horas, el tiempo dedicado a la estancia en prácticas en la entidad Eria Consulting. Estas horas se han repartido durante 13 semanas, con la realización de 25 horas semanales excluyendo los días no laborables y festivos. La estancia en la empresa fue iniciada el 03 de febrero de 2020 y finalizó el 18 de mayo de 2020.

En la tabla 3.2 se puede observar con detalle el resumen de horas dedicadas al proyecto agrupadas semanalmente.

Febrero		Marzo		Abril		Mayo	
03 - 07	25 horas	02 - 06	20 horas	06 - 10	20 horas	04 - 08	25 horas
10 - 14	25 horas	09 - 13	20 horas	20 - 24	25 horas	11 - 15	25 horas
17 - 21	25 horas	23 - 27	25 horas	27 - 01	15 horas	18 - 22	5 horas
24 - 28	20 horas	30 - 03	25 horas				

Tabla 3.2: Calendario del proyecto.

Por otro lado, en la tabla 3.3 se muestra el coste temporal dedicado a la documentación y presentación del proyecto, en el que se incluye la redacción del presente documento.

Tarea	Tarea predecesora	Horas dedicadas
Documentación del proyecto		150
1 Redacción de documentos		114
1.1. Propuesta técnica		10
1.2. Informes quincenales	1.1	14
1.3 Memoria técnica		90
1.3.1 Redacción de la memoria técnica	1.1	90
1.3.2 Entrega de la memoria técnica	1.3.1	0
2 Presentación del proyecto		36
2.1 Preparación de la presentación oral	1	35
2.2 Presentación oral	2.1	1

Tabla 3.3: Distribución temporal de las tareas de documentación del proyecto.

3.3.4. Diagrama de Gantt

Para visualizar mejor la planificación y la dedicación temporal del proyecto, se ha desarrollado un diagrama de Gantt, que se puede observar en la figura 3.3.

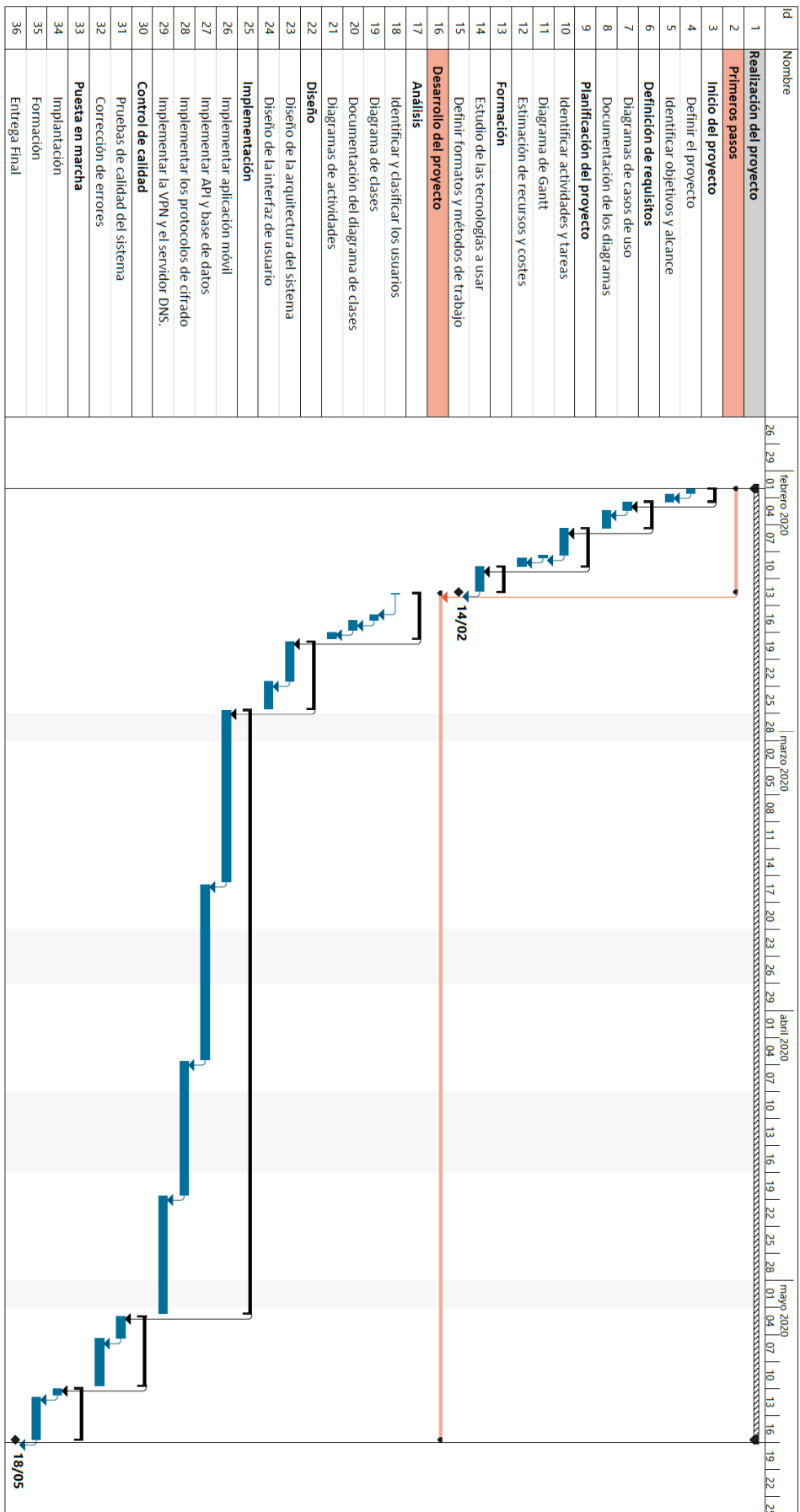


Figura 3.3: Diagrama de Gantt del proyecto.

3.4. Estimación de recursos y costes del proyecto

En este apartado se realiza una estimación de los recursos necesarios para el desarrollo del proyecto, así como el coste que supone el uso de estos. Los recursos utilizados y sus costes asociados se pueden clasificar en tres tipos.

3.4.1. Costes *hardware*

Para llevar a cabo el desarrollo del proyecto, se ha hecho uso de un portátil ASUS Rog Strix valorado en 1000€ y un monitor auxiliar BenQ con un coste de 200€. El periodo de amortización de estos componentes se estima en 5 años (1.825 días) y estos han sido utilizados durante 60 días (300 horas repartidas en 5 horas diarias).

Por otro lado, durante el periodo de pruebas e implantación (20 días), se han utilizado 4 dispositivos móviles, valorados en 200€ cada uno. La amortización de estos se acerca a los 2 años (730 días).

Para calcular el coste del *hardware* en el proyecto, hay que prorratear los precios de adquisición al periodo durante el cual se han utilizado los componentes. En la tabla 3.4 se incluye un resumen con los cálculos y el coste total del *hardware* usado para el proyecto.

Recurso	Precio	Amortización	Coste amortizado por día	Días	Coste total amortizado
Portátil	1000€	1.825 días	0'54€ por día	60	32'4€
Monitor	200€	1.825 días	0'10€ por día	60	6€
Móviles	800€	730 días	1'08€ por día	20	21'6€
					60€

Tabla 3.4: Resumen de costes *hardware*

3.4.2. Costes *software*

En cuanto a los costes del *software* utilizado, hay que analizar las diferentes tecnologías que se han descrito en la sección 2.3. Por un lado, en el proyecto se utilizan programas y tecnologías de código abierto, como Visual Studio Code, OpenVPN, BIND9, MySQL o phpMyAdmin.

No obstante, también se utilizan herramientas con planes de pago específicos para las empresas. Es el caso de Trello [22], GitHub [23] y Postman [24], herramientas en las cuales se paga una cuota mensual por usuario en función de la tarifa contratada.

Mientras que Trello y GitHub han sido utilizadas durante los cuatro meses del proyecto, Postman solo ha sido usada durante los dos últimos, para la implementación y las pruebas de la API. En la tabla 3.5 se observa un resumen con los costes del *software* utilizado durante el proyecto.

Recurso	Tarifa	Precio mensual	Meses	Coste total
Trello	Business Class	12'5 € por mes	4	50 €
GitHub	Team	4 € por mes	4	16 €
Postman	Team	12 € por mes	2	24 €
				90 €

Tabla 3.5: Resumen de costes *software*

3.4.3. Costes humanos

Al analizar los costes generados por los recursos humanos, hay que tener en cuenta los diferentes roles que existen en este proyecto, así como la dedicación horaria de estos.

Por un lado, se encuentra el **analista**, encargado de definir el proyecto y los objetivos, realizar la planificación, identificar los requisitos y analizar y diseñar el sistema. Su sueldo medio es de 22.319 € anuales [25] y su dedicación horaria ha sido de 79 horas.

Por otro lado, el proyecto incorpora un **programador**, encargado de realizar la implementación, las pruebas y la implantación del sistema, además de recibir la formación previa. Tiene un sueldo medio de 18.719 € anuales [26] y ha dedicado 221 horas al proyecto.

Finalmente, también se incluye el rol de **supervisor**, que consiste en un desarrollador especializado que se encarga de supervisar tanto el análisis como el desarrollo del proyecto. Su dedicación ha supuesto un 10 % de la totalidad del proyecto, es decir, 30 horas, mientras que su sueldo medio es de 36.399 € anuales [27].

Para calcular el coste total en salarios, es necesario calcular el salario de cada rol por hora trabajada. Para ello, hay que tener en cuenta que el sueldo mensual se obtiene al repartir el sueldo anual en las 14 pagas reglamentarias y este se corresponde con 160 horas trabajadas. En la tabla 3.6 se visualiza un resumen de los cálculos y el coste total en salarios.

Recurso	Salario anual	Salario mensual	Salario por hora	Horas	Salario total
Analista	22.310 €	1.594 €	10 € por hora	79	790 €
Programador	18.719 €	1.337 €	8'5 € por hora	221	1.878'5 €
Supervisor	36.399 €	2.600 €	16'25 € por hora	30	487'5 €
					3.156 €

Tabla 3.6: Resumen de costes humanos

Además, en los costes humanos hay que tener en cuenta otros costes adicionales como por ejemplo el alquiler de oficinas, la electricidad, Internet o impuestos debidos a la contratación. De forma general, se incluye un 30 % debido a los costes de contratación y un 20 % provocado por otros costes indirectos. En la tabla 3.7 se observa un resumen de estos costes.

Recurso	Coste base	Porcentaje	Coste total
Costes contratación	3.156 €	30 %	946'8 €
Costes indirectos	3.156 €	20 %	631'2 €
			1.578 €

Tabla 3.7: Resumen de costes adicionales

Finalmente, en la tabla 3.8 se resume el coste total del proyecto.

Recurso	Coste
Costes <i>hardware</i>	60 €
Costes <i>software</i>	90 €
Costes humanos	3.156 €
Costes adicionales	1.578 €
	4.884 €

Tabla 3.8: Resumen de costes del proyecto

El coste total de los recursos del proyecto asciende a 4.884 €.

3.5. Seguimiento del proyecto

Durante el desarrollo de este proyecto, se ha llevado a cabo un control constante del trabajo realizado. Al seguir una metodología tradicional, es muy importante revisar la finalización de una tarea para poder avanzar a la siguiente.

Además de la supervisión realizada durante la implementación, se han realizado reuniones de carácter semanal para evaluar la evolución del proyecto, marcar las pautas a seguir y corregir posibles desviaciones en la planificación.

La principal desviación en la planificación del proyecto se produjo el 16 de marzo de 2020, debido a la situación de emergencia sanitaria y estado de alarma provocada por la pandemia del COVID-19.

Para corregir esta alteración, se redefinieron los métodos de trabajo de la siguiente forma:

- El desarrollo del proyecto pasó a realizarse de forma telemática desde el hogar del estudiante.
- La supervisión del proyecto se empezó a llevar a cabo por vía electrónica mediante herramientas de comunicación como Gmail y Hangouts.
- Las reuniones semanales se mantuvieron, realizándose mediante videoconferencias a través de Hangouts.

Sin embargo, la distribución de las tareas no se ha visto prácticamente alterada. Los únicos cambios se han producido en la implementación, la revisión y el despliegue del sistema.

Implementación

Para el desarrollo del proyecto, algunos módulos han requerido más tiempo del planificado, ya que han presentado una complejidad mayor de la esperada. Es el caso de los servidores y la aplicación web. Sin embargo, los protocolos de cifrado y envío de datos han sido desarrollados utilizando menos tiempo del planificado.

Pruebas del sistema

Para verificar y validar el sistema, no se ha necesitado todo el tiempo planificado, ya que no se han encontrado muchos errores o problemas en el sistema.

Puesta en marcha

En el despliegue del sistema, se ha utilizado menos tiempo del esperado, ya que no se ha realizado una formación a los usuarios finales.

En la tabla 3.9 se incluye un resumen con la distribución temporal real de estas tareas:

Tarea	Horas dedicadas
2.3 Implementación	180
2.3.1 Implementar aplicación móvil	68
2.3.2 Implementar API y base de datos	49
2.3.3 Implementar los protocolos de cifrado	20
2.3.4 Implementar la VPN y el servidor DNS	43
2.4 Control de calidad	22
2.4.1 Pruebas de calidad del sistema	10
2.4.2 Corrección de errores	12
2.5 Puesta en marcha	5
2.5.1 Implantación	5
2.5.2 Entrega Final	0

Tabla 3.9: Cambios en la distribución temporal del proyecto.

Por otro lado, en la figura 3.4 se observa el diagrama de Gantt real del proyecto tras el seguimiento.

Capítulo 4

Análisis del sistema

Contenidos

4.1. Actores	47
4.2. Diagrama de casos de uso	47
4.3. Requisitos funcionales	48
4.4. Requisitos de datos	55
4.5. Diagrama de clases	58
4.6. Diagramas de actividades	59

En el presente capítulo se realiza un análisis detallado del sistema a implementar. Para ello, se utilizan técnicas y diagramas UML que permiten identificar los requisitos del sistema y ayudan a modelarlo. Entre estas técnicas, destacan los diagramas de casos de uso, de actividades y de clases.

4.1. Actores

Para poder identificar los requisitos y modelar el sistema, primero es necesario definir los usuarios o actores principales de este. Se definen los actores más importantes del sistema en la tabla 4.1.

Identificador	Actor	Descripción
AC-01	Administrador de votaciones	Este actor representa a cualquier persona con permisos para crear, gestionar e iniciar votaciones, además de controlar los votantes autorizados.
AC-02	Votante	Este actor representa a cualquier persona autorizada para participar en una votación.

Tabla 4.1: Actores principales del sistema.

4.2. Diagrama de casos de uso

El diagrama de casos de uso [28] es una técnica UML que permite identificar los requisitos del sistema de forma visual e intuitiva. Para ello, se representan los diferentes roles y la funcionalidad asociada a cada uno de ellos. En la figura 4.1 se incluye el diagrama de casos de uso que identifica los requisitos de este proyecto.

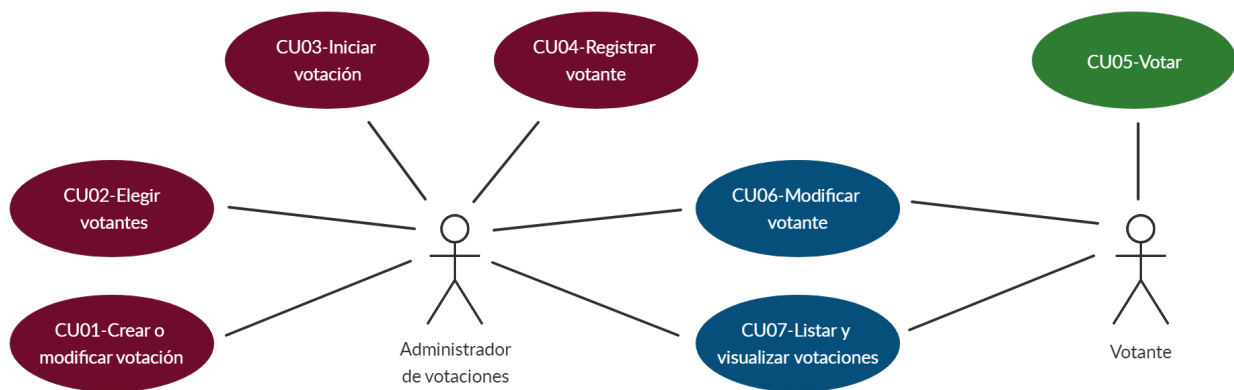


Figura 4.1: Diagrama de casos de uso del sistema.

En la tabla 4.2 se observa un resumen de los requisitos identificados en el diagrama anterior con una descripción de cada uno de ellos.

Actor Principal	Identificador	Caso de uso
AC-01	CU-01	Este caso de uso representa como un administrador de votaciones puede crear una nueva votación o modificar los datos de una existente.
	CU-02	Este caso de uso representa como un administrador de votaciones puede seleccionar los participantes de una votación.
	CU-03	Este caso de uso representa como un administrador de votaciones puede iniciar una votación.
	CU-04	Este caso de uso representa como un administrador de votaciones puede añadir nuevos votantes a la plataforma.
AC-02	CU-05	Este caso de uso representa como un votante puede participar en una votación en la cual tiene acceso.
AC-01 & AC-02	CU-06	Este caso de uso representa como un administrador de votaciones puede modificar los datos de los votantes o un votante puede modificar sus propios datos.
	CU-07	Este caso de uso representa como un administrador de votaciones o un votante pueden consultar el listado y los datos de las votaciones para las que tienen acceso.

Tabla 4.2: Resumen de los casos de uso del sistema.

4.3. Requisitos funcionales

En este apartado se describen con detalle los requisitos funcionales que el sistema debe satisfacer para cumplir con los objetivos del proyecto, que se corresponden con los casos de uso identificados anteriormente.

En las siguientes tablas (desde la 4.3 a la 4.9) se incluye la especificación de estos requisitos.

Requisito funcional	
Identificador	CU-01
Nombre	Crear o modificar una votación
Fecha creación	08/02/2020
Autores	Jordi Miralles Comins
Versión	1
Actor principal	AC-01
Descripción	La aplicación debe permitir a un administrador de votaciones crear una nueva votación o modificar los datos de una existente.
Relaciones	CU-02, CU-07
Precondición	El administrador debe estar conectado a la VPN y autenticado en la aplicación.
Disparador	Un administrador autenticado desea crear una votación.
Secuencia normal	<ol style="list-style-type: none"> 1. El administrador accede a la sección Crear Nueva Votación. 2. El sistema solicita los datos de la nueva votación. 3. El administrador introduce los datos de la votación. 4. El sistema comprueba la validez de los datos, muestra un resumen de estos y solicita la confirmación. 5. El administrador confirma la información. 6. El sistema crea la nueva votación y la almacena en la base de datos.
Secuencia alternativa	<p>Un administrador desea modificar una votación existente.</p> <ol style="list-style-type: none"> 1. El administrador accede a la sección de Votaciones y selecciona la votación a modificar. 2-5. Mismos pasos que en la secuencia normal 6. El sistema guarda los nuevos datos.
Excepciones	Si los datos introducidos por el administrador no son válidos, se muestra un mensaje de error.

Tabla 4.3: Especificación CU-01 - Crear o modificar una votación.

Requisito funcional	
Identificador	CU-02
Nombre	Elegir votantes
Fecha creación	08/02/2020
Autores	Jordi Miralles Comins
Versión	1
Actor principal	AC-01
Actor secundario	AC-02
Descripción	La aplicación debe permitir a un administrador de votaciones seleccionar los participantes de una votación.
Relaciones	CU-01, CU-04, CU-05, CU-07
Precondición	El administrador debe estar conectado a la VPN y autenticado en la aplicación.
Disparador	Un administrador autenticado está creando una nueva votación y desea seleccionar los votantes autorizados para participar en esta.
Secuencia normal	<ol style="list-style-type: none"> 1. El administrador ha introducido los datos de la nueva votación y selecciona la opción para escoger los participantes. 2. El sistema solicita los participantes. 3. El administrador selecciona los participantes. 4. El sistema comprueba la validez de los participantes y continúa con el proceso de creación de una nueva votación.
Excepciones	Si los participantes introducidos por el administrador no son válidos, se muestra un mensaje de error.

Tabla 4.4: Especificación CU-02 - Elegir votantes.

Requisito funcional	
Identificador	CU-03
Nombre	Iniciar votación
Fecha creación	08/02/2020
Autores	Jordi Miralles Comins
Versión	1
Actor principal	AC-01
Descripción	La aplicación debe permitir a un administrador de votaciones iniciar una votación.
Relaciones	CU-05, CU-07
Precondición	El administrador debe estar conectado a la VPN y autenticado en la aplicación.
Disparador	Un administrador visualiza el listado de votaciones y desea iniciar una votación.
Secuencia normal	<ol style="list-style-type: none"> 1. El administrador accede a la sección de Votaciones, al Listado y selecciona una votación para ver sus datos. 2. Dentro de esta votación, el administrador selecciona la opción de Iniciar Votación. 3. El sistema muestra el estado de los participantes, es decir, si están o no conectados. 4. Cuando están todos conectados, el sistema habilita la opción para iniciar. 5. El administrador selecciona la opción para iniciar. 6. El sistema inicia la votación, permitiendo a los votantes acceder a esta.
Excepciones	El administrador puede parar la votación en cualquier momento.

Tabla 4.5: Especificación CU-03 - Iniciar votación.

Requisito funcional	
Identificador	CU-04
Nombre	Registrar votante
Fecha creación	08/02/2020
Autores	Jordi Miralles Comins
Versión	1
Actor principal	AC-01
Actor secundario	AC-02
Descripción	La aplicación debe permitir a un administrador de votaciones añadir nuevos votantes a la plataforma.
Relaciones	CU-02, CU-05
Precondición	El administrador debe estar conectado a la VPN y autenticado en la aplicación.
Disparador	Un administrador autenticado desea registrar un nuevo votante en el sistema.
Secuencia normal	<ol style="list-style-type: none"> 1. El administrador accede a la sección de Participantes y selecciona Registrar Votante. 2. El sistema solicita los datos del nuevo votante. 3. El administrador introduce los datos del votante. 4. El sistema comprueba la validez de los datos, muestra un resumen de estos y solicita la confirmación. 5. El administrador confirma la información. 6. El sistema registra el nuevo votante y lo almacena en la base de datos.
Excepciones	Si los datos introducidos por el administrador no son válidos, se muestra un mensaje de error.

Tabla 4.6: Especificación CU-04 - Registrar votante.

Requisito funcional	
Identificador	CU-05
Nombre	Votar
Fecha creación	08/02/2020
Autores	Jordi Miralles Comins
Versión	1
Actor principal	AC-02
Descripción	La aplicación debe permitir a un votante participar en una votación en la cual tiene acceso.
Relaciones	CU-02, CU-03, CU-04, CU-07
Precondición	El votante debe estar conectado a la VPN y autenticado en la aplicación. Por otro lado, el administrador debe haber iniciado la votación y el votante debe tener permiso para participar en ella.
Disparador	Un votante desea participar en una votación en la cual tiene acceso.
Secuencia normal	<ol style="list-style-type: none"> 1. El votante accede a la sección de Votaciones, selecciona la votación en la que desea participar y selecciona la opción Participar. 2. El sistema se queda en espera hasta que el administrador inicia la votación. 3. El votante escoge la opción deseada en la votación y envía el voto. 4. Después del proceso de cifrado y envío, se muestra el resultado de la votación.
Excepciones	Si se produce algún fallo o se detecta alguna alteración durante la votación, se anula la votación y se muestra un mensaje a todos los votantes.

Tabla 4.7: Especificación CU-05 - Votar.

Requisito funcional	
Identificador	CU-06
Nombre	Modificar votante
Fecha creación	08/02/2020
Autores	Jordi Miralles Comins
Versión	1
Actores principales	AC-01 & AC-02
Descripción	La aplicación debe permitir a un administrador de votaciones modificar los datos de un votante o a un votante modificar su propio perfil.
Precondición	El administrador o el votante debe estar conectado a la VPN y autenticado en la aplicación.
Disparador	Un administrador autenticado desea modificar los datos de un votante.
Secuencia normal	<ol style="list-style-type: none"> 1. El administrador accede a la sección de Participantes y selecciona un votante. 2. Dentro de este votante, el administrador selecciona la opción de Modificar Votante. 3. El sistema muestra los datos actuales del votante. 4. El administrador modifica los datos del votante. 5. El sistema comprueba la validez de los datos, muestra un resumen de estos y solicita la confirmación. 6. El administrador confirma la información. 7. El sistema modifica el votante y lo almacena en la base de datos.
Secuencia alternativa	<p>Un votante desea modificar su perfil.</p> <ol style="list-style-type: none"> 1. El votante accede a su perfil. <p>2-7. Mismos pasos que en la secuencia normal</p>
Excepciones	Si los datos introducidos no son válidos, se muestra un mensaje de error.

Tabla 4.8: Especificación CU-06 - Modificar votante.

Requisito funcional	
Identificador	CU-07
Nombre	Listar y visualizar votaciones
Fecha creación	08/02/2020
Autores	Jordi Miralles Comins
Versión	1
Actores principales	AC-01 & AC-02
Descripción	La aplicación debe permitir a un administrador o un votante consultar el listado y los datos de las votaciones a las que tiene acceso.
Relaciones	CU-01, CU-02, CU-03, CU-05
Precondición	El administrador o el votante debe estar conectado a la VPN y autenticado en la aplicación.
Disparador	Un administrador o votante desea visualizar el listado de votaciones a las cuales tiene acceso.
Secuencia normal	<ol style="list-style-type: none"> 1. El administrador/votante accede a la sección de Votaciones. 2. En esta sección, puede visualizar la lista de votaciones para las cuales tiene acceso. Los administradores visualizan todas las votaciones del sistema. 3. Desde aquí, puede acceder a una votación concreta para poder visualizar todos sus datos con detalle.
Comentarios	Se puede filtrar el listado por diferentes criterios como el estado, el departamento, el ámbito o la fecha.

Tabla 4.9: Especificación CU-07 - Listar y visualizar votaciones.

4.4. Requisitos de datos

Además de definir la funcionalidad básica del sistema, los casos de uso permiten identificar los datos, entidades y atributos más importantes que el sistema debe almacenar. En la tabla 4.10 se incluye un resumen con los requisitos de datos identificados en este sistema.

Identificador	Requisito	Descripción
RD-01	Votación	Este requisito de datos representa los datos que deben almacenarse sobre una votación.
RD-02	Votante	Este requisito de datos representa los datos que deben almacenarse sobre un votante.
RD-03	Administrador de votaciones	Este requisito de datos representa los datos que deben almacenarse sobre un administrador de votaciones.

Tabla 4.10: Resumen de los requisitos de datos del sistema.

En las siguientes tablas (desde la 4.11 a la 4.13) se realiza la especificación de estos requisitos de datos.

Requisito de datos	
Identificador	RD-01
Nombre	Votación
Fecha creación	09/02/2020
Autores	Jordi Miralles Comins
Versión	1
Fuente	AC-01
Tipos	Datos de entrada y de salida
Datos específicos	Pregunta de la votación, descripción, estado, fecha de votación, fecha de creación, fechas de apertura y cierre, ámbito, departamento, listado de opciones, listado de participantes y resultados.
Comentarios	<p>Estado: Puede ser:</p> <ul style="list-style-type: none"> ▪ Creada → La votación ha sido creada pero aún no se ha iniciado. ▪ Activa → La votación ha sido iniciado y los votantes autorizados pueden participar. ▪ Finalizada → La votación ha sido finalizada. <hr/> <p>Fecha de votación: Fecha en la que se realizará la votación.</p> <hr/> <p>Fecha de creación: Fecha en la que el administrador crea la votación.</p> <hr/> <p>Fecha de apertura: Fecha en la que el administrador inicia la votación, momento en el cual se puede empezar a votar.</p> <hr/> <p>Fecha de cierre: Fecha en la que se obtienen los resultados de la votación y esta queda cerrada.</p> <hr/> <p>Ámbito: Puede ser:</p> <ul style="list-style-type: none"> ▪ Pública → Cualquier votante puede consultar la votación, las opciones y los resultados. ▪ Privada → Solo los participantes de la votación pueden consultar los datos de la votación. ▪ Departamento → Solo los votantes que pertenezcan al departamento de la votación pueden consultar los datos de la votación. ▪ Oculto → Ningún votante puede consultar los datos de la votación.

Tabla 4.11: Especificación RD-01 - Votación.

Requisito de datos	
Identificador	RD-02
Nombre	Votante
Fecha creación	09/02/2020
Autores	Jordi Miralles Comins
Versión	1
Fuente	AC-01 & AC-02
Tipos	Datos de entrada y de salida
Datos específicos	Nombre, apellidos, mail, DNI, teléfono, cargo, departamento, contraseña, fecha de registro, DNI_admin, claves RSA y IP_VPN.
Comentarios	Cargo: Cargo que ocupa en la empresa u organización. (<i>e.g.</i> secretario, presidente, etc)
	DNI_admin: DNI del administrador que registra el votante.
	Claves RSA: Claves pública y privada RSA asociadas al votante. Estas serán utilizadas para la votación.
	IP_VPN: IP asignada al votante dentro de la red VPN.

Tabla 4.12: Especificación RD-02 - Votante.

Requisito de datos	
Identificador	RD-03
Nombre	Administrador de votaciones
Fecha creación	09/02/2020
Autores	Jordi Miralles Comins
Versión	1
Fuente	AC-01 & Administrador del sistema
Tipos	Datos de entrada y de salida
Datos específicos	Nombre, apellidos, mail, DNI, teléfono, cargo, departamento, contraseña, fecha de autorización, claves RSA y IP_VPN.
Comentarios	Cargo, claves RSA y IP_VPN: Mismos datos que en los votantes.
	Fecha de autorización: Fecha en la que se le autorizó la gestión de votaciones.

Tabla 4.13: Especificación RD-03 - Administrador de votaciones.

4.5. Diagrama de clases

En el presente apartado se muestra el diagrama de clases del sistema, modelado tras identificar los requisitos funcionales y las entidades más importantes del sistema. Este [28] permite mostrar las diferentes clases, sus atributos, sus operaciones básicas y las relaciones entre estas.

En la figura 4.2 se incluye este diagrama.

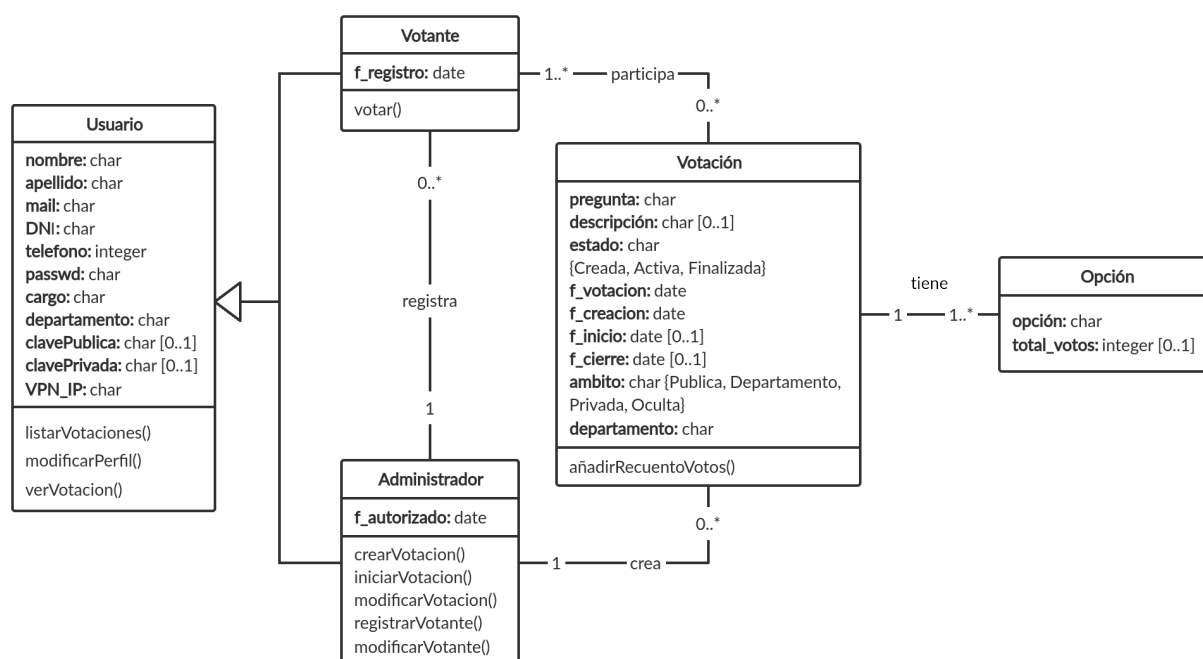


Figura 4.2: Diagrama de clases del sistema.

En este diagrama, se pueden visualizar cinco clases básicas que modelan el sistema implementado. Estas clases son:

- **Usuario:** Esta clase representa todos los usuarios registrados en el sistema. Las claves pública y privada RSA son opcionales, ya que se generan tras el primer acceso del usuario.
- **Votante:** Esta clase representa todos los usuarios del sistema que pueden participar en procesos de votación.
- **Administrador:** Esta clase representa todos los usuarios del sistema con permisos de administrador para poder crear y gestionar votaciones y votantes.
- **Votación:** Esta clase representa todas las votaciones del sistema. La descripción no es un atributo obligatorio y las fechas de inicio y fin son opcionales porque se añaden en la apertura y cierre respectivamente.
- **Opción:** Esta clase representa las opciones y los resultados de cada una de las votaciones del sistema. El recuento de votos no es obligatorio porque se introduce al finalizar la votación.

Además, estas clases se encuentran relacionadas mediante cuatro asociaciones que representan la funcionalidad del sistema:

- **Registra:** Esta asociación representa la relación entre los administradores y los votantes que registra. Tiene una multiplicidad de uno a muchos porque un administrador puede registrar uno o más votantes y un votante es registrado por un administrador.
- **Crea:** Esta asociación representa la relación entre un administrador y las votaciones que crea. Su multiplicidad es uno a muchos porque un administrador crea una o más votaciones y una votación es creada por un administrador.
- **Participa:** Esta asociación representa la relación entre un votante y las votaciones en las que participa. Tiene una multiplicidad de muchos a muchos porque un votante participa en muchas votaciones y en una votación participan muchos votantes.
- **Tiene:** Esta asociación representa la relación entre las votaciones y las opciones que esta presenta. Su multiplicidad es uno a muchos porque una votación tiene muchas opciones y una opción pertenece a una votación.

4.6. Diagramas de actividades

Un diagrama de actividades o de flujo [28] es una técnica UML que permite representar de forma gráfica un proceso o una secuencia de pasos. En la figura 4.3 se observa el diagrama de flujo que representa el acceso autorizado al sistema.

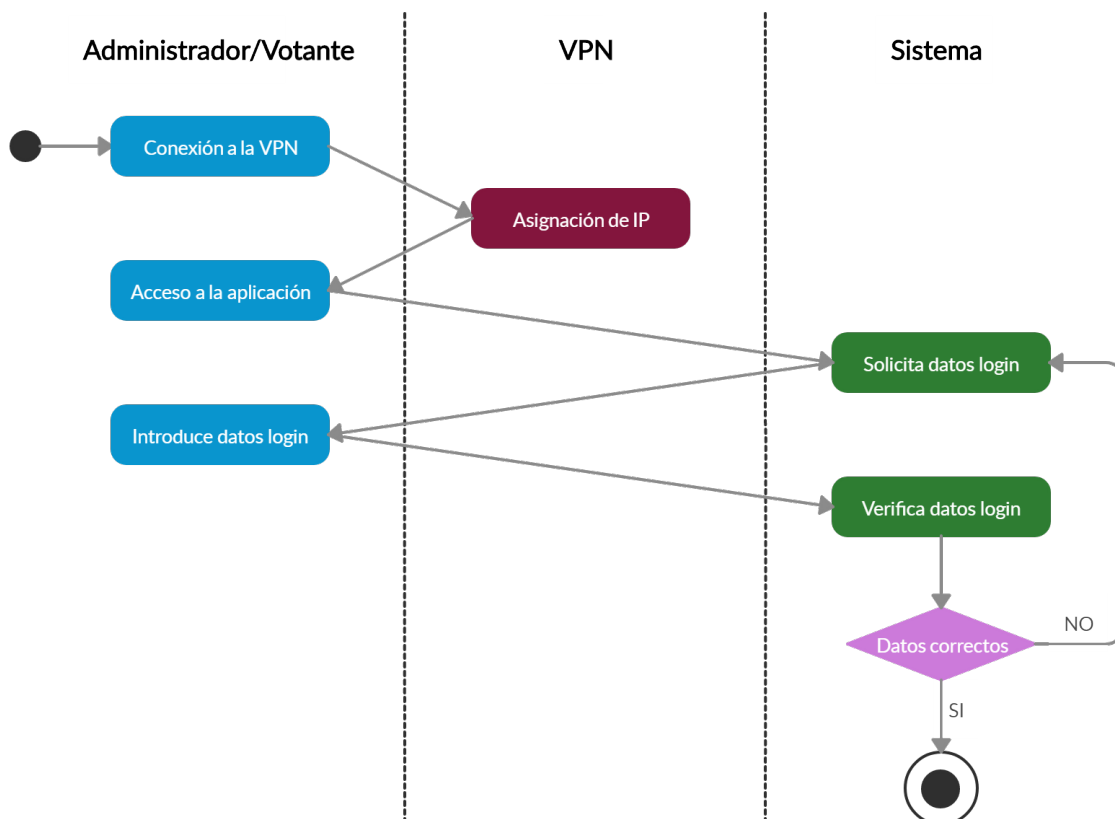


Figura 4.3: Diagrama de actividades - Acceso al sistema.

Por otro lado, en la figura 4.4 se puede visualizar el diagrama de actividades que modela los pasos en un proceso de votación.

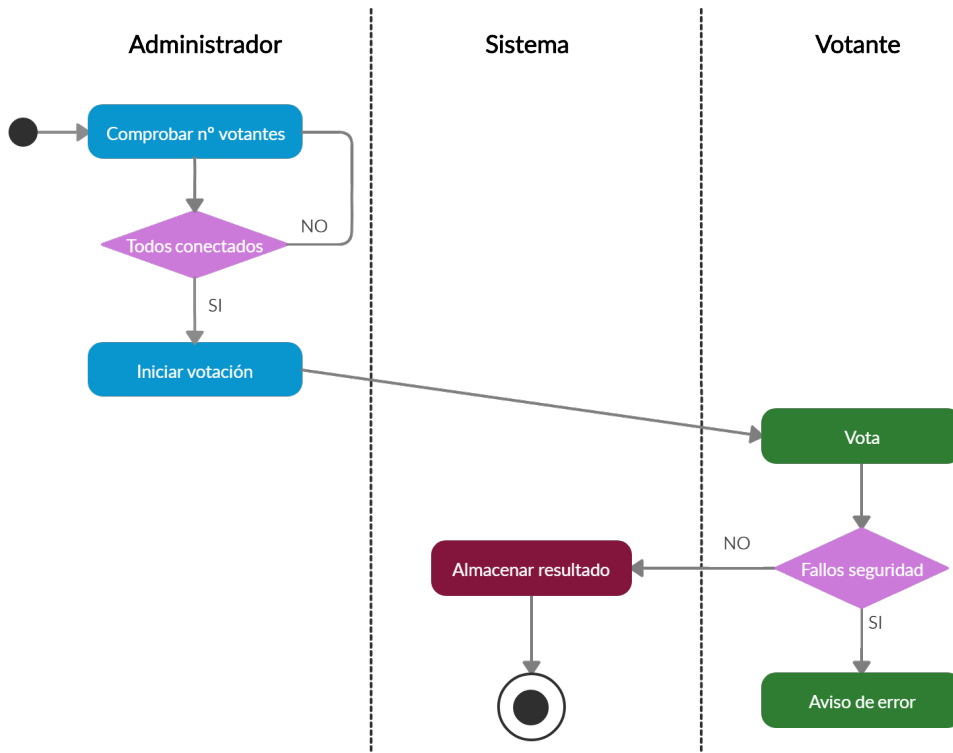


Figura 4.4: Diagrama de actividades - Proceso de votación.

Finalmente, en la figura 4.5 se visualiza el diagrama de flujo que representa el proceso de creación de una votación o el registro de un nuevo votante por parte de un administrador.

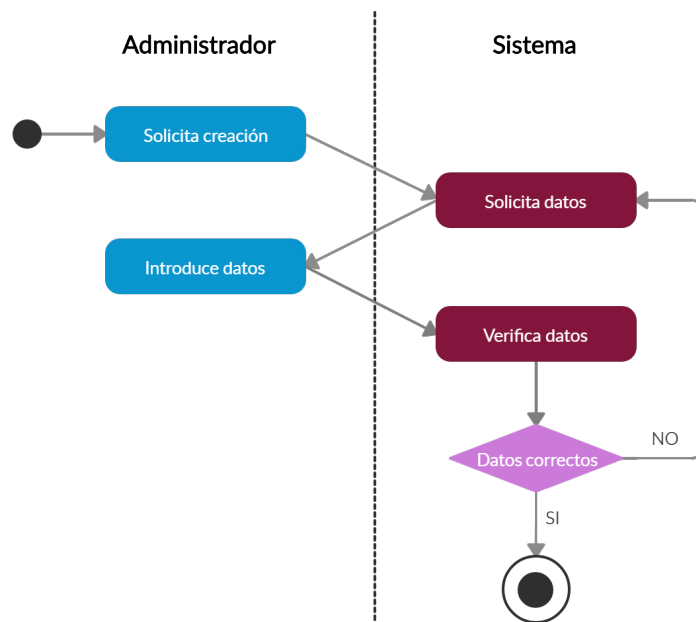


Figura 4.5: Diagrama de actividades - Nueva votación/votante.

Capítulo 5

Diseño del sistema

Contenidos

5.1. Diseño de la arquitectura del sistema	61
5.2. Diseño de la base de datos	62
5.2.1. Diseño conceptual	62
5.2.2. Diseño lógico	62
5.2.3. Diseño físico	64
5.3. Diseño de la interfaz de usuario	66
5.3.1. <i>Sitemap</i>	66
5.3.2. Guía de estilo	66
5.3.3. Prototipos	68

En el presente capítulo se describe el diseño detallado del sistema implementado. Por un lado, se muestra el diseño de la arquitectura del sistema, que incluye el modelado de la base de datos que almacena toda la información. Por otro lado, se incorpora un estudio del estilo y el diseño de las interfaces de usuario de la aplicación web.

5.1. Diseño de la arquitectura del sistema

Tal como se ha visto en la figura 1.2 y en la sección 2.2.3, el sistema se compone de distintos servidores que aportan una funcionalidad específica.

Sin embargo, es importante destacar que estos servidores se encuentran dentro de la VPN. En otras palabras, estos servicios se despliegan dentro de la red privada virtual y solo se puede acceder a ellos a través de esta. De esta forma, se consigue un aumento en el nivel de seguridad del sistema.

En la figura 5.1 se representa esta arquitectura.

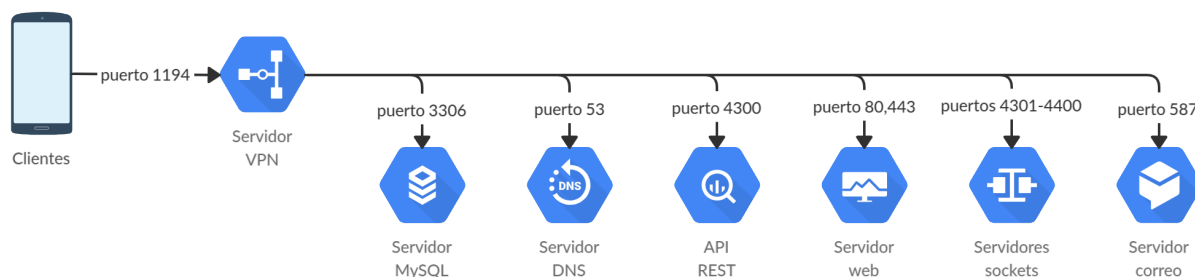


Figura 5.1: Diseño de la arquitectura del sistema.

5.2. Diseño de la base de datos

La base de datos es un módulo importante en la arquitectura del sistema. Por ello, es necesario realizar un diseño adecuado de esta.

5.2.1. Diseño conceptual

En primer lugar, en la figura 5.2 se muestra el diseño conceptual de la base de datos [29], un diagrama entidad-relación¹ creado a partir del diagrama de clases y los requisitos del sistema.

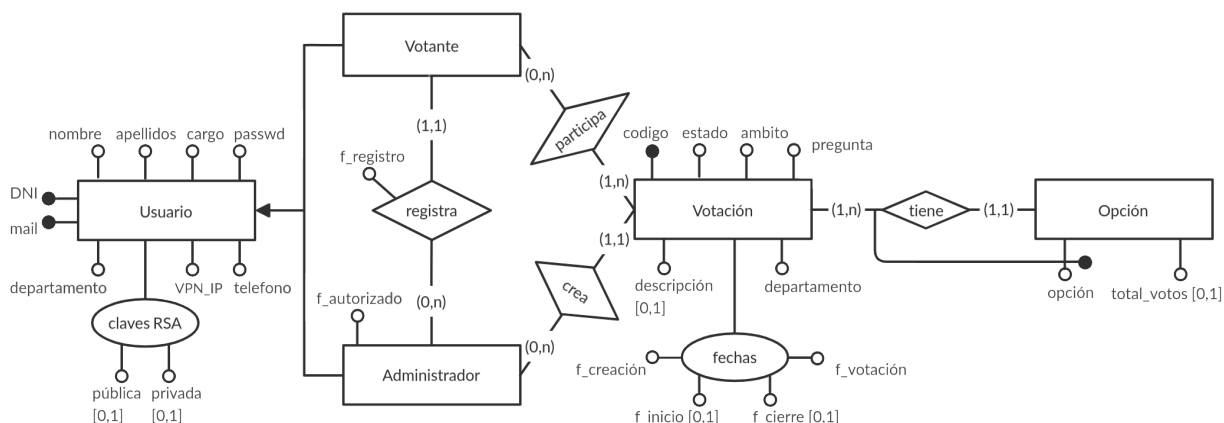


Figura 5.2: Diseño conceptual de la base de datos.

5.2.2. Diseño lógico

A partir del modelo conceptual, se construye el esquema lógico relacional de la base de datos [29], una fuente de información para la implementación de la base de datos. En la figura 5.3 se observa un esquema del diseño lógico del sistema.

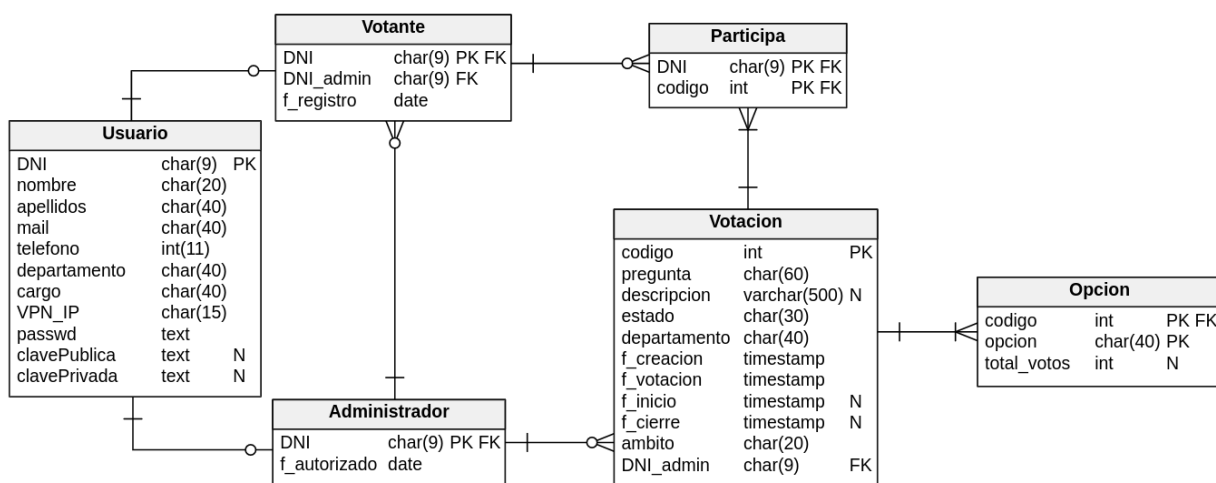


Figura 5.3: Diseño lógico de la base de datos.

¹ El modelo **Entidad-Relación** permite representar las entidades principales de una base de datos.

A continuación, se incluye la definición del diseño lógico mostrado en la figura anterior.

USUARIO (DNI, nombre, apellidos, mail, telefono, departamento, cargo, VPN_IP, passwd, clavePublica, clavePrivada)

mail es clave alternativa

clavePublica y clavePrivada aceptan nulos

VOTANTE (DNI, DNI_admin, f_registro)

VOTANTE $\xrightarrow{\text{DNI}}$ USUARIO **N P P**

VOTANTE $\xrightarrow{\text{DNI_admin}}$ ADMINISTRADOR **N P R**

ADMINISTRADOR (DNI, f_authorized)

ADMINISTRADOR $\xrightarrow{\text{DNI}}$ USUARIO **N P P**

VOTACION (codigo, pregunta, descripcion, estado, departamento, f_creacion, f_votacion, f_inicio, f_cierre, ambito, DNI_admin)

descripcion, f_inicio y f_cierre aceptan nulos

ambito \in {Publica, Departamento, Privada, Oculta}

estado \in {Creada, Activa, Finalizada}

VOTACION $\xrightarrow{\text{DNI_admin}}$ ADMINISTRADOR **N P R**

PARTICIPA (DNI, codigo)

PARTICIPA $\xrightarrow{\text{DNI}}$ VOTANTE **N P R**

PARTICIPA $\xrightarrow{\text{codigo}}$ VOTACION **N P R**

OPCION (codigo, opcion, total_votos)

total_votos acepta nulos

OPCION $\xrightarrow{\text{codigo}}$ VOTACION **N P P**

5.2.3. Diseño físico

Finalmente, se realiza el diseño físico de la base de datos [29], que representa la implementación de esta en el SGBD escogido, es decir, en MySQL. En los siguientes códigos (5.1 a 5.6) se incluye este diseño.

```
1 CREATE TABLE Usuario (  
2     DNI char(9) NOT NULL,  
3     nombre char(20) NOT NULL,  
4     apellidos char(40) NOT NULL,  
5     mail char(40) NOT NULL,  
6     telefono int(11) NOT NULL,  
7     departamento char(40) NOT NULL,  
8     cargo char(40) NOT NULL,  
9     VPN_IP char(15) DEFAULT NULL,  
10    passwd text NOT NULL,  
11    clavePublica text NULL,  
12    clavePrivada text NULL,  
13    UNIQUE INDEX Usuario_ak_1 (mail),  
14    CONSTRAINT Usuario_pk PRIMARY KEY (DNI)  
15 )
```

Código Fuente 5.1: Diseño físico - Usuario

```
1 CREATE TABLE Administrador (  
2     DNI char(9) NOT NULL,  
3     f_authorized datetime NOT NULL DEFAULT CURRENT_TIMESTAMP,  
4     CONSTRAINT Administrador_pk PRIMARY KEY (DNI)  
5 )  
6  
7 ALTER TABLE Administrador  
8     ADD CONSTRAINT Administrador_Usuario FOREIGN KEY (DNI) REFERENCES  
9     ↪ Usuario (DNI) ON DELETE CASCADE ON UPDATE CASCADE;
```

Código Fuente 5.2: Diseño físico - Administrador

```
1 CREATE TABLE Votante (  
2     DNI char(9) NOT NULL,  
3     DNI_admin char(9) NOT NULL,  
4     f_registro datetime NOT NULL DEFAULT CURRENT_TIMESTAMP,  
5     CONSTRAINT Votante_pk PRIMARY KEY (DNI)  
6 )  
7  
8 ALTER TABLE Votante  
9     ADD CONSTRAINT Admin_Votante FOREIGN KEY (DNI_admin) REFERENCES  
10    ↪ Administrador (DNI) ON DELETE CASCADE ON UPDATE RESTRICT,  
11    ADD CONSTRAINT Votante_Usuario FOREIGN KEY (DNI) REFERENCES  
12    ↪ Usuario (DNI) ON DELETE CASCADE ON UPDATE CASCADE;
```

Código Fuente 5.3: Diseño físico - Votante


```

1 CREATE TABLE Votacion (
2     codigo int(11) NOT NULL AUTO_INCREMENT,
3     pregunta char(60) NOT NULL,
4     descripcion varchar(500) DEFAULT NULL,
5     estado char(30) NOT NULL,
6     departamento char(40) DEFAULT NULL,
7     f_creacion datetime NOT NULL DEFAULT CURRENT_TIMESTAMP,
8     f_votacion datetime NOT NULL,
9     f_inicio datetime DEFAULT NULL,
10    f_cierre datetime DEFAULT NULL,
11    ambito char(20) NOT NULL,
12    DNI_admin char(9) NOT NULL,
13    CHECK (ambito IN ('Publica','Departamento','Privada','Oculto')),
14    CHECK (estado IN ('Creada', 'Activa', 'Finalizada')),
15    CONSTRAINT Votacion_pk PRIMARY KEY (codigo)
16 )
17
18 ALTER TABLE Votacion
19     ADD CONSTRAINT Admin_Votacion FOREIGN KEY (DNI_admin) REFERENCES
    → Administrador (DNI) ON DELETE CASCADE ON UPDATE RESTRICT;

```

Código Fuente 5.4: Diseño físico - Votación

```

1 CREATE TABLE Opcion (
2     codigo int(11) NOT NULL,
3     opcion char(40) NOT NULL,
4     total_votos int(11) DEFAULT NULL,
5     CONSTRAINT Opcion_pk PRIMARY KEY (codigo,opcion)
6 )
7
8 ALTER TABLE Opcion
9     ADD CONSTRAINT Opcion_Votacion FOREIGN KEY (codigo) REFERENCES
    → Votacion (codigo) ON DELETE CASCADE ON UPDATE CASCADE;

```

Código Fuente 5.5: Diseño físico - Opción

```

1 CREATE TABLE Participa (
2     DNI char(9) NOT NULL,
3     codigo int(11) NOT NULL,
4     CONSTRAINT Participa_pk PRIMARY KEY (DNI,codigo)
5 )
6
7 ALTER TABLE Participa
8     ADD CONSTRAINT Participa_Votante FOREIGN KEY (DNI) REFERENCES
    → Votante (DNI) ON DELETE CASCADE ON UPDATE RESTRICT,
9     ADD CONSTRAINT Votacion_Participa FOREIGN KEY (codigo) REFERENCES
    → Votacion (codigo) ON DELETE CASCADE ON UPDATE RESTRICT;

```

Código Fuente 5.6: Diseño físico - Participa

5.3. Diseño de la interfaz de usuario

La aplicación web es una de las partes más importantes de este proyecto, por lo que se requiere un diseño adecuado. Para ello, es necesario realizar un proceso centrado en el usuario que esté respaldado por las ocho reglas de oro del diseño web definidas por B. Shneiderman [30].

En este diseño, se ofrece una consistencia visual entre las diferentes páginas, manteniendo los mismos colores, fuentes y tamaños. Además, tal como se observa más adelante, se utilizan iconos y elementos que facilitan el uso de la aplicación. Por otro lado, también se muestran mensajes de ayuda y advertencias para evitar que sucedan errores e informar sobre las acciones realizadas.

5.3.1. Sitemap

El *sitemap* o mapa de navegación muestra las diferentes páginas que componen una aplicación web y las transiciones entre ellas. En la figura 5.4 podemos ver el mapa de la aplicación web del proyecto.

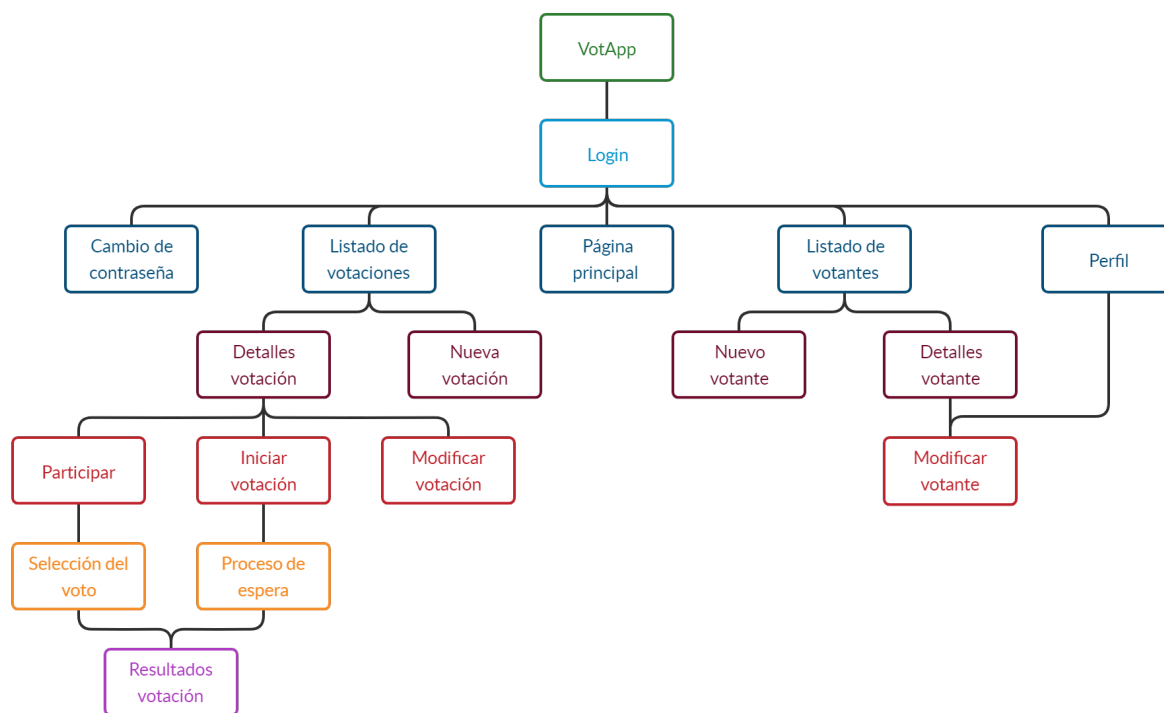


Figura 5.4: *Sitemap* de la aplicación web.

5.3.2. Guía de estilo

Una guía de estilos es una recopilación de las reglas y los estándares seguidos para el desarrollo de una interfaz web. Esta permite mantener la coherencia del sitio web a través de las diferentes páginas.

Colores

En la figura 5.5 se muestran los colores principales utilizados en la web.



Figura 5.5: Guía de estilos - Paleta de colores.

Fuente y tamaño

La fuente utilizada en la aplicación web ha sido **Titillium Web**, que se puede encontrar en:

<https://fonts.google.com/specimen/Titillium+Web?query=Titillium>

En la figura 5.6 se observa esta fuente y los diferentes tamaños utilizados en las interfaces. En general, se han utilizado los dos tamaños más grandes para los títulos y subtítulos, el tercero para los datos de las votaciones y el cuarto para el texto general. El tamaño más pequeño se ha utilizado en los mensajes de error y las alertas.

VotApp - 32px
VotApp - 24px
VotApp - 20px
VotApp - 16px
VotApp - 11px

Figura 5.6: Guía de estilos - Fuente y tamaños.

Iconos

En la interfaz web se ha hecho uso de diferentes iconos que guían las acciones de los usuarios de forma visual. Para ello, se ha utilizado la herramienta de Font Awesome [31], que permite añadir iconos de forma sencilla y gratuita.

En la figura 5.7 se incluyen algunos de los iconos utilizados en la aplicación:



Figura 5.7: Guía de estilos - Iconos de Font Awesome.

Además, la aplicación incluye un logotipo que ha sido creado manualmente y se puede ver en la figura 5.8.



Figura 5.8: Guía de estilos - Logotipo de la aplicación.

Elementos

Para facilitar el diseño de la web se ha utilizado la herramienta Angular Material [32], la cual permite integrar componentes con estilo pre configurado al proyecto. En la figura 5.9 se incluyen algunos de los elementos utilizados en el proyecto.

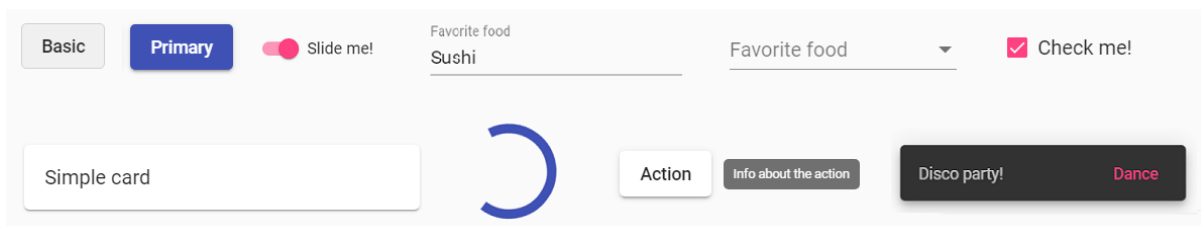


Figura 5.9: Guía de estilos - Elementos de Angular Material.

5.3.3. Prototipos

Tras analizar e identificar los requisitos funcionales y de datos del sistema, se han diseñado unos prototipos que muestran la apariencia general de la aplicación web.

En primer lugar, en la figura 5.10 se observa un prototipo del panel principal que se visualiza tras el inicio de sesión de un administrador. Desde aquí, este puede acceder por ejemplo al panel de registro de nuevos votantes, que se visualiza también en la figura.

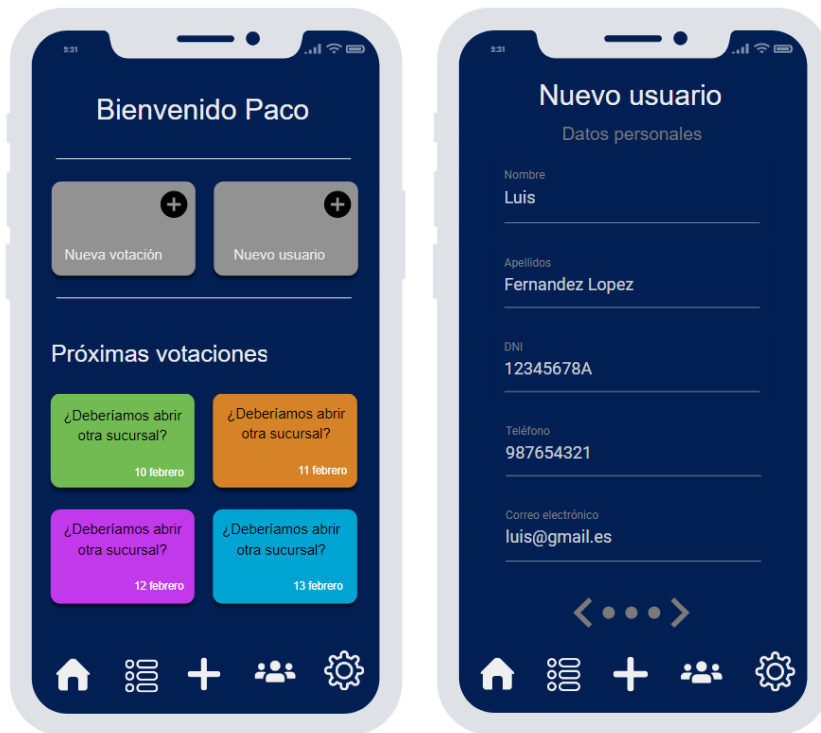


Figura 5.10: Prototipos - Panel principal y Nuevo Votante.

A continuación, en la figura 5.11 se observa el panel con el listado de los usuarios registrados en el sistema. Este panel sólo es accesible para administradores. También se observa un prototipo de la interfaz con el perfil y los datos de los usuarios.

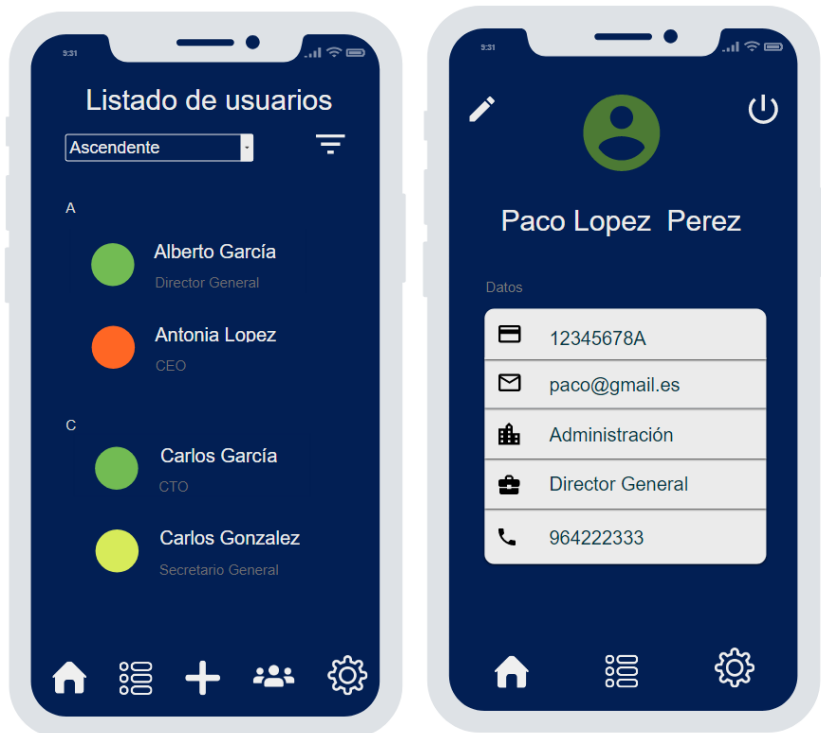


Figura 5.11: Prototipos - Listado usuarios y Perfil.

Finalmente, en la figura 5.12 se incluyen algunos prototipos de la sección de votaciones, la más importante de la aplicación. Se observa el listado con todas las votaciones, los detalles de una votación concreta, el panel para participar en una votación y la interfaz con los resultados de esta.



Figura 5.12: Prototipos - Listado votaciones, Detalles votación, Votar y Resultados.

Capítulo 6

Implementación del sistema

Contenidos

6.1. Patrones diseño	71
6.2. Implementación de servicios	73
6.2.1. Red Privada Virtual	73
6.2.2. Servidor de Nombres de Dominio	74
6.2.3. Base de datos	75
6.3. Implementación del <i>backend</i>	76
6.4. Implementación del <i>frontend</i>	82
6.4.1. Módulo de inicio de sesión	85
6.4.2. Módulo de votaciones	85
6.4.3. Módulo de usuarios	90
6.4.4. Módulo para los procesos de votación	92
6.5. Implementación del protocolo de votación	94
6.5.1. Proceso de cifrado	94
6.5.2. Proceso de envíos	96

En el presente capítulo se muestra el proceso de implementación del sistema que se ha identificado en los casos de uso y se ha diseñado posteriormente. Se describen los patrones de diseño de *software* utilizados durante el desarrollo y las decisiones más importantes que se han tomado durante la construcción de los diferentes módulos.

También se muestra el resultado final del desarrollo mediante imágenes de las interfaces implementadas y de la aplicación en funcionamiento.

6.1. Patrones diseño

Un patrón de diseño [33] es una técnica de *software* que permite resolver problemas conocidos durante el desarrollo de un proyecto informático. De esta forma, se puede estructurar y reutilizar el código facilitando la programación de los sistemas.

En este proyecto, se ha hecho uso del patrón de diseño MVC (Modelo-Vista-Controlador), un patrón muy utilizado en el desarrollo de proyectos informáticos con interfaz de usuario.

MVC [33] define tres actores principales con una funcionalidad específica, separando el proyecto en tres capas distintas.

- **Modelo:** Se encarga de la gestión de la información y los datos del sistema, es decir, de la capa de acceso a datos.
- **Vista:** Representa la interfaz gráfica de la aplicación con la que interaccionan los usuarios, y por tanto, gestiona la capa de aplicación.
- **Controlador:** Es el responsable de controlar la comunicación entre la vista y el modelo, es decir, entre las acciones de los usuarios y los datos del sistema. Se conoce como la capa de lógica de negocio.

El patrón MVC se ha utilizado tanto en la implementación de la aplicación web o *frontend* como en el desarrollo del servidor o *backend* que gestiona el sistema.

En el *backend*, el código fuente se ha repartido en tres directorios distintos, como se observa en la figura 6.1. El directorio **Routes** contiene todas las rutas de la API y representa la vista. Por otro lado, en el directorio **Models** se incluyen todas las consultas sobre la base de datos (inserción, consulta, borrado..). La lógica que gestiona la comunicación entre estos dos se encuentra en el directorio **Controllers**.

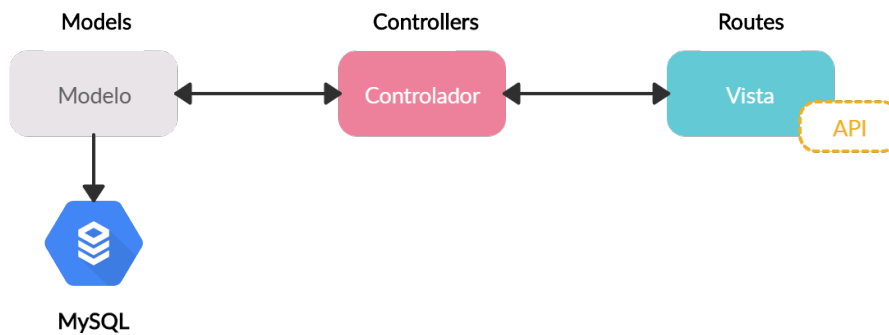


Figura 6.1: Patrón MVC en el *backend*.

Por otro lado, en el *frontend* también existen tres tipos de ficheros distintos, como se observa en la figura 6.2. Los ficheros HTML y CSS implementan la vista de la aplicación, mientras que los servicios que realizan las peticiones HTTP a la API representan el papel del modelo. La comunicación entre ambos se realiza a través de los ficheros TypeScript de Angular.

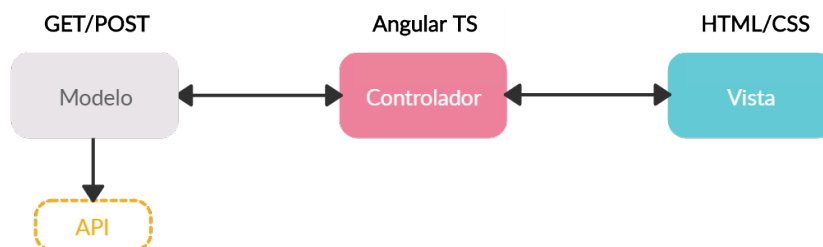


Figura 6.2: Patrón MVC en el *frontend*.

6.2. Implementación de servicios

En este apartado, se describen los pasos llevados a cabo para desplegar los servicios que permiten el funcionamiento del sistema: la VPN, el servidor de DNS y la base de datos.

6.2.1. Red Privada Virtual

Como ya se ha comentado anteriormente, la VPN se ha desarrollado con la herramienta de código abierto OpenVPN en un servidor de tipo Unix. Para implementar una red privada virtual funcional, se realizaron los pasos explicados a continuación.

Configurar y gestionar los certificados

OpenVPN funciona mediante túneles TLS para transportar la información de forma segura y cifrada. Por ello, el primer paso consistió en crear una CA¹ propia que se encarga de generar y firmar los diferentes certificados.

Para crear y configurar la CA, se ha hecho uso de la utilidad `easy-rsa`, que permite gestionar una Autoridad Certificadora y generar certificados de forma sencilla. En general, es necesario generar un certificado para la CA, otro para el servidor VPN y uno por cada cliente o usuario que vaya a hacer uso de la VPN.

Configurar y desplegar el servicio OpenVPN

Tras gestionar todos los aspectos relacionados con los certificados, se configuran los parámetros del servidor de VPN mediante los ficheros de configuración. Entre los múltiples aspectos que se pueden configurar en el servicio de OpenVPN, en el proyecto destacan los que se observan en el código fuente 6.1.

```
1 ca "/etc/openvpn/certificates/ca.pem"
2 cert "/etc/openvpn/certificates/cert.pem"
3 key "/etc/openvpn/certificates/key.pem"
4 dh "/etc/openvpn/certificates/dh.pem"
5 tls-auth "/etc/openvpn/certificates/takey.pem"
6 port 1194
7 proto udp
8 server 192.168.210.0 255.255.255.0
9 push "route 192.168.210.0 255.255.255.0"
10 push "dhcp-option DNS 192.168.210.1"
```

Código Fuente 6.1: Configuración del servidor OpenVPN.

¹ Una CA (Autoridad Certificadora) es una entidad de confianza que emite certificados electrónicos.

Estas directivas representan la siguiente funcionalidad:

- 1-3. Indican las rutas del certificado de la CA y el certificado y la llave del servidor VPN.
- 4-5. Establece la ruta a la llave fuerte Diffie-Hellman² y la firma HMAC³ del servidor.
- 6-7. Establecen el puerto y el protocolo de las conexiones con los clientes.
8. Indica la red y la máscara de la red privada virtual.
9. Indica a los clientes que todo el tráfico dirigido a la red indicada (la de la propia red privada) se debe enviar a través del túnel.
10. Indica a los clientes que servidor DNS deben usar. En este caso, se indica la dirección IP del servidor DNS creado en el proyecto.

Una vez configurados todos los parámetros y arrancado el servicio de OpenVPN, la red privada virtual ya se encuentra operativa. Sin embargo, es necesario activar la redirección de paquetes para que funcionen las conexiones a través de la red. Para ello, basta con añadir la directiva `net.ipv4.ip_forward=1` al fichero de configuración del sistema `/etc/sysctl.conf`.

Configurar y crear nuevos usuarios

Como ya se ha comentado anteriormente, es necesario generar un certificado para cada uno de los usuarios que quiera hacer uso del servicio de VPN.

Una vez generado el certificado, se debe crear un fichero con extensión `.ovpn` que contenga el certificado y la llave del cliente, el certificado del servidor y la firma HMAC. Además, este fichero debe incluir también algunas directivas de configuración que le indiquen, entre otras cosas, la dirección del servidor OpenVPN, el puerto y el protocolo.

6.2.2. Servidor de Nombres de Dominio

Tal como se ha comentado en la sección 2.3.4, en el proyecto se implementa un servidor DNS para poder desplegar la aplicación web progresiva sobre el protocolo HTTPS. Por ese motivo, se ha desarrollado un servidor de nombres de dominio que permite acceder al servidor web a través de su dominio, en este caso, `jordi.tfg`.

Para desplegar el servicio se ha hecho uso de la utilidad de *software* libre BIND9 sobre un servidor de tipo Unix. Esta herramienta permite montar un servidor DNS siguiendo unos sencillos pasos.

En primer lugar, es necesario indicar al propio servidor DNS que se debe utilizar a sí mismo para resolver las peticiones. Para ello, se sustituye el contenido del fichero `/etc/resolv.conf` por las directivas `nameserver 127.0.0.1` y `search tfg`.

² Una **llave fuerte Diffie-Hellman** facilita el intercambio de llaves entre el servidor y cada cliente.

³ Una **firma HMAC** (*Hashed Message Authentication Code*) permite verificar la integridad de los datos entre cliente y servidor.

A continuación, se modifica el archivo de configuración `/etc/bind/named.conf.local`, indicando la zona DNS⁴ que el servidor abarca y su zona correspondiente para resoluciones inversas⁵, tal como se muestra en el código fuente 6.2.

```
1     zone "tfg" {
2         type master;
3         file "/etc/bind/db.tfg";
4     };
5     zone "210.168.192.in-addr.arpa" {
6         type master;
7         file "/etc/bind/db.192";
8     };
```

Código Fuente 6.2: Configuración de las zonas DNS.

Como se puede observar, se indican las zonas a abarcar, el tipo de servidor DNS (en este caso maestro⁶) y el fichero en el que se incluye la configuración de cada zona.

El último paso consiste en configurar los ficheros que se encargan de gestionar cada zona definida anteriormente (`/etc/bind/db.tfg` y `/etc/bind/db.192`), incluyendo la correspondencia entre las direcciones IP y los nombres de dominio. En este caso, se ha relacionado el dominio `jordi.tfg` con la IP 192.168.210.1, correspondiente al servidor web y la API dentro de la VPN.

6.2.3. Base de datos

Para implementar la base de datos, basta con realizar una instalación de los paquetes MySQL y phpMyAdmin en Unix. Después, desde una terminal, se accede al servidor de bases de datos y se crea un usuario para poder acceder desde phpMyAdmin, utilizando los siguientes comandos:

- `CREATE USER 'jordi'@'localhost' IDENTIFIED BY 'password';`
- `GRANT ALL PRIVILEGES ON *.* TO 'jordi'@'localhost' WITH GRANT OPTION;`

A continuación, ya en la herramienta de phpMyAdmin, se crea una nueva base de datos y se introducen en esta las sentencias del diseño físico modelado en la sección 5.2.3. No obstante, este diseño se ha visto modificado durante la implementación debido a ciertas necesidades. En concreto, se han añadido dos campos nuevos:

- En la tabla **Votación**, se ha añadido un campo `OptionalError` de tipo texto y que acepta nulos. Este campo representa los posibles errores que una votación puede presentar y facilita la gestión de estos errores en la aplicación web.
- En la tabla **Participa**, se ha añadido un campo `temporal_socket` de tipo texto y que acepta nulos. Este permite a los votantes consultar el *socket* al que deben conectarse durante el proceso de votación.

⁴ Una **zona DNS** es un conjunto de dominios sobre los que un servidor DNS tiene control (*e.g.* zona `.com`).

⁵ Una **resolución DNS inversa** es aquella en la que se obtiene el dominio a partir de la dirección IP.

⁶ Un **servidor DNS maestro** es aquel que almacena los ficheros de configuración de las zonas que controla.

6.3. Implementación del *backend*

Como ya se ha comentado anteriormente, el desarrollo del controlador del lado del servidor se ha realizado usando el *framework* Node.js. Entre otras cosas, se ha desplegado el servidor web de la aplicación, la API y los métodos de interacción con la base de datos, los servidores de *sockets* y la autenticación de los usuarios mediante *tokens* JWT.

Este proyecto se ha estructurado en diferentes directorios, como se observa en la figura 6.3.

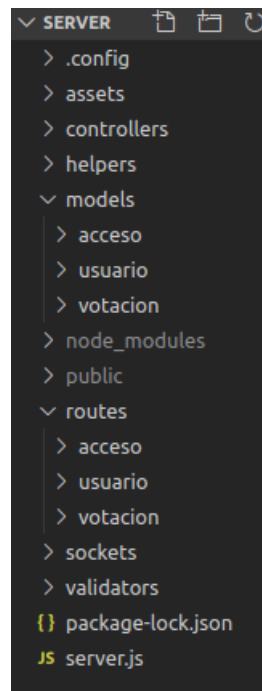


Figura 6.3: Estructura de los directorios del proyecto en el *backend*.

Cada uno de estos directorios contiene el código fuente con una funcionalidad específica:

- El directorio `.config` contiene las variables de entorno del proyecto y los certificados y llaves utilizados para el servidor web, los *tokens* JWT y las notificaciones *push*.
- En `assets` se incluyen las plantillas utilizadas para el envío de correos electrónicos.
- Como se ha comentado en la sección 6.1, el directorio `controllers` contiene el código fuente que gestiona la interacción entre la API y el acceso a la base de datos.
- `Helpers` incluye el código auxiliar que añade funcionalidad al proyecto, como el envío de correos, la generación de *tokens* JWT o el cifrado de contraseñas.
- El modelo que gestiona el acceso y la interacción con la base de datos se incluye en el directorio `models`. Este se divide en varios subdirectorios de forma que se organiza mejor el código.

- **Public** contiene todo el código fuente que el servidor web sirve a los clientes cuando acceden a la aplicación web.
- Como también se comentó, **routes** contiene todas las rutas de acceso a la API, es decir, los *endpoints* o las URI a los que se puede acceder.
- El directorio **Sockets** incluye todo el código fuente que permite gestionar y controlar el funcionamiento de los *sockets* durante los procesos de votación.
- **Validators** contiene la funcionalidad que permite llevar un control sobre los datos introducidos por el usuario antes de interactuar con la base de datos. En general, se controla que todos los campos estén bien formados y no superen el número máximo de caracteres.

Servidor web y API

En la estructura anterior, además del listado de directorios, se puede ver un archivo `server.js`, el cual incluye el código fuente necesario para desplegar tres servidores web:

- Un servidor web HTTPS en el puerto **443**, que sirve el contenido de la aplicación web a los clientes.
- Un servidor web HTTP en el puerto **80**, que redirige el tráfico al servidor HTTPS anterior.
- Un servidor web HTTPS en el puerto **4300**, que aloja la API con las rutas de acceso.

En la figura 6.4 se muestra la relación entre estos servidores.

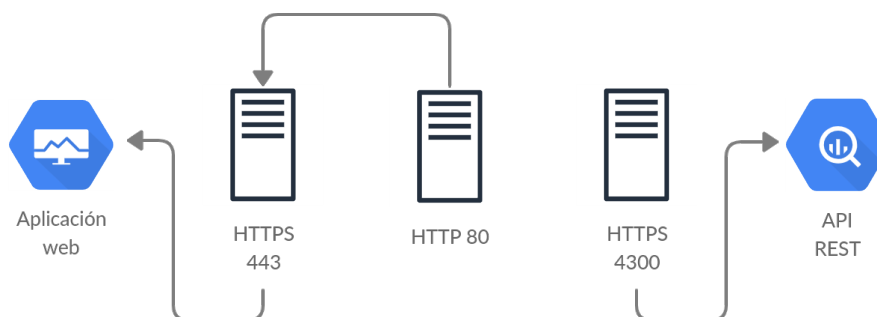


Figura 6.4: Esquema de los servidores web del proyecto.

Para desplegar estos servidores, se han utilizado las librerías de `spdy` y `express` de Node.js.

En primer lugar, se debe crear una instancia de la biblioteca `express`. A continuación, se añaden los *middleware*⁷ deseados, como la biblioteca `cors` o un controlador que comprueba la presencia de *tokens* JWT.

Finalmente, se construye un servidor con `spdy` que utilice la instancia de `express`. En los servidores HTTPS, hay que indicar el certificado y la llave RSA a utilizar.

⁷ Un *middleware* actúa como enlace entre dos componentes *software* y permite la comunicación o la ejecución de ciertas acciones.

Sockets

En cada proceso de votación, se crea un servidor de *sockets* para cada uno de los participantes y otro para el administrador. Cada uno de estos servidores se aloja en un puerto aleatorio entre el 4301 y el 4400. Al finalizar la votación, estos *sockets* se cierran y los respectivos puertos quedan libres de nuevo.

Tal como se ha comentado anteriormente, el código para gestionar los *sockets* se incluye en el directorio `sockets`, el cual contiene 5 ficheros distintos, como se observa en la figura 6.5.

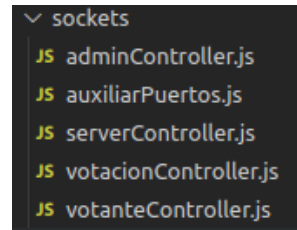


Figura 6.5: Estructura de los ficheros para la gestión de *sockets*.

Cada fichero implementa una funcionalidad específica.

- El código fuente en `votacionController.js` se encarga de crear los servidores y comunicarles los datos necesarios para la votación.
- El fichero `serverController.js` incluye todo lo necesario para la creación de los servidores y `auxiliarPuertos.js` se encarga de la asignación y la liberación de puertos.
- Tanto `votanteController.js` como `adminController.js` contienen la lógica necesaria para gestionar las diferentes fases del proceso de votación en los servidores de los votantes y el administrador respectivamente. Entre otras cosas, se encargan del reenvío de mensajes, la gestión de errores o el almacenamiento de los resultados.

Para construir un servidor de *sockets*, el primer paso consiste en crear un servidor web HTTPS como los anteriores, mediante las librerías de `spdy` y `express`. A continuación, utilizando la librería `websocket` de Node.js, se crea el servidor de *websockets* al que se le asigna el servidor web anterior. A partir de este momento, el servidor se encuentra activo y a la espera de nuevos eventos como la conexión de un usuario o la llegada de un mensaje.

En el código fuente 6.3 se muestra como se construye un servidor de *websockets*, se controlan y aceptan nuevas conexiones y se reciben y envían nuevos mensajes.

```
1 wsServer = new WebSocketServer({ httpServer: server });
2
3 wsServer.on( 'request' , function( request ) {
4     var connection = request.accept( null , request.origin );
5     connection.on( 'message' , function( message ) {
6         // ... LÓGICA DE NEGOCIO ...
7         connection.sendUTF( JSON.stringify ( "RESPUESTA" ) );
```

Código Fuente 6.3: Esquema para la creación de *websockets*.

Notificaciones por correo

Para mejorar la experiencia de los usuarios, se ha implementado un módulo que permite enviar notificaciones a los usuarios. Para ello, se ha utilizado la librería `nodemailer`, que permite enviar correos electrónicos utilizando plantillas HTML.

Como también se ha comentado anteriormente, estas plantillas se encuentran en el directorio `assets`. En el proyecto, el envío de correos se realiza solo al dar de alta a un usuario o cuando un usuario modifica su contraseña. En la figura 6.6 se muestra un ejemplo del correo recibido por un votante al ser dado de alto por un administrador.

Has sido dado de alta en VotApp por el administrador

votapp.noreply@gmail.com

para mí ▾



Figura 6.6: Ejemplo de los correos enviados en el proyecto.

Notificaciones *push*

A diferencia de la tecnología *pull*, en la cual son los clientes los que envían peticiones a un servidor; en la tecnología *push* es el servidor el que se comunica con el cliente. De esta forma [34], el servidor puede enviar notificaciones a los clientes cuando sucede algún evento o acción importante, sin la necesidad que los clientes estén a la espera. Para que los clientes puedan recibir estas notificaciones, hay que seguir los siguientes pasos:

1. El cliente debe otorgar permisos a la aplicación para poder mostrar las notificaciones.
2. El cliente genera y envía una suscripción *push* al servidor, la cual contiene la información necesaria para las notificaciones posteriores.
3. El servidor recibe y almacena la suscripción *push*.

A partir de este momento, cuando el servidor quiere enviar un mensaje a un cliente, lo realiza a través de la información que ha recibido en la suscripción. A su vez, en el cliente, la notificación se recibe y se muestra a través de un *service worker* que se ejecuta en segundo plano en la aplicación web progresiva.

En este proyecto, se ha utilizado esta tecnología para notificar a los votantes del inicio de una votación en la que participan. Para ello, se ha hecho uso de la librería *web-push* de Node.js.

En la figura 6.7 se muestra un ejemplo de las notificaciones que puede recibir un votante.



Figura 6.7: Ejemplo de notificación recibida por un votante.

Autenticación mediante *tokens* JWT

Como ya ha sido comentado con anterioridad, para controlar la autenticación de los clientes se hace uso de la tecnología JWT, mediante la librería `jsonwebtoken` de Node.js. El proceso de autenticación es el siguiente:

1. Cuando un usuario inicia sesión con sus credenciales, si estas son correctas, se genera un nuevo *token* JWT con un tiempo de expiración determinado.
 - Para los administradores, el periodo es de 1 hora.
 - Para los votantes, el periodo es de 20 minutos.
2. Este *token* se envía al usuario junto con el resto de información del inicio de sesión.
3. En cada petición que el usuario realiza a la API para obtener datos, se verifica si este posee un *token* válido y que no haya expirado. Además, también se comprueba que el usuario pertenece a un rol con permisos para realizar la acción deseada.

A continuación, en el código fuente 6.4, se muestra un ejemplo de la creación de un *token* para un administrador. En primer lugar se crea la carga útil o *payload*, que como ya ha sido explicado en la sección 2.3.2, contiene la información del usuario. En este caso, se incluye el DNI y el rol del usuario.

Después se construyen las opciones, con información útil para la cabecera como el algoritmo JWT utilizado o el tiempo de expiración. Como se utiliza el algoritmo RS256, es necesario el uso de un par de claves RSA para firmar los *tokens*. Con esto, ya se puede generar y firmar un *token*, como se observa en la línea subrayada del código fuente 6.4.

```
1     let payload = { sub: DNI, admin: true };
2     let options = { expiresIn: config.JWT_TIME_ADMIN, algorithm: 'RS256' }
3     let PRIVATE_KEY = readFile( config.JWT_PRIVATE_KEY )
4     let token = JWT.sign( payload , PRIVATE_KEY , options )
```

Código Fuente 6.4: Esquema para la creación de *tokens* JWT.

Al recibir una nueva petición que incluye un *token* JWT, se verifica la validez de este, como se observa en el código fuente 6.5. En primer lugar, se generan las opciones y se obtiene la clave pública RSA complementaria a la privada con la que se firman los *tokens*.

Con esto, se comprueba la validez del *token*. En caso de haber expirado o no estar bien formado, se genera una excepción y se rechaza la petición. Si el *token* es válido, se extrae la carga útil para conocer la identidad del usuario.

```
1     try {
2         let options = { algorithm: 'RS256' }
3         let PUBLIC_KEY = readFile(config.JWT_PUBLIC_KEY)
4         let payload = JWT.verify( token , PUBLIC_KEY , options );
5         let response = { DNI: payload.sub , admin: payload.admin };
6     } catch(error) {
7         reject('Token Not Valid')
8     }
```

Código Fuente 6.5: Esquema para la verificación de *tokens* JWT.

6.4. Implementación del *frontend*

Para la implementación de la aplicación web progresiva, se ha utilizado el *framework* de Angular 9, el cual genera automáticamente una estructura de ficheros y directorios necesarios. En la figura 6.8 se observa la estructura final del proyecto.

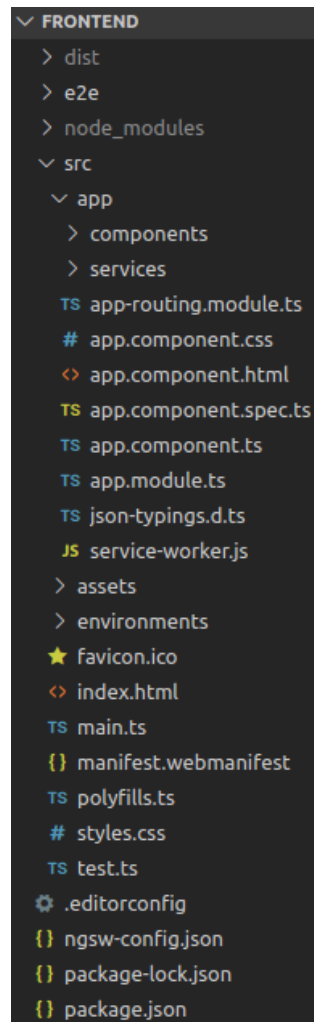


Figura 6.8: Estructura de los directorios del proyecto en el *frontend*.

A continuación, se explica el contenido de los directorios y los ficheros más importantes de la estructura anterior.

- El directorio `dist` contiene el código fuente tras compilar el proyecto. Este debe trasladarse al directorio `public` del *backend* para servir la aplicación web.
- En `src` se sitúa todo el código funcional del proyecto.
- `Components` contiene los diferentes componentes del proyecto Angular, que incluyen las vistas y la lógica de negocio.

- En el directorio `services` se incluyen los servicios creados, que añaden funcionalidad a la aplicación.
- Los ficheros `app.component` representan el componente global del proyecto, a partir del cual se incluyen el resto de componentes.
- En `app.module` se importan y especifican todos los módulos, librerías externas y proveedores utilizados en la aplicación.
- `App-routing.module` incluye las rutas que permiten navegar entre las diferentes páginas de la aplicación. Para ello, se hace uso de la utilidad `Angular-Router`.
- El directorio `assets` incluye los ficheros de configuración del sistema y todas las imágenes utilizadas en la aplicación.
- En los ficheros `index.html` y `styles.css` se incluye la inicialización de la interfaz web y los estilos globales de esta.
- Tanto `manifest` como `ngsw-config` son ficheros necesarios para construir y configurar la aplicación como una PWA e inicializar los *service workers*.

En el directorio `components`, se incluyen diferentes subdirectorios que representan cada uno de los módulos o secciones de la aplicación, tal como se puede ver en la figura 6.9.

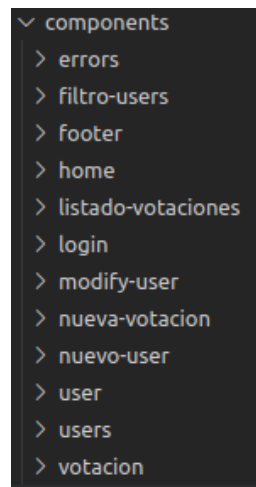


Figura 6.9: Estructura de los componentes en Angular.

Cada directorio contiene uno o varios componentes que representan una página o una parte de esta en la aplicación. Cada uno de estos componentes se compone de 3 ficheros distintos:

- Un fichero `.html`, que contiene todos los elementos gráficos que se visualizan.
- Un fichero `.css` con las reglas de estilo que se aplican al componente.
- Un fichero `.ts`, que se encarga de obtener la información y controlar la interacción de los usuarios con la vista.

En cuanto al directorio `services`, también se pueden encontrar varios subdirectorios que incluyen diferentes servicios para cubrir la funcionalidad de la aplicación. En la figura 6.10 se puede ver la estructura de subdirectorios, cuyo contenido se explica a continuación.

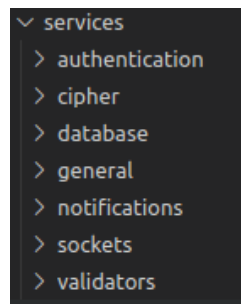


Figura 6.10: Estructura de los servicios en Angular.

- En el directorio `authentication` se incluyen 2 servicios distintos.
 - Un controlador que se encarga de gestionar las peticiones de inicio y cierre de sesión.
 - Un controlador que gestiona y almacena la información de sesión del usuario, incluyendo el nombre, el DNI, el *token* JWT y el rol.
- `Cipher` es un directorio muy importante para los procesos de cifrado en RSA y AES. Se compone de:
 - Un servicio para cifrar y descifrar mensajes mediante el algoritmo AES.
 - Un servicio que permite cifrar, descifrar, firmar y validar firmas mediante RSA.
 - Un controlador que permite generar un par de claves RSA o crear llaves AES mediante derivación de contraseñas.
 - Un servicio que gestiona el cifrado y descifrado en AES de las llaves RSA.
 - Un servicio que gestiona las diferentes fases de cifrado y descifrado de los votos.
- En el directorio `database` se encuentran los servicios que facilitan las consultas y peticiones sobre la API.
 - Un servicio con los métodos para realizar peticiones HTTP *GET* o *POST*.
 - Un servicio que agrupa las distintas peticiones que se pueden realizar a la API.
- `General` incluye varios servicios que permiten obtener la configuración y las variables de entorno del proyecto.
- En la carpeta `notifications` se incluye el servicio que se encarga de enviar las subscripciones *push* al servidor para poder recibir notificaciones.
- `Validators` contiene un grupo de servicios que permiten validar la información introducida por los usuarios antes de enviarla al servidor. Se validan los diferentes campos de cada votación, usuario o contraseña.
- El directorio `sockets` se compone de 4 servicios que facilitan el envío de mensajes a través de los *sockets* durante los procesos de votación.
 - Dos servicios que controlan el proceso de envíos de un administrador o un votante.
 - Un servicio que almacena información relevante sobre el proceso de votación.
 - Un servicio que gestiona el envío de mensajes en cada fase del proceso.

Como resultado de este proyecto en Angular, se ha desarrollado una aplicación web que cumple con los requisitos funcionales y de datos identificados en el capítulo 4. A continuación, se muestran los resultados obtenidos en los distintos módulos de la aplicación.

6.4.1. Módulo de inicio de sesión

Este módulo representa el proceso en el cual un usuario, sea votante o administrador, inicia sesión y accede a la aplicación. En primer lugar, el usuario observa una vista en la cual le solicitan los datos de acceso. Si las credenciales son correctas, el usuario inicia sesión y accede a la aplicación. En ese momento, visualiza el panel principal, el cual es diferente según el rol del usuario.

Si se trata de un administrador, el panel contiene dos accesos directos para crear nuevas votaciones o registrar nuevos votantes. Además, el administrador observa las 4 votaciones más cercanas a la fecha actual. En el caso de los votantes, se visualiza un panel con las votaciones más cercanas en las que participan. Si alguna se encuentra activa para participar, está se muestra destacada arriba.

En la figura 6.11 se incluyen las 3 vistas de este módulo.

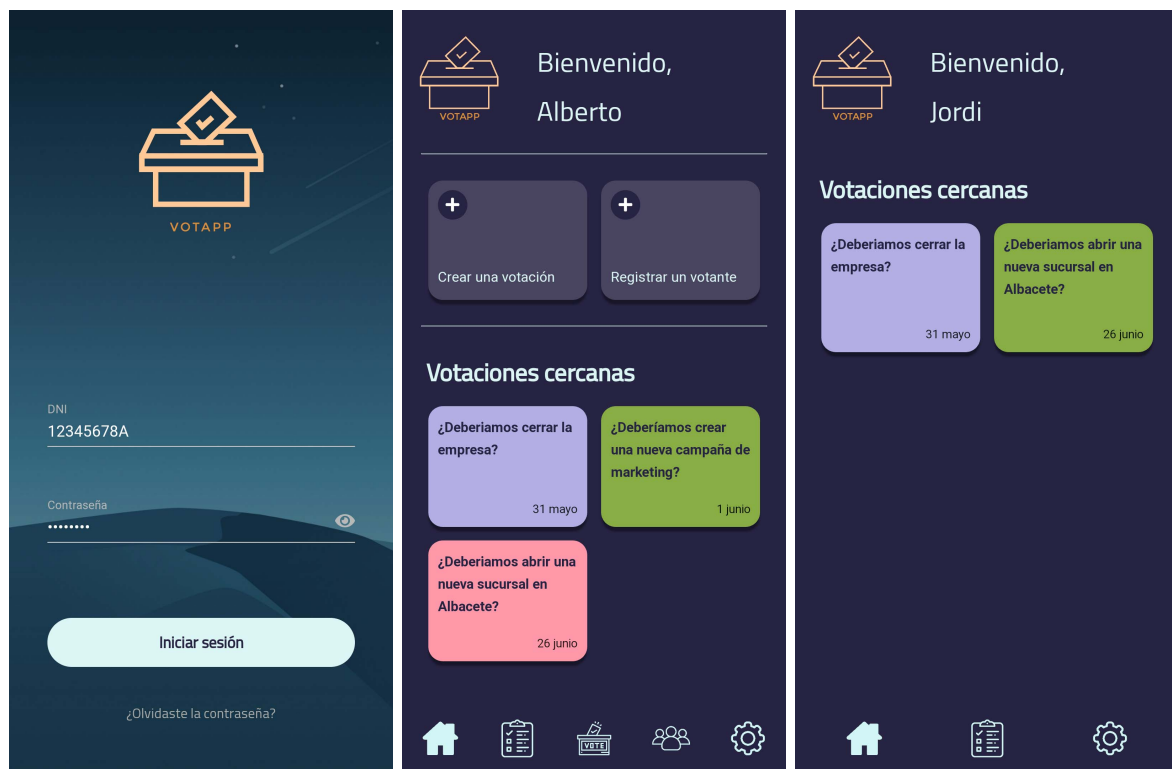


Figura 6.11: Interfaces - Inicio de sesión y panel principal.

6.4.2. Módulo de votaciones

Este módulo se compone de las diferentes vistas que permiten crear, gestionar y visualizar las votaciones del sistema.

Una funcionalidad importante en la aplicación es la creación de nuevas votaciones. Para ello, hay que seguir varios pasos, en los que se introducen los datos básicos como la pregunta o la fecha, las opciones disponibles y los usuarios que participan. Finalmente, se muestra un resumen con los datos de la votación y se confirma la inserción. En la figura 6.12 se pueden observar las diferentes interfaces que se visualizan durante la creación de una votación.

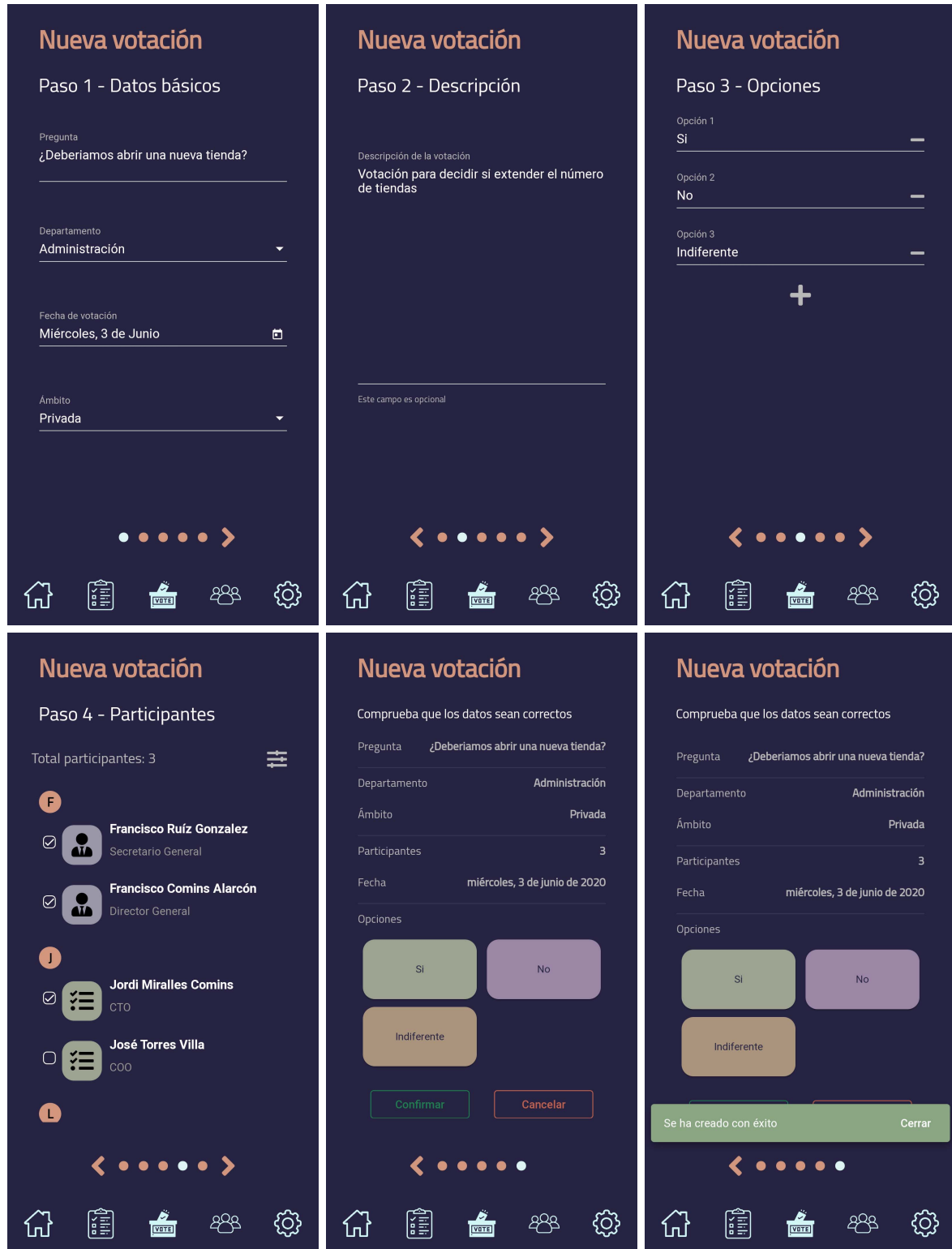


Figura 6.12: Interfaces - Creación de una nueva votación.

Una vez creada la votación, esta puede ser visualizada en el listado de votaciones. En este listado, los administradores visualizan todas las votaciones del sistema, mientras que los votantes solo aquellas a las que tienen permiso, es decir:

- Todas las votaciones de ámbito público.
- Todas las votaciones privadas en las que participen.
- Todas las votaciones de ámbito departamento que pertenezcan al mismo departamento que el votante.
- Las votaciones ocultas en las que participen, solo cuando están activas.

El listado se puede filtrar por diferentes campos, como la pregunta, el estado o el departamento. Además, los votantes pueden seleccionar la opción que muestra solo las votaciones en las que participan.



En la figura 6.13 se incluyen algunos ejemplos de esta interfaz.



Figura 6.13: Interfaces - Listado de votaciones.

A partir de este listado, se puede acceder al panel de una votación concreta, en la cual se ven los detalles y la información de esta.

Esta interfaz modifica su apariencia en función del estado y el rol del usuario que la visualiza. Cuando la votación está creada, los votantes pueden visualizar los datos de esta y ampliar los detalles pulsando sobre las flechas.

Además, un administrador también puede modificar los datos de esta mediante los botones con el icono  o iniciarla mediante el botón  de la esquina superior derecha.

Cuando la votación está iniciada, el administrador ya no puede modificar los datos, mientras que los votantes, si participan en ella, pueden acceder a esta mediante el botón de Participar. Al finalizar la votación, tanto el administrador como los votantes pueden acceder a los resultados.

En la figura 6.14 se pueden observar algunos ejemplos de la apariencia de esta interfaz.

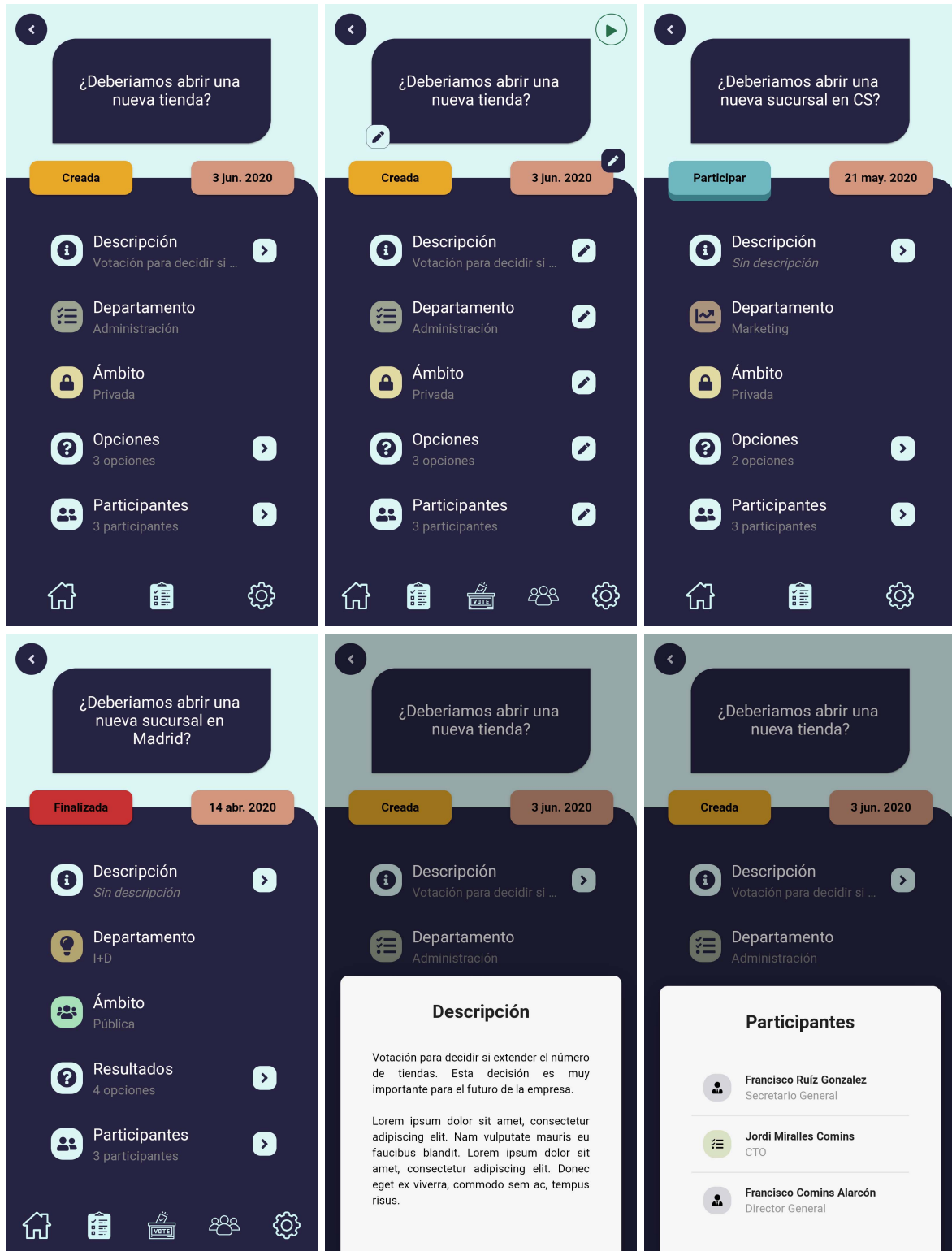


Figura 6.14: Interfaces - Detalles de una votación.

Como se ha comentado anteriormente, si la votación se encuentra en el estado 'Creada', el administrador puede modificar algunos de sus datos, como la pregunta, la fecha, el departamento o los participantes.

La figura 6.15 incluye algunos ejemplos de las vistas que permiten modificar una votación.

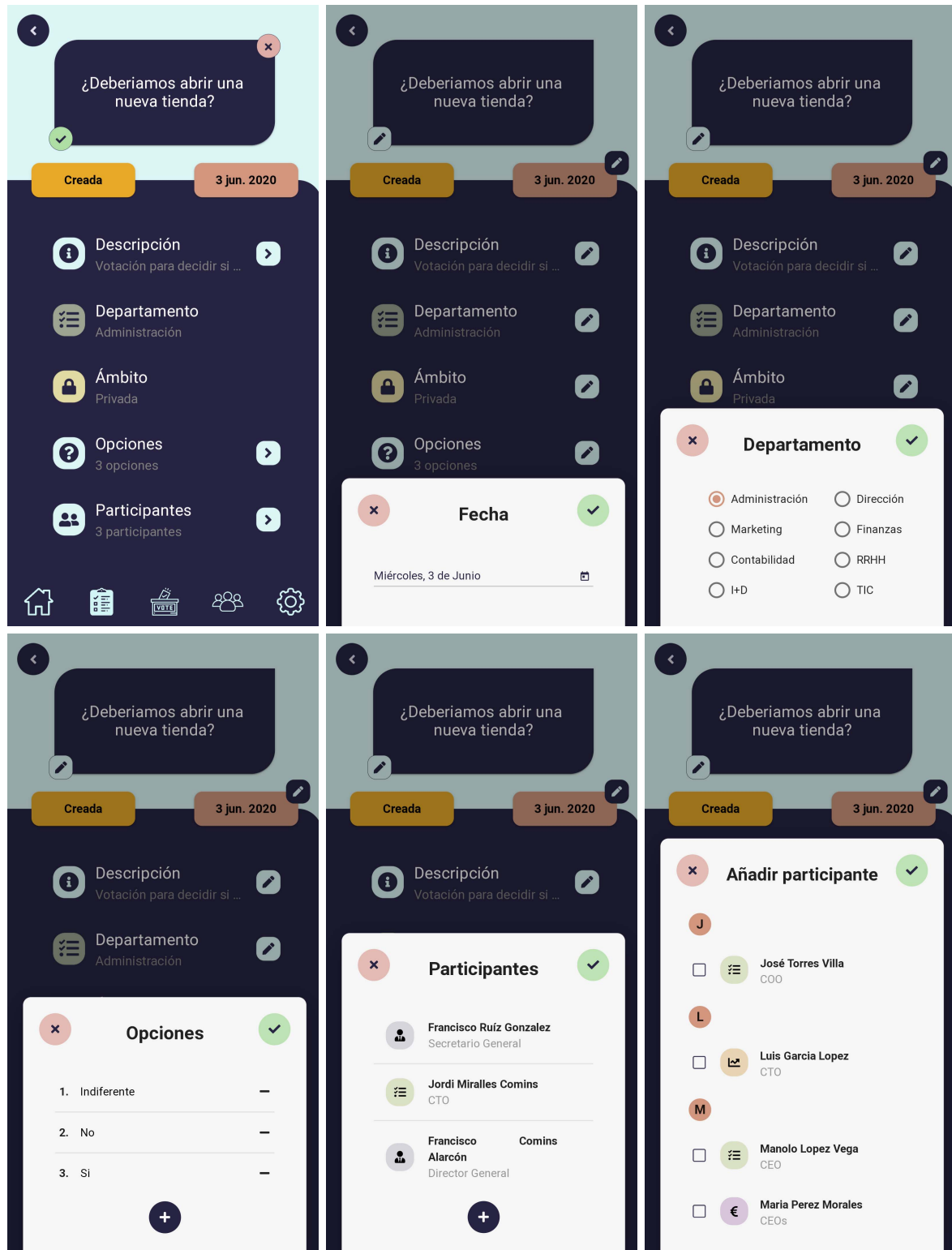


Figura 6.15: Interfaces - Modificar los datos de una votación.

6.4.3. Módulo de usuarios

En este módulo se incluyen todas las interfaces utilizadas para registrar, modificar y listar los participantes en el sistema.

Al crear un nuevo usuario, hay que completar todos los datos de este, tanto los personales (*e.g.* nombre, DNI...) como los de contacto (*e.g.* teléfono, correo electrónico, departamento...). Finalmente, se muestra un resumen de estos datos y se confirma el registro.

En la figura 6.16 se pueden ver las vistas asociadas al registro de un votante.

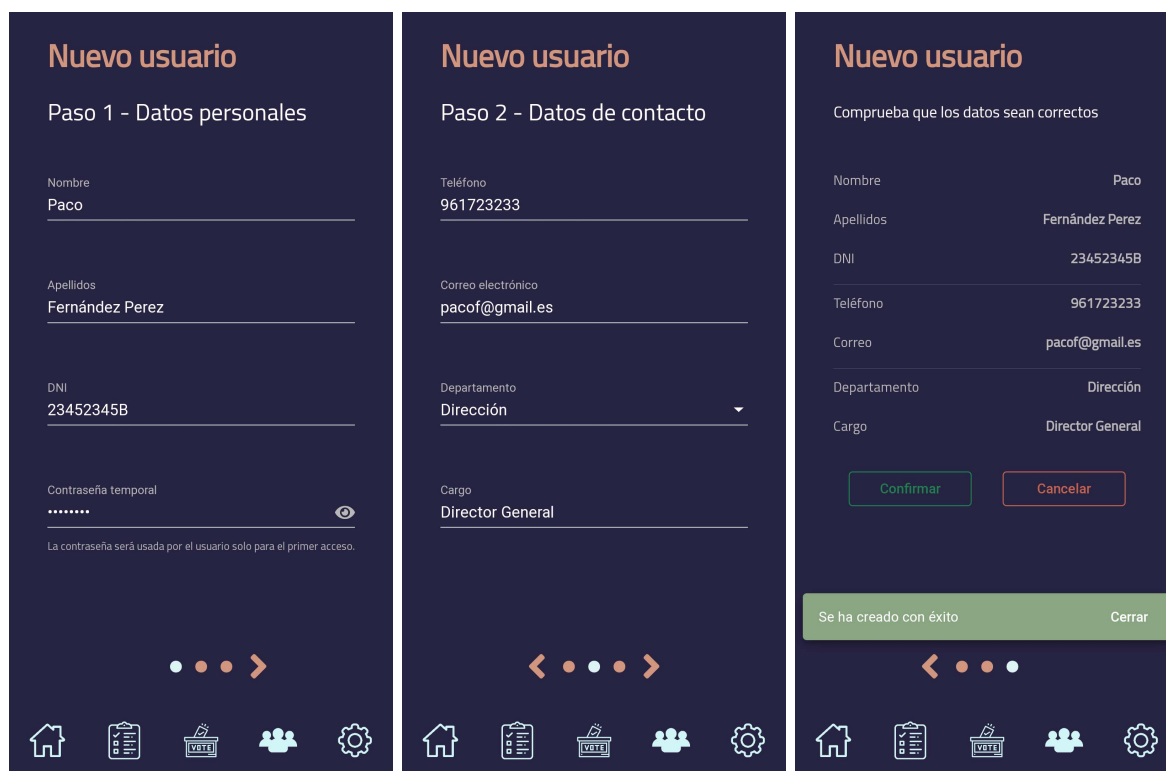


Figura 6.16: Interfaces - Registrar un nuevo votante.

A continuación, el administrador puede visualizar y gestionar el listado de votantes dados de alta en el sistema. Además, puede filtrar este listado por diferentes campos como el nombre, el cargo o el departamento al que pertenecen los votantes.

Desde el listado, el administrador puede acceder al perfil de un votante concreto y visualizar o modificar sus datos, exceptuando la contraseña de acceso.

Por otro lado, tanto el administrador como los votantes pueden acceder a su propio perfil y modificarlo, incluyendo la contraseña.

Además, cuando un votante inicia sesión por primera vez, debe modificar su contraseña, ya que el administrador le asigna una temporal al darlo de alta.

En la figura 6.17 se observan las interfaces con el listado de usuarios y la gestión de estos por parte del administrador, mientras que en la 6.18 se incluyen las interfaces con el perfil del usuario y la vista que permite modificar la contraseña en el primer inicio.



Figura 6.17: Interfaces - Gestionar los votantes del sistema.

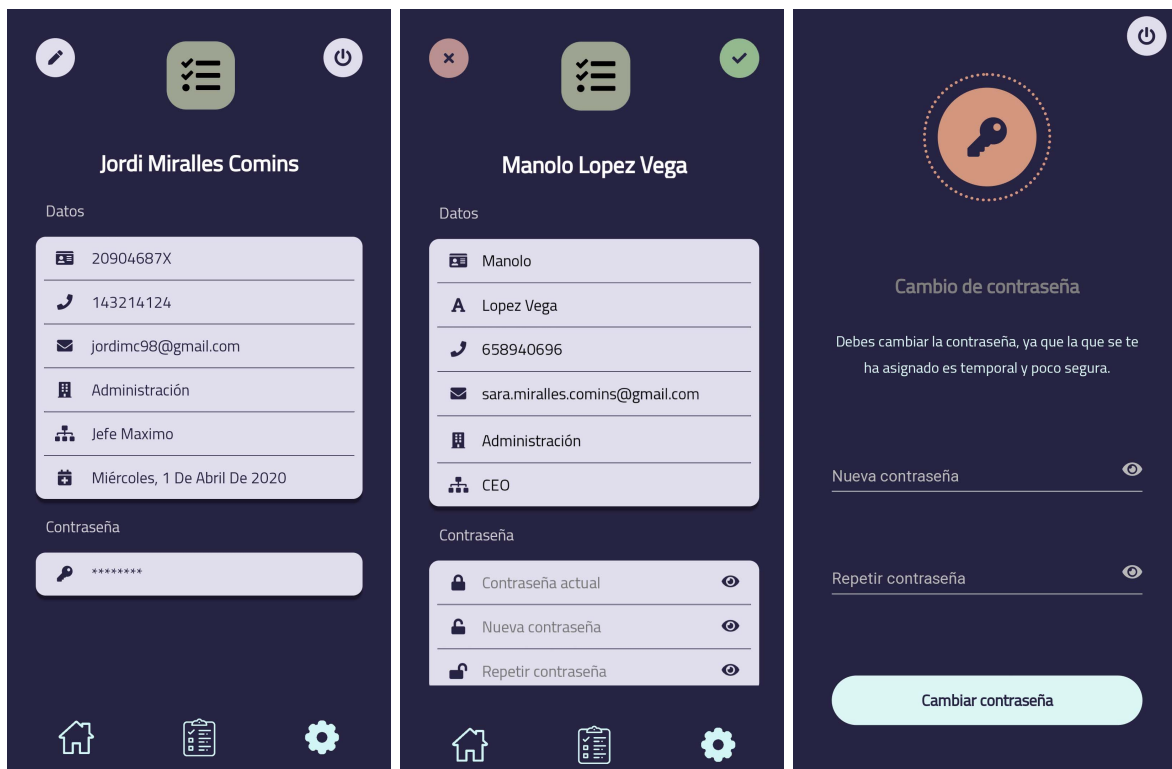



Figura 6.18: Interfaces - Gestionar el perfil de un usuario.

6.4.4. Módulo para los procesos de votación

Este módulo incluye la funcionalidad más importante de la aplicación, ya que cumple con el objetivo principal del proyecto, la realización de votaciones. En primer lugar, es necesario que el administrador inicie la votación. Para ello, primero debe introducir su contraseña para descifrar su clave privada RSA.

A continuación, el administrador visualiza un panel en el que se muestra el estado de los votantes. Además de la notificación enviada al arrancar la votación, el administrador puede enviar más notificaciones mediante el icono . El administrador también puede parar la votación en cualquier momento, de forma que esta vuelve al estado de 'Creada'.

Cuando todos los votantes se han conectado, el administrador puede arrancar definitivamente la votación, permitiendo a los votantes emitir su voto. A partir de este momento, el administrador visualizará una pantalla de carga hasta que finalice el recuento de votos. En la figura 6.19 se muestran las interfaces que permiten a un administrador iniciar la votación.



Figura 6.19: Interfaces - Iniciar una votación.

Como se ha visto anteriormente, si la votación está iniciada, los votantes participantes pueden acceder a esta. No obstante, mientras no se hayan conectado todos los participantes, los votantes conectados visualizarán una pantalla de espera.

Una vez se conectan todos y el administrador arranca la votación, los votantes pueden seleccionar y emitir su voto. Durante el recuento de los votos, tanto los votantes como el administrador observan otra pantalla de espera.

En la figura 6.20 se muestran representadas las pantallas de espera y el panel de votación.

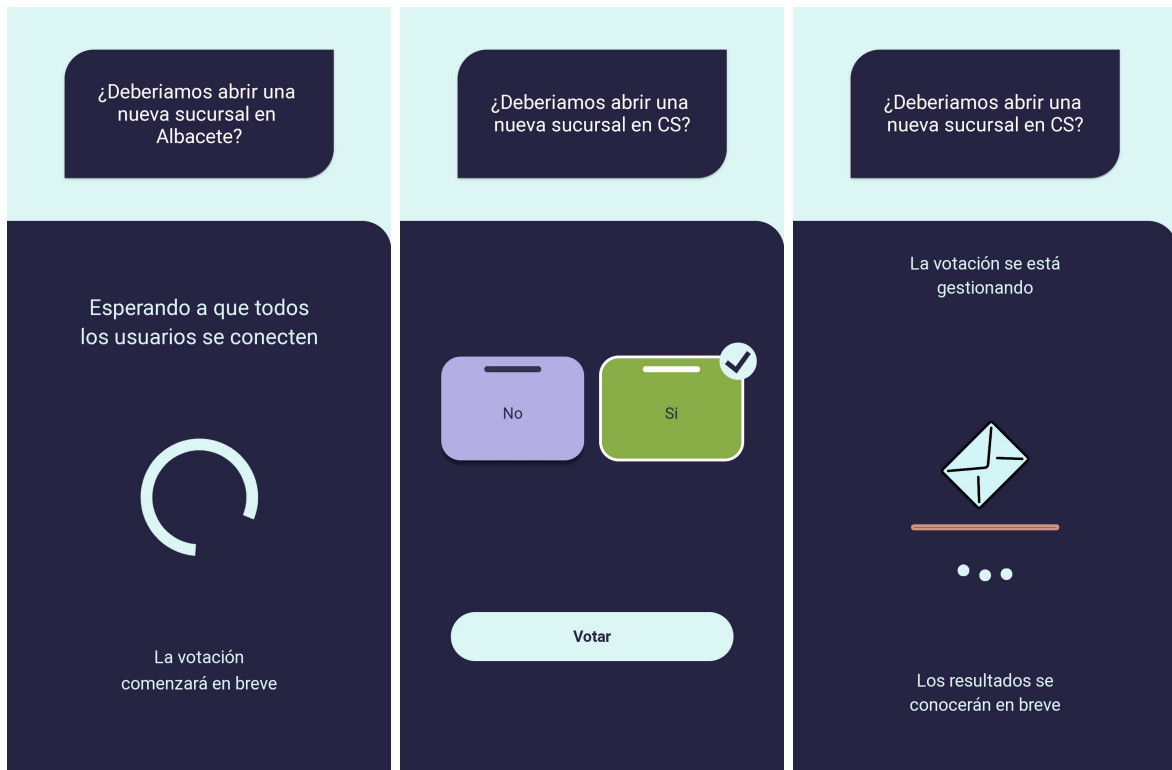


Figura 6.20: Interfaces - Proceso de votación.

Al finalizar la votación, si no se produce ningún error o problema, se muestran los resultados de esta tanto para el administrador como para los votantes, como se observa en la figura 6.21.



Figura 6.21: Interfaces - Resultados de una votación.

6.5. Implementación del protocolo de votación

Como se ha descrito en la sección 2.2.2, el sistema utiliza como algoritmo de votación el protocolo de Merritt, basado en RSA. Este protocolo se divide en dos procesos distintos.

6.5.1. Proceso de cifrado

El primer paso consiste en el cifrado del voto emitido por cada votante. Este proceso es realizado por todos los votantes de forma independiente. Para ello, es necesario que todos los votantes dispongan de la clave pública RSA del resto de participantes y también del administrador.

El proceso se compone de diferentes fases y se utiliza una estructura de datos para almacenar los resultados de cada fase. Durante las diferentes fases, se utilizan varias cadenas de caracteres ($R_0 R_1 \dots R_N$) generadas aleatoriamente, que permiten a los votantes comprobar la presencia de su voto durante el recuento.

Primera Fase

En primer lugar, se genera una cadena aleatoria R_0 , la cual se almacena en la estructura de datos y se añade al final del voto emitido por el votante. Posteriormente, se cifra el voto y la cadena R_0 con la clave pública del administrador y se almacena también el resultado.

En la ecuación 6.1 se observa el resultado de esta fase de forma visual, donde Cif_x representa el proceso de cifrado con la clave pública de X.

$$Fase_1 = Cif_{Adm} (Voto + R_0) \quad (6.1)$$

Segunda Fase

Para la segunda etapa, se ordenan los votantes de forma aleatoria. Todos los votantes siguen el mismo orden establecido a la hora de cifrar. A continuación, se cifra de forma escalonada con las claves de todos los votantes, siguiendo el orden establecido.

Por ejemplo, en el caso de haber 3 votantes (A B C), el proceso sería:

1. El resultado de la primera fase se cifra con la clave de C.
2. El resultado del cifrado con la clave de C se cifra con la clave de B.
3. El resultado del cifrado con la clave de B se cifra con la clave de A.

El resultado de los 3 cifrados se almacena también en la estructura de datos. Este proceso queda resumido de forma matemática en la ecuación 6.2.

$$Fase_2 = Cif_A (Cif_B (Cif_C (Fase_1))) \quad (6.2)$$

Tercera Fase

La tercera fase es idéntica a la segunda, por lo que se cifra de forma escalonada con las claves de todos los votantes. Sin embargo, antes de cada cifrado, se genera y se añade una nueva cadena aleatoria, la cual se almacena también en la estructura de datos.

En resumen, se sigue el siguiente proceso, con el mismo orden que en la fase anterior:

1. Se genera una cadena aleatoria R_1 y se añade al resultado de la segunda fase. A continuación, se cifra con la clave de C.
2. Al resultado del cifrado anterior, se le añade una cadena aleatoria R_2 y se cifra con B.
3. De la misma forma, se añade una cadena R_3 y se cifra con A.

En la ecuación 6.3 se observa el esquema de esta fase, que es la última del proceso de cifrado.

$$Fase_3 = Cif_A (R_3 + Cif_B (R_2 + Cif_C (R_1 + Fase_2))) \quad (6.3)$$

El resultado final de esta fase es el mensaje que se utiliza a continuación en el siguiente proceso, el de envíos mediante *sockets*. Por otro lado, la estructura de datos contiene los siguientes datos, utilizados posteriormente para verificar la integridad de la votación:

- El resultado del cifrado de la primera fase, en la que se usa la clave del administrador.
- Los resultados de los cifrados de la segunda fase (cifrados de A B y C).
- La cadena aleatoria inicial unida al voto (R_0) y las cadenas aleatorias añadidas en la tercera fase (R_1 R_2 y R_3).

En el código fuente 6.6 se muestra la implementación de las diferentes fases del proceso de cifrado.

```
1 //FASE 1
2 this.store["f1"]["cadena"] = this.KeyGenerator.generateRandom();
3 voto += "###" + this.store["f1"]["cadena"];
4 voto = await this.RSACipher.encrypt(voto, "admin");
5 this.store["f1"]["cifrado"] = voto;
6
7 //FASE 2
8 for (var id_vot of Object.keys(this.lista).reverse()) {
9     voto = await this.RSACipher.encrypt(voto, id_vot);
10    this.store["f2"][id_vot] = voto;
11 }
12
13 //FASE 3
14 for (var id_vot of Object.keys(this.lista).reverse()) {
15    this.store["f3"][id_vot] = this.KeyGenerator.generateRandom();
16    voto += "###" + this.store["f3"][id_vot]
17    voto = await this.RSACipher.encrypt(voto, id_vot);
18 }
```

Código Fuente 6.6: Implementación del proceso de cifrado.

Tal como se comenta en la sección 6.4, se han implementado varios servicios en Angular para gestionar el proceso de cifrado. En este caso, se utiliza el servicio `KeyGenerator` para generar las cadenas aleatorias y `RSACipher` para cifrar en RSA. Este último hace uso de la librería `jsencrypt` de JavaScript, la cual permite crear y gestionar un cifrador RSA para cifrar el voto y el resto de mensajes escalonados.

6.5.2. Proceso de envíos

Tras cifrar los votos, cada votante posee un mensaje con un esquema similar a la ecuación 6.3. A partir de aquí, se inicia el proceso de comunicación entre los votantes, que permite enviar y descifrar los votos para el posterior recuento. Este puede dividirse también en 3 etapas que están relacionadas con las fases de cifrado anteriores, pero en orden inverso.

Primera Fase

En primer lugar, todos los votantes envían al votante A los mensajes cifrados. Este, al recibirlos, los baraja de forma aleatoria para perder el rastro y desconocer la procedencia de cada mensaje.

Posteriormente, descifra todos los mensajes, extrae las cadenas aleatorias que se habían añadido (R_3) y comprueba que alguna de ellas coincide con la que tiene almacenada en la estructura de datos.

Si todo es correcto, envía los mensajes descifrados al votante siguiente, es decir, a B. Tanto B como el resto de votantes realizan el mismo proceso que A: barajar los votos, descifrarlos, extraer las cadenas, comprobar estas (R_2 R_1) y enviar los mensajes descifrados al votante siguiente.

Cuando el último votante, en este caso C, finaliza correctamente la comprobación de las cadenas, este dispone de una lista de mensajes con el esquema de la ecuación 6.2. En resumen, esta etapa se encarga de procesar y eliminar todos los cifrados de la fase 3 anterior.

En la figura 6.22 se incluye un esquema de esta fase de envíos, con la fórmula matemática que representa la estructura de cada mensaje.

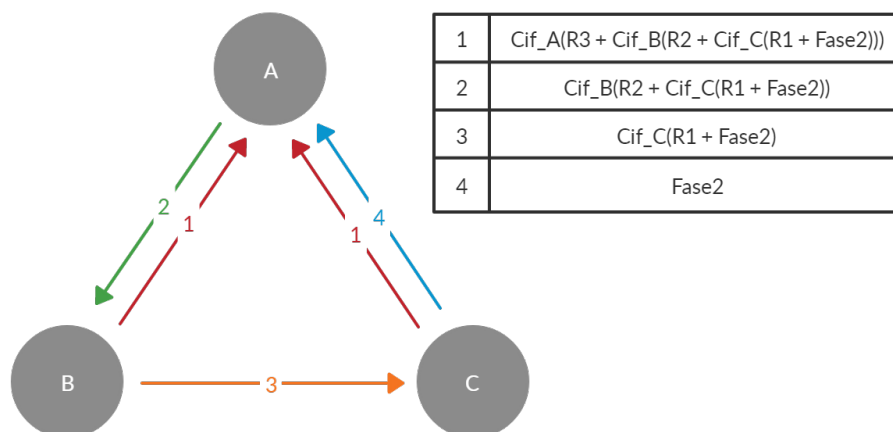


Figura 6.22: Esquema de la primera fase del proceso de envío de votos.

Segunda Fase

Esta fase se inicia cuando A recibe el último mensaje de la fase anterior. En primer lugar, este comprueba que alguno de los votos cifrados que recibe coinciden con el que se almacenó en la segunda etapa de cifrados.

A continuación, descifra todos los votos y firma el resultado con su clave pública. La firma permite al resto de votantes comprobar el origen del mensaje. Finalmente, envía este mensaje firmado al resto de participantes.

Si el votante que lo recibe no es el siguiente (en este caso C), simplemente comprueba la validez de la firma y comprueba que entre los mensajes cifrados se encuentra su cifrado almacenado en la etapa 2.

Si por el contrario, el votante es el siguiente (en este caso B), además de lo anterior, también realiza los mismos pasos que A. Esto es, descifra, firma y envía los votos al resto de participantes.

Cuando el último votante, en este caso C, descifra todos los mensajes, este dispone de una lista de mensajes con el esquema de la ecuación 6.1, por lo que en resumen, esta etapa se encarga de procesar y eliminar todos los cifrados de la fase 2.

El último votante envía a todos los participantes, incluido él mismo, la lista de votos firmados con su clave. Además, envía también la lista firmada al administrador. Esta etapa se ve representada en el esquema de la figura 6.23. En las fórmulas matemáticas, $Firm_x$ representa la firma de un mensaje con la clave pública de X.

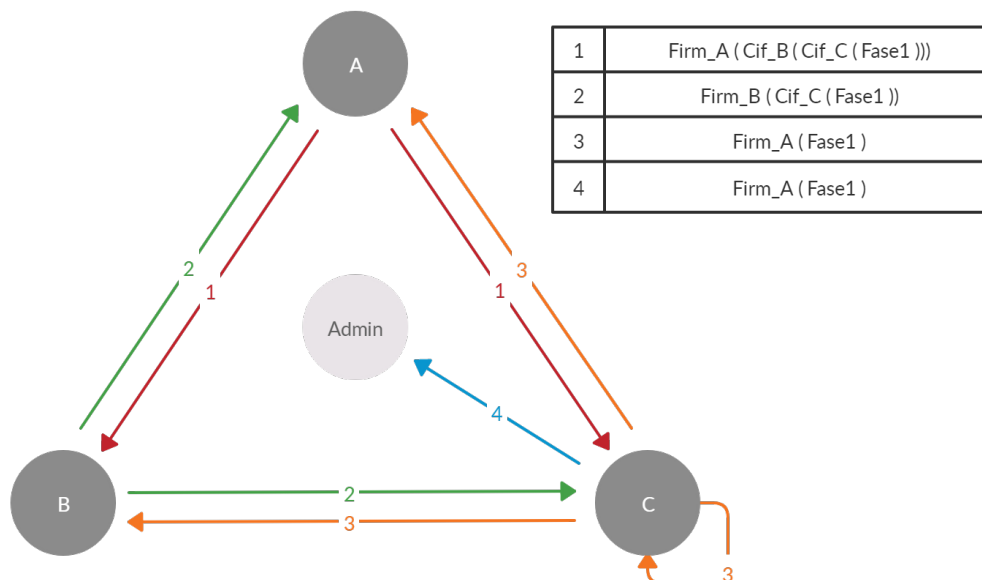


Figura 6.23: Esquema de la segunda fase del proceso de envío de votos.

Tercera Fase

Esta fase se inicia cuando todos los participantes reciben la lista de votos firmados por el último (en este caso C).

Entonces, estos comprueban la validez de la firma y comprueban también que en la lista se encuentra el voto cifrado que habían almacenado en la etapa 1 del cifrado.

Por otro lado, el administrador también recibe la lista de votos firmados. Este comprueba también la validez de la firma y descifra todos los votos con su clave privada. En este momento, el administrador dispone de una lista en la que los votos ya no se encuentran cifrados e incluyen la primera cadena aleatoria (R_0).

A continuación, el administrador realiza el recuento de votos y guarda los resultados de forma temporal. Antes de almacenarlos definitivamente en la base de datos, debe recibir el consenso de todos los votantes.

Para ello, firma la lista con su clave pública y la envía a todos los votantes. Estos comprueban la validez de la firma y la presencia de su cadena aleatoria R_0 en la lista. En caso afirmativo, responden con un mensaje OK al administrador, el cual almacena los resultados de forma definitiva.

En la figura 6.24 se representa el esquema de esta etapa de envíos y descifrados.

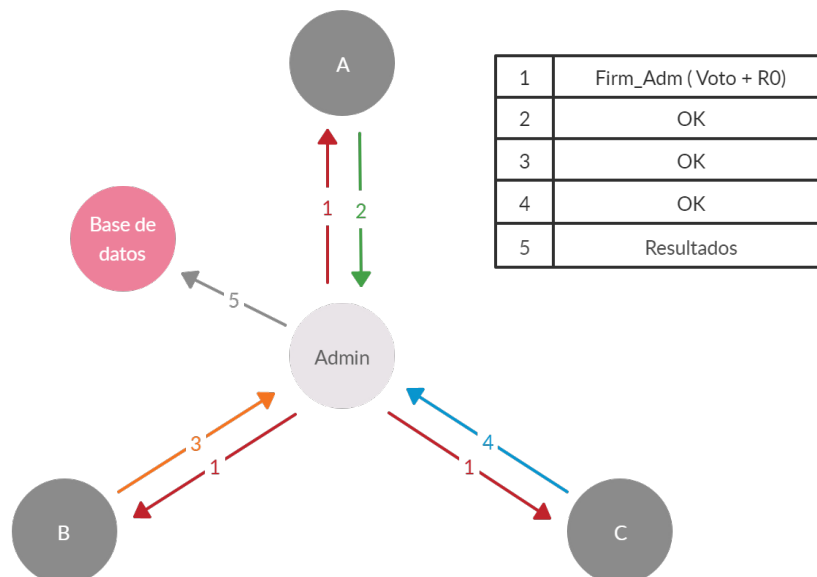


Figura 6.24: Esquema de la tercera fase del proceso de envío de votos.

Además de estas 3 fases de envíos, también se utilizan los canales de *sockets* para enviar otros mensajes útiles durante el proceso de votación, como por ejemplo:

- Notificar al administrador cuando todos los votantes están conectados.
- Notificar a los votantes cuando el administrador arranca la votación.
- Notificar al resto de votantes y el administrador cuando se produce cualquier error, excepción o alteración durante la votación.

Del mismo modo que en la etapa de cifrados, se utiliza la librería `jsencrypt` para descifrar, firmar y validar firmas en RSA. Por otro lado, se utiliza la biblioteca `WebSocket` de JavaScript para comunicarse con el servidor de *sockets* y permitir el envío y la recepción de mensajes.

Capítulo 7

Verificación y validación del sistema

Contenidos

7.1. Verificación	99
7.2. Validación	102
7.2.1. Validación de requisitos	102
7.2.2. Validación de interfaces	103
7.2.3. Validación de la seguridad	105

En el presente capítulo se describen las herramientas utilizadas para comprobar el correcto funcionamiento del sistema.

Además, se verifica el cumplimiento de los requisitos funcionales identificados en el capítulo 4 y se comprueba que el producto final presenta la fiabilidad y la calidad esperada.

7.1. Verificación

La verificación de un programa informático consiste en la comprobación del correcto funcionamiento de este. Este proceso permite detectar y corregir errores que impiden que la aplicación funcione de la forma esperada.

Para verificar el sistema, se han realizado pruebas unitarias sobre cada módulo y pruebas globales sobre el *software* completo. Para la realización de estas, se han utilizado las herramientas para desarrolladores que proporciona el navegador Google. Entre las múltiples utilidades que se incluyen, se han utilizado las siguientes.

Toggle device toolbar

Esta funcionalidad permite visualizar la apariencia de una página web en diferentes dispositivos. Esta herramienta ha sido muy útil para programar la aplicación web sin la necesidad de conectar un dispositivo móvil

Consola

La consola es una de las herramientas más importantes en el desarrollo de una aplicación web. Esta permite visualizar mensajes que informan del funcionamiento y los errores que presenta el producto.

En la figura 7.1 se muestra un ejemplo en el que se visualiza la apariencia de la web en un dispositivo móvil y la terminal con algunos mensajes sobre las fases de cifrado y envío.

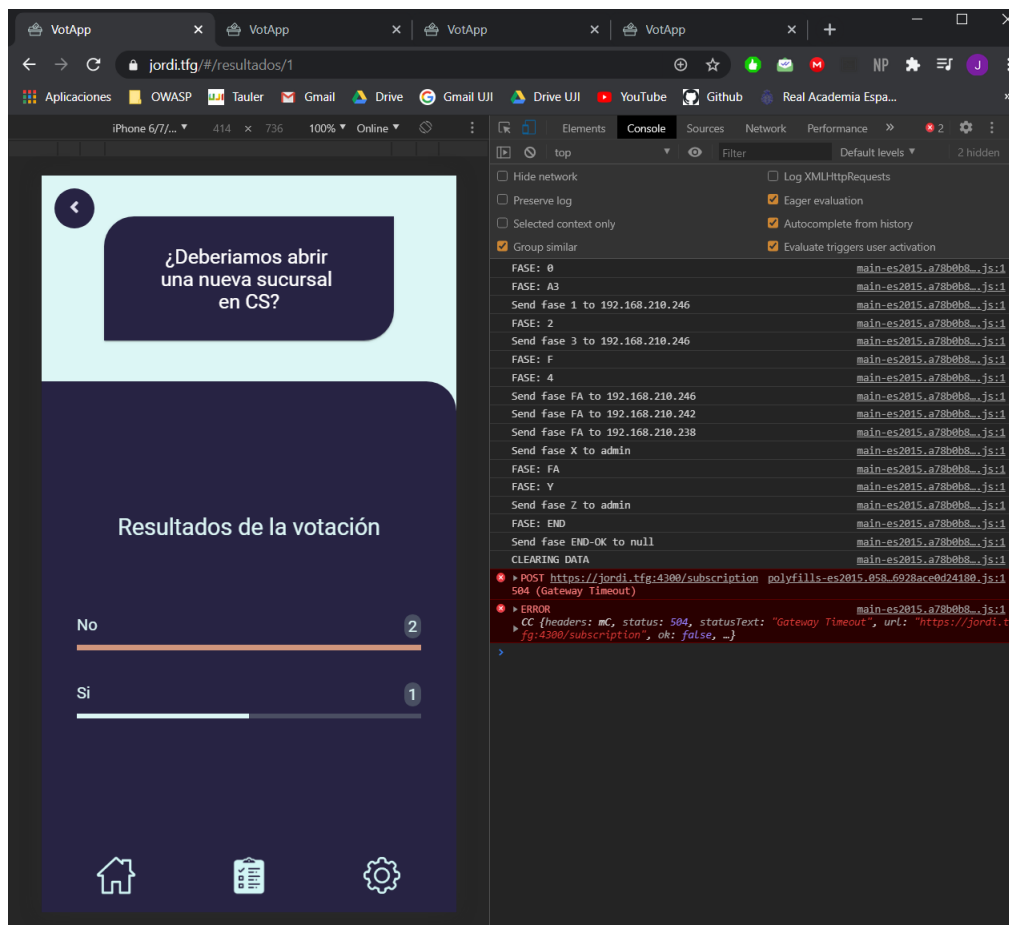


Figura 7.1: Ejemplo de uso de la consola de Google Developers.

Application

Una de las funcionalidades que incluyen las aplicaciones web progresivas son los *service workers*, que trabajan en segundo plano ejecutando ciertas acciones. En esta sección se puede gestionar y controlar el estado de estos. Además, se pueden comprobar otros aspectos como los metadatos de la PWA o los datos almacenados en la sesión.

Security

Permite controlar algunos aspectos de seguridad de la aplicación, como la validez del certificado del servidor.

Lighthouse

Esta herramienta permite realizar auditorías sobre una página web. Entre los diferentes aspectos que se pueden auditar, el más importante para este proyecto ha sido el que verifica que la aplicación cumple con todos los requisitos para ser una PWA.

En la figura 7.2 se muestran algunas de las funcionalidades de la sección *Application*.

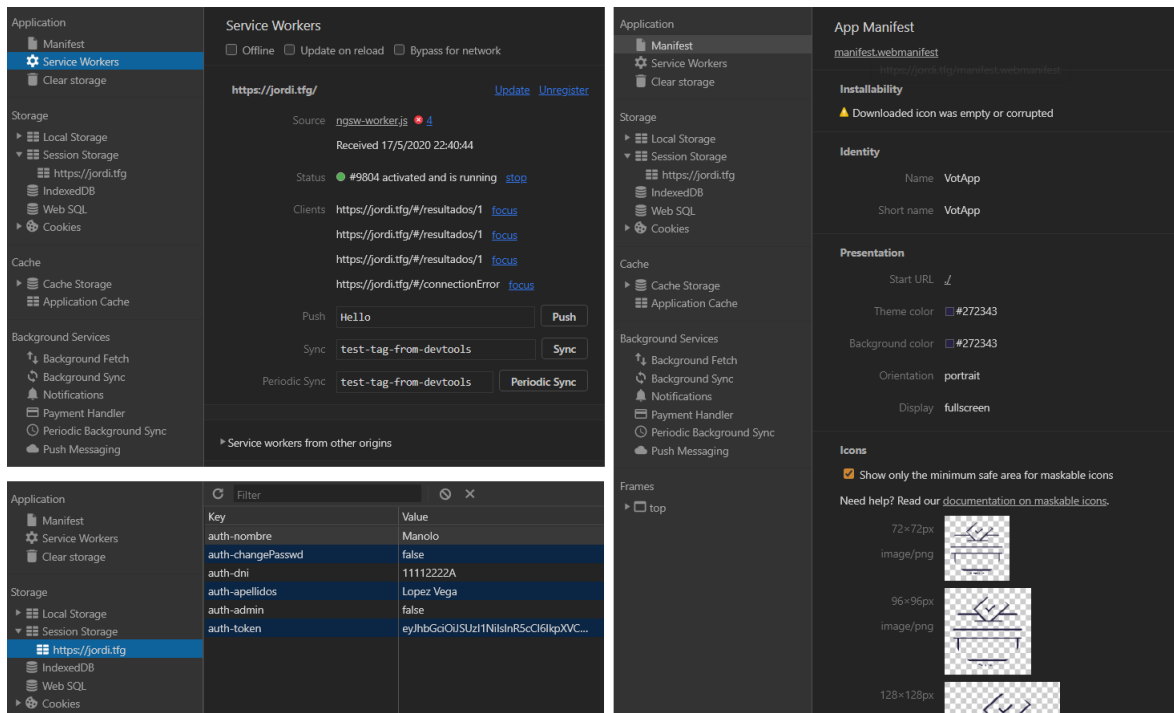


Figura 7.2: Ejemplo de uso de la sección *Application* de Google Developers.

En la figura 7.3 se observa el resultado de la auditoría sobre la aplicación implementada.

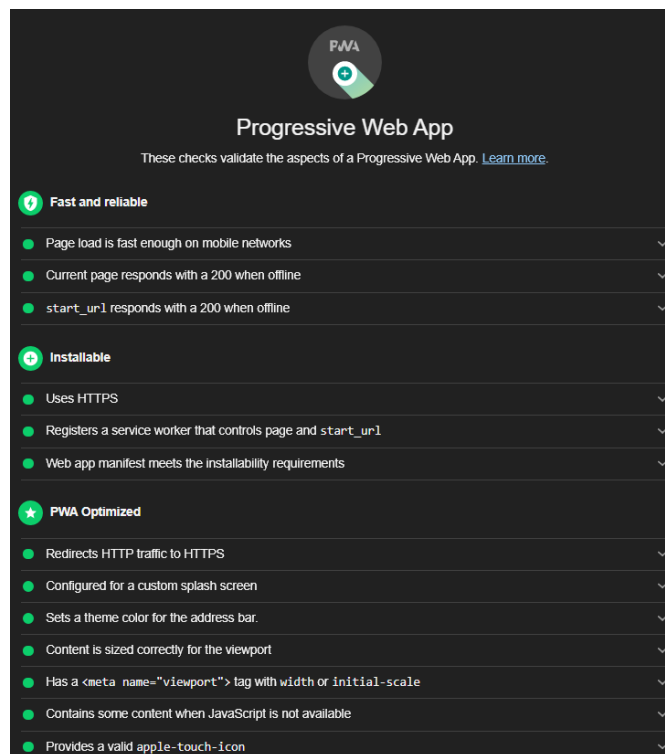


Figura 7.3: Resultado de la auditoría Lighthouse sobre la PWA implementada.

7.2. Validación

La validación del *software* consiste en comprobar que el sistema funciona acorde con los objetivos y los requisitos de este, es decir, que cumple con la funcionalidad esperada.

7.2.1. Validación de requisitos

Para validar los requisitos, hay que comprobar si el sistema implementado soluciona o cubre todos los casos de uso identificados en el apartado 4.2.

CU01 - Crear o modificar votaciones

Cualquier usuario con el rol de administrador puede crear nuevas votaciones en el sistema con los datos específicos. Los votantes no pueden realizar esta operación.

Por otro lado, los administradores también pueden modificar los datos de una votación, mientras esta no se encuentre activa o finalizada.

CU02 - Elegir votantes

Los administradores, al crear una votación, pueden decidir quién participa en esta y quién no. Además, mediante el ámbito de la votación, también deciden quién puede visualizar los datos de la votación.

Al modificar los datos de una votación, también tienen la posibilidad de añadir o eliminar participantes, siempre dentro de los límites (mínimo 3 y máximo 6).

CU03 - Iniciar votación

Los administradores pueden iniciar una votación en cualquier momento. A partir de este momento, pueden visualizar el estado de los votantes y arrancar la votación definitivamente cuando están todos conectados.

Además, el administrador puede parar la votación y reiniciar su estado a ‘Creada’ en cualquier momento.

CU04 - Registrar votante

Un administrador tiene la posibilidad de añadir nuevos votantes al sistema, introduciendo todos sus datos. También debe otorgarle una contraseña temporal para el primer acceso a la aplicación.

CU05 - Votar

Los votantes que el administrador haya escogido para participar en una votación pueden acceder a esta cuando está iniciada. En el momento en el que el administrador la arranca definitivamente, pueden emitir su voto y ver el resultado.

CU06 - Modificar votante

Por un lado, los administradores pueden modificar todos los datos personales de cualquier votante, exceptuando la contraseña de acceso. Por otro lado, cualquier votante o administrador puede modificar todos los datos de su propio perfil.

CU07 - Listar y visualizar votaciones

Los administradores tienen la posibilidad de visualizar el listado completo de votaciones del sistema y acceder a los detalles de cada una de estas. Además, puede filtrar el listado por diferentes campos para reducir la búsqueda.

Por otro lado, los votantes también pueden visualizar el listado de votaciones a los cuales tienen acceso, sea de participación o solo de visualización. También pueden filtrar el listado y tienen la posibilidad de mostrar solo aquellas votaciones en las que participan.

7.2.2. Validación de interfaces

Para evitar errores y futuros problemas, el sistema implementa validadores que comprueban los datos, tanto en el *frontend* como en el *backend*.

Estos validadores verifican que los diferentes campos de información de las votaciones y los usuarios sean correctos y cumplan todos los requisitos. Algunos de los aspectos que se validan son:

- Se verifica que todos los campos obligatorios estén presentes
- Se comprueba que los campos no superen la longitud máxima o no lleguen a la longitud mínima.
- Se comprueba que las claves únicas como el DNI o el correo electrónico no estén repetidos.
- Se verifica que las contraseñas, los correos o los DNI tienen un formato adecuado.

En la figura 7.4 se incluyen algunos de los mensajes de error que se muestran en los diferentes formularios de las interfaces, como la página de inicio, la edición del perfil o la inserción de una nueva votación.

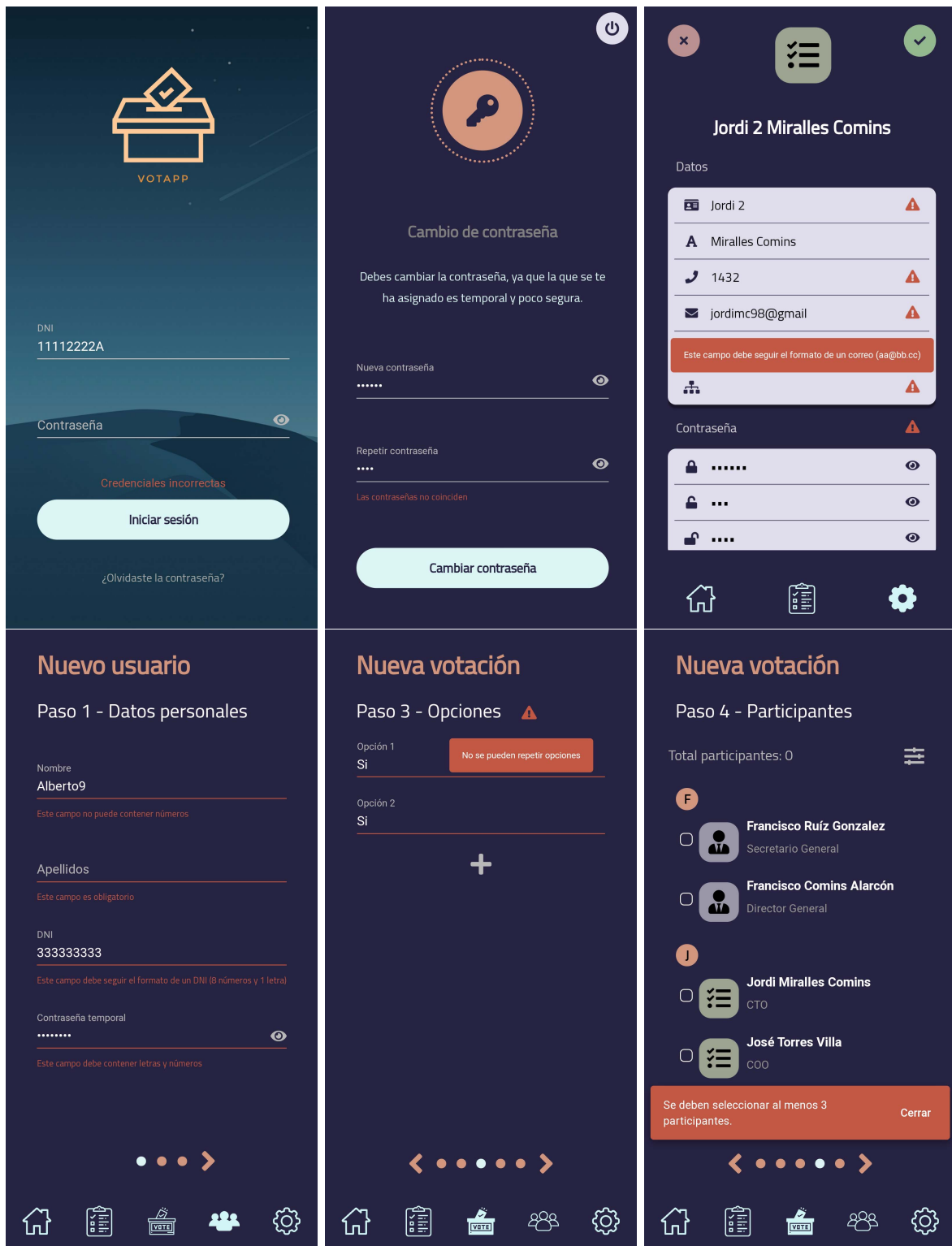


Figura 7.4: Interfaces - Validadores de campos.

7.2.3. Validación de la seguridad

Como ya se ha comentado anteriormente, este proyecto tiene como objetivo el desarrollo de una aplicación web que permita realizar votaciones garantizando la confidencialidad y la seguridad. Por ello, es importante establecer técnicas que proporcionen un alto nivel de seguridad.

En este proyecto, se han utilizado herramientas relevantes que permiten garantizar la seguridad de las votaciones. Entre ellas, destacan:

- Acceso a la aplicación mediante una red privada virtual en la que cada votante tiene una IP fija. Por tanto, en una votación solo pueden participar aquellos dispositivos cuya IP dentro de la VPN esté asociada a un votante autorizado. Este procedimiento dificulta notablemente la manipulación de la votación.
- Además, para acceder a los módulos de la aplicación y participar en las votaciones, es necesario un *token* JWT válido proporcionado por la aplicación tras iniciar sesión con credenciales válidas.
- Para garantizar la confidencialidad, se utilizan los protocolos HTTPS y WSS (Websocket *Secure*). Esto implica que todas las conexiones entre cliente-servidor o cliente-cliente se realizan cifradas.
- A esto se le añade la propia naturalidad del protocolo de votación, en el cual los mensajes viajan cifrados con las claves públicas de los diferentes participantes.

Aun así, el protocolo incluye mecanismos para detectar algún problema o alteración durante las votaciones, como por ejemplo el almacenamiento de los pasos intermedios en una estructura de datos.

En general, existen tres tipos de errores o excepciones que se pueden dar:

- El administrador detiene la votación.
- Se produce alguna alteración en la votación. Esta se detecta mediante las comprobaciones con los datos de la estructura almacenada.
- Se produce algún error en el canal de comunicación de algún participante o el administrador.

En los tres casos, se notifica a todos los participantes y se cierra la votación. En la figura 7.5 se observan las interfaces con los tres posibles mensajes de error.

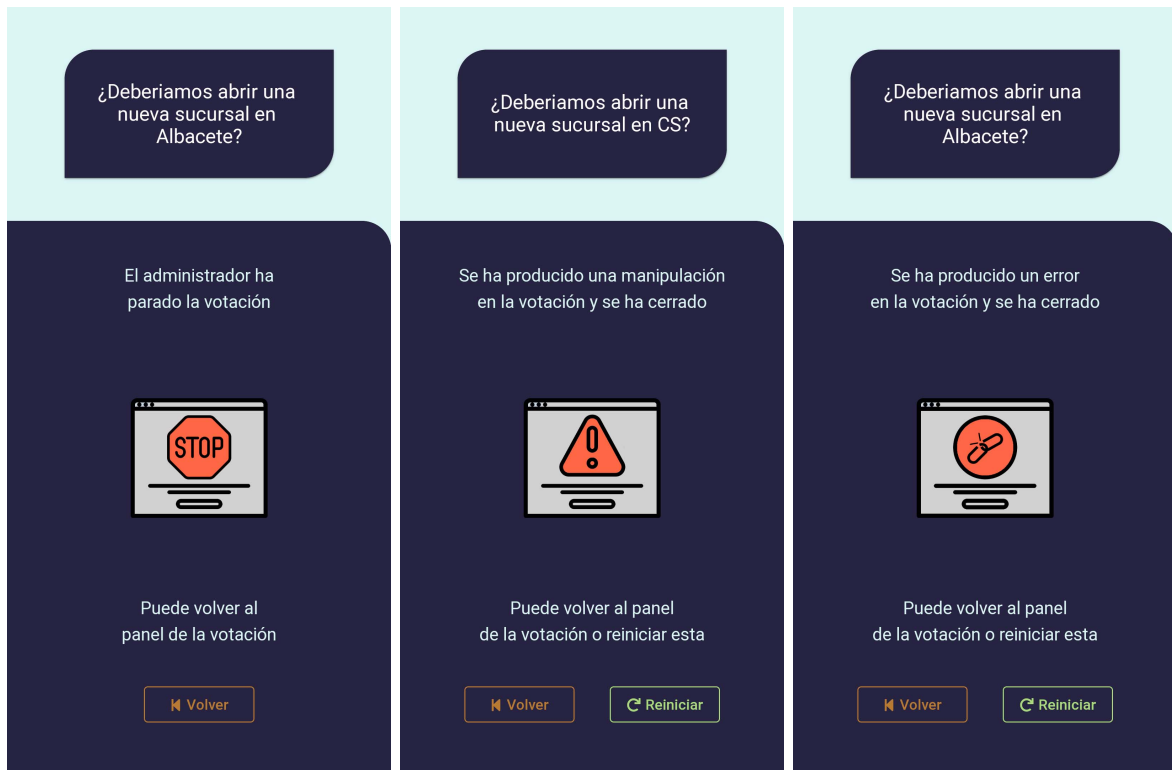


Figura 7.5: Interfaces - Mensajes de error durante la votación.

Finalmente, como se puede ver en la figura 7.6, se han implementado algunas interfaces que muestran problemas o errores globales del sistema, como cuando un usuario intenta acceder a un recurso prohibido o inexistente o cuando hay problemas con la conexión al servidor.

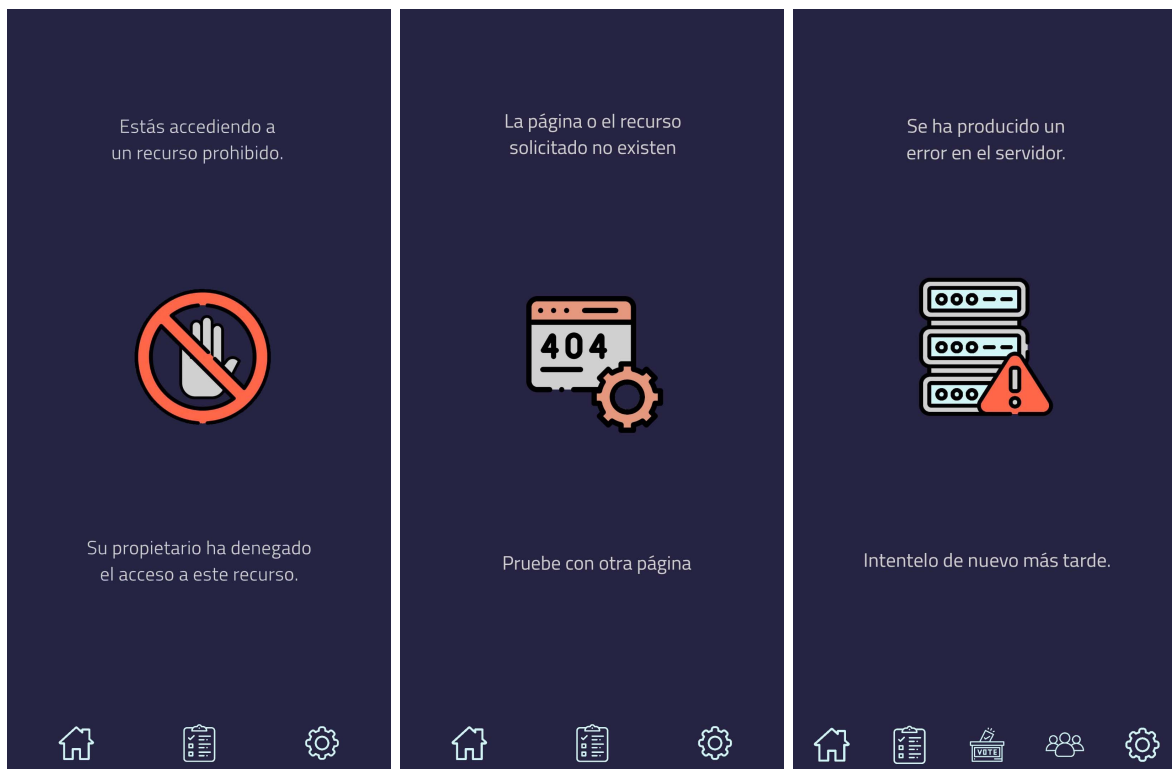


Figura 7.6: Interfaces - Mensajes de error globales del sistema.

Capítulo 8

Conclusiones y mejoras

Contenidos

8.1. Conclusiones	107
8.1.1. Conclusiones técnicas	107
8.1.2. Conclusiones personales	108
8.2. Posibles mejoras	108

El presente capítulo concluye la memoria técnica del proyecto. En este, se incluye la valoración sobre el proyecto realizado, tanto a nivel técnico como personal.

Además, se describen las posibles mejoras y el trabajo futuro que se pueden realizar sobre el proyecto para aumentar su funcionalidad.

8.1. Conclusiones

Tal como ha sido explicado anteriormente, este proyecto tiene como objetivo desarrollar una aplicación web que permita la realización de votaciones telemáticas seguras dentro de una organización.

8.1.1. Conclusiones técnicas

El resultado del proyecto ha sido totalmente satisfactorio, ya que el sistema desplegado cubre todas las necesidades y los requisitos identificados durante las fases iniciales. Además, se han implementado diferentes técnicas de seguridad que permiten realizar las votaciones de forma segura y confidencial, aumentando así la fiabilidad del producto.

Las tecnologías escogidas han supuesto una gran ventaja para la implementación del sistema, tanto por su facilidad de uso como por su gran funcionalidad. Tanto Angular para el *frontend* como Node.js para el *backend* han demostrado ser *frameworks* de programación web muy robustos y completos.

Además, los servicios de DNS y VPN, mediante las utilidades de BIND9 y OpenVPN, han sido desarrollados de forma correcta y su funcionamiento ha sido el esperado.

En resumen, se ha desarrollado un sistema completo que incluye, dentro de una red privada virtual, una aplicación web progresiva, una base de datos funcional y un servidor de nombres. Mediante este sistema, los usuarios pueden participar en votaciones privadas de forma segura.

8.1.2. Conclusiones personales

A nivel personal, el proyecto también ha sido satisfactorio y beneficioso en diferentes ámbitos.

En el **ámbito formativo**, el proyecto ha permitido aumentar mis habilidades y capacidades informáticas notablemente. He podido formarme en el uso de nuevos algoritmos y lenguajes de programación como Angular, JavaScript, Node.js o RSA. Además, he podido adquirir aptitudes y conocimientos para desplegar diferentes servicios de red como una VPN o un servidor DNS.

Es importante destacar que la formación recibida en el grado a través de las diferentes asignaturas ha facilitado el desarrollo y la implementación de este proyecto.

En el **ámbito profesional**, el proyecto y la estancia en prácticas me han permitido conocer y familiarizarme con un entorno laboral y colaborativo. En general, estoy muy satisfecho con la entidad Eria Consulting y todo el grupo de compañeros.

Cabe destacar también que el hecho de haber desarrollado de forma satisfactoria un proyecto ideado y propuesto por mí hace que la experiencia sea aún más enriquecedora.

En conclusión, la valoración del proyecto y del trabajo realizado es positiva y altamente recomendable, ya que permite entrar en contacto con nuevos métodos de trabajo profesionales y conocer nuevas tecnologías informáticas.

8.2. Posibles mejoras

Para concluir la memoria de este proyecto, se resumen algunas de las posibles acciones o mejoras que no han podido ser implementadas durante el proyecto. Estas propuestas podrán desarrollarse en un futuro.

- Formar a los futuros usuarios en el uso de la aplicación.
- Pruebas de *pentesting*¹ que permitan verificar de forma más exhaustiva el nivel de seguridad de la aplicación. Se podría comprobar el nivel de respuesta frente ataques del tipo DoS² o de inyección SQL³.
- Añadir nuevas funcionalidades como por ejemplo:
 - Permitir duplicar o copiar el contenido de una votación para facilitar la creación de nuevas.
 - Permitir agrupar a los votantes en grupos para facilitar la elección de estos al crear una votación. (*e.g.* miembros del consejo).

¹ Las **pruebas de *pentesting*** son ataques intencionados sobre un sistema informático para detectar debilidades y fallos de seguridad.

² Un **ataque DoS** (*Denial of Service*) es un tipo de ataque informático en el que se realizan múltiples peticiones sobre un servidor para sobrecargarlo y colapsar su funcionamiento.

³ Un **ataque de inyección SQL** es un tipo de ataque informático en el cual se introduce código SQL a través de los campos de un formulario. Se utiliza para realizar operaciones de corrupción, modificación o borrado de datos.

Bibliografía

- [1] Eria Consulting: *Logotipo de la empresa*. <http://www.eriaconsulting.com>. [Fecha de consulta: 23 de Junio de 2020].
- [2] Flaticon: *Flaticon, the largest database of free vector icons*. <https://www.flaticon.com>. [Fecha de consulta: 23 de Junio de 2020].
- [3] Wikipedia: *Seguridad informática*. https://es.wikipedia.org/wiki/Seguridad_inform%C3%A1tica, 2020. [Fecha de consulta: 23 de Junio de 2020].
- [4] ISO/IEC: *Tecnología de la información - Técnicas de seguridad - Sistemas de Gestión de la Seguridad de la Información (SGSI) - Visión de conjunto y vocabulario*. ISO/IEC 27000:2018, Suiza, 2018.
- [5] Carracedo Gallardo, J.: *Seguridad en redes telemáticas*, capítulo 11.5, páginas 495–501. McGraw-Hill, 1ª edición, 2004.
- [6] MDN: *Progressive web apps (PWAs)*. https://developer.mozilla.org/en-US/docs/Web/Progressive_web_apps, 2020. [Fecha de consulta: 23 de Junio de 2020].
- [7] Lucena López, Manuel J.: *Criptografía y Seguridad en Computadores*. Universidad de Jaén, 4ª edición, 2011.
- [8] Schneier, B.: *Applied Cryptography: Protocols, Algorithms, and Source Code in C*, capítulo 6.1, páginas 130–133. Wiley, 2ª edición, 1996.
- [9] Microsoft: *Virtual Private Networking: An Overview*. [https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-2000-server/bb742566\(v=technet.10\)](https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-2000-server/bb742566(v=technet.10)), 2009. [Fecha de consulta: 23 de Junio de 2020].
- [10] Atlassian: *¿Qué es Trello?* <https://trello.com/es/about>. [Fecha de consulta: 23 de Junio de 2020].
- [11] GitHub: *GitHub features: the right tools for the job.?* <https://github.com/features>. [Fecha de consulta: 23 de Junio de 2020].
- [12] JSON.org: *Introducción a JSON*. <https://www.json.org/json-es.html>. [Fecha de consulta: 23 de Junio de 2020].
- [13] JWT.io: *JSON Web Tokens Introduction*. <https://jwt.io/introduction>. [Fecha de consulta: 23 de Junio de 2020].
- [14] World Wide Web Consortium: *HTML & CSS - W3C*. <https://www.w3.org/standards/webdesign/htmlcss>. [Fecha de consulta: 23 de Junio de 2020].
- [15] MDN: *JavaScript*. <https://developer.mozilla.org/es/docs/Web/JavaScript>, 2020. [Fecha de consulta: 23 de Junio de 2020].
- [16] Microsoft: *Typed JavaScript at Any Scale*. <https://www.typescriptlang.org>. [Fecha de consulta: 23 de Junio de 2020].
- [17] Angular.io: *Introduction to Angular concepts*. <https://angular.io/guide/architecture>. [Fecha de consulta: 23 de Junio de 2020].

- [18] Node.js: *Acerca de Node.js*. <https://nodejs.org/es/about>. [Fecha de consulta: 23 de Junio de 2020].
- [19] Oracle: *MySQL Database Service*. <https://www.oracle.com/mysql>. [Fecha de consulta: 23 de Junio de 2020].
- [20] OpenVPN: *What is OpenVPN?* <https://openvpn.net/faq/what-is-openvpn>. [Fecha de consulta: 23 de Junio de 2020].
- [21] phpMyAdmin: *Bringing MySQL to the web*. <https://www.phpmyadmin.net>. [Fecha de consulta: 23 de Junio de 2020].
- [22] Atlassian: *Precios de Trello*. <https://trello.com/es/pricing>. [Fecha de consulta: 23 de Junio de 2020].
- [23] GitHub: *Plans for all developers*. <https://github.com/pricing>. [Fecha de consulta: 23 de Junio de 2020].
- [24] Postman: *Plans & Pricing*. <https://www.postman.com/pricing>. [Fecha de consulta: 23 de Junio de 2020].
- [25] Indeed: *Salarios para empleos de Analista programador/a junior en España*. <https://es.indeed.com/salaries/analista-programador-junior-Salaries>, 2020. [Fecha de consulta: 23 de Junio de 2020].
- [26] Indeed: *Salarios para empleos de Programador/a junior en España*. <https://es.indeed.com/salaries/programador-junior-Salaries>, 2020. [Fecha de consulta: 23 de Junio de 2020].
- [27] Indeed: *Salarios para empleos de Desarrollador/a senior en España*. <https://es.indeed.com/salaries/desarrollador-senior-Salaries>, 2020. [Fecha de consulta: 23 de Junio de 2020].
- [28] UML.org: *What is UML?* <https://www.uml.org/what-is-uml.htm>, 2005. [Fecha de consulta: 23 de Junio de 2020].
- [29] Marqués, M: *Bases de datos*, capítulo 5.3, páginas 101–103. Universitat Jaume I, Servei de Comunicació i Publicació, 1^a edició, 2011.
- [30] Shneiderman, B: *Designing the User Interface: Strategies for Effective Human-Computer Interaction*. Addison-Wesley Longman Publishing, 3^a edició, 1997.
- [31] Font Awesome: *Free Icons*. <https://fontawesome.com>. [Fecha de consulta: 23 de Junio de 2020].
- [32] Angular.io: *Material Design components for Angular*. <https://material.angular.io>. [Fecha de consulta: 23 de Junio de 2020].
- [33] Belmonte, O., Granell C & Erdozain M.C.: *Desarrollo de proyectos informáticos con tecnología Java*. Universitat Jaume I, Servei de Comunicació i Publicació, 1^a edició, 2012.
- [34] Google Developers: *Introduction to Push Notifications*. <https://developers.google.com/web/ilt/pwa/introduction-to-push-notifications>, 2019. [Fecha de consulta: 23 de Junio de 2020].