



UNIVERSIDAD NACIONAL DE CÓRDOBA
FACULTAD DE CIENCIAS EXACTAS FÍSICAS Y NATURALES

SISTEMA DE GESTIÓN DE CASOS JUDICIALES

Realizado por
Julieta Abigail Prieto

Matrícula 42304327

Mail julieta.prieto@mi.unc.edu.ar

Contacto +54 9 351 211-2630

Proyecto Integrador de la Carrera
Ingeniería en Computación

Dirigido por
Danilo Páez

Co-dirigido por
Laura Díaz

Córdoba, Argentina
Marzo del 2024

Agradecimientos

Quiero expresar mi sincero agradecimiento a quienes han contribuido de manera significativa a mi trayectoria académica y personal durante el desarrollo de esta carrera.

En primer lugar, agradezco profundamente a mi madre por su inquebrantable apoyo y por haber creído en mi capacidad para alcanzar cada uno de mis objetivos. A mi padre por enseñarme la perseverancia y la disciplina. A mi familia y amigos, les agradezco por su constante compañía y aliento a lo largo de este exigente camino.

A mi director Danilo Paez, a mi co-directora Laura Díaz y al profesor Daniel Britos por su valiosa guía durante este proyecto.

A mis valiosos compañeros, quienes no solo fueron un apoyo fundamental, sino que también añadieron alegría a esta experiencia y poder compartir este trayecto educativo y el conocimiento adquirido con ellos ha sido una experiencia enriquecedora y significativa.

Mi reconocimiento también se extiende a los respetados profesores que dejaron una marca indeleble en mi formación. A aquellos que, con entusiasmo y alegría, transmitieron su pasión por el conocimiento. A los que dedicaron paciencia infinita para explicar conceptos hasta lograr una comprensión completa. Les agradezco por sus valiosos consejos, por su dedicación a nuestra educación, por las horas adicionales de consulta y por su constante esfuerzo por avivar la chispa del aprendizaje en cada uno de nosotros.

Agradezco a mi país y a esta prestigiosa universidad por ofrecerme la oportunidad de cursar esta carrera. Me comprometo a retribuir este privilegio contribuyendo al servicio de la sociedad.

Finalmente, mi gratitud se dirige a mi abuela, cuya presencia he sentido a lo largo de toda mi carrera. Su apoyo desde el cielo ha sido una guía constante.

Estoy profundamente agradecido a todos por formar parte de este viaje educativo y formativo que ha enriquecido mi vida.

Resumen

Esta iniciativa se encuadra como un proyecto integrador en la carrera de Ingeniería en Computación de la Facultad de Ciencias Exactas, Físicas y Naturales, impulsado por el laboratorio de IALAB.

El enfoque principal de este proyecto es la creación de un sistema de gestión integral que ofrece una plataforma web para optimizar la asignación de casos a diversas comisiones. Cada comisión representa una subdivisión de una asignatura, donde un grupo de docentes, responsables de una materia, guían a los estudiantes de último año de la carrera de Abogacía de la Facultad de Derecho durante sus prácticas, proporcionando este servicio de manera gratuita a la comunidad.

Este sistema gestiona y organiza los casos, que son en esencia consultas patrocinadas, dentro de cada comisión. Además, simplifica el control y la supervisión a cargo de los responsables del Patrocinio con paneles administrativos, tableros amigables, alertas y notificaciones. También integra Google Forms para el registro eficiente de nuevos consultantes y consultas.

Palabras clave: System software, Web sites, Sockets, Databases, Selenium, Google.

Abstract

This initiative falls under the category of an integrative project in the Computer Engineering program at the Facultad de Ciencias Exactas, Físicas y Naturales, driven by the IALAB laboratory.

The primary focus of this project is the development of a comprehensive management system that provides a web platform to streamline the allocation of cases to various committees. Each committee represents a subdivision of a subject, where a group of teachers, responsible for a subject, guide final-year students in the Law degree program at the Faculty of Law during their internships, providing this service free of charge to the community.

This system manages and organizes cases, essentially sponsored consultations, within each committee. Additionally, it simplifies control and supervision under the sponsorship leaders with administrative panels, user-friendly dashboards, alerts, and notifications. It also integrates Google Forms for the efficient registration of new clients and consultations.

Keywords: System software, Web sites, Sockets, Databases, Selenium, Google.

Índice general

1. Introducción	1
1.1. Introducción	1
1.2. Contexto del PI	1
1.3. Objetivo General del Proyecto	1
1.4. Objetivos Específicos del Proyecto	2
1.5. Alcance del Proyecto	2
1.6. Acrónimos y Abreviaturas	2
2. Marco Teórico	4
2.1. API REST	4
2.2. Integración Continua y Entrega o Despliegue continuo - CI/CD	4
2.3. Falsificación de Petición en Sitios Cruzados - CSRF	5
2.4. App Scripts	5
2.4.1. Secciones de App Script	6
2.4.2. Contexto del Proyecto	6
2.5. Web Sockets	6
2.6. Patrón de Diseño Pub/Sub	7
2.7. Docker Swarm	8
2.8. Normalización de la Base de Datos	8
2.8.1. Primera Forma Normal (1FN)	9
2.8.2. Segunda Forma Normal (2FN)	9
2.8.3. Tercera Forma Normal (3FN)	9
2.8.4. Forma Normal de Boyce-Codd (FNBC)	9
2.9. Propiedades ACID en Bases de Datos	9
3. Gestión de las Configuraciones	11
3.1. Introducción	11
3.2. Herramientas para la Administración de Configuraciones	11
3.3. Herramienta de Control de Versiones	12
3.3.1. Dirección de Acceso	12
3.3.2. Plan del Esquema de Ramas	12
3.3.3. Normas de Etiquetado y de Nombramiento de los Archivos . . .	12
3.3.4. Políticas de Fusión de Archivos y Etiquetado en Conformidad con el Progreso de Calidad en los Entregables	13
3.3.5. Convención de Commits y Merge Request	13
3.4. Estructura de Directorios	14
3.4.1. Repositorio <i>cms-deploy</i>	14
3.4.2. Repositorio <i>cms-docs</i>	15
3.4.3. Repositorio <i>proyecto-patrocinio</i>	15
3.4.4. Repositorio <i>cms-test</i>	16
3.5. Herramientas de Gestión de Tareas y Defectos	17
3.5.1. Confluence	17

3.5.2.	Issues	17
3.5.3.	Épicas	18
3.6.	Herramienta de Integración Continua	20
3.7.	Formato de Entrega	21
3.8.	Change Control Board - CCB	21
3.8.1.	Metodología Ágil	21
3.8.2.	Miembros y Funciones	22
4.	Requerimientos y Especificaciones	24
4.1.	Introducción	24
4.2.	Alcance del Producto	24
4.3.	Referencias	24
4.4.	Descripción General	26
4.4.1.	Perspectiva del Producto	26
4.4.2.	Funciones del Producto	26
4.4.3.	Tipos de Usuarios y Características	26
4.4.4.	Entorno Operativo	27
4.5.	Restricciones de Diseño e Implementación	28
4.6.	Modelo Conceptual de Datos - DER	28
4.7.	Requerimientos de Interfaces Externas	30
4.7.1.	Consultoría	31
4.7.2.	Tablero de Trabajo para Cada Comisión	32
4.7.3.	Inicio de Sesión	33
4.7.4.	Información de la Consulta	33
4.7.5.	Tablas de Control	34
4.8.	Requerimientos Funcionales	35
4.8.1.	Sistema de Solicitudes de Asignación de Casos	36
4.8.2.	Sistema de Gestión de Casos por Comisión	37
4.8.3.	Sistema de Registro de Clientes y Consultas	37
4.8.4.	Sistema de Notificaciones y Alertas	38
4.8.5.	Registro y Autenticación de Usuarios	38
4.9.	Requerimientos No Funcionales	38
4.9.1.	Requerimientos del producto	39
4.9.2.	Requerimientos de la organización	39
4.9.3.	Requerimientos externos	40
4.10.	Definición de Roles y Responsabilidades	41
4.11.	Diagrama de Caso de Uso del Sistema	41
4.12.	Diagrama de Secuencia Nominal Simplificado	43
4.13.	Matriz de Trazabilidad	45
5.	Arquitectura y Diseño	46
5.1.	Introducción	46
5.2.	Modelado del sistema	46
5.2.1.	Modelo de Contexto	46
5.2.2.	Modelo de Contenedores	47
5.2.3.	Modelo de Componentes	48
5.3.	Arquitectura	50
5.4.	Desarrollo Modular de la Arquitectura REST en DRF	51

5.5.	Patrón de Arquitectura Mediador (Broker pattern)	51
5.6.	Implementación del Patrón de Diseño Pub/Sub sobre el Protocolo de comunicación Web Sockets	52
5.7.	Implementación Arrastrar y soltar	53
6.	Implementación y Prueba	55
6.1.	Introducción	55
6.2.	Etapas de Implementación	55
6.2.1.	Stack Tecnológico Seleccionado	55
6.2.2.	Diagrama de Paquetes en el Backend	55
6.2.3.	Diagrama de Paquetes en el Frontend	56
6.2.4.	Integración con Google Forms	57
6.2.5.	Nginx como Servidor y Reverse Proxy	63
6.2.6.	Integración Continua	64
6.2.7.	Protección contra Ataques CSRF:	66
6.2.8.	Certificación de Seguridad y Cifrado SSL/TLS	66
6.3.	Diseño e Informe de Pruebas	67
6.3.1.	Pruebas de Caja Blanca	67
6.3.2.	Pruebas de Caja Negra	67
6.3.3.	Pruebas Unitarias	68
6.3.4.	Diseño de Escenarios de Prueba	68
7.	Despliegue y Operación	84
7.1.	Introducción	84
7.2.	Instalación de la Plataforma	84
7.2.1.	Configuración de Templates	84
7.2.2.	Configuración de Variables de Entorno en el Archivo <code>.env</code>	85
7.2.3.	Configuración de Variables de Entorno en <code>backend.env</code>	86
7.2.4.	Configuración de Variables de Entorno en <code>frontend.env</code>	87
7.2.5.	Configuración de Variables de Entorno para la Base de Datos	88
7.2.6.	Archivo de Configuración de Nginx	88
7.2.7.	Archivo <code>terms_and_policies</code>	89
7.3.	Despliegue de la Plataforma	89
7.4.	Configuración	90
7.5.	Registro de Usuarios	92
7.6.	Permisos de Usuario	96
7.7.	Permisos de Usuario	97
7.8.	Administración	97
7.9.	Tablero de Trabajo de Consultoría	98
7.10.	Panel de Control	100
7.11.	Tablero de Trabajo para la Comisión	102
7.12.	Detalles de la Consulta	104
7.13.	Configuración de Cuenta	107
8.	Trabajos Futuros	108
8.1.	Servicio Unificado de Logs	108
8.2.	Cambio en la Interfaz de Consultoría	108
8.3.	Tests de Performance y Estrés	108

8.4. Almacenamiento Encriptado	108
8.5. Panel de Estadísticas y Campos Adicionales	109
8.6. Backup Automático	109
8.7. Gestión de Secretos	109
9. Conclusiones	110
A. Diccionario de Base de Datos	112
A.1. Tabla Board	112
A.2. Tabla Board-User	112
A.3. Tabla Calendar	113
A.4. Tabla Event	113
A.5. Tabla Card	113
A.6. Tabla Child	113
A.7. Tabla Client	115
A.8. Tabla Tel	117
A.9. Tabla Comment	117
A.10. Tabla File	117
A.11. Tabla Consultation	118
A.12. Tabla Request Consultation	119
A.13. Tabla Panel	120
B. Matriz de Relaciones	121
C. Configuración de Nginx	125
C.1. Configuración del Nginx como Reverse Proxy	125
C.2. Configuración del Nginx como Servidor	127
D. Endpoints de Formularios	128
D.1. Endpoint del Formulario Registro de Cliente	128
D.2. Endpoint del Formulario Registro de Hijo	130
D.3. Endpoint del Formulario Consulta	131
E. Archivos de Configuración para el Despliegue de la Plataforma	133
E.1. Archivo de configuración .env	133
E.2. Archivo de configuración backend.env	133
E.3. Archivo de configuración frontend.env	134
E.4. Archivo de configuración portgres.env	135
F. Test Unitarios	136
F.1. Ejemplo de Test Unitario en Backend	136
F.2. Ejemplo de Test Unitario en Frontend	137
G. Bibliografía	138

Índice de figuras

2.1. Ataque CSRF	5
2.2. Web Sockets	7
2.3. Publicador Suscriptor	8
3.1. Ramas de Git	12
3.2. Estructura de directorios del repositorio <i>cms-deploy</i>	15
3.3. Estructura de directorios del repositorio <i>proyecto-patrocinio</i>	16
3.4. Estructura del directorio de pruebas en <i>cms-test</i>	16
3.5. CI/CD	21
4.1. Diagrama Entidad Relación	29
4.2. Implementación del Modelo Entidad Relación	30
4.3. Mockup de la Interfaz Gráfica de Usuario (GUI) para la Página Consultoría	31
4.4. Mockup de la Interfaz Gráfica de Usuario (GUI) para el Tablero de Comisión	32
4.5. Mockup de la Interfaz de Inicio de Sesión	33
4.6. Mockup de la Interfaz de Información de Consulta	34
4.7. Mockup de la Interfaz para las Tablas de Consultas	35
4.8. Diagrama Caso de Uso	42
4.9. Diagrama de Secuencia Para Registro de Usuario	43
4.10. Diagrama de Secuencia Simplificado	44
5.1. Diagrama de Contexto C4	47
5.2. Diagrama de Contenedores C4	48
5.3. Diagrama de Componentes C4 del Contenedor CMS Core	49
5.4. Diagrama de Componentes C4 del Contenedor Web App	50
5.5. Diagrama de un Módulo en DRF	51
5.6. Diagrama Implementación del Patrón Pub/Sub	52
5.7. Estructura Drag and Drop	53
5.8. Diagrama de Componentes Drag and Drop	54
6.1. Diagrama de Paquetes del Backend	56
6.2. Diagrama de Paquetes del Frontend	56
6.3. Captura de pantalla del código en Google Apps Script.	59
6.4. Configuración de activadores.	59
6.5. Activador para el envío del formulario a la API.	60
6.6. Ubicación del Plugin de Menú en Google Forms	61
6.7. Interfaz de Configuración del Plugin de Menú	62
6.8. Activador para el Refresco Automático del Token	63
6.9. Implementación de Pipelines en Github Actions	65
6.10. Certificado Let's Encrypt obtenido con éxito	67
7.1. Diagrama de Despliegue	90
7.2. Configuración de Sitios en el Panel de Administración.	91

7.3. Creación de Pizarras en la Sección de Administración.	91
7.4. Pantalla de Registro de Usuarios.	92
7.5. Confirmación por Correo Electrónico.	93
7.6. Configuración de Permisos de Usuario.	94
7.7. Creación de Relación Usuario-Comisión.	94
7.8. Lista de Relaciones Usuario-Comisión.	95
7.9. Página de Inicio de Sesión.	96
7.10. Interfaz de Administración de Django.	97
7.11. Página de Consultoría	98
7.12. Historial de Asignaciones de una Comisión	99
7.13. Panel Expandido de Historial de Asignaciones de una Comisión	99
7.14. Formulario para Crear Consulta en la Página de Consultoría	100
7.15. Tabla de Consultas en el Panel de Control.	101
7.16. Tabla de Clientes en el Panel de Control.	101
7.17. Aplicación de un Filtro en la Columna Progress State	102
7.18. Página Board de la Comisión	103
7.19. Botones para eliminar o crear un Panel del Board	103
7.20. Ventana de Información de Consulta.	104
7.21. Sección de Comentarios y Archivos.	105
7.22. Calendario de Consulta.	105
7.23. Detalle de Evento del Calendario de Consulta.	106
7.24. Vista Agenda de Calendario de Consulta.	106
7.25. Página de Configuración de Cuenta.	107

Índice de extractos de código

C.1. Configuración de Nginx Reverse Proxy	125
C.2. Configuración de Nginx en el Servidor	127
D.1. Body de Ejemplo	128
D.2. Body de Ejemplo	131
D.3. Body de Ejemplo	131
E.1. Archivo de configuración .env	133
E.2. Archivo de configuración backend.env	133
E.3. Archivo de Configuración frontend.env	134
E.4. Archivo de configuración portgres.env	135
F.1. Test Unitario Backend	136
F.2. Test Unitario Frontend	137

1. Introducción

1.1. Introducción

En este capítulo se delimita el contexto de la tesis, al mismo tiempo que se describen los objetivos, el alcance y la estructura de este documento.

1.2. Contexto del PI

El origen de este producto surgió a raíz del contacto con el laboratorio IALAB, quienes solicitaron el desarrollo de un sistema de gestión diseñado específicamente para potenciar y optimizar el Patrocinio Jurídico de la Facultad de Derecho de la Universidad de Buenos Aires.

El “Patrocinio Jurídico Gratuito”, con casi un siglo de dedicación, se erige como un proyecto comprometido con la provisión de asistencia legal a aquellos en situación de vulnerabilidad económica y social. Esta iniciativa, integrada como práctica profesional en el plan de estudios de la carrera de abogacía, se distingue como un servicio singular en el país.

La problemática identificada en la administración de casos por parte de esta entidad se centra en la utilización actual de hojas de cálculo y herramientas de pago empleadas para la gestión de casos, y Google Forms como sistema de recopilación de datos. Sin embargo, estas soluciones no satisfacen completamente las necesidades específicas del organismo, que requiere un software integral para la gestión, organización y automatización parcial de sus procesos.

En esta fase del proyecto, se busca reemplazar y unificar el sistema de gestión de casos, como así también automatizar eficientemente la carga de formularios en el sistema, contribuyendo un producto de valor al Proyecto Patrocinio.

El desarrollo de este sistema representa la concepción de una primera base que servirá como cimiento para futuras expansiones. Este enfoque busca establecer una plataforma modular y versátil que aborde las necesidades actuales del patrocinio jurídico de la UBA.

1.3. Objetivo General del Proyecto

Diseñar e implementar un software integral de gestión de casos para el patrocinio jurídico de la Universidad de Buenos Aires (UBA).

1.4. Objetivos Específicos del Proyecto

1. Realizar un análisis exhaustivo para comprender a fondo las limitaciones al proceso actualmente en uso para el procesamiento de los casos, así como al sistema de recolección de datos basado en Google Forms.
2. Releva, analizar y construir los requisitos tanto funcionales como no-funcionales solicitados por el cliente.
3. Diseñar e implementar una arquitectura modular que permita una gestión eficiente y escalable de casos.
4. Diseñar una estructura de base de datos que permita una gestión eficaz de la información.

1.5. Alcance del Proyecto

En el marco de este proyecto, se abordarán los siguientes procesos:

1. Diseño, Desarrollo e Implementación del Software de Gestión de Casos:

El enfoque principal estará en la concepción, desarrollo y despliegue de un software de gestión de casos diseñado específicamente para satisfacer las necesidades del patrocinio jurídico de la Universidad de Buenos Aires (UBA).

2. Integración con Google Forms:

Se incorporará la integración de Google Forms para automatizar la carga eficiente de formularios en el sistema, mejorando así la recopilación de datos y simplificando el proceso para el patrocinio jurídico.

3. Optimización del Sistema de Solicitud y Control Histórico:

El software se orientará hacia la facilitación y optimización del sistema de solicitud, asegurando un control histórico efectivo de los casos por comisión. Esto se traducirá en una gestión más eficiente de los procesos asociados con la presentación y seguimiento de casos.

4. Registro Histórico por Comisión:

Se contemplará la mejora del registro histórico mediante la posibilidad de cargar archivos relevantes y comentarios asociados a cada caso. Esto permitirá un seguimiento más detallado y completo de la evolución de los casos a lo largo del tiempo.

1.6. Acrónimos y Abreviaturas

En el presente documento, se utilizan los siguientes acrónimos y abreviaturas:

Acrónimos	Descripción
API	Interfaz de Programación de Aplicaciones (por sus siglas en inglés, <i>Application Programming Interface</i>)
HTTP	Protocolo de Transferencia de Hipertexto (por sus siglas en inglés, <i>Hypertext Transfer Protocol</i>)
IP	Protocolo de Internet (<i>Internet Protocol</i>)
JSON	Notación de Objetos de JavaScript (<i>JavaScript Object Notation</i>)
REST	Transferencia de Estado Representacional (<i>Representational State Transfer</i>)
DNS	Sistema de Nombres de Dominio (<i>Domain Name System</i>)
SSL/TLS	Capa de Conexión Segura / Protocolo de Seguridad de la Capa de Transporte (<i>Secure Sockets Layer / Transport Layer Security</i>)
SQL	Lenguaje de Consulta Estructurada (<i>Structured Query Language</i>)
CORS	Intercambio de recursos de origen cruzado (<i>Cross Origin Resource Sharing</i>)
CSRF	Falsificación de Petición en Sitios Cruzados (<i>Cross-Site Request Forgery</i>)
NIST	Instituto Nacional de Estándares y Tecnología (<i>National Institute of Standards and Technology</i>)
CI	Integración Continua (<i>Continuous Integration</i>)
CD	Entrega Continua (<i>Continuous Delivery</i>)
UBA	Universidad de Buenos Aires
UNC	Universidad Nacional de Córdoba
ASGI	Interfaz de Puerta de Enlace Asíncrona del Servidor (<i>Asynchronous Server Gateway Interface</i>)
WSGI	Interfaz de Puerta de Enlace del Servidor Web (<i>Web Server Gateway Interface</i>)
PK	Clave Primaria (<i>Primary Key</i>).

Cuadro 1.1: Lista de Acrónimos y Abreviaturas

2. Marco Teórico

2.1. API REST

Según RedHat [4] se explica los siguientes conceptos relacionados a las API REST:

API (Interfaz de Programación de Aplicaciones): Una API es un conjunto de reglas y definiciones que permite que distintos software se comuniquen entre sí. Sirve como intermediario para que una aplicación pueda utilizar las funciones o servicios de otra aplicación, facilitando la integración y la interacción.

API REST (Interfaz de Programación de Aplicaciones basada en Transferencia de Estado Representacional): Una API REST es una implementación de una API que sigue los principios de la arquitectura REST. Se caracteriza por utilizar estándares y restricciones definidos por REST para facilitar la comunicación eficiente entre sistemas distribuidos. Utiliza solicitudes HTTP para realizar operaciones sobre recursos, y la transferencia de representaciones de estados de recursos se lleva a cabo en formatos como JSON, HTML, XML, entre otros.

REST (Transferencia de Estado Representacional): REST es un conjunto de principios arquitectónicos que definen cómo deben comunicarse los sistemas en una red. Se basa en la idea de que cada recurso (como un objeto o servicio) tiene una representación única y direccionable a través de un URI (Identificador de Recurso Uniforme). REST utiliza operaciones estándar de HTTP (como GET, POST, PUT, DELETE) para manipular estos recursos. Su enfoque en la simplicidad, escalabilidad y la independencia entre el cliente y el servidor lo hace ampliamente adoptado en el diseño de servicios web.

2.2. Integración Continua y Entrega o Despliegue continuo - CI/CD

Según Github [3] se explica los siguientes conceptos:

CI/CD, que abrevia Integración Continua y Entrega Continua, o en ocasiones Despliegue Continuo, constituye un conjunto de prácticas destinadas a automatizar el proceso de desarrollo de software, desde la integración de código hasta la entrega y, en algunos casos, el despliegue en entornos de producción.

En **Continuous Delivery**, entrega automáticamente cambios de código a entornos listos para su aprobación en producción, dejando el resto de los pasos manuales. En **Continuous Deployment**, implementa automáticamente cambios de código directamente a los clientes.

Los **pipelines** en GitHub se refieren a flujos de trabajo automatizados que implementan CI/CD. Estos pipelines permiten la ejecución automática de tareas como

pruebas, construcción y despliegue, asegurando la consistencia y eficiencia en el ciclo de vida del desarrollo de software.

En el contexto de este proyecto, se implementará un pipeline de CI/CD que utiliza Continuous Delivery, la automatización se detiene en la entrega de las imágenes de docker tagadas en DockerHub.

2.3. Falsificación de Petición en Sitios Cruzados - CSRF

En términos de seguridad informática, CSRF o Cross-Site Request Forgery, puede entenderse como una “vulnerabilidad que explota la confianza que un sitio web tiene en las solicitudes originadas desde el navegador del usuario” [7].

En otras palabras, CSRF implica que un atacante logra engañar al navegador del usuario para que realice acciones no deseadas en otro sitio web donde el usuario ya ha iniciado sesión. Este tipo de ataque se basa en la premisa de que, si un usuario ya ha autenticado su identidad en un sitio web, su navegador enviará solicitudes confiables a ese sitio, incluso si esas solicitudes son iniciadas desde otro sitio web malicioso. La Figura 2.1 ilustra un ejemplo de un ataque CSRF.

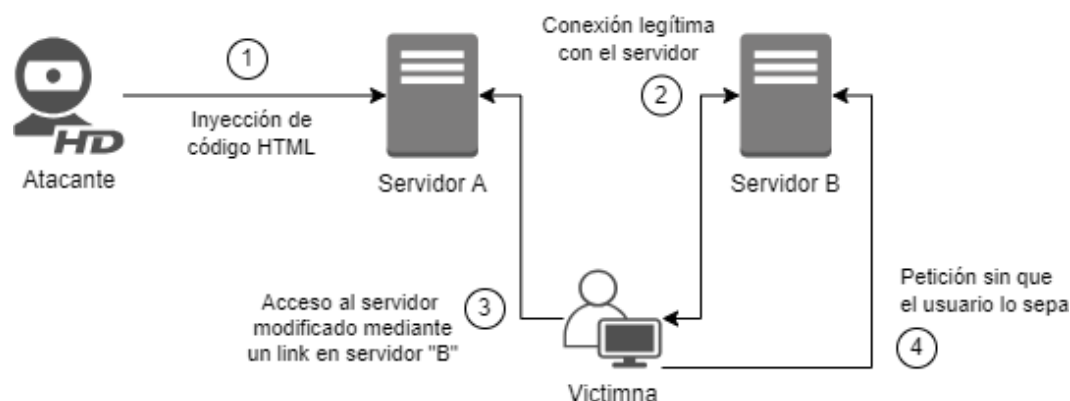


Figura 2.1: Ataque CSRF

En el contexto de este proyecto, para mitigar este riesgo, se implementó medidas de seguridad como tokens CSRF. Estos tokens actúan como códigos secretos que deben incluirse en cada solicitud que puede realizar cambios en el estado del servidor (POST, PATCH, PUT, DELETE) y son verificados por el servidor para garantizar la legitimidad de la solicitud, evitando así que se realicen acciones no autorizadas en nombre del usuario.

2.4. App Scripts

“Apps Script es la única plataforma con bajo nivel de codificación que agiliza y facilita la compilación de soluciones empresariales que integran, automatizan y extienden

Google Workspace.

Apps Script le permite compilar con HTML, CSS y JavaScript, sin que tenga que aprender un nuevo marco de trabajo exclusivo.” [12].

Apps Script ofrece una variedad de funcionalidades, permitiéndote realizar:

- Automatizaciones
- Creación de complementos
- Desarrollo de funciones personalizadas
- Creación de Web Apps Script, entre otras opciones.

2.4.1. Secciones de App Script

Dentro del archivo script creado con Google Apps Script, encontramos varias secciones que facilitan el desarrollo y la gestión de nuestros proyectos:

- **Información General:** Esta sección proporciona detalles sobre el script, la posibilidad de crear copias, visualizar el uso y los alcances utilizados.
- **Editor de Código:** La sección principal donde programamos e implementamos nuestro código.
- **Activadores:** Aquí creamos activadores o triggers basados en eventos o tiempo para ejecutar el script.
- **Ejecuciones:** Esta sección muestra el historial de ejecuciones, logs y resultados obtenidos.
- **Configuración:** Contiene configuraciones avanzadas para ajustar el comportamiento de nuestro script.

2.4.2. Contexto del Proyecto

En el marco de este proyecto, se empleó Google Apps Script como una solución integral para automatizar la carga de formularios en el sistema. Además, se utilizó para desarrollar plugins que mejoran la experiencia de gestión del usuario administrador mediante un menú y una interfaz gráfica intuitiva para la configuración eficiente de opciones.

2.5. Web Sockets

Según lo desarrollado en el artículo de IONOS [9] se explican los siguientes conceptos.

Un **socket** es un mecanismo de comunicación que permite que dos procesos en diferentes máquinas o en la misma máquina se comuniquen entre sí. En términos simples, un socket proporciona una interfaz para la comunicación entre procesos.

Un **WebSocket** es un protocolo de comunicación bidireccional en tiempo real sobre un único socket TCP. A diferencia de HTTP, que sigue un modelo de solicitud-respuesta, los WebSockets permiten la comunicación en ambas direcciones en cualquier momento. La Figura 2.2 ilustra el concepto de Web Sockets.

El proceso típico para establecer una conexión WebSocket es:

- **Handshake (apretón de manos):** El cliente envía una solicitud HTTP al servidor solicitando el establecimiento de una conexión WebSocket. Si el servidor acepta, se produce un “apretón de manos” y la conexión se actualiza a WebSocket.
- **Comunicación bidireccional:** Después del “apretón de manos”, ambas partes pueden enviar y recibir datos de manera bidireccional en tiempo real a través del mismo socket.

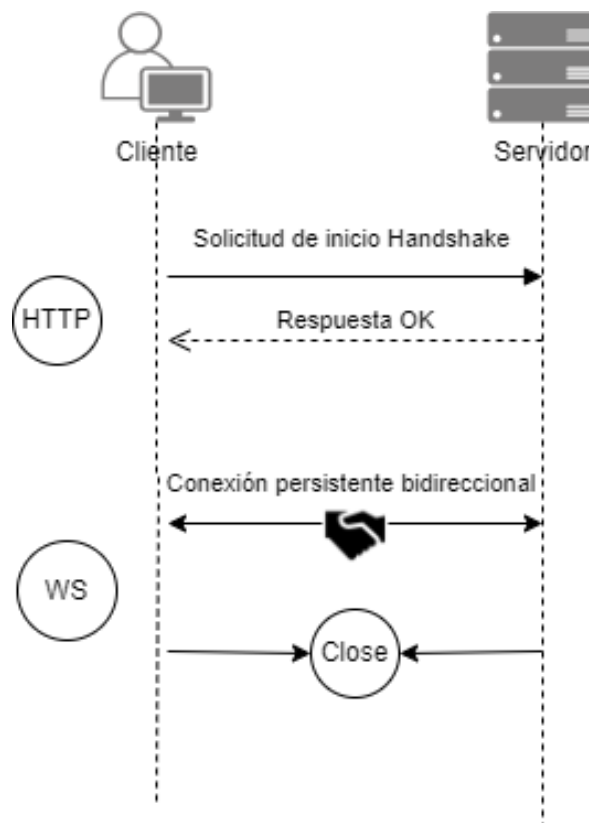


Figura 2.2: Web Sockets

2.6. Patrón de Diseño Pub/Sub

El patrón de diseño Publicador-Suscriptor, es una solución que permite a una aplicación comunicar eventos de manera asincrónica a múltiples consumidores interesados,

evitando la necesidad de establecer una conexión directa entre los emisores y receptores. La Figura 2.3 muestra una representación gráfica del patrón, un componente llamado “publicador” emite eventos a través de canales temáticos, y los “suscriptores” que están interesados en eventos específicos se registran en dichos canales.

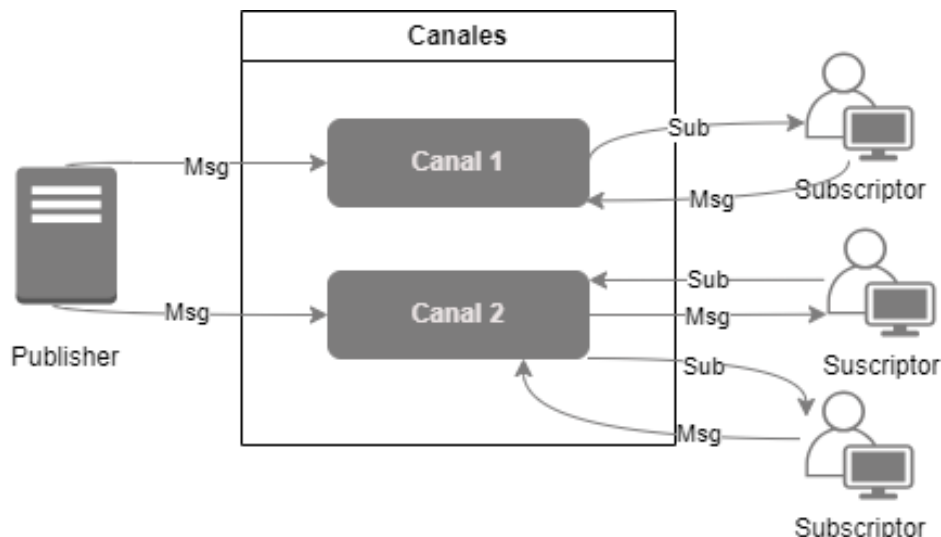


Figura 2.3: Publicador Suscriptor

En el contexto de este proyecto, se utiliza este patrón con el uso de web sockets en el sistema de notificaciones en tiempo real y actualizaciones de vistas.

2.7. Docker Swarm

Docker es una plataforma de código abierto diseñada para facilitar la creación, implementación y ejecución de aplicaciones en entornos aislados llamados “contenedores”. Estos contenedores son unidades ligeras y portátiles que contienen todo lo necesario para que una aplicación se ejecute, incluidas bibliotecas, dependencias y código, de manera eficiente y coherente en cualquier entorno que admita Docker.

Docker Swarm es una herramienta de orquestación integrada en Docker que facilita la administración y escalabilidad de aplicaciones distribuidas que se ejecutan en contenedores Docker. Swarm permite la creación y administración de clústeres de Docker, donde múltiples nodos pueden trabajar juntos para proporcionar una plataforma robusta y escalable para la implementación de aplicaciones en contenedores.

2.8. Normalización de la Base de Datos

La normalización de una base de datos es un proceso esencial en el diseño de sistemas de gestión de bases de datos. Cabe destacar que cada forma normal depende de que se cumpla la forma normal anterior. Esto asegura que el proceso de normalización sea progresivo y que cada nivel proporcione una mayor reducción de redundancia y

una mejora en la organización de los datos. A continuación, se describen las formas normales principales:

2.8.1. Primera Forma Normal (1FN)

La 1FN elimina los grupos repetidos de las tablas individuales. En otras palabras, cada celda de la tabla debe contener un único valor atómico, evitando así la repetición de grupos de datos.

2.8.2. Segunda Forma Normal (2FN)

La 2FN establece que, si un atributo no forma parte de la clave primaria, debe proporcionar un hecho que dependa de la clave completa. Esto elimina dependencias parciales y contribuye a una mejor organización de los datos.

2.8.3. Tercera Forma Normal (3FN)

La 3FN va un paso más allá y elimina los campos que no dependen directamente de la clave primaria. Esto ayuda a asegurar que cada campo en una tabla contribuya únicamente a la información específica de esa clave.

2.8.4. Forma Normal de Boyce-Codd (FNBC)

La Forma Normal de Boyce-Codd (FNBC) es una extensión de la 3FN. Si no existen claves candidatas compuestas, una tabla se considera en FNBC. Sin embargo, si hay claves candidatas compuestas con un elemento común, puede no estar en FNBC, a menos que cada determinante en las dependencias funcionales sea una clave candidata.

2.9. Propiedades ACID en Bases de Datos

En el contexto de bases de datos, las propiedades ACID son fundamentales para garantizar la integridad y confiabilidad de las transacciones.

- **Atomicidad (A):** Asegura que una transacción se realice de manera completa o no se realice en absoluto.
- **Consistencia (C):** Garantiza que la base de datos pase de un estado válido a otro válido después de que una transacción se haya completado.
- **Aislamiento (I):** Asegura que las transacciones en ejecución sean independientes entre sí, de modo que el resultado de una transacción no afecte el resultado de otras transacciones concurrentes.

- **Durabilidad (D):** Garantiza que una vez que una transacción se ha completado con éxito, sus cambios son permanentes y persisten incluso en caso de fallo del sistema.

3. Gestión de las Configuraciones

3.1. Introducción

En este capítulo se presenta el plan de gestión de configuraciones para el sistema de automatización y gestión del Patrocinio Jurídico de la UBA. El objetivo principal es dar a conocer las políticas, estrategias y métodos empleados para mantener la integridad del proyecto, mejorar su desarrollo y garantizar un producto de calidad.

3.2. Herramientas para la Administración de Configuraciones

En esta sección se describen las herramientas y procesos utilizados para gestionar las configuraciones del proyecto. La Tabla 3.1 resume las principales herramientas y sus propósitos.

Herramienta/Proceso	Propósito
Visual Studio Code	Editor de código fuente
GitHub	Control de versiones y gestión de cambios
Jira Software	Administración de tareas y defectos
GitHub Actions	Integración continua
Google Drawio	Creación de diagramas UML
Mockflow	Creación de prototipos GUI
Docker	Contenerización de servicios y aplicaciones
Docker Swarm	Orquestación de contenedores
Apps Script	Automatización y generación de extensiones para Google Forms
Overleaf	Documentación en Latex
Conda	Creación de entornos virtuales en desarrollo
Confluence	Documentación y registros internos

Cuadro 3.1: Herramientas para la administración de configuraciones

3.3. Herramienta de Control de Versiones

3.3.1. Dirección de Acceso

La herramienta de control de versiones utilizada en este proyecto es GitHub. El acceso al repositorio se realiza a través del siguiente enlace:

[Repositorio](#)

3.3.2. Plan del Esquema de Ramas

Se establece el siguiente plan para el esquema de ramas, como se muestra en la Figura 3.1:

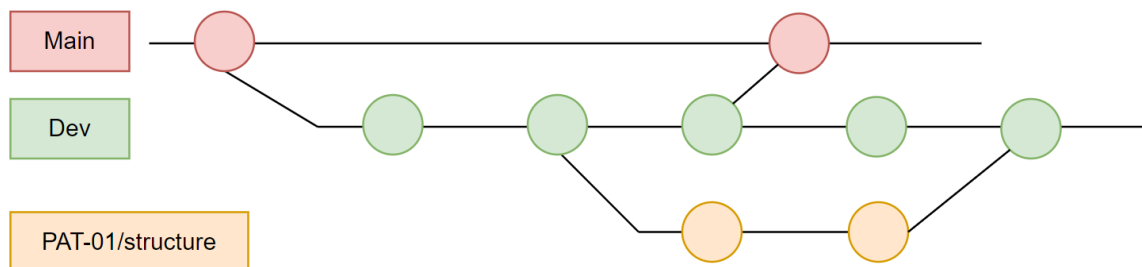


Figura 3.1: Ramas de Git

- **Principal (Main):** Rama principal del proyecto, refleja la versión estable y funcional del código.
- **Desarrollo (Dev):** Rama de desarrollo, donde se integran las nuevas características y mejoras.
- **Funcionalidades (PAT-***):** Ramas temporales para el desarrollo de características específicas, fusionadas con la rama de desarrollo al completarse. Esta rama consta del código la issue en Jira seguido de un slash '/' y una breve descripción en inglés. Ejemplo: *PAT-01/structure_directory*.

Las ramas Feature o funcionalidades utilizan la rama develop como rama primaria. Cuando una feature está terminada, se fusiona con la rama Dev. Las features no deben interactuar nunca directamente con Main.

3.3.3. Normas de Etiquetado y de Nombramiento de los Archivos

Las versiones se numeran con la siguiente nomenclatura **MAYOR.MENOR.PATCH** siguiendo el estándar de versionamiento semántico 2.0.0 (Sember) [6]:

- **PATCH:** Aumenta sólo cuando se corrigen errores o se realiza un refactor que no afecta al funcionamiento, es decir, no realizan cambios en el comportamiento.
- **MENOR:** Se incrementa cuando se añade una nueva funcionalidad compatible con la versión anterior, si algún método se marca como obsoleto debe aumentarse la versión menor.
- **MAYOR:** Se incrementa cuando se produce un cambio que es incompatible con alguna versión anterior, pueden incluir cambios menor y patch.

3.3.4. Políticas de Fusión de Archivos y Etiquetado en Conformidad con el Progreso de Calidad en los Entregables

A continuación, se detallan las políticas establecidas para la gestión de versiones y el control de calidad en el proceso de desarrollo de software:

- **Control de Versiones en Main:** Únicamente se permitirá la inclusión de versiones en producción en la rama principal (Main). Se incluirá código debidamente testeado, dejando evidencias de las pruebas realizadas en cada entrega.
- **Numeración de Versiones en Releases:** En la fase de release, se generará el tag correspondiente en la rama main, siguiendo la estructura estándar de versionamiento semántico mencionada en la sección 3.3.3.
- **Etiquetado de Versiones:** Cada versión debe ser etiquetada, proporcionando una referencia clara y accesible para su seguimiento. Este etiquetado permitirá una identificación rápida y precisa de los puntos de lanzamiento.
- **Integración con Pruebas:** Únicamente se autorizará la fusión de código que haya superado satisfactoriamente las pruebas correspondientes. Esta medida garantiza la estabilidad y calidad del código integrado en la rama principal.
- **Estándar de Commits Semánticos:** Cada commit que se envíe debe cumplir con el estándar de commits semánticos establecido. Este enfoque proporciona una estructura clara y consistente en los mensajes de commit, facilitando la comprensión del historial de cambios.

3.3.5. Convención de Commits y Merge Request

Para mantener un estándar, se hace uso de commits semánticos, basados en la Convención de Commits Convencionales 1.0.0 [1]. Estos siguen la siguiente estructura:

PREFIX(CODE): DESCRIPTION

Ejemplo para commits: feat(PAT-05): Add function to search for files

Ejemplo para merge request: feat[PAT-05]: Add Calendar

Puede ver un ejemplo en el siguiente link [Merge Request](#).

A continuación se listan los posibles prefijos:

- **feat**: Una nueva característica para el usuario.
- **fix**: Arregla un bug que afecta al usuario.
- **imp**: Cambios que mejoran el funcionalidades o rendimientos.
- **build**: Cambios en las tareas de despliegue o instalación.
- **docs**: Cambios en la documentación.
- **refactor**: Refactorización del código como cambios de nombre de variables o funciones.
- **test**: Añade tests o refactoriza uno existente.

3.4. Estructura de Directorios

El diseño del proyecto se organiza en tres repositorios principales alojados en GitHub, cada uno cumpliendo un rol específico en la implementación y gestión del sistema. A continuación, se detallan estos repositorios junto con sus respectivos enlaces:

- **proyecto-patrocinio** [[GitHub-unit](#)]: Este repositorio alberga tanto el núcleo del proyecto como su interfaz de usuario. Se estructura en carpetas específicas para el backend, frontend y archivos de script destinados a Google App Script.
- **cms-deploy** [[GitHub-deploy](#)]: Este repositorio se centra en la orquestación y configuración del despliegue de la plataforma mediante Docker Swarm. Contiene archivos de composición para Docker, plantillas y configuraciones necesarias para asegurar un despliegue efectivo.
- **cms-docs** [[GitHub-docs](#)]: Aquí se encuentra toda la documentación del proyecto. Este repositorio incluye información detallada sobre la arquitectura del sistema, sus componentes, guías de instalación y uso, entre otros documentos esenciales para el desarrollo y mantenimiento del proyecto.
- **cms-test** [[GitHub-test](#)]: En este repositorio se encuentran los test de sistema. Éste se utiliza en el proceso de verificación y validación del software para garantizar la calidad del mismo.

3.4.1. Repositorio *cms-deploy*

El repositorio *cms-deploy* se encarga específicamente de la configuración de despliegue. Incluye archivos de composición para Docker Swarm, plantillas y configuraciones necesarias para garantizar un despliegue eficiente y escalable (ver Figura 3.2).

En la raíz del directorio, destaca el archivo de composición para Docker Swarm, que define el stack de contenedores. Además, se encuentra la carpeta **resources**, que contiene archivos de configuración y variables de entorno para los contenedores. La subcarpeta **templates** dentro de **resources** se subdivide en dos directorios: **account** y **notifications**. El directorio **account** almacena plantillas HTML relacionadas con

los correos electrónicos enviados para el registro de cuentas y cambios de contraseñas. Por otro lado, en el directorio **notifications**, se encuentran los archivos HTML que se enviarán por correo electrónico para notificar las solicitudes de casos a las comisiones, así como las respuestas a estas solicitudes.

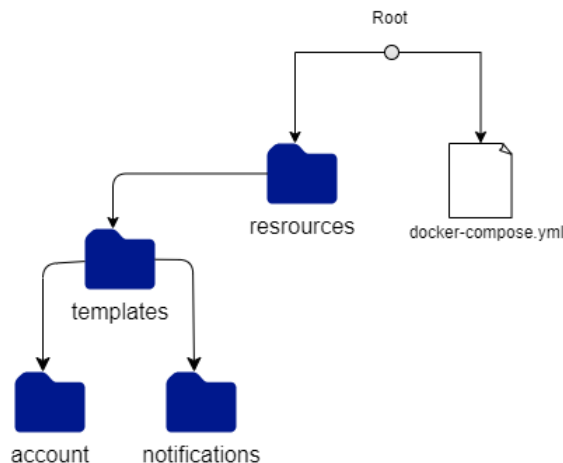


Figura 3.2: Estructura de directorios del repositorio *cms-deploy*.

3.4.2. Repositorio *cms-docs*

En el repositorio *cms-docs*, se concentra toda la documentación del proyecto. Desde detalles técnicos hasta guías de usuario, este repositorio actúa como una fuente centralizada de información para desarrolladores, administradores y usuarios finales.

3.4.3. Repositorio *proyecto-patrocinio*

El repositorio *proyecto-patrocinio* constituye el núcleo esencial del proyecto, integrando tanto el backend como el frontend, junto con los scripts para Google App Script.

A continuación, se presenta una representación visual de la estructura de directorios específica del repositorio *proyecto-patrocinio*, ofreciendo una visión detallada de la distribución de archivos y carpetas clave para comprender la organización del código y los recursos del proyecto (ver Figura 3.3).

Dentro del directorio **backend**, se encuentran dos carpetas principales: **.env**, que alberga todas las variables de entorno configurables, y **app**, donde se desarrolla la API REST en Django. Este último contiene toda la lógica de negocio de la plataforma.

En el directorio **frontend**, la carpeta **app** almacena la interfaz de usuario Reactiva, con una mínima lógica necesaria para obtener y visualizar datos.

La carpeta **scripts-forms** se subdivide en cinco directorios. **Child** contiene scripts y archivos necesarios para el formulario de registro de un nuevo hijo para un consultante existente. **Client** incluye scripts y archivos para el formulario de registro

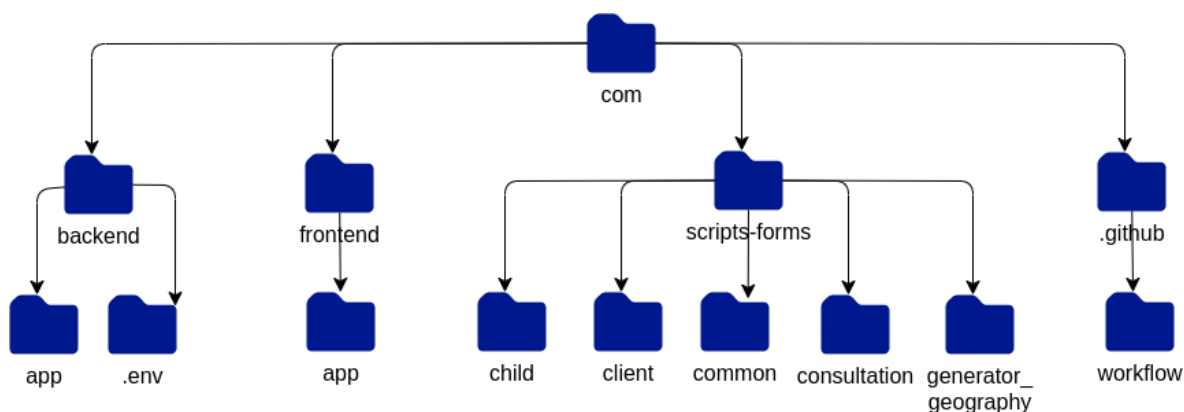


Figura 3.3: Estructura de directorios del repositorio *proyecto-patrocinio*.

de un nuevo consultante. **Consultation** alberga scripts y archivos para el formulario de creación de una nueva consulta a un consultante existente. La carpeta **common** contiene archivos compartidos por los formularios mencionados anteriormente. La carpeta **generator-geography** incluye la hoja de cálculo y scripts necesarios para generar nacionalidades, provincias y localidades en los cuestionarios de los formularios mencionados.

Dentro del directorio **.github**, la carpeta **workflows** contiene los pipelines para la integración continua del proyecto. Esta configuración garantiza una gestión eficiente del desarrollo y la implementación continua del proyecto.

3.4.4. Repositorio *cms-test*

En el repositorio *cms-test*, se encuentran todos los casos de prueba automatizados realizados con Robot Framework y Selenium. La estructura del directorio de pruebas se puede visualizar en la Figura 3.4.

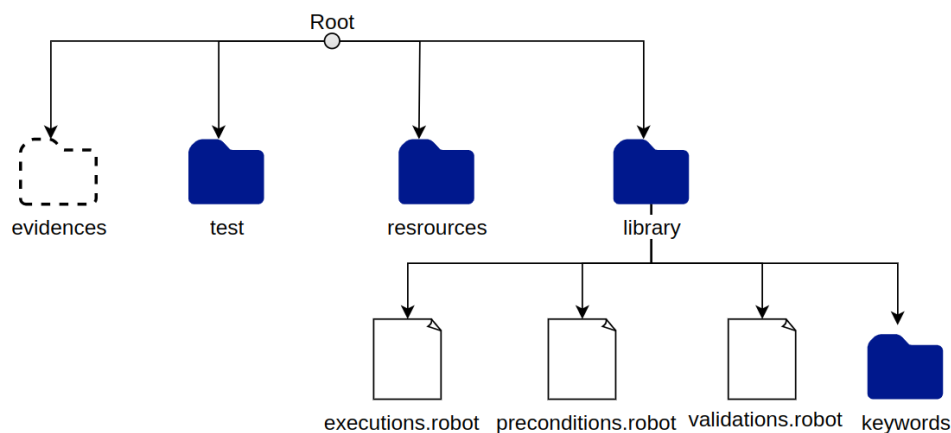


Figura 3.4: Estructura del directorio de pruebas en *cms-test*.

Dentro del directorio *test*, se ubican los casos de prueba específicos. En el directorio *resources* se almacenan los archivos utilizados en las pruebas. En el directorio *library*, se encuentra la implementación de las *keywords*, divididas en tres archivos: *preconditions.robot*, *executions.robot* y *validations.robot*. Cada archivo contiene las *keywords* relacionadas con las precondiciones (Given), acciones (When) y validaciones (Then) respectivamente, siguiendo la sintaxis de Gherkin [10]. Por otro lado, en la carpeta *keywords*, se agrupan las *keywords* con la lógica para realizar diversas tareas.

Por último, la carpeta *evidences* se genera automáticamente después de ejecutar los tests, en la cual se recopilan las evidencias generadas para cada caso de prueba.

3.5. Herramientas de Gestión de Tareas y Defectos

Para la supervisión y gestión eficiente de tareas y defectos, se empleó JIRA ([Link Jira](#)). Esta plataforma proporciona tanto el backlog como el board del sprint activo, ofreciendo una vista integral del progreso del proyecto. JIRA no solo facilita la asignación y seguimiento de tareas, sino que también ofrece una interfaz interactiva que simplifica la colaboración y la toma de decisiones.

3.5.1. Confluence

La integración con Confluence añade un valor adicional al proceso de gestión. Con la capacidad de agregar páginas Markdown directamente al proyecto, se promueve la documentación detallada y la contextualización de las issues. Esto no solo mejora la comprensión de los problemas abordados, sino que también facilita la colaboración al proporcionar un contexto más amplio para la toma de decisiones.

3.5.2. Issues

Cada Issue consta de:

- Una descripción detallada.
- Un etiquetado que indica el tipo de problema (bug, feat, task).
- Un enlace directo a Commits y Merge Request/rama asociados. Para establecer conexiones efectivas entre la Issue, los commits y los Merge Requests o ramas, es esencial agregar el código de la Issue en cada uno de ellos.
- Una asignación a persona responsable de hacer avanzar el problema.
- Comentarios de cualquier persona con acceso a Jira.
- Épica a la cual pertenece.

Se ilustra un ejemplo en el siguiente enlace: [Link a la ISSUE](#).

3.5.3. Épicas

En el ámbito de la gestión de proyectos, una épica se refiere a una categoría o tema amplio que engloba un conjunto de tareas o historias de usuario relacionadas. En el contexto de este proyecto, se han identificado diversas épicas que se especifican a continuación.

Calendario

Se centra en la integración de calendarios, programación de eventos y cualquier otra característica asociada al seguimiento temporal de los casos.

Documentación

Abarca actividades destinadas a mejorar la comprensión y el mantenimiento del proyecto. Incluye la configuración de logs para facilitar la monitorización y depuración del sistema, la actualización regular de los archivos README para proporcionar información actualizada y relevante, el refinamiento y ampliación de las docstrings en el código fuente para mejorar la comprensión de las funciones y métodos, la elaboración de documentación técnica detallada que explique la arquitectura, diseño y decisiones técnicas del proyecto, la creación y actualización de manuales de usuario que guíen a los usuarios finales sobre el uso efectivo del sistema, y la generación de informes detallados que aborden aspectos específicos del proyecto.

Testing

Se enfoca en la implementación y mejora de pruebas para garantizar la calidad del software. En este proyecto, la atención se centra principalmente en dos aspectos: pruebas unitarias de backend y frontend, y pruebas de sistema.

Infraestructura

Se enfoca en la gestión y mejora de los recursos tecnológicos subyacentes del proyecto. Las tareas dentro de esta épica se dividen en dos aspectos clave:

Desarrollo de Pipelines: Se realiza el diseño, desarrollo y optimización de pipelines que automatizan procesos relacionados con la implementación de integración y entrega continua del proyecto.

Despliegue en Docker Swarm: Se establecen y mejoran los procedimientos de despliegue de la aplicación en el entorno Docker Swarm.

Notificaciones

Aborda la implementación de sistemas de notificación eficientes. Esto incluye alertas y correos electrónicos que informe sobre eventos relevantes en el sistema.

Comentarios

Se dedica a la implementación de la funcionalidad de comentarios dentro de las consultas, incluyendo la carga de documentos.

Panel de Control

Se centra en la creación de las páginas de panel de control, el cual consta de tablas para visualizar los datos y descargarlos.

Integración con Google Forms

Busca facilitar la conexión y utilización de formularios de Google dentro del proyecto. Esto implica la automatización de la recopilación de datos y la integración de formularios en procesos específicos.

Administración

Se centra en la implementación de la administración básica como el registro de usuarios, gestión de permisos y la carga de datos iniciales al sistema.

Consultorio

Abarca las tareas vinculadas a la creación de un tablero de consultoría. Este tablero tiene como objetivo gestionar los casos judiciales y facilitar el proceso de envío de casos a las comisiones correspondientes para su patrocinamiento. Se centra en la implementación de funcionalidades específicas que agilizan y optimizan la administración de casos, desde su registro inicial hasta su asignación a las comisiones pertinentes.

Tablero

Engloba las tareas relacionadas con la creación y desarrollo de un tablero destinado a la comisión. Este tablero tiene como objetivo fundamental la gestión integral de casos, incluyendo su ingreso, su seguimiento, y la organización eficiente de cada uno de ellos dentro del sistema. Se enfoca en la implementación de funcionalidades clave que permiten a la comisión llevar a cabo sus responsabilidades de manera efectiva, proporcionando una interfaz intuitiva y completa para la administración de los casos asignados.

Seguridad y Privacidad

Se dedica a fortalecer la seguridad y privacidad del sistema. Incluye la implementación de medidas de seguridad, la gestión de accesos y la protección de datos.

3.6. Herramienta de Integración Continua

GitHub Actions es una poderosa plataforma de Integración Continua y Despliegue Continuo (CI/CD) que posibilita la automatización de pipelines para la construcción, prueba y despliegue de proyectos. A través de GitHub Actions, se pueden crear flujos de trabajo personalizables que se encargan de compilar y probar cada pull request enviado al repositorio, así como desplegar las contribuciones fusionadas en la rama principal hacia los entornos de producción.

Un **Workflow** o flujo de trabajo es un proceso automatizado configurable que ejecuta uno o más trabajos (jobs). Estos flujos de trabajo se definen mediante un archivo de configuración YAML que se incorpora al repositorio. La ejecución del flujo de trabajo puede desencadenarse por eventos específicos dentro del repositorio, como cambios en el código, pull requests, o de manera manual.

En el contexto de este proyecto, se han empleado dos flujos de trabajo, uno dedicado al frontend y otro al backend, alojados en el repositorio *proyecto-patrocinio* (ver 3.4).

Para ambos pipelines, se establecen disparadores que inician el flujo de trabajo compuesto por dos etapas principales: **ci** y **cd** (consulte la Figura 3.5).

- **triggers:** La ejecución de los pipelines se configura en respuesta a eventos específicos. Estos eventos incluyen el lanzamiento de una nueva versión (*release*) o la realización de un *push* a las ramas **main** o **dev**. En el caso de las ramas **dev** o **main**, los pasos del flujo presentan una ligera variación para adecuarse a la gestión de versiones y etiquetas de las imágenes destinadas a Docker Hub.

- **ci:** Este trabajo realiza un conjunto de pasos esenciales, incluyendo la verificación de la calidad del código, la configuración del entorno de la API, la instalación de dependencias, y la ejecución de pruebas, entre otros. Garantiza que el código enviado cumpla con los estándares definidos antes de proceder con la siguiente etapa.

- **cd:** La etapa de entrega continua se activa tras el éxito de la etapa de integración continua (**ci**). En esta fase, se realiza la construcción y la publicación de la imagen de Docker en Docker Hub. La imagen se etiqueta con la versión actual y la etiqueta "latest". Este proceso permite que la imagen esté lista para ser implementada en entornos de producción.

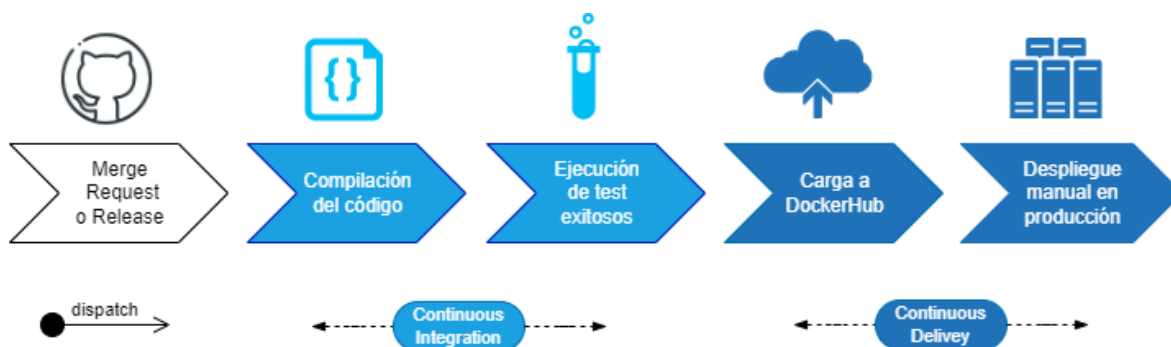


Figura 3.5: CI/CD

Consulte la sección [GitHub Actions](#) del repositorio para ver los detalles de los reportes.

3.7. Formato de Entrega

La entrega del proyecto se realizará vía email adjuntando los siguientes archivos:

- Un archivo comprimido denominado `cms-deploy.zip`, que abarca los componentes esenciales para el despliegue del proyecto.
- La documentación en formato PDF.
- Un archivo comprimido `proyecto-patrocinio.zip`, que contiene la unidad del proyecto.

Además, se proporcionarán los nombres de las imágenes correspondientes al backend y frontend.

3.8. Change Control Board - CCB

El Comité de Control de Cambios (CCB) se configura como un grupo de partes interesadas (stakeholders) encargadas de supervisar y tomar decisiones relacionadas con propuestas de cambios, requerimientos, hotfixes y otros aspectos inherentes al desarrollo de software. Su objetivo principal es garantizar la coherencia y alineación de estas modificaciones con los objetivos y requisitos establecidos, asegurando la implementación de modificaciones de manera estructurada y controlada.

3.8.1. Metodología Ágile

La elección del marco de trabajo SCRUM se fundamentó en la idoneidad de su enfoque ágil, especialmente en proyectos con requisitos no completamente definidos.

SCRUM ofrece una estructura que favorece el progreso incremental a través de sprints y se ajusta bien a la gestión de tareas en entornos dinámicos.

La implementación de SCRUM en este proyecto requirió adaptaciones específicas debido al contexto particular de este proyecto. Se realizaron sprints con una duración flexible de 2 a 4 semanas, ajustando el tiempo según la disponibilidad durante el año lectivo. Adicionalmente, se programaron reuniones de revisión y planificación al final y al inicio de cada sprint, respectivamente, para evaluar avances y definir las metas para el próximo ciclo.

Además, se optó por simplificar los roles de Product Owner, Scrum Master y Developer considerando las limitaciones de recursos de personal.

3.8.2. Miembros y Funciones

A continuación se tabulan los roles de los miembros del proyecto (ver [3.2](#)).

Además, se realiza un reconocimiento especial a Alberto Germán Sotelo y Agustina Cuello por su colaboración como becarios del laboratorio IALAB durante la fase inicial del proyecto realizando tareas de desarrollo backend.

Rol	Función	Responsable
Directores del proyecto	Revisión del proyecto. Aprobación de decisiones importantes. Gestiona el buen funcionamiento del proyecto, quien controla y administra los recursos (tanto personales como económicos) con el fin de cumplir el plan y el objetivo definido. Se encargan de que todo funcione según lo establecido, resolver desviaciones en el plan, y hacer que los diferentes equipos del proyecto se sincronicen y trabajen juntos.	Director, Danilo Paez y Co-directora, Laura Díaz
Product Owner	Deben conocer perfectamente el producto para garantizar que el equipo de desarrollo cumpla con las expectativas y necesidades.	Carina Papini y Melisa Raban
Asesor/Consultor	Proporciona asesoramiento y orientación técnica. Está disponible para consultas y resolución de dudas relacionadas con aspectos específicos del proyecto.	Daniel Britos
Desarrollo Full Stack e Ingeniería	Detección y documentación de requerimientos. Diseño y planificación del software en sprints. Encargado de tomar el control de la herramienta de gestión de defecto, documentando específicamente sobre cada problema en el que se trabaja. Se encargará de crear un incremento terminado a partir de los elementos del Product Backlog seleccionados por sprint.	Julieta Abigail Prieto

Cuadro 3.2: Tablero de Miembros y Funciones

4. Requerimientos y Especificaciones

4.1. Introducción

En este capítulo se detalla en su totalidad el producto de la plataforma de gestión y automatización para el proceso de selección de los casos del Patrocinio Jurídico Gratuito de la UBA. En este se describen sus funcionalidades, especificaciones y limitaciones en cuanto a su utilización.

4.2. Alcance del Producto

Este producto está orientado a profesores y tomadores de casos del Patrocinio Jurídico Gratuito, con el fin de facilitar y mejorar la eficiencia del sistema de solicitud y control histórico de los casos, asegurando una experiencia intuitiva en su utilización.

El alcance de esta plataforma abarca el proceso desde la recepción de datos a través de Google Forms, la integración de la información en el sistema, la gestión de los casos en la asignación a comisiones y, finalmente, el seguimiento y gestión de los casos por parte de los profesores de la comisión.

4.3. Referencias

A continuación, se presenta una lista detallada de las tecnologías utilizadas en el desarrollo del proyecto, acompañadas de enlaces a sus respectivas documentaciones. Estos recursos fueron seleccionados estratégicamente para facilitar el desarrollo, la implementación y el mantenimiento del sistema.

Recurso	Sitio Web
Documentación de Python 3.	Python3
Documentación de Django 4.1.	Django
Documentación de Django Rest Framework, una extensión de Django para construir APIs.	Django-Rest
Documentación de PostgreSQL para la base de datos.	PostgreSQL
Documentación Django-docutils para la automatización de la documentación de la API.	Django-docutils
Documentación del lenguaje Gherkin para listar los requerimientos.	Gherkin
Documentación de Nginx para un servidor web y/o proxy inverso.	Nginx
Documentación de Gunicorn, un servidor HTTP Python WSGI para UNIX.	Gunicorn
Documentación de React, una biblioteca de JavaScript para construir interfaces Reactivas de usuario.	React
Documentación de Material-UI, una biblioteca de componentes de interfaz de usuario para React.	MUI
Documentación de Beautiful Drag and Drop para implementar funcionalidades de arrastrar y soltar.	React-Beautiful-DND
Documentación de Django Channels para el manejo de conexiones en tiempo real a Django.	Channels
Documentación de Daphne, un servidor ASGI (Asynchronous Server Gateway Interface) para aplicaciones asincrónicas con Django.	Daphne
Documentación de Docker.	Docker
Documentación de Docker Swarm para la orquestación de contenedores.	Docker-Swarm
Documentación de Robot Framework para la automatización de test.	Robot-Framework
Documentación de Selenium para la automatización de tareas desde el browser.	Selenium

Cuadro 4.1: Recursos y Enlaces de Documentación

4.4. Descripción General

4.4.1. Perspectiva del Producto

El producto surge para abordar una necesidad administrativa identificada en la organización del Patrocinio Jurídico de la UBA. Actualmente, los formularios de los solicitantes se envían automáticamente por Google Forms mediante correo electrónico y posteriormente, se gestionan de manera manual utilizando hojas de cálculo, Excel u otras herramientas de pago, sin contar con un sistema centralizado. Esta metodología presenta desafíos de integración y no se ajusta completamente a los requisitos específicos de la organización. La solución propuesta busca mejorar la organización y gestión de casos mediante un software que se integre de manera eficiente con Google Forms.

El sistema se integrará sin inconvenientes con Google Forms para facilitar la captura de datos relacionados con casos judiciales y nuevos consultantes. Este enfoque optimizará el proceso de solicitud, selección y gestión de casos, brindando a los usuarios finales una experiencia más sencilla y ágil. La interfaz de usuario, diseñada intuitiva y amigablemente, seguirá las mejores prácticas de diseño centrado en el usuario, asegurando eficiencia y transparencia en la experiencia del usuario.

4.4.2. Funciones del Producto

El sistema desarrollado proporciona un conjunto integral de funciones diseñadas para satisfacer las necesidades específicas de la gestión de casos judiciales y su asignación a las comisiones. Integra de manera fluida con Google Forms, permitiendo la captura eficiente de datos relacionados con casos judiciales y nuevos consultantes. Facilita la selección y asignación de casos a comisiones. La interfaz de usuario intuitiva permite el seguimiento detallado de cada caso a lo largo de su ciclo de vida, facilitando una gestión efectiva y transparente por parte de los profesores de las comisiones. Incluye un sistema de notificaciones vía email y alertas para anunciar ciertos eventos. La interfaz de usuario, es intuitiva, fácil de navegar y brinda una experiencia eficiente, contribuyendo a mejorar la eficiencia, la transparencia y la efectividad en la gestión de casos y la selección de comisiones.

4.4.3. Tipos de Usuarios y Características

El sistema contempla varios tipos de usuarios, cada uno con funciones y características específicas para satisfacer sus necesidades dentro del proceso. Los roles principales incluyen:

Administradores del Sistema: Tienen acceso completo al sistema y la capacidad de gestionar usuarios y acceder a todas las funcionalidades. Su función principal es garantizar el correcto funcionamiento y la configuración adecuada del sistema.

Profesores de Comisión: Estos usuarios son responsables de revisar, evaluar y gestionar los casos asignados a su comisión. Pueden acceder a la información deta-

llada de cada caso, realizar comentarios, asignar tareas y seguir el progreso de manera integral.

Solicitantes: Son los usuarios encargados de presentar casos judiciales mediante Google Forms. Aunque no tienen acceso directo a la plataforma, desempeñan un papel crucial al ingresar casos y nuevos consultantes a través de Google Forms, contribuyendo así al flujo eficiente de datos en el sistema.

Administradores de Casos: Más conocidos como **Tomadores de Caso**, este rol se ocupa de la gestión específica de los casos, desde su recepción hasta su asignación a una comisión. Pueden revisar la información proporcionada por los solicitantes y asignar casos a las comisiones correspondientes.

4.4.4. Entorno Operativo

La plataforma estará diseñada para operar en un entorno que cumpla con los siguientes requisitos:

Requisitos de Servidor

La aplicación está diseñada para ser accesible desde cualquier dispositivo con capacidad para ejecutar contenedores Docker, lo que incluye sistemas operativos como Linux, Windows y macOS. Además, gracias a la naturaleza de Docker, la aplicación es altamente compatible con entornos en la nube, lo que significa que puede ejecutarse en servicios cloud. La orquestación de contenedores también se simplifica mediante tecnologías como Kubernetes; por ejemplo, puede implementarse y gestionarse en servicios como Amazon EKS en AWS.

Se recomienda empezar al menos con:

- 100 GB de almacenamiento libre
- 6 GB de RAM
- 2 núcleo de CPU

Estos requisitos proporcionarán una base para el despliegue inicial de la aplicación, asegurando un funcionamiento estable y eficaz. Se debe tener en cuenta que estos son requisitos mínimos y se puede considerar la posibilidad de aumentar estos recursos en función de la carga de trabajo y el crecimiento futuro de la aplicación.

Navegadores Soportados

- Google Chrome. Se probó en la versión Versión 120.0.6099.130.
- Mozilla Firefox. Se probó en la versión 121.0.

Conectividad

Se requiere una conexión a Internet estable para el correcto funcionamiento de la integración con Google Forms y el acceso a la plataforma. La velocidad de la conexión afectará directamente la eficiencia en la carga y manejo de datos.

Dispositivos Compatibles

La plataforma está diseñada para ser accesible desde dispositivos con pantallas de tamaño mediano a grande, como computadoras de escritorio, laptops y tabletas. El acceso desde dispositivos móviles puede ser posible, pero la experiencia de usuario puede variar según el tamaño de la pantalla.

4.5. Restricciones de Diseño e Implementación

- **Limitaciones a nivel de Interfaz:** Se busca una interfaz gráfica simple e intuitiva que sea semejante a la interfaz de Trello.
- **Limitaciones presupuestarias:** Debido a que el organismo solicitante del producto no tiene presupuesto para la creación y mantenimiento del sistema. Este debe tener un costo bajo o nulo.
- **Seguridad y Privacidad** La plataforma debe seguir las mejores prácticas de seguridad para proteger la información confidencial de los casos judiciales y los datos de los usuarios. Se deben implementar mecanismos de autenticación y autorización, y se debe garantizar el cifrado de la comunicación entre el cliente y el servidor.

4.6. Modelo Conceptual de Datos - DER

En la construcción del Modelo Conceptual de Datos, se tomó la decisión de utilizar PostgreSQL como motor de búsqueda para la base de datos, basándonos en su destacada capacidad para respaldar las propiedades fundamentales del modelo ACID (Atomicidad, Consistencia, Aislamiento y Durabilidad). Estas características son esenciales para garantizar la integridad y confiabilidad de las transacciones en la base de datos.

Posteriormente, al interpretar la matriz de relaciones, se simplificó la visualización al omitir algunas tablas intermedias, para facilitar la comprensión del esquema general de la base de datos de manera más accesible.

Los diagramas subsiguientes se derivan de un modelo de base de datos que ha sido procesado y refinado hasta alcanzar la tercera forma normal, una práctica común con el propósito de optimizar la organización y estructura de los datos, evitando redundancias y asegurando consistencia.

Adicionalmente, como estrategia para simplificar el diseño y reducir la complejidad asociada al manejo de grandes volúmenes de datos, se ha decidido tratar las calles como cadenas de texto independientes de las localidades. Esta decisión se basa en la idea de no vincularlas geográficamente, evitando así la carga adicional que implicaría gestionar todas las calles de Argentina, la cual podría resultar significativa en términos de volumen de datos y complejidad operativa.

Consulte en el anexo, los Diagramas B.1, B.2 y B.3 para visualizar las matrices de relaciones de entidades.

A continuación se presenta el diagrama de Entidad-Relación (Figura 4.1). En celeste se resalta la relación muchos a muchos, la cual ha sido reemplazada por una tabla intermedia denominada "board-user". Y en un azul mas oscura se representa la herencia de Person con User y Client.

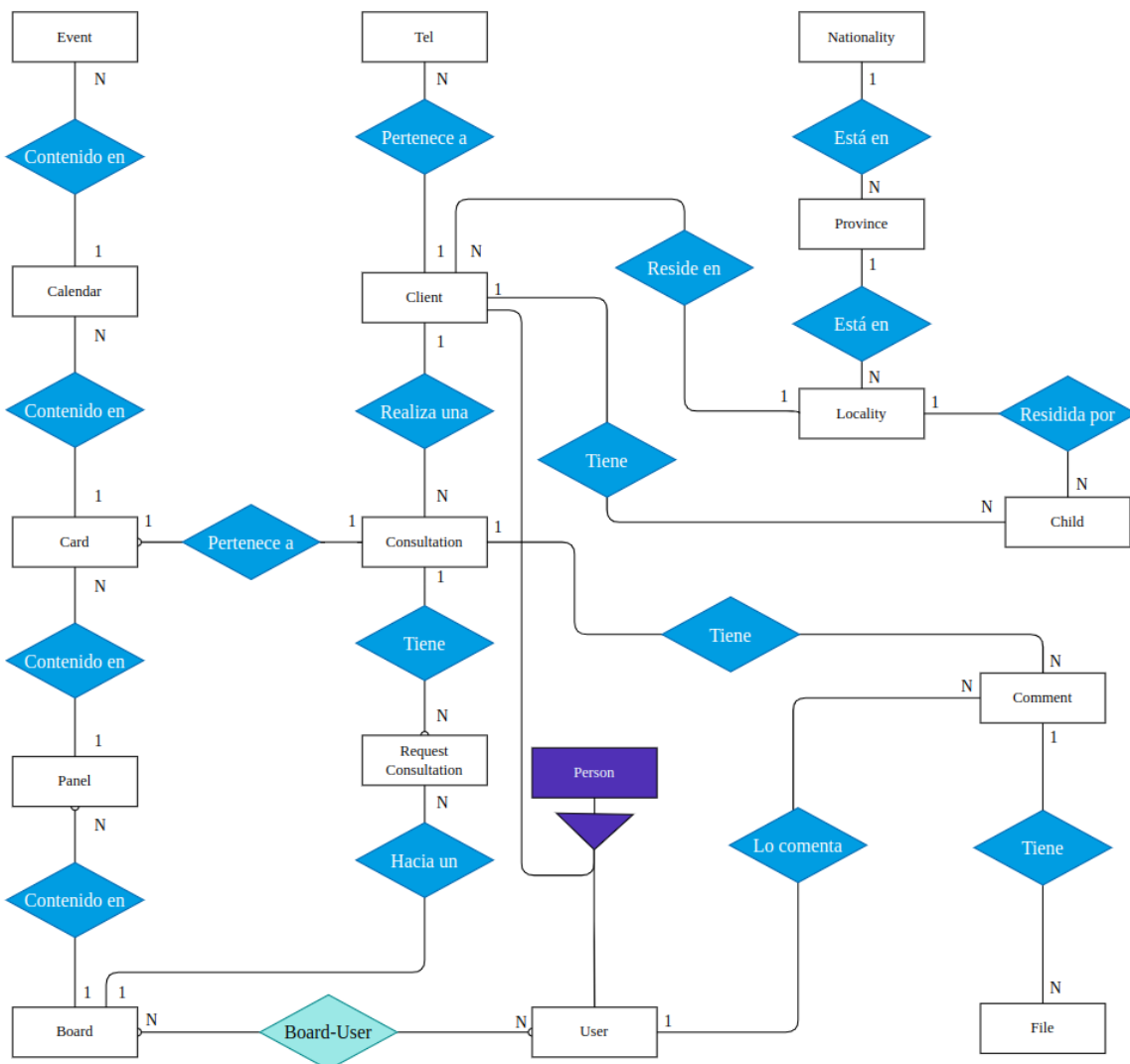


Figura 4.1: Diagrama Entidad Relación

La implementación del modelo se lleva a cabo en Django. Sin embargo, para simplificar la explicación, se han omitido otras entidades relacionadas al usuario propias de Django, como tokens, grupos, etc. A continuación, en la figura 4.2 se presenta el resultado visualizado mediante DBeaver, utilizado como gestor de la base de datos.

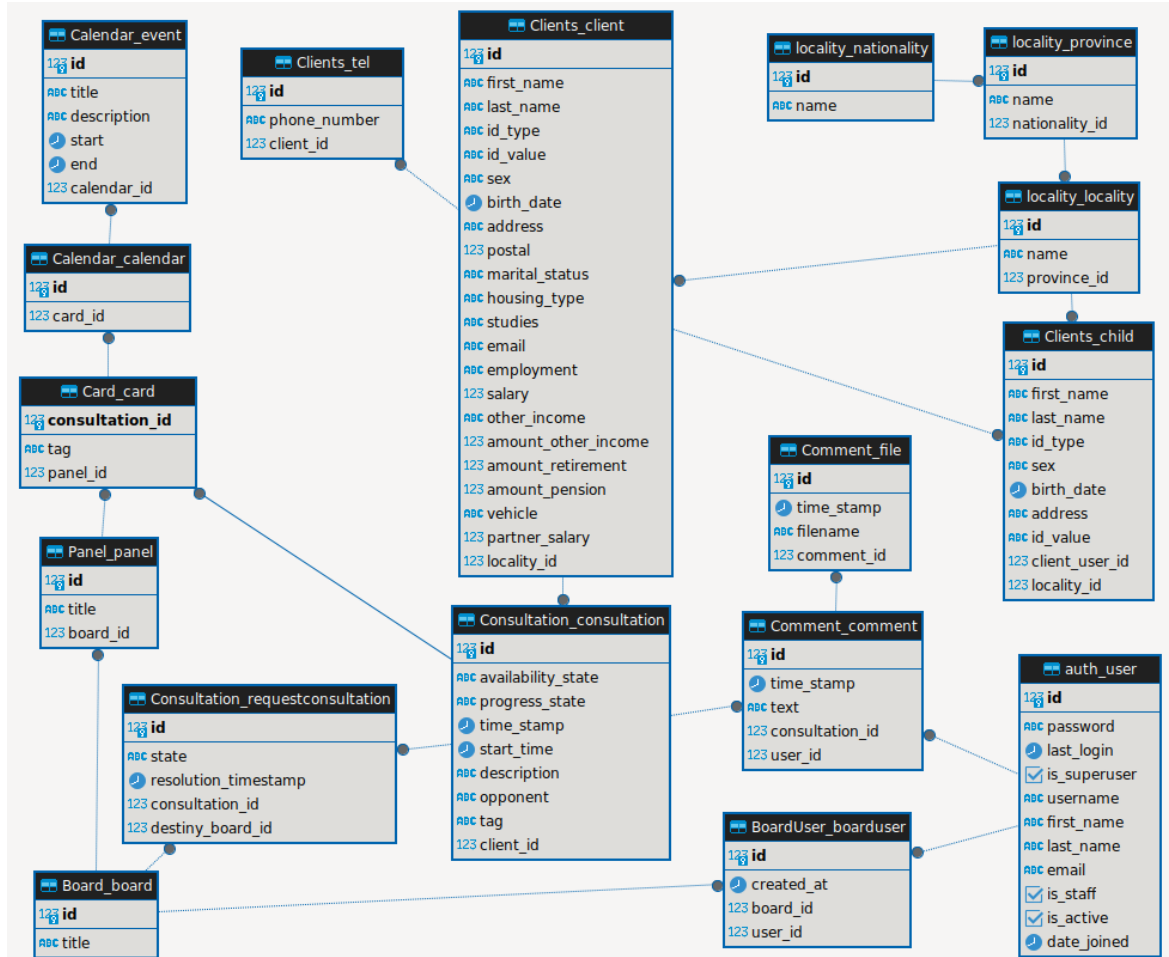


Figura 4.2: Implementación del Modelo Entidad Relación

Para obtener información detallada sobre la base de datos, consulte el anexo A, donde se encuentra el diccionario de la base de datos con la definición de las tablas y sus atributos.

4.7. Requerimientos de Interfaces Externas

En esta sección, se presenta los requerimientos de de Interfaces, que sirve como un elemento para la visualización y estructuración de la interfaz de usuario. A continuación, se exhiben los mockup representativos del diseño propuesto para la interfaz gráfica de usuario (GUI).

4.7.1. Consultoría

La Figura 4.3 exhibe un mockup que encapsula la apariencia y disposición previstas de la interfaz de usuario, específicamente en el contexto de la funcionalidad de consultoría.

La similitud con Trello, fue determinada en la etapa de gobernanza de datos y reingeniería de procesos de IALAB, donde participó el equipo del Patrocinio Jurídico Gratuito de la Facultad de Derecho.

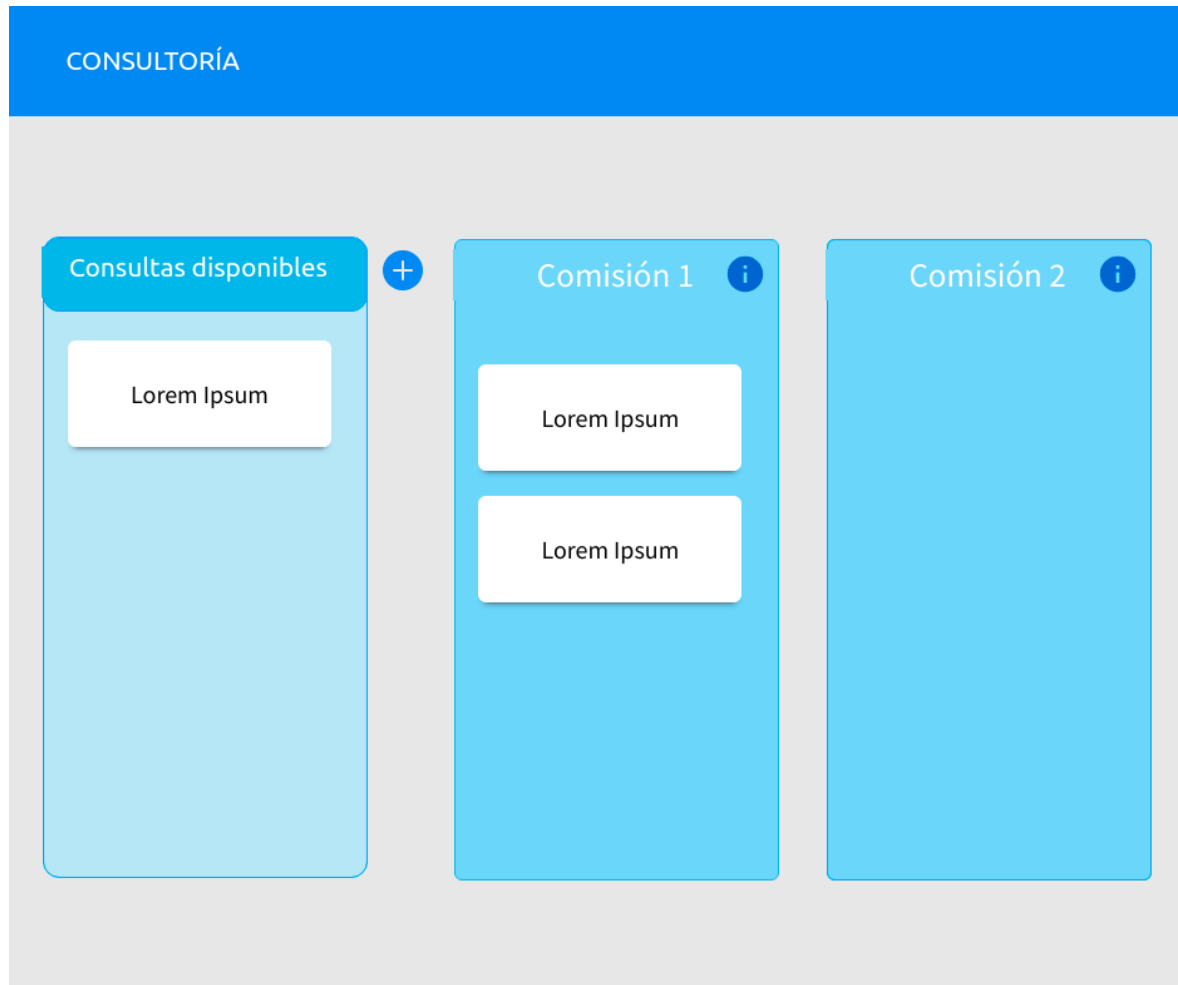


Figura 4.3: Mockup de la Interfaz Gráfica de Usuario (GUI) para la Página Consultoría

A la izquierda, se ubica el panel de casos sin asignar, mientras que a la derecha se presentan los paneles destinados para enviar las solicitudes de asignación del caso a las diversas comisiones (tableros). Cada tarjeta en estos tableros representa una consulta específica.

Adicionalmente, se incorporan dos elementos clave para la interacción:

- El botón “+” facilita la creación de nuevas consultas.

- El botón de información proporciona detalles específicos sobre la comisión correspondiente.

4.7.2. Tablero de Trabajo para Cada Comisión

El diseño del tablero de trabajo adopta una lógica similar a la implementada en la funcionalidad de consultoría, manteniendo paneles organizativos que permiten la manipulación intuitiva de tarjetas mediante el sistema de arrastre y soltar.

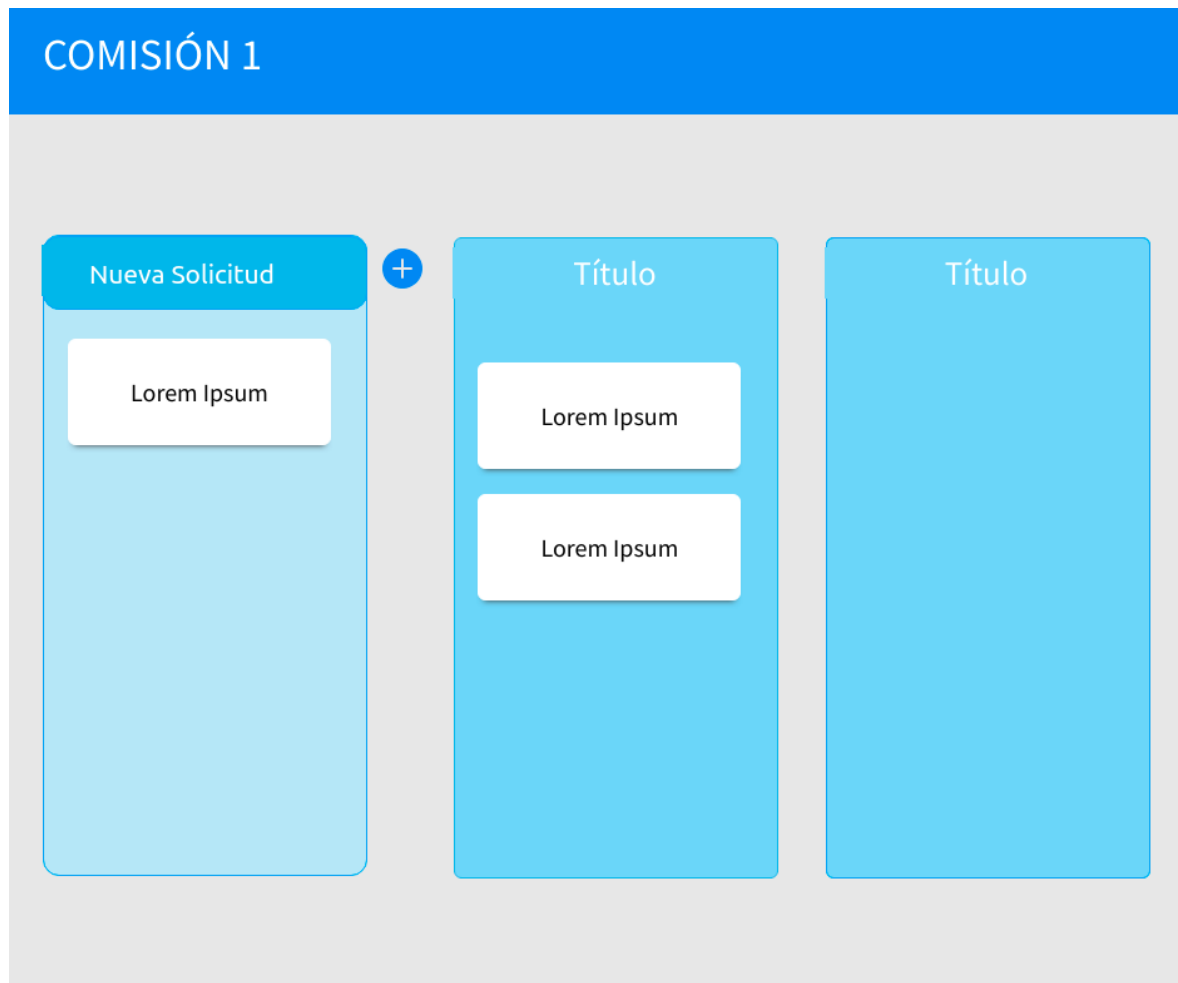


Figura 4.4: Mockup de la Interfaz Gráfica de Usuario (GUI) para el Tablero de Comisión

En el lado izquierdo de la de la figura (4.4), se sitúa el panel de entrada, destinado a recibir las solicitudes entrantes. A la derecha, se encuentran paneles específicos de la comisión, ajustados según las necesidades particulares. La funcionalidad del botón "+" simplifica la creación de paneles adicionales. Por ejemplo, se podrían generar paneles para representar el estado de la consulta o agruparlos según profesor.

Finalmente, las tarjetas blancas dentro de los paneles representan cada consulta con una etiqueta referente.

4.7.3. Inicio de Sesión

Para garantizar la seguridad y autenticación de los usuarios, se ha implementado se tiene como requisito una interfaz de inicio de sesión al sistema. La Figura 4.5 presenta el diseño visual asociado con la página de inicio de sesión.

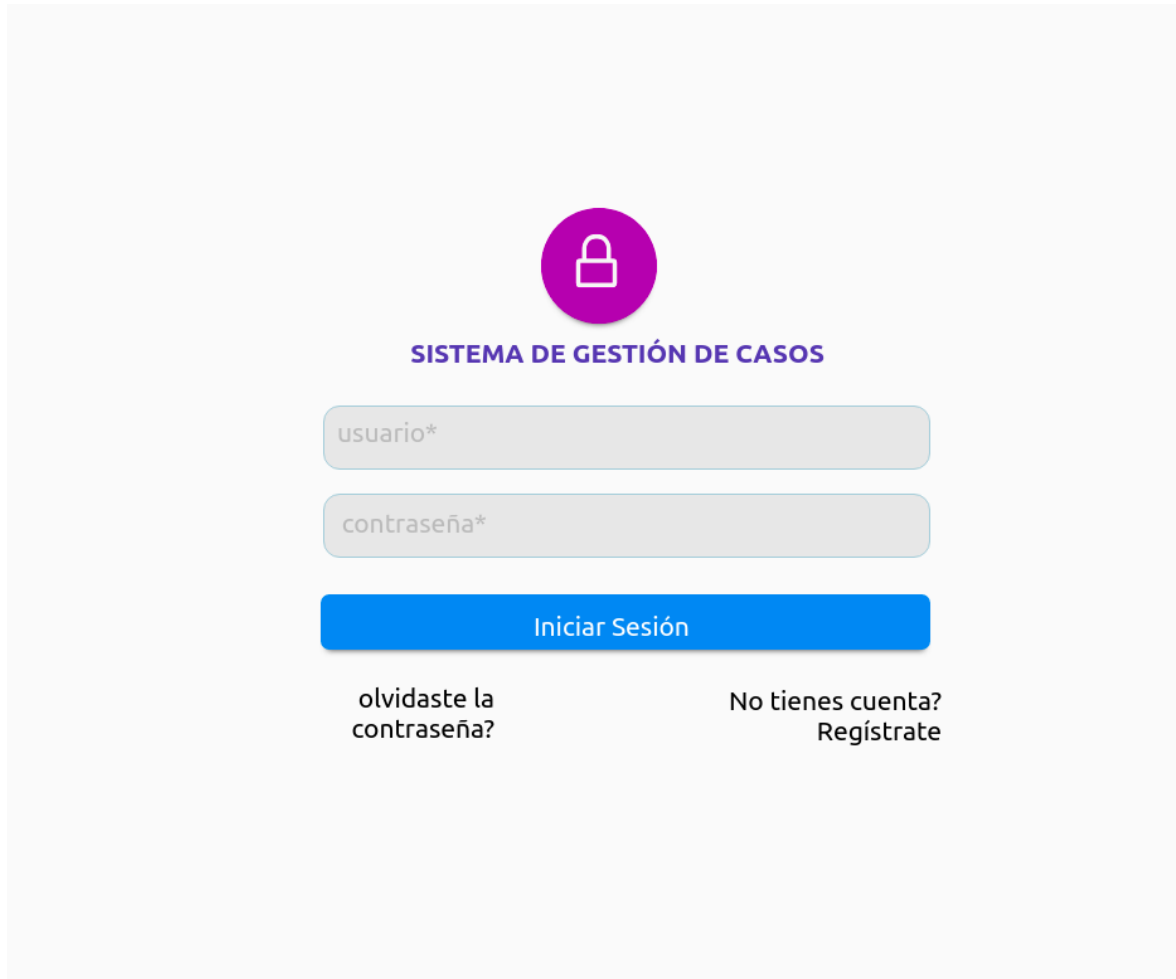
The image shows a login interface for a system titled "SISTEMA DE GESTIÓN DE CASOS". At the top center is a purple circular icon containing a white padlock. Below the icon, the title "SISTEMA DE GESTIÓN DE CASOS" is displayed in a bold, purple, sans-serif font. Underneath the title are two light gray input fields with rounded corners. The first field is labeled "usuario*" in a light gray font, and the second field is labeled "contraseña*" in a light gray font. Below these fields is a solid blue button with the text "Iniciar Sesión" in white. At the bottom of the interface, there are two links: "olvidaste la contraseña?" on the left and "No tienes cuenta? Regístrate" on the right, both in a standard black font.

Figura 4.5: Mockup de la Interfaz de Inicio de Sesión

Como se evidencia en esta interfaz, el requisito de inicio de sesión solicita la introducción de un usuario y contraseña para acceder al sistema. Además, se ha incorporado la opción de recuperar la contraseña en caso de olvido, así como la capacidad de crear una nueva sesión para los usuarios que aún no cuentan con una cuenta registrada.

4.7.4. Información de la Consulta

Un requisito fundamental del sistema es la capacidad de visualizar la información detallada de cada consulta. Esta sección abarca diversos aspectos esenciales, incluyendo datos específicos de la consulta, una sección dedicada a comentarios, y una interfaz de calendario para gestionar fechas clave.

La Figura 4.6 presenta un mockup que ilustra la propuesta visual para la interfaz de información de consulta.



Figura 4.6: Mockup de la Interfaz de Información de Consulta

Esta interfaz permite interactuar a través de la sección de comentarios, permitiendo a los usuarios intercambiar información relevante, registrar datos y adjuntar archivos. Por otro lado, el calendario facilita la organización y seguimiento de eventos asociados a la consulta.

4.7.5. Tablas de Control

Se han requerido tablas para la visualización integral de todos los registros de consultas y consultantes. En este contexto, se presenta un ejemplo con la tabla específica de consultas en la Figura 4.7.

Referencia	Función
RF.1	Sistema de solicitudes de asignación de casos
RF.2	Sistema de gestión de casos por comisión
RF.3	Sistema de registro de consultantes y consultas
RF.4	Sistema de alertas y notificaciones
RF.5	Registro y Autenticación de Usuarios

Cuadro 4.2: Requerimientos Funcionales

4.8.1. Sistema de Solicitudes de Asignación de Casos

Referencia	Función
RF.1.1	Un Tomador de Caso podrá revisar la información de una consulta.
RF.1.2	Un Tomador de Caso podrá revisar la cantidad de casos con sus estados de actividad de una comisión.
RF.1.3	Un Tomador de Caso podrá revisar el historial reciente de solicitudes de una comisión.
RF.1.4	Un Tomador de Caso podrá enviar una solicitud de asignación de un caso a una comisión.
RF.1.5	Un Tomador de Caso podrá deshacer una solicitud de asignación.
RF.1.6	Un Tomador de Caso podrá revisar la totalidad de casos.
RF.1.7	Un Tomador de Caso podrá revisar la totalidad de consultantes.

Cuadro 4.3: Requerimientos Funcionales para Tomadores de Caso

4.8.2. Sistema de Gestión de Casos por Comisión

Requerimiento	Función
RF.2.1	Un usuario de una comisión tendrá la capacidad de aceptar o rechazar una solicitud de asignación de un nuevo caso a su comisión.
RF.2.2	Un usuario de una comisión podrá visualizar los casos asignados a la comisión, incluyendo su estado y detalles relevantes.
RF.2.3	Un usuario de una comisión podrá realizar modificaciones en los casos asignados a la comisión, actualizando la información según sea necesario.
RF.2.4	Un usuario de una comisión podrá comentar en los casos asignados a la comisión, proporcionando información adicional o aclaraciones con la posibilidad de adjuntar archivos.
RF.2.5	Un usuario de una comisión tendrá la opción de agregar y quitar eventos en los casos asignados a la comisión, permitiendo un seguimiento detallado de las actividades relacionadas con cada caso.

Cuadro 4.4: Requerimientos Funcionales del Sistema de Gestión de Casos por Comisión

4.8.3. Sistema de Registro de Clientes y Consultas

Requerimiento	Descripción
RF.3.1	Un Tomador de Caso o Super Usuario podrá almacenar los datos de un consultante en el sistema.
RF.3.2	Un Tomador de Caso o Super Usuario podrá registrar los detalles de una consulta de un consultante en el sistema.
RF.3.3	El sistema permitirá cargar consultantes mediante la integración con Google Forms, agilizando el ingreso de datos al sistema.
RF.3.4	El sistema permitirá cargar consultas mediante la integración con Google Forms, facilitando el registro eficiente de información relacionada con las consultas de los consultantes.

Cuadro 4.5: Requerimientos del Sistema de Registro de Clientes y Consultas

4.8.4. Sistema de Notificaciones y Alertas

Requerimiento	Descripción
RF.4.1	El sistema podrá enviar correos electrónicos para notificar a los Tomadores de Caso cuando una solicitud es aceptada o rechazada.
RF.4.2	El sistema podrá enviar correos electrónicos a los usuarios de una comisión al llegar una nueva solicitud de asignación de caso.
RF.4.3	El sistema enviará alertas en tiempo real a los Tomadores de Caso cuando una solicitud es aceptada o rechazada.
RF.4.4	El sistema enviará alertas a los usuarios de una comisión al llegar una solicitud de asignación de caso.
RF.4.5	El sistema enviará alertas cuando ingresa un nuevo consultante o consulta a través de Google Forms, manteniendo a los usuarios informados sobre las actualizaciones en el sistema.

Cuadro 4.6: Requerimientos del Sistema de Notificaciones y Alertas.

4.8.5. Registro y Autenticación de Usuarios

Requerimiento	Descripción
RF.5.1	Un usuario podrá acceder a la plataforma mediante un nombre de usuario y contraseña.
RF.5.2	Un usuario podrá registrarse en la plataforma a través de un proceso de autenticación vía correo electrónico. El acceso a la plataforma estará condicionado a la aceptación por parte de un usuario administrador.
RF.5.3	Un usuario tendrá la capacidad de cambiar su contraseña en caso de olvidarla.
RF.5.4	La plataforma permitirá a los usuarios visualizar únicamente las páginas para las cuales tengan permisos.

Cuadro 4.7: Requerimientos de Registro y Autenticación de Usuarios.

4.9. Requerimientos No Funcionales

Según Sommerville [11], los requerimientos no funcionales son “Son limitaciones sobre servicios o funciones que ofrece el sistema. Incluyen restricciones tanto de temporización y del proceso de desarrollo, como impuestas por los estándares. Los requerimientos no funcionales se suelen aplicar al sistema como un todo, más que a características o a servicios individuales del sistema.”

Referencia	Función
RNF.1	Requerimientos del producto
RNF.2	Requerimientos de la organización
RNF.3	Requerimientos externos

Cuadro 4.8: Requerimientos No Funcionales

4.9.1. Requerimientos del producto

Requerimiento	Descripción
RNF.1.1	El sistema se podrá acceder a través de la web desde una computadora con acceso a internet.
RNF.1.2	La interfaz de usuario deberá ser intuitiva y fácil de usar, siguiendo los principios de diseño de experiencia de usuario (UX).
RNF.1.3	El sistema deberá ser capaz de manejar simultáneamente por 5 a 10 usuarios sin degradación del rendimiento.
RNF.1.4	El acceso a ciertas funcionalidades estará restringido a usuarios autorizados mediante roles y permisos.

Cuadro 4.9: Requerimientos del Producto.

4.9.2. Requerimientos de la organización

Requerimiento	Descripción
RNF.2.1	La PC del usuario deberá tener un mínimo de 4GB de RAM y tarjeta de red para la conexión a internet.
RNF.2.2	La plataforma será compatible con versiones estables de Chrome $\geq 120.0.6099.*$ y accesible a través de pantallas medianas a grandes.
RNF.2.3	El código fuente deberá seguir buenas prácticas de desarrollo y ser fácilmente mantenible.

Cuadro 4.10: Requerimientos de la Organización.

4.9.3. Requerimientos externos

Requerimiento	Descripción
RNF.3.1	Un usuario deberá aceptar los términos y condiciones para el registro a la plataforma.
RNF.3.2	El sistema debe aplicar una política de complejidad de contraseñas que incluya la longitud mínima, uso de mayúsculas, minúsculas, números y caracteres especiales.
RNF.3.3	Se debe implementar una protección efectiva contra ataques de fuerza bruta en las interfaces administrativas, incluyendo mecanismos de bloqueo temporal de cuentas.
RNF.3.4	La integración con Google Forms y la API debe utilizar tokens de seguridad para autenticar y autorizar las solicitudes, manteniendo un flujo seguro de información.
RNF.3.5	La plataforma deberá implementar medidas de seguridad contra ataques CSRF Token.
RNF.3.6	La comunicación entre el servidor y los usuarios debe estar cifrada mediante un certificado SSL proporcionado por Let's Encrypt y configurado en el servidor Nginx.

Cuadro 4.11: Requerimientos Externos.

4.10. Definición de Roles y Responsabilidades

Tomador de Caso	Este usuario tiene acceso al tablero de la consultoría y al panel de control. Su responsabilidad principal es agregar consultantes y consultas de forma manual, además de asignar las consultas a una comisión específica. Los Tomadores de Caso pueden visualizar el estado de cada comisión, revisando la cantidad de casos activos y las últimas solicitudes enviadas.
Integrante de Comisión (Profesor)	Este rol se asigna a los integrantes de una comisión con acceso al tablero. Sus responsabilidades incluyen aceptar o rechazar solicitudes de asignación de casos o consultas. Además, se encargan de gestionar el seguimiento y progreso de cada caso asignado a su comisión.
Jefe de Patrocinio (Administrador)	También conocido como administrador, este usuario tiene privilegios que le otorgan acceso total a la plataforma. Sus responsabilidades principales incluyen la gestión de usuarios, permisos y roles, asegurando el correcto funcionamiento y la seguridad de la plataforma.
Cliente (por Google Forms)	Consultante que busca patrocinio gratuito. Este actor no tiene acceso directo a la plataforma y no ingresa a ella. Su interacción se limita a completar formularios de registro de consultante y consulta, los cuales se cargan automáticamente en la plataforma desde Google Forms .

Cuadro 4.12: Roles y Responsabilidades de los Actores en la Plataforma

4.11. Diagrama de Caso de Uso del Sistema

A continuación, se presentan los diagramas de caso de uso [4.8](#) para identificar a los actores implicados en una interacción. Si bien en este caso no se relacionan los casos de uso a alto nivel con diferentes actores, no significa que no exista una relación. Por otro lado, debe entenderse que Google Forms también cumple un rol e interactúa con el sistema principal Case Management System.

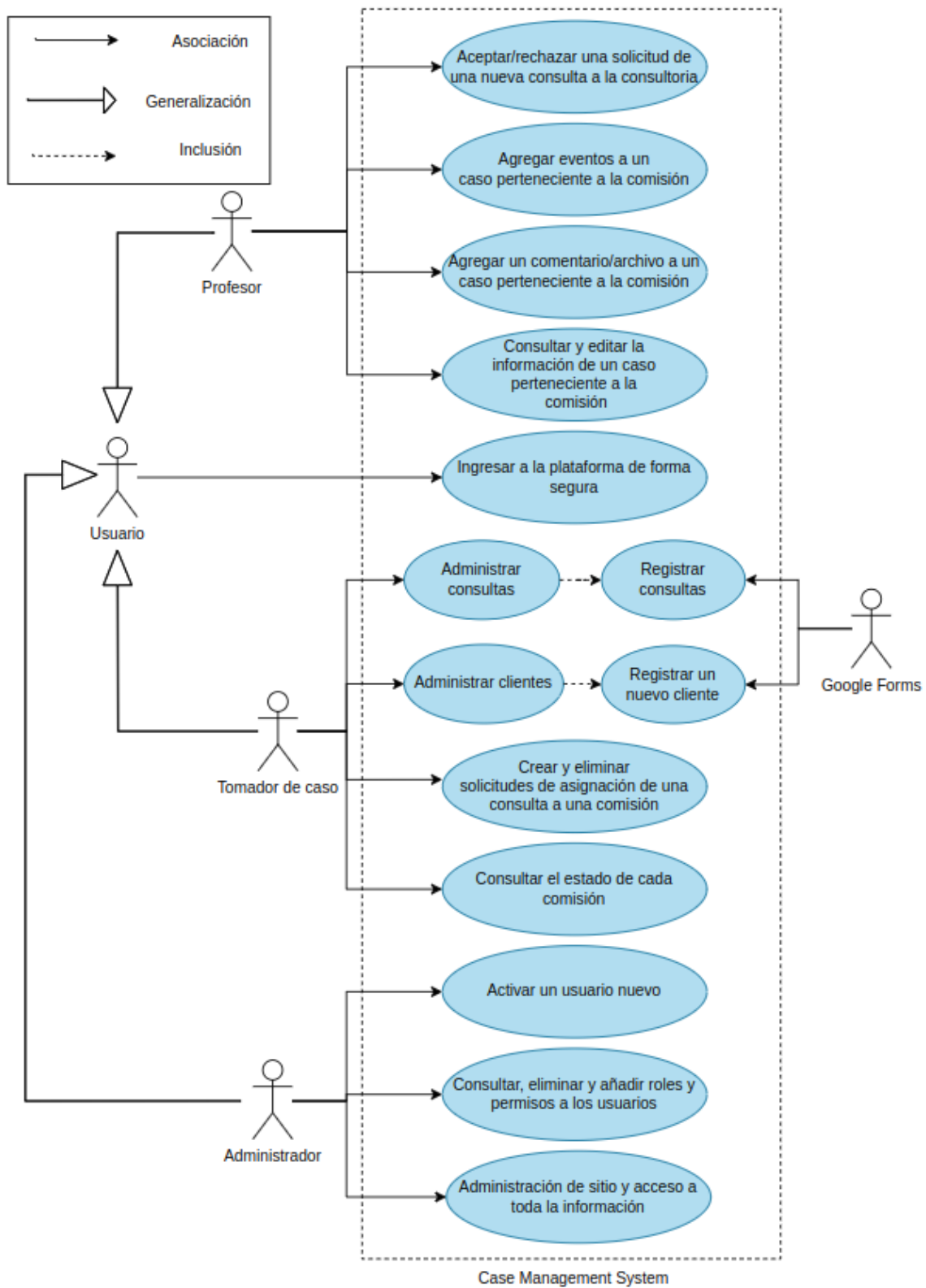


Figura 4.8: Diagrama Caso de Uso

- CU.1: Aceptar/rechazar una solicitud de una nueva consulta a la consultoría.
- CU.2: Agregar eventos a un caso perteneciente a la comisión.

- **CU.3:** Agregar un comentario/archivo a un caso perteneciente a la comisión.
- **CU.4:** Ingresar a la plataforma de forma segura.
- **CU.5:** Administrar consultas.
- **CU.6:** Administrar consultantes.
- **CU.7:** Crear y eliminar solicitudes de asignación de una consulta a una comisión.
- **CU.8:** Ver el estado de cada comisión.
- **CU.9:** Activar a un usuario nuevo.
- **CU.10:** Administrar roles y permisos de los usuarios.
- **CU.11:** Administración de sitio y acceso a toda la información.

4.12. Diagrama de Secuencia Nominal Simplificado

A continuación, se incluyen los diagramas de secuencia 4.9 y 4.10 de alto nivel para agregar comprensión a la funcionalidad de cada actor y su interacción con otros actores.

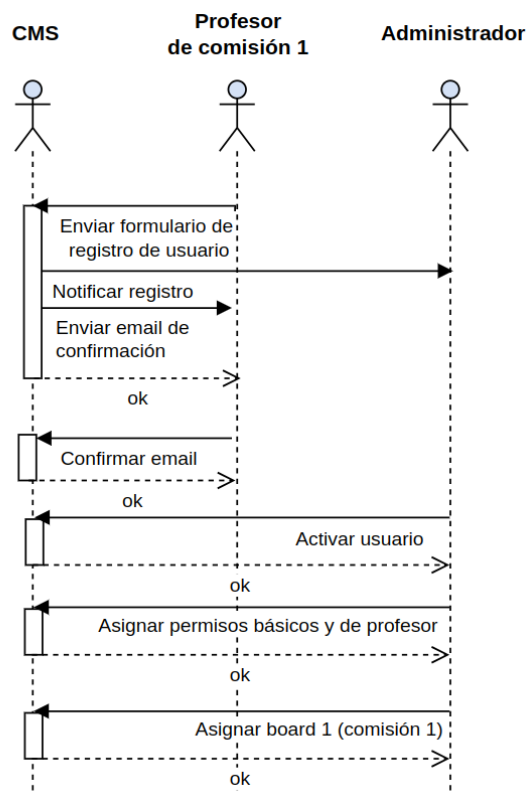


Figura 4.9: Diagrama de Secuencia Para Registro de Usuario

En este diagrama, participa el sistema Case Management System, el profesor de la comisión número uno y el jefe de patrocinio o administrador.

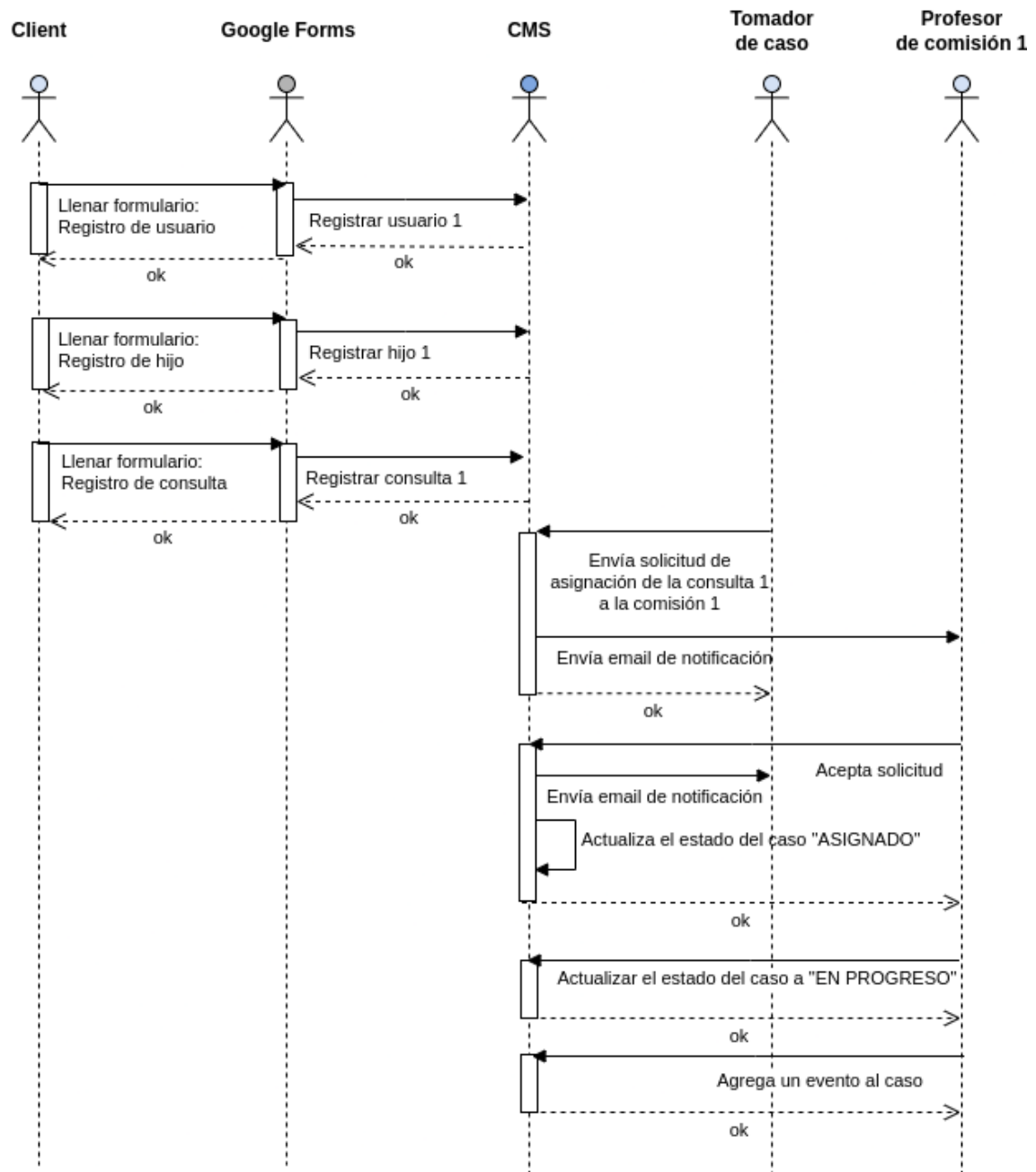


Figura 4.10: Diagrama de Secuencia Simplificado

En este diagrama, participan varios actores, incluyendo un consultante con un solo hijo, el sistema externo Google Forms, el sistema Case Management System, el usuario tomador de caso, el profesor de la comisión número uno y el jefe de patrocinio o administrador.

4.13. Matriz de Trazabilidad

La siguiente matriz de trazabilidad relaciona cada uno de los requerimientos con los casos de uso.

	CU1	CU2	CU3	CU4	CU5	CU6	CU7	CU8	CU9	CU10	CU11
RF.1.1						•					
RF.1.2									•		
RF.1.3									•		
RF.1.4								•			
RF.1.5								•			
RF.1.6						•					
RF.1.7							•				
RF.2.1	•										
RF.2.2					•						
RF.2.3					•						
RF.2.4			•								
RF.2.5		•									
RF.3.1							•				•
RF.3.2						•					•
RF.3.3							•				
RF.3.4						•					
RF.4.1	•										
RF.4.2								•			
RF.4.3	•										
RF.4.4								•			
RF.4.5							•				
RF.5.1				•							
RF.5.2				•							
RF.5.3				•							
RF.5.4										•	
RNF.1.1											
RNF.1.2											
RNF.1.3											
RNF.1.4										•	
RNF.2.1											
RNF.2.2											
RNF.2.3											
RNF.3.1											
RNF.3.2											
RNF.3.3											
RNF.3.4											
RNF.3.5											
RNF.3.6											

Cuadro 4.13: Matriz de Trazabilidad

5. Arquitectura y Diseño

5.1. Introducción

En este capítulo, se presenta la arquitectura y el diseño del sistema implementado. Se abordan los principios y decisiones arquitectónicas que guiaron el desarrollo del sistema, así como los modelos y diagramas que ilustran la estructura y comportamiento del mismo.

5.2. Modelado del sistema

Según Somerville, el modelado de sistemas es un proceso para desarrollar modelos abstractos que representan diferentes perspectivas de un sistema. Estos modelos, que a menudo utilizan notaciones gráficas basadas en el Lenguaje de Modelado Unificado (UML), desempeñan un papel importante en la ingeniería de requerimientos y el diseño del sistema [11]. En esta sección se modelan las perspectivas de contexto o entorno, de interacción, estructural y de comportamiento para proporcionar una comprensión integral del sistema.

5.2.1. Modelo de Contexto

En la figura 5.1, se identifican dos sistemas externos que interactúan con la plataforma:

- **Google Forms:** Utilizado para el envío de formularios, facilitando el ingreso de consultantes y consultas.
- **Email:** El sistema de correo electrónico, como Gmail, se conecta a la plataforma para el envío de notificaciones y el registro de nuevos usuarios.

Por otro lado, se encuentran los distintos tipos de usuarios que acceden a la plataforma:

- **Tomadores de Caso:** Administran los casos, asignándolos a diferentes comisiones.
- **Jefes de Patrocinio:** Ingresan a la página de administración para gestionar permisos de usuarios, aprobar nuevos ingresos, visualizar información general y editar configuraciones.
- **Integrantes de Comisión:** Profesores, jefes de trabajo práctico y/o jefe de comisión que acceden al tablero de su comisión para gestionar los casos.
- **Alumnos:** Los alumnos no acceden directamente a la plataforma; en su lugar, envían información al profesor, quien gestiona los casos.

- **Clientes:** No ingresan a la plataforma; en su lugar, completan formularios que se envían al sistema.

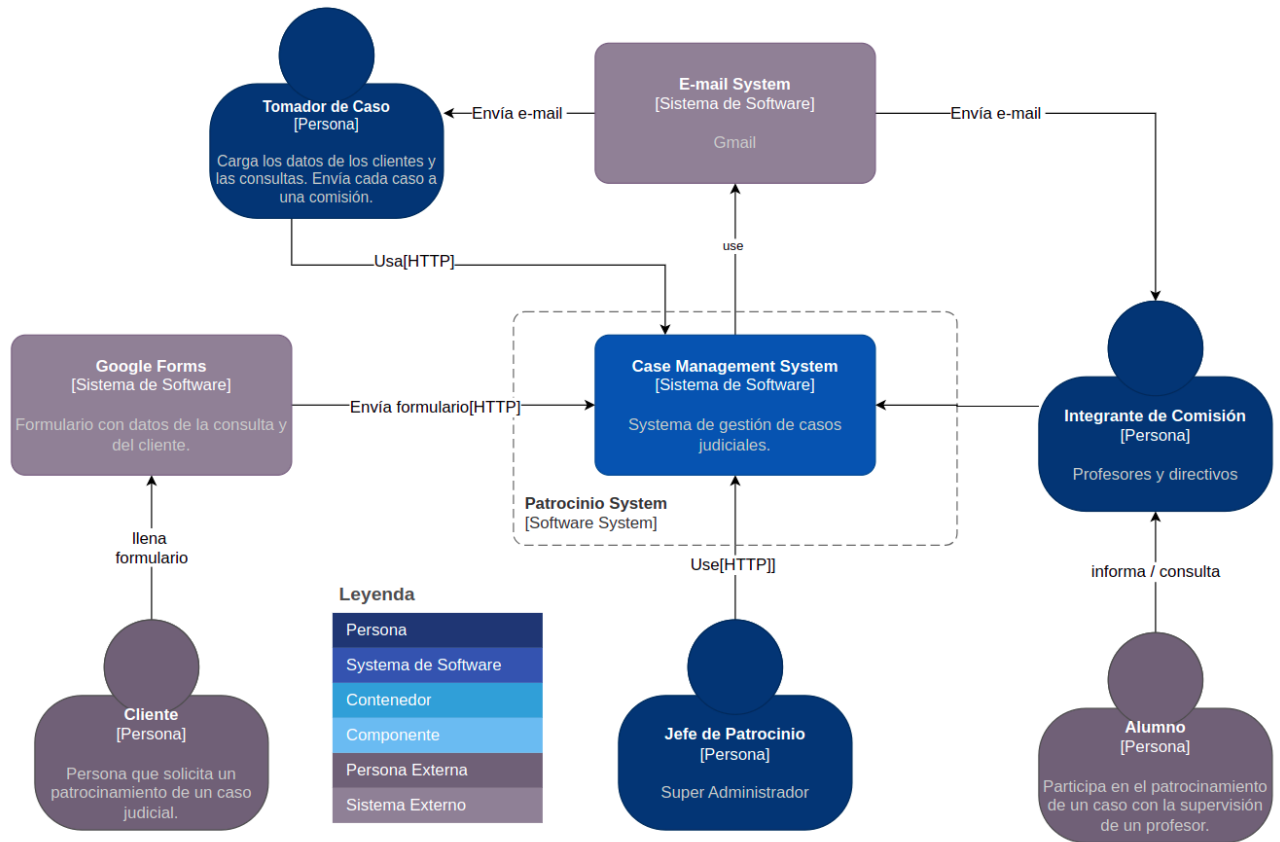


Figura 5.1: Diagrama de Contexto C4

5.2.2. Modelo de Contenedores

Si profundizamos, observamos cómo está conformado el sistema a nivel de contenedores.

Un “contenedor” no se limita a la noción convencional de un contenedor Docker; más bien, representa una aplicación, un almacén de datos o cualquier entidad desplegable de manera independiente, abarcando desde aplicaciones web hasta sistemas de archivos.

La figura 5.2 muestra el diagrama de contenedores que se explicará a continuación.

- **Web App:** Proporciona una interfaz web para el sistema de gestión de casos destinado a las comisiones y consultoría.
- **Real-time Notifications Service:** Utiliza Django ASGI con WebSockets para la entrega de notificaciones en tiempo real.
- **Redis:** Actúa como un almacén en memoria y se utiliza para gestionar las comunicaciones en tiempo real a través de Pub/Sub.

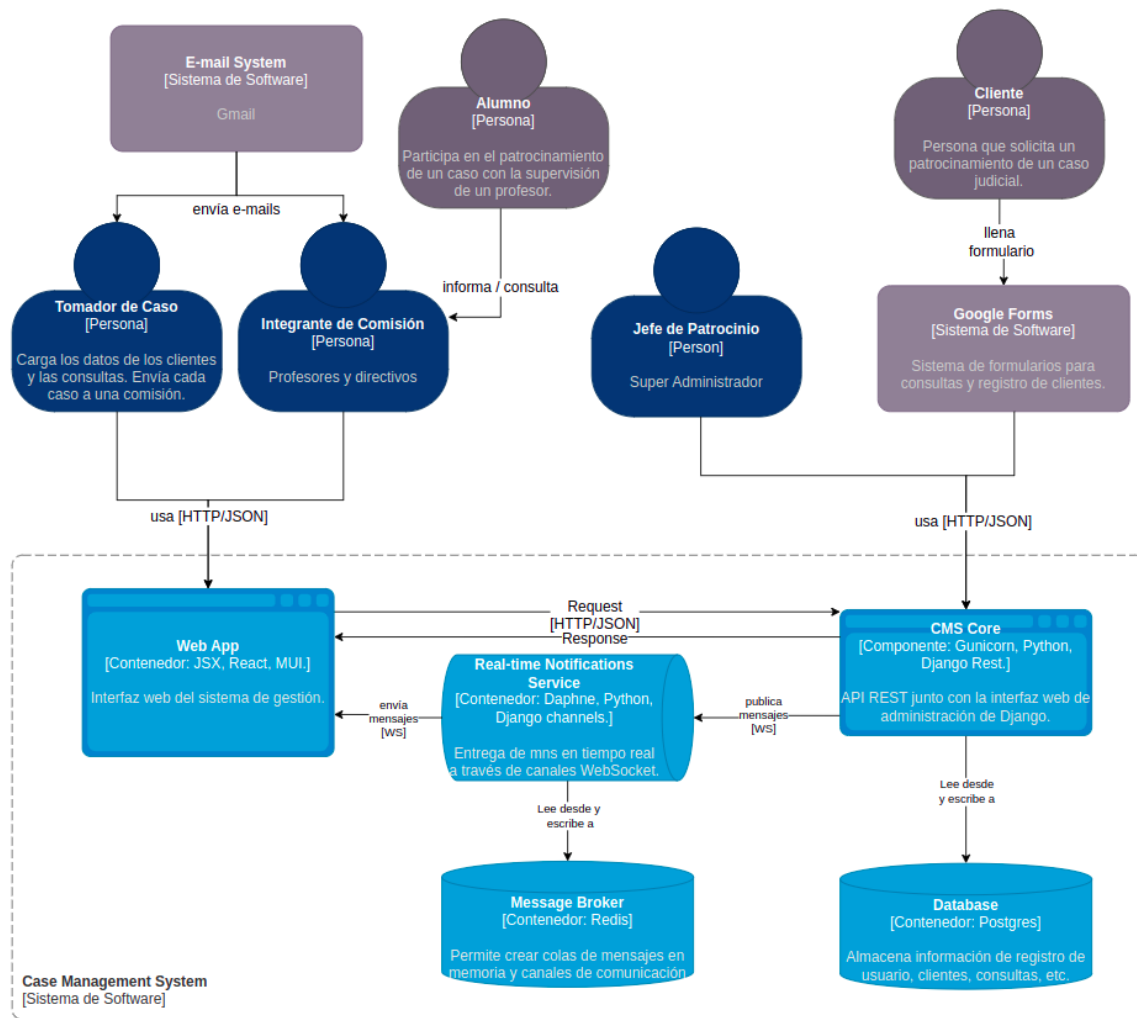


Figura 5.2: Diagrama de Contenedores C4

- **CMS Core:** Proporciona una API REST en Django con la lógica de negocio. También ofrece la interfaz de administración de Django.

5.2.3. Modelo de Componentes

A continuación, se desglosan algunos contenedores para identificar los componentes estructurales principales y sus interacciones mediante los siguientes diagramas.

En el diagrama del contenedor CMS Core (Figura 5.3), se presentan los componentes clave y sus relaciones internas.

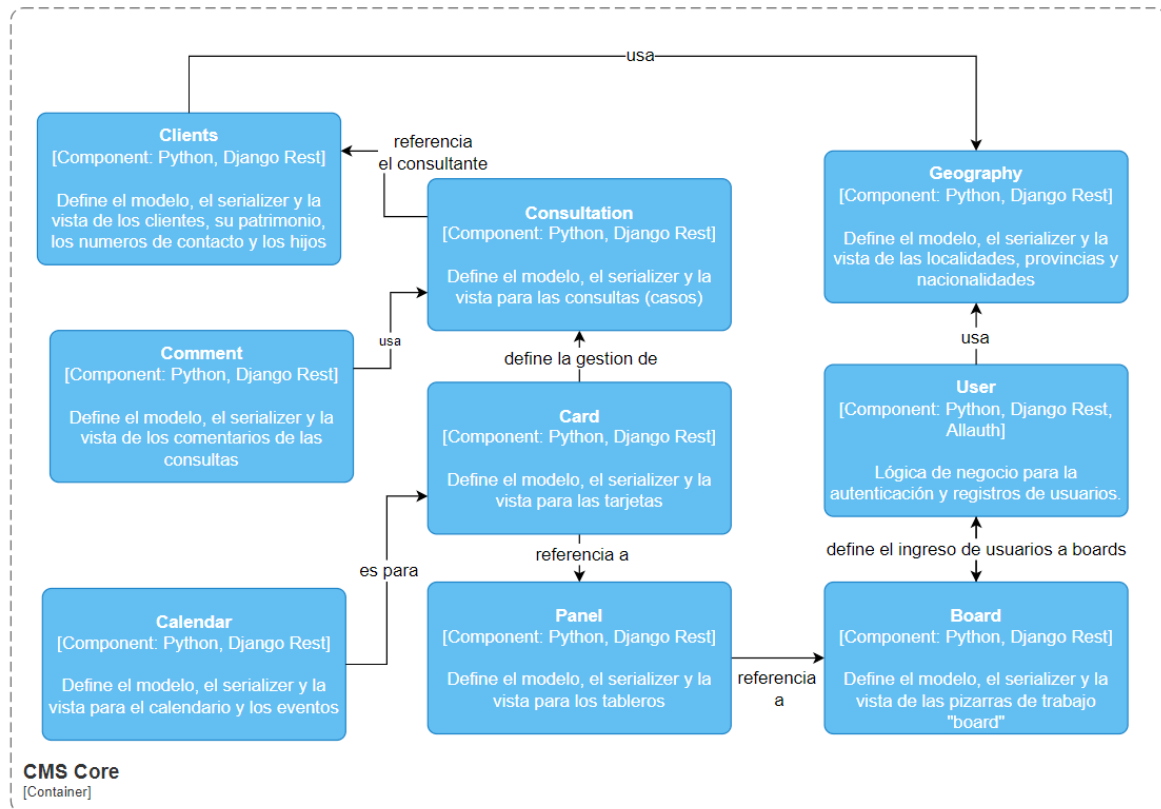


Figura 5.3: Diagrama de Componentes C4 del Contenedor CMS Core

Este diagrama ilustra cómo el contenedor CMS Core incorpora cada componente de la API REST de manera modular, junto con la interfaz de usuario proporcionada por Django para la administración.

Asimismo, a continuación, se presenta el diagrama de componentes del contenedor Web App con sus principales elementos (ver Figura 5.4).

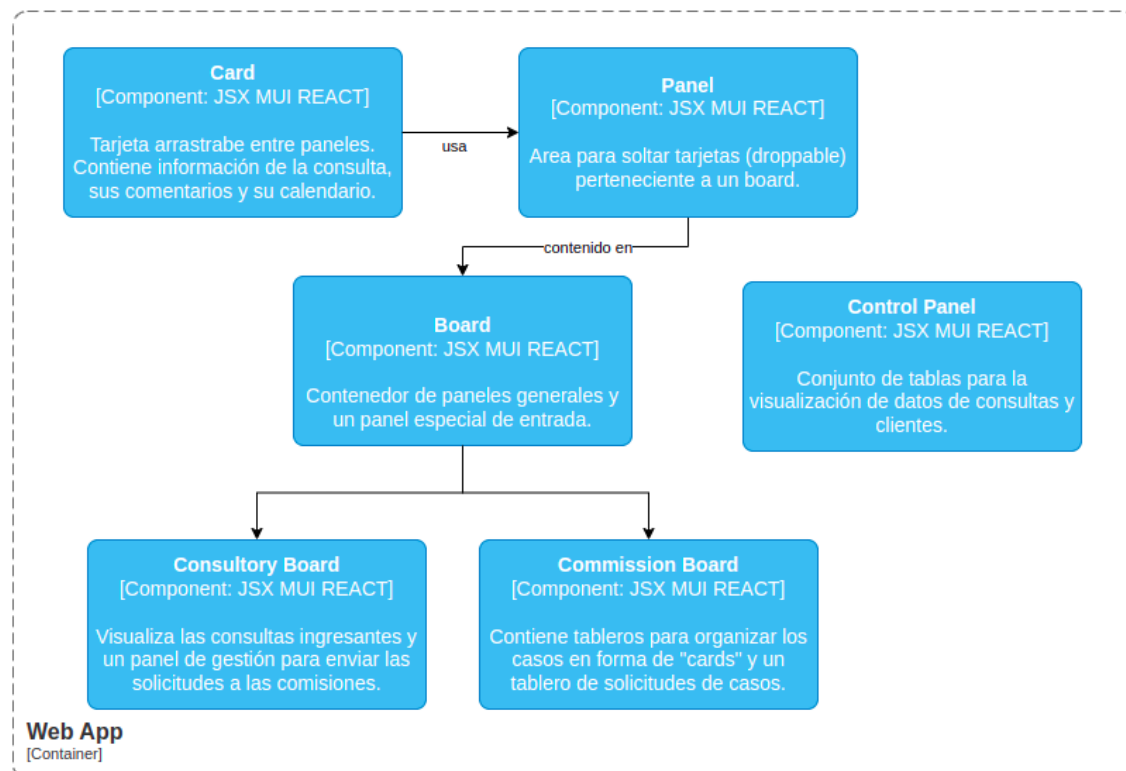


Figura 5.4: Diagrama de Componentes C4 del Contenedor Web App

En este diagrama, se detalla la estructura del tablero (Board), que contiene paneles y, a su vez, cada panel alberga tarjetas (Cards). Además, se distinguen dos tipos de tableros: “Consultancy” para la consultoría y “Commission Board” para la pizarra de trabajo de cada comisión. Por separado, se encuentra el “Control Panel”, que consta de dos tablas para consultas y consultantes.

5.3. Arquitectura

La implementación del software seguirá el patrón de arquitectura cliente-servidor. Esta elección se fundamenta en la necesidad de alojar la aplicación web en el servidor de aplicaciones, permitiendo a los usuarios acceder a ella desde cualquier dispositivo con conexión a internet. Se optará por la arquitectura REST para la comunicación cliente-servidor. Además, para el sistema de notificaciones, se aplicará el patrón de arquitectura Mediador (Broker Pattern), haciendo uso del diseño de publicador-suscriptor.

5.4. Desarrollo Modular de la Arquitectura REST en DRF

Cuando se menciona que la API REST está estructurada de forma modular, se refiere a que cada componente se organiza como un módulo independiente (ver 2.1). Cada uno de estos módulos define las rutas, modelos y vistas necesarias para su funcionamiento.

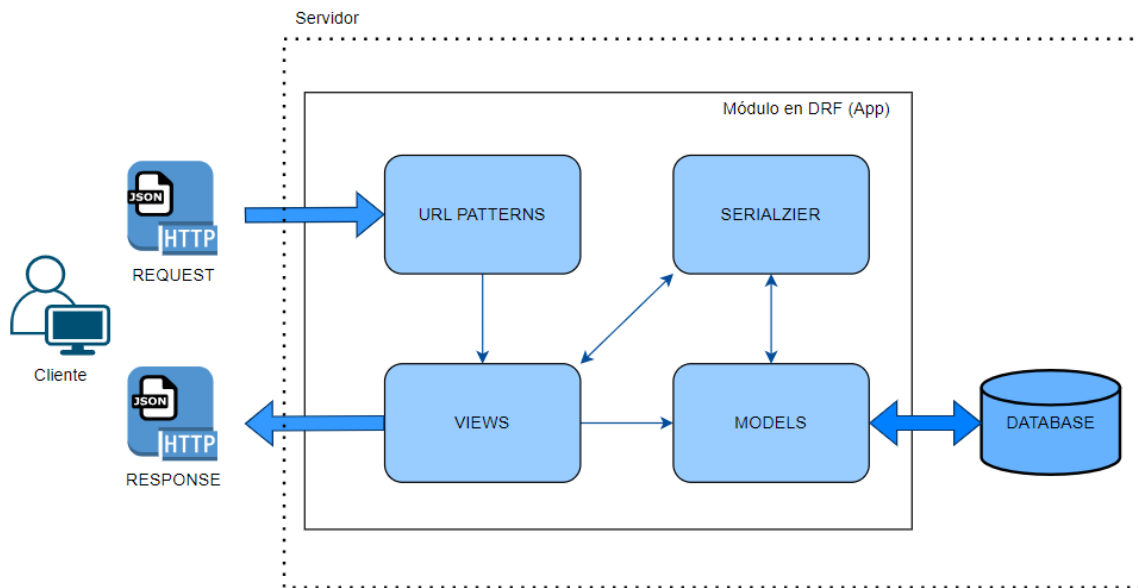


Figura 5.5: Diagrama de un Módulo en DRF

En la figura 5.5, los serializadores traducen la información de los modelos a un formato serializado (JSON) y viceversa. Las vistas establecen la lógica de negocio para manipular y procesar los datos antes de enviar una respuesta al cliente. Por otro lado, los modelos representan la estructura de los datos almacenados en la base de datos, proporcionando una abstracción entre la base de datos y las representaciones de datos expuestas a través de la API.

5.5. Patrón de Arquitectura Mediador (Broker pattern)

La implementación del patrón de arquitectura Mediador (Broker pattern) se logró mediante el uso de Django Channels y Redis como el componente Broker que coordina la comunicación entre los diversos elementos del sistema de notificaciones en tiempo real. En este contexto, el servidor publica sus servicios en el Broker, representados como “canales” (un canal para la consultoría y un canal por comisión).

Cuando un cliente requiere un servicio específico, se comunica con el Broker, el cual redirige al cliente hacia el servicio correspondiente según su registro.

5.6. Implementación del Patrón de Diseño Pub/Sub sobre el Protocolo de comunicación Web Sockets

En relación a la sección anterior, respecto al sistema de notificaciones en tiempo real, la implementación del patrón de diseño Publicador-Suscriptor (Pub/Sub [2.6](#)) sobre el protocolo de comunicación WebSockets se utiliza en la suscripción de un cliente a un canal y la publicación de mensajes del servidor a dicho canal.

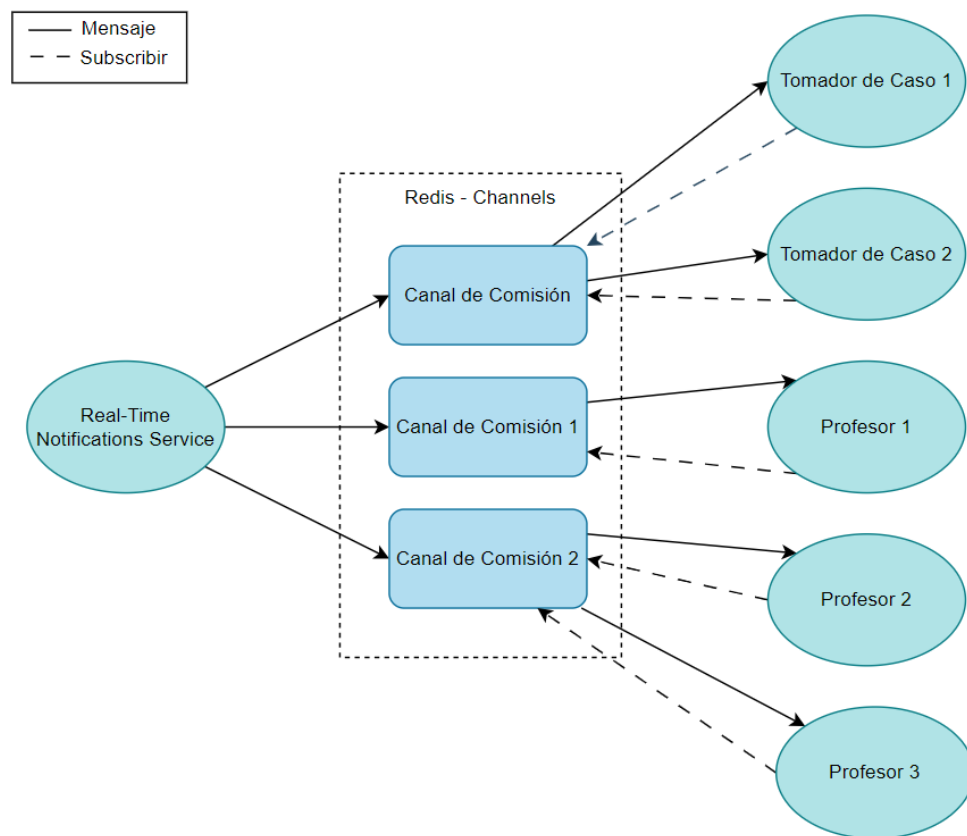


Figura 5.6: Diagrama Implementación del Patrón Pub/Sub

Como se observa en la figura [5.6](#), los canales representan una consultoría y un canal por cada comisión.

5.7. Implementación Arrastrar y soltar

Se utilizó el framework de JavaScript *React-beautiful-dnd* para la creación de los “Boards”, permitiendo al usuario tener una experiencia más intuitiva y rápida con la función de “Arrastrar y soltar”. La similitud con Trello fue determinada en la etapa de gobernanza de datos y reingeniería de procesos, donde participó el equipo del Patrocinio Jurídico Gratuito de la Facultad de Derecho.

La estructura de “Arrastrar y soltar” se muestra en la Figura 5.7.

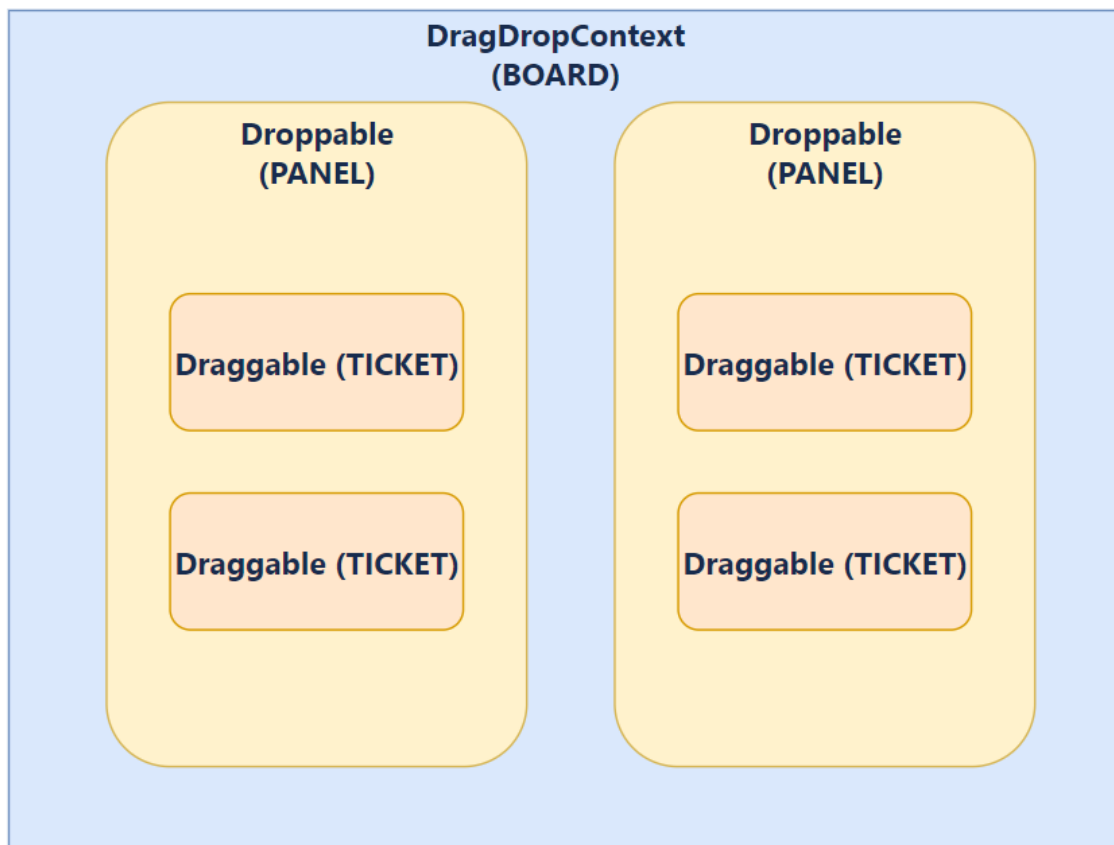


Figura 5.7: Estructura Drag and Drop

- **DragDropContext:** Envuelve la parte de la aplicación para la que desea habilitar la función de arrastrar y soltar.
- **Droppable:** Esta es el área donde puede soltar y contiene.
- **Draggable:** Estos son los elementos que se pueden arrastrar.

A continuación, se presenta un diagrama de componentes que ilustra la estructura y relación de los elementos clave en la implementación del patrón de “Arrastrar y soltar” en el frontend.

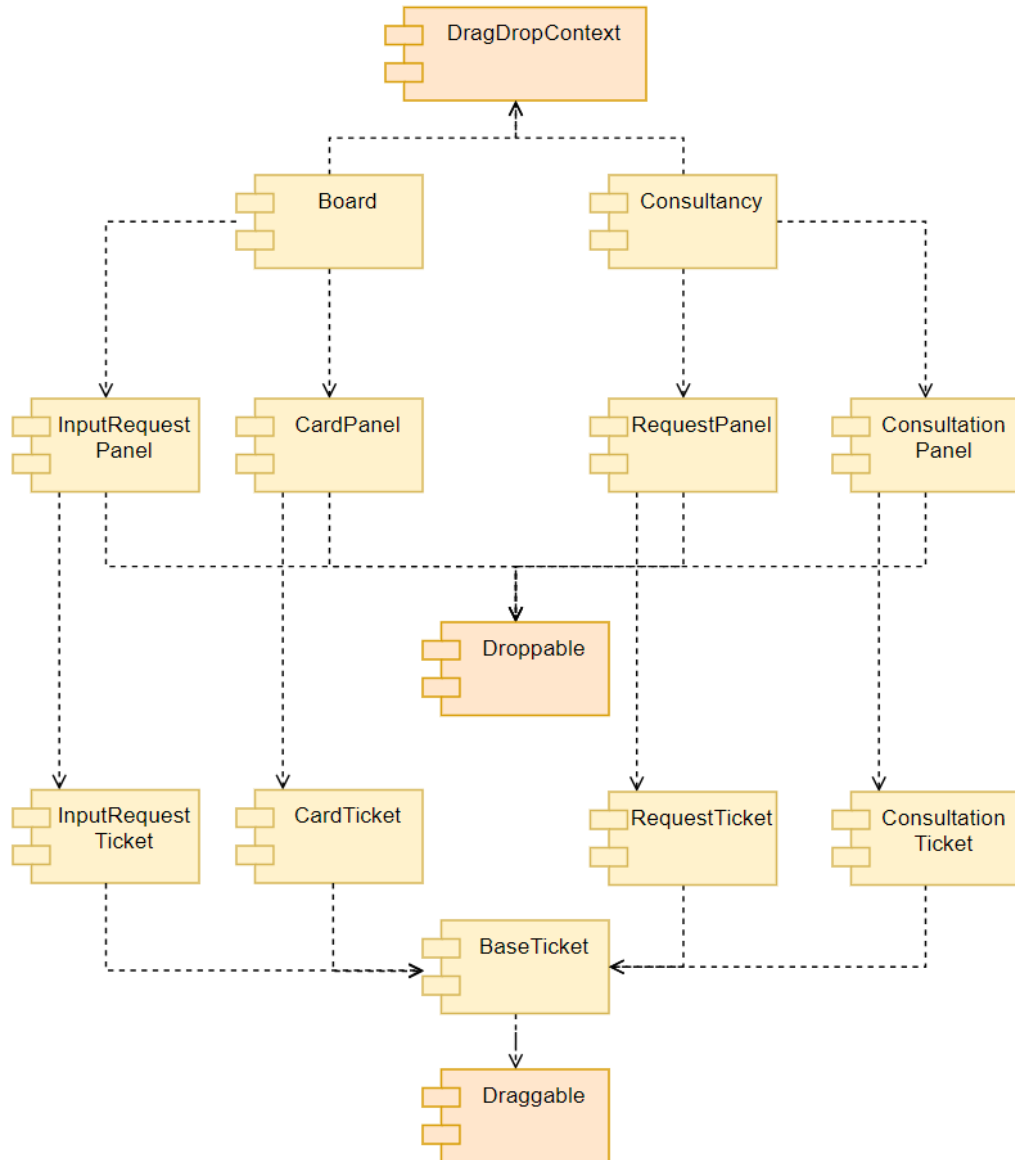


Figura 5.8: Diagrama de Componentes Drag and Drop

En la figura 5.8, se pueden observar diferentes tipos de pizarras, paneles y tickets. Cada pizarra representa un área de trabajo donde se organizan los paneles. Para cada pizarra, existen dos tipos de paneles: uno para la entrada de tickets y otro para la administración de los tickets propios de la consultoría o comisión, según corresponda. Los tickets se dividen en tres categorías: Solicitud, Consulta y Card. Los tickets de Solicitud se refieren a la solicitud de asignación de una consulta. Los tickets de Consulta son aquellos que se encuentran en la consultoría sin asignar, mientras que los tickets Card se refieren a los tickets asignados a una comisión.

6. Implementación y Prueba

6.1. Introducción

Este capítulo se centra en la implementación y validación del componente desarrollado. Se presentan los resultados más relevantes derivados de las pruebas realizadas, garantizando así el correcto funcionamiento del sistema y su conformidad con los requisitos establecidos por el cliente.

6.2. Etapa de Implementación

6.2.1. Stack Tecnológico Seleccionado

La elección del stack tecnológico se fundamentó en la perspectiva de mantener y continuar el desarrollo de la plataforma a largo plazo. Considerando que el laboratorio IALAB ofrece cursos de Python, Javascript y React, se optó por utilizar estos lenguajes, anticipando que los graduados de dichos cursos podrían constituir una valiosa fuente de recursos para posiciones de becarios en el proyecto.

Además de Python, Javascript y React, se seleccionaron otras tecnologías clave para la implementación de la plataforma. Para la base de datos, se eligió PostgreSQL, un sistema de gestión de bases de datos relacional robusto y de código abierto. En cuanto al componente de almacenamiento en memoria y coordinación de comunicaciones en tiempo real, se incorporó Redis. Para el desarrollo de la API REST, se adoptó Django Rest Framework (DRF), una potente herramienta que facilita la creación de servicios web con Django.

6.2.2. Diagrama de Paquetes en el Backend

En el diagrama de paquetes del backend (Figura 6.1), se presenta una estructura clara y organizada del sistema implementado con Django. El directorio principal, denominado *api patrocinio*, se destaca en azul y alberga la definición de todas las URL y la configuración global del proyecto.

En color amarillo, se identifican los módulos de la API REST, encapsulando las diferentes funcionalidades de la aplicación. En verde, se encuentran los archivos de plantillas y recursos que contribuyen a la presentación de la interfaz y el envío de emails.

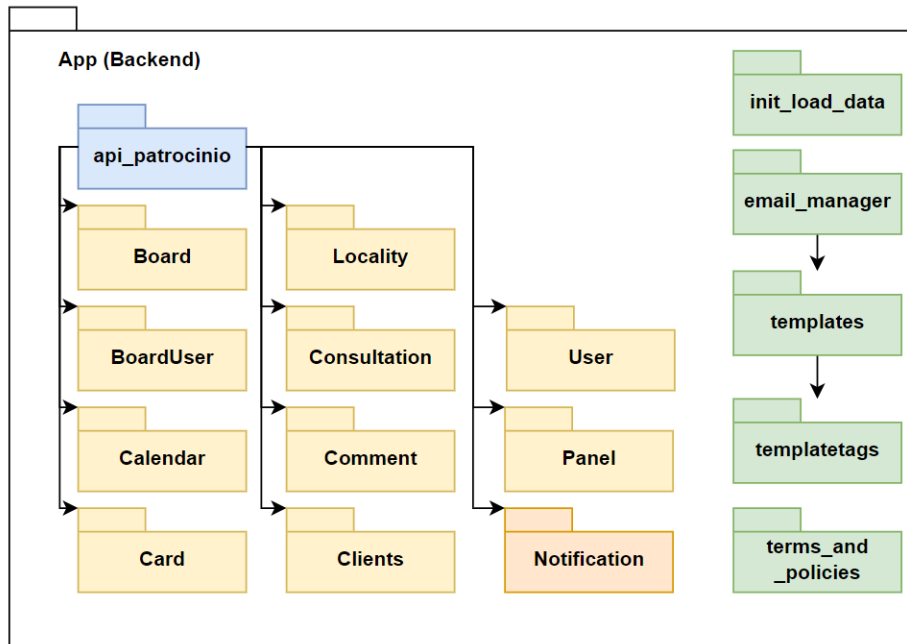


Figura 6.1: Diagrama de Paquetes del Backend

6.2.3. Diagrama de Paquetes en el Frontend

En el diagrama de paquetes del frontend (Figura 6.2), se observa la estructura organizada de la interfaz de usuario desarrollada en React. El directorio `pages`, en amarillo, contiene las definiciones de las páginas, que a su vez alojan contenedores, pudiendo contener otros contenedores y/o componentes.

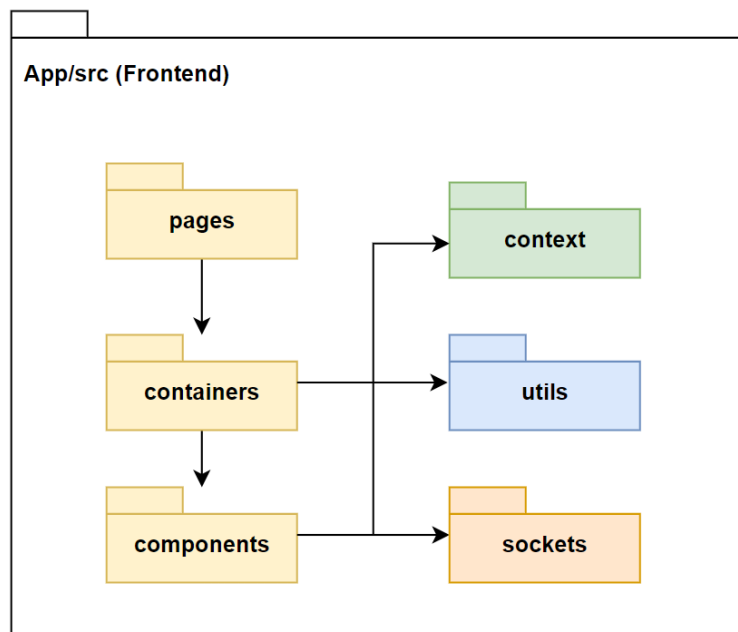


Figura 6.2: Diagrama de Paquetes del Frontend

El directorio *pages* encapsula definiciones simples de cada página, gracias a la reutilización de código proporcionada por los contenedores y componentes. Dentro del directorio *containers*, se encuentran los contenedores que agrupan un conjunto de componentes destinados a una interfaz o funcionalidad específica. Por otro lado, el directorio *components* contiene las definiciones de los componentes individuales.

En verde, el directorio *context* almacena los proveedores utilizados en el proyecto React, como el proveedor de información del usuario. Este enfoque permite gestionar eficientemente el estado de la aplicación y compartir información relevante entre componentes.

La lógica para el procesamiento y la obtención de datos desde el backend se encuentra en el directorio *utils*, resaltado en azul. Este directorio actúa como un repositorio centralizado para las funciones y utilidades esenciales utilizadas en todo el proyecto, promoviendo la coherencia y el mantenimiento eficiente del código.

Finalmente, en naranja, el directorio *sockets* contiene los consumidores del sistema de notificaciones en tiempo real, lo que contribuye significativamente a la implementación de esta funcionalidad clave. La capacidad de actualización dinámica de las vistas por eventos y el envío de alertas se logra mediante estos consumidores, mejorando la experiencia del usuario al mantenerlo informado en tiempo real sobre eventos relevantes.

6.2.4. Integración con Google Forms

La integración con Google Forms se ha llevado a cabo mediante el desarrollo de plugins que optimizan la recolección y organización de datos a través de formularios. A continuación, se detallan los aspectos clave de esta integración:

Características de Google Forms:

Google Forms se presenta como una herramienta versátil y gratuita para la recopilación de datos a través de formularios. Su facilidad de uso y la familiaridad que tienen los usuarios con esta plataforma, gracias a su implementación en el ámbito del patrocinio jurídico de la UBA, fueron determinantes en su elección. Además, los formularios de Google pueden integrarse en correos electrónicos y páginas web, ampliando su versatilidad.

Desafíos y Soluciones en la Estructuración de Preguntas:

Uno de los desafíos encontrados fue la limitación en la diversidad de tipos de datos disponibles en los formularios. Esto se volvió problemático cuando un conjunto de preguntas dependía de otra, como en el caso de la selección de la nacionalidad, provincia y localidad. La solución adoptada fue conceptualizar cada grupo como una sección independiente, facilitando la navegación del usuario a través de las diferentes opciones.

Dada la complejidad del proceso, la cantidad de volumen inusable proporcionado a la base de datos si se tenían en cuenta todas las localidades del mundo, se decidió restringir a provincias y localidades de Argentina. Para países extranjeros, se permite recuperar únicamente la nacionalidad, registrando la provincia y la localidad como campos de texto en Google Forms. Aunque estos últimos no son recolectados por el sistema para garantizar la compatibilidad con la base de datos, pueden ser ingresados manualmente por un administrador si es necesario.

Automatización de la Carga de Preguntas:

La carga manual de datos para las localidades, provincias y nacionalidades se optimizó mediante una hoja de cálculo con opciones de filtro por nacionalidad y un script generado en Google Apps Script [2.4.1](#). Estos recursos, están disponibles en el repositorio de la unidad (ver [3.3.1](#)), adicionalmente se deberá establecer vínculos manuales entre preguntas para redirigir a los usuarios a las secciones correspondientes según sus elecciones.

Manejo de Formularios Específicos:

Para casos específicos, como el registro de hijos de un consultante, se creó un formulario separado debido a las limitaciones de Google Forms en la gestión dinámica de la cantidad de hijos que un consultante podría tener. Además, se diseñó un formulario exclusivo para consultas, vinculando la consulta con el consultante mediante su documento de identidad (DNI o pasaporte).

Envío de Datos al Sistema Case Management System:

Para el envío de la información recopilada a través de los formularios, se implementaron endpoints dedicados para cada tipo de formulario en la API REST. Además, se diseñaron funciones en Google Apps Script que se ejecutan al enviar un formulario, encargadas de recuperar y transformar los datos antes de transmitirlos a la API REST del proyecto. En caso de fallos en este proceso, el sistema notificará a través de correo electrónico sobre la imposibilidad de enviar los datos, asegurando una comunicación eficiente en caso de inconvenientes.

Para obtener información adicional sobre las interfaces, consulte la sección de anexos [D](#).

La implementación de estas funcionalidades se puede encontrar en el directorio [proyecto-patrocinio/com/script-forms/](#) del repositorio del proyecto. Para incorporar estos archivos al formulario correspondiente, simplemente abra el editor del formulario, seleccione los tres puntos, vaya a “Editor de secuencias de comandos” y agregue los archivos correspondientes.

La Figura [6.3](#) muestra una captura de pantalla del código en Google Apps Script.

Editar Activador de Patrocinio- client - form

Seleccionar qué función ejecutar

onFormSubmit ▼

Qué debe ejecutarse durante el despliegue

Principal ▼

Selecciona la fuente del evento

De un formulario ▼

Selecciona el tipo de evento

Al enviarse el formulario ▼

Ajustes de notificación de errores +

Notifícame inmediatamente

Cancelar Guardar

Figura 6.5: Activador para el envío del formulario a la API.

Seguridad y Autenticación:

Se estableció un rol especial para Google Forms con el fin de garantizar la conexión segura con el sistema. Se requiere proporcionar las credenciales de un usuario con este rol en cada formulario. Para facilitar la gestión de estas credenciales, se implementó un plugin de menú mediante Google Apps Script, ofreciendo opciones para cargar credenciales y refrescar tokens de forma intuitiva.

La implementación de esta funcionalidad implica agregar los archivos del directorio [proyecto-patrocinio/com/script-forms/common](#) del repositorio en los tres formularios.

En la Figura 6.6, se muestra la ubicación del plugin del menú en la esquina superior derecha del formulario.

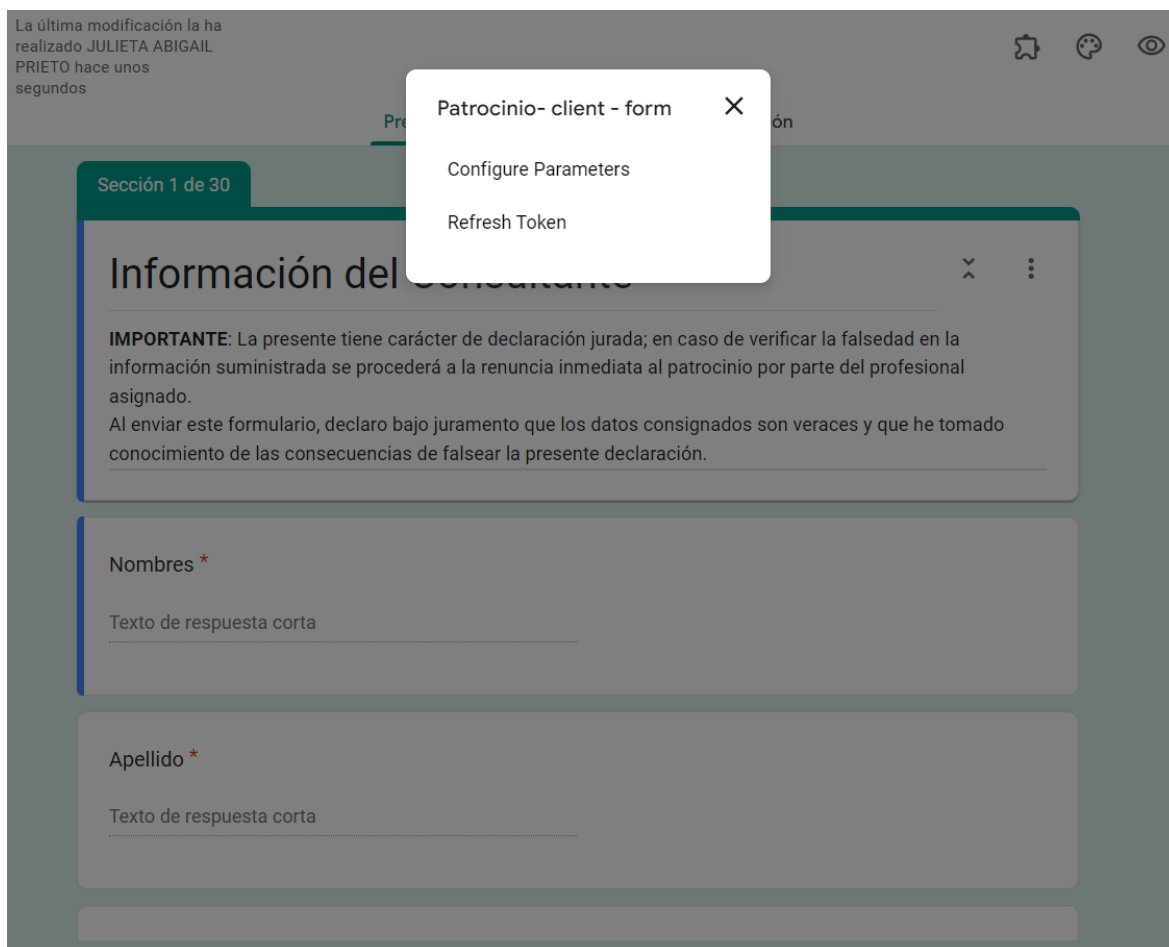


Figura 6.6: Ubicación del Plugin de Menú en Google Forms

En la Figura 6.7, se presenta la interfaz de configuración del plugin. Aquí, se debe establecer, por única vez, el nombre de usuario y la contraseña del usuario de Case Management System con permisos específicos para “forms”, la URL para iniciar sesión, es decir <https://{{domain}}/api/auth/login/>, y la URL del endpoint correspondiente, como por ejemplo <https://{{domain}}/api/consultations/consultation/form/> para el formulario de consultantes.

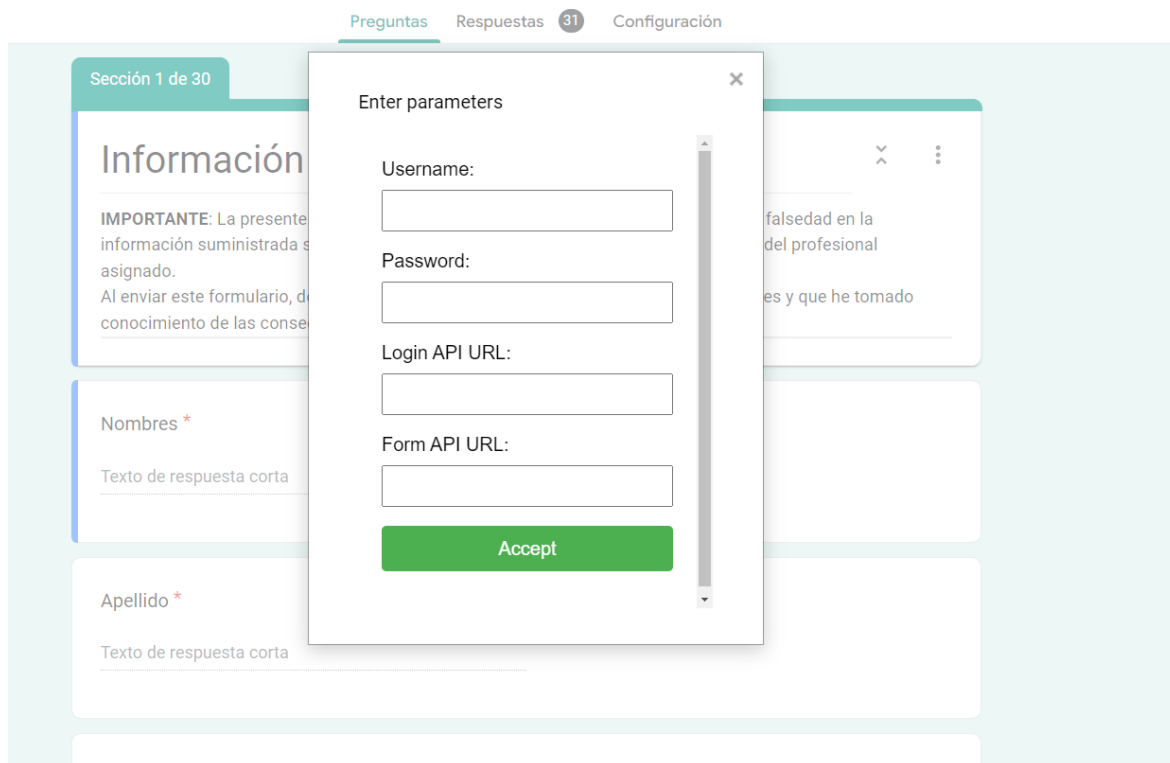


Figura 6.7: Interfaz de Configuración del Plugin de Menú

Para obtener información detallada, consulte el README y los archivos de notas en cada directorio correspondiente al formulario en el repositorio.

Refresco Automático de Tokens:

Con el objetivo de mejorar la eficiencia del sistema, se ha implementado un activador que ejecuta automáticamente el proceso de refresco del token en intervalos regulares. Aunque es posible realizar este proceso manualmente, se recomienda configurar la opción automática para garantizar la continuidad segura de la conexión.

Para implementar esta funcionalidad, además de agregar los archivos del directorio [proyecto-patrocinio/com/script-forms/common](https://github.com/proyecto-patrocinio/com/script-forms/common) en el repositorio, es necesario crear manualmente el activador. Se sugiere configurar el activador para que ejecute el refresco del token diariamente, preferiblemente durante un horario nocturno.

En la Figura 6.8, se muestra cómo crear el activador de forma manual.

Editar Activador de Patrocinio - Consulta

Seleccionar qué función ejecutar	Ajustes de notificación de errores +
<div>updateCredentials ▼</div>	<div>Notifícame cada día ▼</div>
Qué debe ejecutarse durante el despliegue	
<div>Principal ▼</div>	
Selecciona la fuente del evento	
<div>Según tiempo ▼</div>	
Selecciona el tipo de activador basado en la hora	
<div>Temporizador por días ▼</div>	
Selecciona la hora	
<div>De 3:00 a 4:00 h ▼</div>	
(GMT-03:00)	
	<div>Cancelar</div> <div>Guardar</div>

Figura 6.8: Activador para el Refresco Automático del Token

6.2.5. Nginx como Servidor y Reverse Proxy

En la infraestructura de la plataforma, se utiliza Nginx como reverse proxy para dirigir el tráfico a los diferentes servicios. La estructura general se compone de tres servicios principales: el frontend con Node, la API REST de Django alojada en Gunicorn, y el sistema de notificaciones en tiempo real alojado en Dhapne.

Puede encontrar un template del archivo Nginx en el repositorio de deploy referenciado en la sección [3.3.1](#).

Adicionalmente, se ha añadido un archivo de configuración adicional de Nginx en el servidor para integrar el nuevo servicio en el sistema. Ambos archivos de configuración se encuentran detallados en el anexo, específicamente en el Apéndice [C](#).

Limitación de Solicitudes y Conexiones

Se han implementado límites de solicitudes y conexiones utilizando las directivas `limit_req_zone` y `limit_conn_zone` de NGINX. Estas medidas ayudan a mitigar posibles ataques DDoS y garantizan un rendimiento óptimo del sistema al controlar la velocidad de las solicitudes y conexiones desde una sola dirección IP.

Configuración de Redirecciones

NGINX se encarga de redirigir las solicitudes a los servicios correspondientes mediante la configuración de las ubicaciones (`location`). Se han definido ubicaciones específicas para el frontend, la API REST y las conexiones WebSocket, asegurando un enrutamiento adecuado de acuerdo con los requisitos del sistema.

Alias para Recursos Estáticos

Se utiliza la directiva `alias` para configurar NGINX y servir recursos estáticos desde una ubicación específica, facilitando la gestión de archivos estáticos generados por Django para la interfaz de administración.

Consideraciones Adicionales

El archivo de configuración en el servidor está configurado para manejar solicitudes HTTPS en el puerto 443 utilizando certificados SSL proporcionados por Certbot. Proporciona una capa adicional de seguridad al cifrar la comunicación entre el cliente y el servidor.

6.2.6. Integración Continua

Para asegurar la calidad del código y facilitar la entrega continua, se implementó un sistema de integración continua utilizando GitHub Actions (ver [2.2](#)). Este sistema automatiza la verificación del código, la ejecución de pruebas y la generación de imágenes de Docker. Se establecieron dos pipelines independientes para el frontend y el backend de la aplicación.

Ambos pipelines se inician en respuesta a eventos específicos, el push a las ramas `main` y `dev`, la creación de nuevos `tags`, o la publicación de releases. A continuación, se describen las fases clave del pipeline:

- CI: Se encarga de realizar verificaciones de salud en el código, ejecutar pruebas y garantizar el formato adecuado del mismo.

- CD: Se encarga de buildear y publicar las imágenes de Docker en Docker Hub.

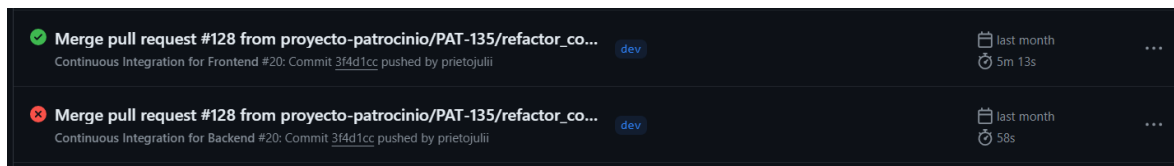


Figura 6.9: Implementación de Pipelines en Github Actions

En la figura 6.9, se observa cómo el pipeline del frontend se activó correctamente tras la fusión de la rama PAT-135 en la rama dev, mientras que el pipeline de la rama backend experimentó un fallo. Este enfoque resulta práctico e intuitivo, ya que GitHub proporciona una interfaz visual que permite verificar fácilmente el estado de los pipelines y revisar los registros en caso de errores.

Política de Contraseñas Seguras

Para fortalecer la seguridad de la plataforma, se ha establecido una política rigurosa con respecto a las contraseñas de los usuarios. Con el objetivo de proteger adecuadamente las cuentas, se requiere que las contraseñas cumplan con los siguientes criterios mínimos:

- Longitud mínima de 8 caracteres.
- Inclusión de letras minúsculas y mayúsculas.
- Uso de al menos un número.
- Inclusión de al menos un carácter especial.

Esta política se implementa durante la creación y modificación de contraseñas, asegurando que los usuarios establezcan credenciales robustas que contribuyan a la protección de los datos. La exigencia de contraseñas seguras mitiga posibles riesgos de seguridad y garantiza un entorno confiable para todos los usuarios de la plataforma.

Además de manera predeterminada, Django emplea el algoritmo **PBKDF2** con un hash **SHA256**, un estándar de extensión de contraseña recomendado por el **Instituto Nacional de Estándares y Tecnología (NIST)**. Esta elección proporciona un alto nivel de seguridad y resiste eficazmente los intentos de descifrado, ya que requiere considerables recursos computacionales para ser comprometido. Por lo tanto las contraseñas se almacenan de forma segura en la base de datos.

Autenticación Reforzada y Activación de Usuarios

Con el propósito de elevar aún más los estándares de seguridad, se implementaron medidas adicionales en el proceso de creación y acceso de usuarios:

- **Autenticación Via Email:** Durante el proceso de creación de usuarios, se ha incorporado un mecanismo de autenticación a través del correo electrónico. Los usuarios recibirán un enlace de verificación en su dirección de email registrado, el cual deben confirmar para completar el registro y activar su cuenta.
- **Activación Manual por el Administrador:** Una vez que los usuarios han completado la autenticación vía email, no podrán acceder a la plataforma de inmediato. Se ha implementado un sistema donde la activación final de las cuentas está sujeta a la aprobación manual por parte del administrador. Esto asegura un control riguroso sobre el acceso a la plataforma, permitiendo al administrador verificar y autorizar cada cuenta antes de su utilización.

6.2.7. Protección contra Ataques CSRF:

CSRF (ver [2.3](#)) es un tipo de ataque en el que un sitio malicioso realiza acciones no deseadas en un sitio web en el que un usuario está autenticado, utilizando las credenciales del usuario sin su conocimiento.

En este contexto, se implementó la protección CSRF utilizando el middleware ‘CsrfViewMiddleware’ proporcionado por Django, el framework utilizado para la API Rest. Un middleware en Django es un componente que procesa las solicitudes y las respuestas a medida que atraviesan la cadena de procesamiento de Django.

El middleware ‘CsrfViewMiddleware’ genera una cookie CSRF con un valor secreto aleatorio que otros sitios no pueden acceder y agrega un campo oculto llamado ‘csrfmiddlewaretoken’ a todos los formularios POST salientes.

El middleware verifica la presencia y corrección de la cookie CSRF y del campo ‘csrfmiddlewaretoken’ en todas las solicitudes entrantes que no utilizan los métodos seguros HTTP GET, HEAD, OPTIONS o TRACE. Si falta alguno de ellos, el usuario recibe un error 403.

Además, ‘CsrfViewMiddleware’ verifica la cabecera ‘Origin’ (si es proporcionada por el navegador) contra el host actual y la configuración CSRF_TRUSTED_ORIGINS, lo que brinda protección contra ataques entre subdominios.

6.2.8. Certificación de Seguridad y Cifrado SSL/TLS

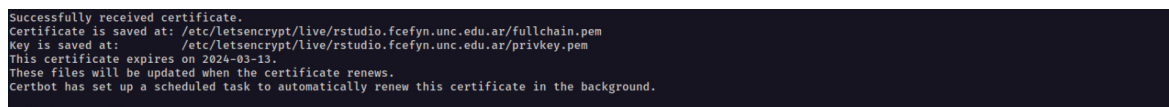
La seguridad y privacidad de las comunicaciones son elementos críticos para garantizar una conexión segura entre los usuarios y nuestro servidor, por lo que se implementó el cifrado SSL/TLS mediante certificados proporcionados por Let’s Encrypt, una autoridad de certificación que ofrece certificados gratuitos y automatiza el proceso de emisión y renovación. Esto, mitiga significativamente el riesgo de ataques de “Man-in-the-Middle” (Hombre en el Medio), garantizando que la información compartida entre el cliente y el servidor permanezca privada y no sea susceptible a manipulaciones indebidas.

Se decidió integrar Let's Encrypt directamente en el servidor y redireccionar las solicitudes HTTP a través de HTTPS utilizando un proxy en el servidor NGINX existente (ver C.).

La implementación de este proceso se llevó a cabo siguiendo el tutorial proporcionado por NGINX [8].

La Figura 6.10 muestra la salida del comando utilizado para solicitar y configurar el certificado, luego de haber configurado correctamente Nginx, mediante el siguiente comando:

```
sudo certbot --nginx -d proyecto-patrocinio.fcefyn.unc.edu.ar
```



```
Successfully received certificate.
Certificate is saved at: /etc/letsencrypt/live/rstudio.fcefyn.unc.edu.ar/fullchain.pem
Key is saved at: /etc/letsencrypt/live/rstudio.fcefyn.unc.edu.ar/privkey.pem
This certificate expires on 2024-03-13.
These files will be updated when the certificate renews.
Certbot has set up a scheduled task to automatically renew this certificate in the background.
Renewing certificate
```

Figura 6.10: Certificado Let's Encrypt obtenido con éxito

6.3. Diseño e Informe de Pruebas

El proceso de prueba de software tiene como objetivos primordiales demostrar que el programa cumple con los requerimientos y detectar situaciones donde su comportamiento sea incorrecto o no esté alineado con la especificación. Para lograr estos objetivos, se llevan a cabo pruebas utilizando datos artificiales y se verifica el correcto funcionamiento del software. Este proceso se centra en dos metas específicas: demostrar el cumplimiento de requerimientos tanto al desarrollador como al cliente, y encontrar y corregir situaciones donde el comportamiento del software no sea el esperado, eliminando así posibles defectos. (Somerville, 2011, P.206 [11])

6.3.1. Pruebas de Caja Blanca

Las pruebas de caja blanca implican la evaluación directa del código fuente para garantizar que la operación interna se ajuste a las especificaciones [5]. Mediante este enfoque, el ingeniero del software puede generar casos de prueba que abarquen todos los caminos independientes de cada módulo, exploren las decisiones lógicas en sus vertientes verdaderas y falsas, ejecuten bucles en sus límites operacionales y evalúen las estructuras internas de datos para asegurar su validez.

6.3.2. Pruebas de Caja Negra

Las pruebas de caja negra, también conocidas como pruebas funcionales o de entrada y salida, se ejecutan sobre la interfaz del software [5]. Estas pruebas examinan las funcionalidades del sistema sin considerar su comportamiento interno. Su objetivo

es demostrar que las funciones del sistema son operativas, que la entrada se acepta adecuadamente y que se produce una salida correcta.

6.3.3. Pruebas Unitarias

Son pruebas de caja blanca que se realizan con el objetivo de detectar errores de implementación en el componente desarrollado. Además, se identifican errores de entrada y salida de datos.

Backend: Pruebas Unitarias en Django

En el backend del sistema, se han empleado pruebas unitarias para evaluar la funcionalidad de cada módulo en el framework Django, utilizando la librería integrada en Django Rest Framework `rest_framework.test`. Estas pruebas se han organizado en un archivo `test.py` asociado a cada aplicación de Django.

Especialmente, se han enfocado en testear las vistas, asegurando que la lógica del servidor maneje adecuadamente las solicitudes y produzca las respuestas esperadas.

Se puede encontrar un ejemplo detallado de la implementación de pruebas unitarias en el Apéndice [F.1](#).

Frontend: Pruebas Unitarias en React

En el frontend, se han implementado pruebas unitarias utilizando la biblioteca Testing Library en conjunto con React. Estas pruebas se centran en evaluar la funcionalidad de los componentes de React, asegurando que la interfaz de usuario responda correctamente a las interacciones del usuario.

Se puede encontrar un ejemplo detallado de la implementación de pruebas unitarias en el Apéndice [F.2](#).

6.3.4. Diseño de Escenarios de Prueba

En la fase de diseño de pruebas, se proponen escenarios específicos que abarcan diversas situaciones para evaluar el software. Cada escenario se identifica con un ID único y se describe detalladamente para facilitar la ejecución y el análisis de las pruebas.

Estas pruebas, realizadas a nivel de sistema, adoptan un enfoque de caja negra. Se centran en examinar el comportamiento del sistema desde una perspectiva externa, sin acceder a los detalles internos de su implementación.

Se ha optado por utilizar el lenguaje Gherkin para expresar los escenarios de prueba. Gherkin es un lenguaje de formato sencillo que facilita la escritura de casos de prueba de manera comprensible tanto para técnicos como para no técnicos.

En cuanto a las pruebas automáticas, se implementan utilizando el framework de prueba Robot Framework en conjunto con Selenium. Robot Framework es una herramienta de automatización de pruebas de código abierto que permite escribir pruebas de aceptación de manera clara y legible. Selenium, por su parte, se utiliza para la automatización de navegadores web, facilitando la interacción con la interfaz de usuario y la validación del comportamiento del sistema en un entorno real.

Cuadro 6.1: Escenarios de Prueba propuestos.

ID	Escenario	Descripción
PAT-SYS-1	Registro de un Nuevo Usuario	<ul style="list-style-type: none"> ▪ Given Se accedió a la página “SignUp”. ▪ And Se completó el formulario con los datos del usuario. ▪ And Se aceptaron los términos y condiciones. ▪ When Se presiona el botón SignUp. ▪ Then Debería recibir un correo electrónico con el enlace de confirmación. ▪ And Debería ser redirigido a la página de inicio de sesión. ▪ And Debería recibir un error al intentar iniciar sesión. ▪ And En la base de datos debería existir el nuevo usuario registrado SIN ACTIVAR.
PAT-SYS-2	Activación de un Usuario Registrado	<ul style="list-style-type: none"> ▪ Given Existe un superusuario administrador. ▪ And Existe un usuario registrado sin activar. Existe un superusuario administrador ▪ And Se accedió a la plataforma como usuario “administrador”. ▪ And Se ingresó a la página de administración. ▪ And Se navegó a la pestaña “Usuarios”. ▪ When Se edita el estado del usuario “nuevo” a “Activo”. ▪ And Se desloguea de la página de administración. ▪ Then El usuario “nuevo” debería poder iniciar sesión en la plataforma con éxito.

ID	Escenario	Descripción
PAT-SYS-3	Visibilidad de Pestañas para Usuario Tomador de Caso.	<ul style="list-style-type: none"> ■ Given Existe un usuario registrado activo con permisos “common” y “case.taker” en la DB. ■ When Se accede a la plataforma como el usuario “Tomador de Caso”. ■ Then La pestaña “Consultoría” debería estar visible. ■ And Las pestañas “Consultas” y “Consultantes” del “Panel de Control” deberían estar visibles. ■ And La pestaña “Tableros” no debería estar visible.
PAT-SYS-4	Visibilidad de Pestañas para Usuario Profesor.	<ul style="list-style-type: none"> ■ Given Existe un usuario registrado activo con permisos “common” y “professor” en la DB. ■ When Se accede a la plataforma como el usuario “Profesor”. ■ Then La pestaña “Tableros” debería estar visible. ■ And La pestaña “Consultoría” no debería estar visible. ■ And Las pestañas “Consultas” y “Consultantes” del “Panel de Control” no deberían estar visibles.

ID	Escenario	Descripción
PAT-SYS-5	Creación y visualización de una consulta como usuario Tomador de Caso	<ul style="list-style-type: none"> ■ Given Existe un usuario registrado activo con permisos “common” y “case.taker” en la DB. ■ And Existe un consultante con DNI “11111111” en la base de datos. ■ And Ese accedió a la plataforma como usuario “Tomador de Caso”. ■ And Se navegó a la pestaña “Consultoría”. ■ When Se crea la consulta “Garantía” con Cliente “11111111”, oponente “Samsung” y descripcion “Dummy”. ■ Then La consulta “Garantía” para el consultante con DNI “11111111” debería existir en base de datos. ■ And El ticket “Garantía” debería estar visible en el panel de entrada del tablero “Consultoría”. ■ And La información de la consulta “Garantía” debería contener el consultante con DNI “11111111”. ■ And La información de la consulta “Garantía” debería contener el campo “Descripción” en “Dummy”. ■ And La información de la consulta “Garantía” debería contener el campo “Estado de progreso” en “Por hacer”. ■ And La información de la consulta “Garantía” debería contener el campo “Oponente” en “Samsung”. ■ And La información de la consulta “Garantía” debería contener el campo “Estado de disponibilidad” en “Creado, sin asignar”.

ID	Escenario	Descripción
PAT-SYS-6	Visualización de una consulta como usuario Profesor	<ul style="list-style-type: none"> ■ Given Existe el board “Comisión A1” en la DB. ■ And Existe un usuario registrado activo con permisos “common” y “professor” en la DB. ■ And El usuario profesor tiene acceso al board “Comisión A1”. ■ And Existe un consultante con DNI “11111111” en la base de datos. ■ And Existe un panel llamado “Panel A1” para el board de la comisión “Comisión A1” ■ And Existe un ticket para el panel, de la comisión, con tag, DNI del consultante, oponente, descripción y estado: <ul style="list-style-type: none"> ... Panel A1 ... Comisión A1 ... Garantía ... 11111111 ... Samsung ... Dummy ... TODO ■ And Se accedió a la plataforma como usuario “profesor”. ■ When Se navega a la pestaña “Board/Comisión A1” ■ Then El ticket “Garantía” debería estar visible en el panel de entrada del tablero “Comisión A1”. ■ And La información de la consulta “Garantía” debería contener el consultante con DNI “11111111”. ■ And La información de la consulta “Garantía” debería contener el campo “Descripción” en “Dummy”. ■ And La información de la consulta “Garantía” debería contener el campo “Oponente” en “Samsung”. ■ And La información de la consulta “Garantía” debería contener el campo “Estado de disponibilidad” en “Asignado”.

ID	Escenario	Descripción
PAT-SYS-7	Visualización del estado de una comisión	<ul style="list-style-type: none"> ■ Given Existe un usuario registrado activo con permisos “common” y “case.taker” en la DB. ■ And Se accedió a la plataforma como usuario “Tomador de Caso”. ■ And Existe el board “Comisión A1” en la DB. ■ And Existe un panel llamado “Panel A1” para el board de la comisión “Comisión A1”. ■ And Existe un ticket para el panel, de la comisión, con tag, DNI del consultante, oponente, descripción y estado: <ul style="list-style-type: none"> ... Panel A1 ... Comisión A1 ... Garantía1 ... 11111111 ... Samsung ... Dummy ... TODO ■ And Existe un ticket para el panel, de la comisión, con tag, DNI del consultante, oponente, descripción y estado: <ul style="list-style-type: none"> ... Panel A1 ... Comisión A1 ... Garantía2 ... 11111111 ... Samsung ... Dummy ... IN_PROGRESS ■ And Existe un ticket para el panel, de la comisión, con tag, DNI del consultante, oponente, descripción y estado: <ul style="list-style-type: none"> ... Panel A1 ... Comisión A1 ... Garantía3 ... 11111111 ... Samsung ... Dummy ... TODO ■ And Se navegó a la pestaña “Consultoría”. ■ When Se selecciona el botón de información del panel “Comisión A1”. ■ Then El Popper de la comisión debería contener “3 consultas totales”. ■ And El Popper de la comisión debería contener “2 consultas por hacer”. ■ And El Popper de la comisión debería contener “1 consultas en progreso”. ■ And El Popper de la comisión debería contener “0 consultas detenidas”.⁷³

ID	Escenario	Descripción
PAT-SYS-8	Creación de solicitud de asignación de caso a una comisión	<ul style="list-style-type: none"> ■ Given Existe un usuario registrado activo con permisos “common” y “case_taker” en la DB. ■ And Existe un consultante con DNI “11111111” en la base de datos. ■ And Se accedió a la plataforma como usuario “Tomador de Caso”. ■ And Existe el board “Comisión A1” en la DB. ■ And Existe un panel llamado “Panel A1” para el board de la comisión “Comisión A1”. ■ And Existe una consulta con tag, DNI del consultante, oponente, descripción y estado: <ul style="list-style-type: none"> ... Garantía ... 11111111 ... Samsung ... Dummy ... CREATED ■ And Se navegó a la pestaña “Consultoría”. ■ When Se crea la solicitud de asignación de consulta “Garantía” a la comisión “Comisión A1”. ■ Then Debería existir una “solicitud de consulta” de la consulta “Garantía” al board “Comisión A1” en la DB. ■ And El ticket “Garantía” debería estar en el primer panel “Comisión A1” de la comisión. ■ When Se elimina la solicitud de asignación de consulta “Garantía”. ■ Then debería haberse eliminado la “solicitud de consulta” de la consulta “Garantía” de la DB. ■ And el ticket “Garantía” debería estar en el panel de entrada “Consultas disponibles” de la comisión.

ID	Escenario	Descripción
PAT-SYS-9	Visualización y manipulación de la tabla en la ventana consultations	<ul style="list-style-type: none"> ■ Given Existe un usuario registrado activo con permisos “common” y “case.taker” en la DB. ■ And Existe un consultante con DNI “11111111” en la base de datos. ■ And Se accedió a la plataforma como usuario “Tomador de Caso”. ■ And Existe una consulta con tag, DNI del consultante, oponente, descripción y estado: <pre> ... Garantía1 11111111 Samsung ... Dummy TODO CREATED </pre> ■ And Existe una consulta con tag, DNI del consultante, oponente, descripción y estado: <pre> ... Garantía2 11111111 Samsung ... Dummy TODO CREATED </pre> ■ When Se navegó a la pestaña “Control Panel - Consultations” ■ Then La tabla debería contener 2 filas. ■ And La tabla debería contener la consulta: <pre> ... Garantía1 11111111 ... Samsung Dummy </pre> ■ And La tabla debería contener la consulta: <pre> ... Garantía2 11111111 ... Samsung Dummy </pre> ■ When Se descarga el csv de la tabla. ■ Then El archivo se debería haber descargado correctamente. ■ And El archivo de consultas descargado debería ser el esperado ‘expected_consultations.csv’ ■ When Se crea el filtro “Etiqueta” con “Garantía2”. ■ Then La tabla debería contener 1 filas. ■ And la tabla debería contener la consulta: <pre> ... Garantía2 ... 11111111 ... Samsung ... Dummy </pre> ■ When Se descarga el csv de la tabla. ■ Then El archivo se debería haber descargado correctamente. ■ And El archivo de consultas descargado debería ser el esperado ‘expected_filter_consultations.csv’

ID	Escenario	Descripción
PAT-SYS-10	Visualización de la ventana clients	<ul style="list-style-type: none"> ■ Given Existe un usuario registrado activo con permisos “common” y “case.taker” en la DB. ■ And Existe el consultante en la base de datos: <ul style="list-style-type: none"> ... Emily Davis DOCUMENT 11111111 ... FEMALE 1996-06-23 ... "Dummy Street 01" 1111 SINGLE ... HOUSE COMPLETE_UNIVERSITY ... emily96@gmail.com ■ And Existe el consultante en la base de datos: <ul style="list-style-type: none"> ... John Davis DOCUMENT 22145685 ... FEMALE 1980-06-25 ... "Dummy Street 01" 1111 SINGLE ... HOUSE COMPLETE_UNIVERSITY john80@gmail.com ■ And Se accedió a la plataforma como usuario “Tomador de Caso”. ■ When Se navegó a la pestaña “Panel de Control - Clientes”. ■ Then La tabla debería contener 2 filas. ■ And La tabla debería contener el consultante: <ul style="list-style-type: none"> ... 11111111 Emily Davis Document ... Female 1996-06-23 ... Dummy Street 01 1,111 Single House ... Complete University emily96@gmail.com ■ And la tabla debería contener el consultante: <ul style="list-style-type: none"> ... 22145685 John Davis Document Female ... 1980-06-25 Dummy Street 01 1,111 Single ... House Complete University john80@gmail.com ■ When Se descarga el csv de la tabla. ■ Then El archivo se debería haber descargado correctamente. ■ And El archivo de consultantes descargado debería ser el esperado 'expected_clients.csv'. ■ When Se crea el filtro “Nombre” con “Emily”. ■ Then La tabla debería contener 1 filas. ■ And La tabla debería contener el consultante: <ul style="list-style-type: none"> ... 11111111 Emily Davis Document ... Female 1996-06-23 Dummy Street 01 ... 1,111 Single House Complete University ... emily96@gmail.com ■ When Se descarga el csv de la tabla. ■ Then El archivo se debería haber descargado correctamente. ■ And El archivo de consultantes descargado debería ser el esperado 'expected_filter_clients.csv'.

ID	Escenario	Descripción
PAT-SYS-11	Aceptar y eliminar solicitudes de asignación de caso	<ul style="list-style-type: none"> ■ Given Existe el board “Comisión A1” en la DB. ■ And Existe un consultante con DNI “11111111” en la base de datos. ■ And Existe una consulta con tag, DNI del consultante, oponente, descripción y estado: <ul style="list-style-type: none"> ... Garantía1 ... 11111111 ... Samsung ... Dummy ... CREATED ■ And Existe una solicitud de asignación de la consulta “Garantía1” a la comisión “Comisión A1”. ■ And Existe una consulta con tag, DNI del consultante, oponente, descripción y estado: <ul style="list-style-type: none"> ... Garantía2 ... 11111111 ... Samsung ... Dummy ... CREATED ■ And Existe una solicitud de asignación de la consulta “Garantía2” a la comisión “Comisión A1”. ■ And Existe un panel llamado “Panel A1” para el board de la comisión “Comisión A1”. ■ And Existe un usuario registrado activo con permisos “common” y “professor” en la DB. ■ And El usuario profesor tiene acceso al board “Comisión A1”. ■ And Se accedió a la plataforma como usuario “Profesor”. ■ And Se navega a la pestaña “Board/Comisión A1”. ■ When Se acepta la solicitud de asignación de consulta “Garantía1” y se asigna al panel “Panel A1”. ■ Then Debería haberse cambiado la “solicitud” de la consulta “Garantía1” a “ACCEPTED” en la DB. ■ And El ticket “Garantía1” debería estar en el primer panel “Panel A1” del board. ■ When Se selecciona la opción rejected del menu del ticket “Garantía2”. ■ Then Debería haberse eliminado la “solicitud de consulta” de la consulta “Garantía2” de la DB. ■ And No debería existir el ticket “Garantía2” en el board.

ID	Escenario	Descripción
PAT-SYS-12	Creación, edición y eliminación de un comentario de una consulta	<ul style="list-style-type: none"> ■ Given Existe el board “Comisión A1” en la DB. ■ And Existe un usuario registrado activo con permisos “common” y “professor” en la DB. ■ And El usuario profesor tiene acceso al board “Comisión A1”. ■ And Existe un consultante con DNI “11111111” en la base de datos. ■ And Existe un panel llamado “Panel A1” para el board de la comisión “Comisión A1”. ■ And Existe un ticket para el panel, de la comisión, con tag, DNI del consultante, oponente, descripción y estado: <ul style="list-style-type: none"> ... Panel A1 ... Comisión A1 ... Divorcio ... 11111111 ... Samsung ... Dummy ... TODO ■ And Se accedió a la plataforma como usuario “profesor”. ■ And Se navega a la pestaña “Board/Comisión A1”. ■ When Se agrega el comentario “dummy” al ticket “Divorcio”. ■ Then La vista de comentarios de la consulta “Divorcio” debería contener “dummy”. ■ And El comentario “dummy” para la consulta “Divorcio” debería existir en la DB. ■ When Se edita el comentario “dummy” a “lore ipsum” al ticket “Divorcio”. ■ Then La vista de comentarios de la consulta “Divorcio” debería contener “lore ipsum”. ■ And El comentario “lore ipsum” para la consulta “Divorcio” debería existir en la DB. ■ And La vista de comentarios de la consulta “Divorcio” NO debería contener “dummy”. ■ And El comentario “dummy” para la consulta “Divorcio” NO debería existir en la DB. ■ When Se elimina el comentario “lore ipsum” al ticket “Divorcio”. ■ Then La vista de comentarios de la consulta “Divorcio” NO debería contener “lore ipsum” ■ And El comentario “lore ipsum” para la consulta “Divorcio” NO debería existir en la DB.

ID	Escenario	Descripción
PAT-SYS-13	Realizar cambios de una consulta como usuario Profesor	<ul style="list-style-type: none"> ■ Given Existe el board “Comisión A1” en la DB. ■ And Existe un usuario registrado activo con permisos “common” y “professor” en la DB. ■ And El usuario profesor tiene acceso al board “Comisión A1”. ■ And Existe un consultante con DNI “11111111” en la base de datos. ■ And Existe un panel llamado “Panel A1” para el board de la comisión “Comisión A1”. ■ And Existe un ticket para el panel, de la comisión, con tag, DNI del consultante, oponente, descripción y estado: <ul style="list-style-type: none"> ... Panel A1 ... Comisión A1 ... Divorcio ... 11111111 ... Samsung ... Dummy ... TODO ■ And se accedió a la plataforma como usuario “profesor”. ■ And se navega a la pestaña “Board/Comisión A1”. ■ When Se edita el campo “Descripción” a “otra descripcion” del ticket “Divorcio”. ■ And Se edita el campo “Oponente” a “otro oponente” del ticket “Divorcio”. ■ And Se edita el campo “Estado de progreso” seleccionando la opción “IN_PROGRESS” del ticket “Divorcio”. ■ And Se edita el campo “Etiqueta” a “CODE-123: Divorcio” del ticket “Divorcio”. ■ Then No debería existir el ticket “Divorcio” en el board. ■ And El ticket “CODE-123: Divorcio” debería estar visible en el panel de entrada del tablero “Comisión A1”. ■ And La información de la consulta “CODE-123: Divorcio” debería contener el campo “Etiqueta” en “CODE-123: Divorcio”. ■ And La información de la consulta “CODE-123: Divorcio” debería contener el consultante con DNI “11111111”. ■ And La información de la consulta “CODE-123: Divorcio” debería contener el campo “Descripción” en “otra descripcion”. ■ And La información de la consulta “CODE-123: Divorcio” debería contener el campo “Oponente” en “otro oponente”. ■ And La información de la consulta “CODE-123: Divorcio” debería contener el campo “Estado de disponibilidad” en “Asignado”.

ID	Escenario	Descripción
PAT-SYS-14	Creación y eliminación de eventos de una consulta	<ul style="list-style-type: none"> ■ Given Existe el board “Comisión A1” en la DB. ■ And Existe un usuario registrado activo con permisos “common” y “professor” en la DB. ■ And El usuario profesor tiene acceso al board “Comisión A1”. ■ And Existe un consultante con DNI “11111111” en la base de datos. ■ And Existe un panel llamado “Panel A1” para el board de la comisión “Comisión A1”. ■ And Existe un ticket para el panel, de la comisión, con tag, DNI del consultante, oponente, descripción y estado: <ul style="list-style-type: none"> ... Panel A1 ... Comisión A1 ... Divorcio ... 11111111 ... Samsung ... Dummy ... TODO ■ And Se accedió a la plataforma como usuario “profesor”. ■ And Se navega a la pestaña “Board/Comisión A1”. ■ When Se agrega el evento para hoy al ticket “Divorcio” titulado “Junta” con descripción “sucursal principal”. ■ Then La vista calendario de la consulta “Divorcio” debería contener el evento “Junta” el día de la fecha. ■ And El evento “Junta” hoy para la consulta “Divorcio” y descripción “sucursal principal” debería existir en la DB. ■ When Se elimina el evento “Junta” del ticket “Divorcio”. ■ Then La vista calendario de la consulta “Divorcio” NO debería contener el evento “Junta”. ■ And No debería existir el evento “Junta” para la consulta “Divorcio” en la DB.

ID	Escenario	Descripción
PAT-SYS-15	Creación edición y eliminación de un consultante como usuario Tomador de Caso	<ul style="list-style-type: none"> ■ Given Existe un usuario registrado activo con permisos “common” y “case.taker” en la DB. ■ And Se accedió a la plataforma como usuario “Tomador de Caso”. ■ And Se navegó a la pestaña “Panel de Control - Clientes”. ■ When Se crea un nuevo consultante “Dummy Client” con DNI “11111111”. ■ Then El consultante con DNI “11111111” debería existir en DB. ■ When Se edita el campo “partner.salary” a “123” del consultante con DNI “11111111”. ■ Then El campo “partner.salary” del consultante con DNI “11111111” debería ser “123” en DB. ■ When Se elimina el consultante con DNI “11111111”. ■ Then El consultante con DNI “11111111” NO debería existir la DB.
PAT-SYS-16	Calendario desactivado para una consulta sin asignar	<ul style="list-style-type: none"> ■ Given Existe un usuario registrado activo con permisos “common” y “case.taker” en la DB. ■ And Existe un consultante con DNI “11111111” en la base de datos. ■ And Se accedió a la plataforma como usuario “Tomador de Caso”. ■ And Existe una consulta con tag, DNI del consultante, opo- nente, descripción y estado: <ul style="list-style-type: none"> ... Garantía ... 11111111 ... Samsung ... Dummy ... CREATED ■ And Se navegó a la pestaña “Consultoría”. ■ When Se abre el detalle del ticket “Garantía”. ■ Then El botón calendario debería estar desactivado.

ID	Escenario	Descripción
PAT-SYS-17	Calendario desactivado para una consulta con solicitud pendiente de asignación	<ul style="list-style-type: none"> ■ Given Existe un usuario registrado activo con permisos “common” y “case.taker” en la DB. ■ And Existe un consultante con DNI “11111111” en la base de datos. ■ And Se accedió a la plataforma como usuario “Tomador de Caso”. ■ And Existe el board “Comisión A1” en la DB. ■ And Existe un panel llamado “Panel A1” para el board de la comisión “Comisión A1”. ■ And Existe una consulta con tag, DNI del consultante, opo- nente, descripción y estado: <ul style="list-style-type: none"> ... Garantía ... 11111111 ... Samsung ... Dummy ... CREATED ■ And Existe una solicitud de asignación de la consulta “Ga- rantía” a la comision “Comision A1”. ■ And Se navegó a la pestaña “Consultoría”. ■ When Se abre el detalle del ticket “Garantía”. ■ Then El boton calendario deberia estar desactivado.
PAT-SYS-18	[MANUAL]: Creación de un cliente por Google Forms	<ul style="list-style-type: none"> ■ Given Existe un token de sesión para un usuario con permisos de formulario en base de datos. ■ When Se envía un formulario de registro para el cliente “Dummy Client” con DNI “11111111” por API usando el token en el header. ■ Then El cliente con DNI “11111111” debería existir en base de datos.

ID	Escenario	Descripción
PAT-SYS-19	[MANUAL]: Creación de una consulta por Google Forms	<ul style="list-style-type: none"> ■ Given Existe un token de sesión para un usuario con permisos de formulario en base de datos. ■ And Existe el cliente con DNI “11111111” en base de datos. ■ When Se envía un formulario de consulta para el cliente con DNI “11111111” por API usando el token en el header. ■ Then La consulta del cliente con DNI “11111111” debería existir en base de datos.
PAT-SYS-20	[MANUAL]: Creación de un hijo por Google Forms	<ul style="list-style-type: none"> ■ Given Existe un token de sesión para un usuario con permisos de formulario en base de datos. ■ And Existe el cliente con DNI “11111111” en base de datos. ■ When Se envía un formulario de registro de hijos para el hijo con DNI “33333333” para el cliente con DNI “11111111” por API usando el token en el header. ■ Then El hijo con DNI “33333333” debería existir en base de datos.

7. Despliegue y Operación

7.1. Introducción

Este capítulo aborda el proceso integral de instalación y puesta en marcha del servicio, detallando el despliegue de la pila de contenedores en Swarm. Además, se proporciona una guía práctica sobre el uso efectivo de la misma.

7.2. Instalación de la Plataforma

Para iniciar el proceso de instalación, es necesario contar con el repositorio de deploy. Puede clonar el repositorio desde GitHub utilizando el siguiente comando:

```
$ git clone git@github.com:proyecto-patrocinio/proyecto-patrocinio.git
```

Una vez clonado el repositorio, es fundamental realizar la configuración previa de los archivos. A continuación, se proporciona una descripción detallada de los archivos de configuración del repositorio de deploy.

```
.
|-- .env
|-- docker-compose.yml
|-- dotenv.sh
|-- README.md
|-- resources
|   |-- backend.env
|   |-- frontend.env
|   |-- nginx.conf
|   |-- postgres.env
|   |-- templates
|   |   |-- account
|   |   |   |-- email_confirmation_message.html
|   |   |   |-- email_confirmation_signup_message.html
|   |   |-- notifications
|   |   |-- new_request.html
|   |   |-- request_accepted.html
|   |   |-- request_rejected.html
|   |-- terms_and_policies.md
```

7.2.1. Configuración de Templates

1. En la carpeta **templates/account**, se encuentran los siguientes templates:

- `email_confirmation_signup_message`: HTML enviado por email en la confirmación del registro de una cuenta.
- `email_confirmation_message`: HTML enviado por email cuando se solicita reenviar el email para el registro de usuario.
- `password_reset_key_message`: HTML enviado por email cuando se solicitó cambiar contraseña olvidada.

2. En la carpeta **templates/notifications**, se encuentran los siguientes templates:

- `new_request`: HTML enviado por email para notificar al usuario cuando la comisión tiene una nueva solicitud de asignación de caso.
- `request_accepted`: HTML enviado por email en la notificación al tomador de caso cuando una comisión aceptó una solicitud.
- `request_rejected`: HTML enviado por email en la notificación al tomador de caso cuando una comisión rechazó una solicitud.

7.2.2. Configuración de Variables de Entorno en el Archivo `.env`

En el archivo `.env`, se deben establecer las siguientes variables de entorno que serán rerenderizadas por el archivo compose `docker-compose.yml`. A continuación, se presenta una tabla con descripciones de cada variable:

Variable de Entorno	Descripción
CMS_BACKEND_IMAGE	Nombre de la imagen de Docker para el backend.
CMS_FRONTEND_IMAGE	Nombre de la imagen de Docker para el frontend.
CMS_PROXY_PORT	Puerto del servidor proxy Nginx.
CMS_NGINX_CONFIG_FILE	Ruta al archivo de configuración de NGINX.
CMS_BACKEND_ENV_FILE	Ruta al archivo de variables de entorno para el backend.
CMS_POSTGRES_ENV_FILE	Ruta al archivo de variables de entorno para PostgreSQL.
CMS_FRONTEND_ENV_FILE	Ruta al archivo de variables de entorno para el frontend.
CMS_LOGO_FILE	Ruta al archivo del icono de la plataforma.
CMS_TEMPLATES_ACCOUNT_PATH	Ruta al directorio de plantillas de “account”.
CMS_TEMPLATES_NOTIFICATION_PATH	Ruta al directorio de plantillas de “notifications”.
CMS_TERMS_AND_POLICIES_FILE	Ruta al archivo de términos y políticas.

Cuadro 7.1: Configuración de Variables de Entorno en el Archivo `.env`

Consulte el anexo para revisar las variables utilizadas: [E.1](#).

7.2.3. Configuración de Variables de Entorno en `backend.env`

En el archivo **backend.env**, se deben configurar las siguientes variables de entorno. A continuación, se presenta una tabla con descripciones de cada variable:

Variable de Entorno	Descripción
DEBUG	Modo de depuración (0 para desactivado, 1 para activado).
DJANGO_ALLOWED_HOSTS	Lista de hosts permitidos separados por espacios.
SQL_ENGINE	Motor de base de datos para Django. Por ejemplo para postgres es 'django.db.backends.postgresql'.
SQL_DATABASE	Nombre de la base de datos.
SQL_USER	Usuario de la base de datos.
SQL_PASSWORD	Contraseña de la base de datos.
SQL_HOST	Dirección del servidor de base de datos.
SQL_PORT	Puerto del servidor de base de datos.
DATABASE	Tipo de base de datos. En este caso es 'postgres'.
EMAIL_HOST_USER	Usuario del servidor de correo electrónico.
EMAIL_HOST_PASSWORD	Contraseña del servidor de correo electrónico.
CORS_ALLOWED_ORIGINS	Lista de orígenes permitidos para CORS.
HOSTNAME	Nombre del host de la aplicación.
CONSULTANCY_BOARD_NAME	Nombre de la comisión de consultoría.
DEFAULT_HTTP_PROTOCOL	Protocolo HTTP.
CSRF_TRUSTED_ORIGINS	Lista de orígenes confiables para CSRF.
LOG_ROTATE_DAYS	Días antes de rotar los archivos de log.
SECRET_KEY	Clave secreta de Django.
DJANGO_SUPERUSER_USERNAME	Nombre de usuario del superusuario administrador para acceder a la web admin de Django.
DJANGO_SUPERUSER_PASSWORD	Contraseña del superusuario administrador del proyecto.
DJANGO_SUPERUSER_EMAIL	Correo electrónico del superusuario administrador.

Cuadro 7.2: Configuración de Variables de Entorno en `backend.env`

Consulte el anexo para revisar las variables utilizadas: [E.2](#).

7.2.4. Configuración de Variables de Entorno en `frontend.env`

En el archivo `frontend.env`, se deben configurar las siguientes variables de entorno. A continuación, se presenta una tabla con descripciones de cada variable:

Variable de Entorno	Descripción
REACT_APP_URL_BASE_API _REST_PATROCINIO	URL base de la API REST de Patrocinio. Ejemplo <code>https://{{dominio}}/api/</code> .
REACT_APP_WS_NOTIFICATION _PATH_PATROCINIO	Ruta del servidor asincrónico para notificaciones. Ejemplo <code>wss://{{dominio}}/ws/notification/</code> .

Cuadro 7.3: Configuración de Variables de Entorno en `frontend.env`

El resto de variables de entorno de este archivo no deben tocarse, son específicas para acceder a los endpoints del backend.

Consulte el anexo para revisar las variables utilizadas: [E.3](#).

7.2.5. Configuración de Variables de Entorno para la Base de Datos

En el archivo `backend.env`, se deben configurar las siguientes variables de entorno relacionadas con la base de datos. A continuación, se presenta una tabla con descripciones de cada variable:

Variable de Entorno	Descripción
POSTGRES_DB	Nombre de la base de datos de Patrocinio en PostgreSQL.
POSTGRES_USER	Usuario de la base de datos de Patrocinio en PostgreSQL.
PGDATA	Ruta del directorio de datos de PostgreSQL.
POSTGRES_PASSWORD	Contraseña para el usuario de la base de datos de Patrocinio en PostgreSQL.

Cuadro 7.4: Configuración de Variables de Entorno para la Base de Datos

Consulte el anexo para revisar las variables utilizadas: [E.4](#).

7.2.6. Archivo de Configuración de Nginx

El archivo de configuración de Nginx puede obtener más información detallada en la sección correspondiente ([C.1](#)). Asegúrese de revisar esa sección para comprender y ajustar la configuración de Nginx según sea necesario para el correcto funcionamiento de la plataforma.

7.2.7. Archivo `terms_and_policies`

El archivo `terms_and_policies` debe configurarse según las políticas y términos que la plataforma desea establecer y que los usuarios deberán aceptar. Este archivo es crucial para definir las reglas y condiciones de uso de la plataforma.

7.3. Despliegue de la Plataforma

Para llevar a cabo el despliegue de la plataforma, se deben seguir los pasos detallados en el archivo README del repositorio de deploy.

Hasta la fecha actual, se dispone de un único nodo, que también actúa como nodo maestro, en el cual se ha implementado el despliegue mediante Docker Swarm. Este nodo maestro hospeda todos los servicios necesarios para la plataforma.

Inicialmente, el servidor ya contaba con un servidor Nginx en funcionamiento y un servicio de R Studio. Se procedió a integrar el servicio Case Management System en el servidor Nginx existente.

En la Figura 7.1 se presenta un diagrama que ilustra los servicios proporcionados por el servidor, destacando la composición del servicio Case Management System, definido a través de un Docker Stack.

Es importante tener en cuenta que si se decide agregar nodos adicionales al clúster de Docker Swarm, será necesario actualizar el controlador de volúmenes de Docker Swarm. Esto se realiza para permitir el intercambio de volúmenes entre los diferentes nodos del clúster.

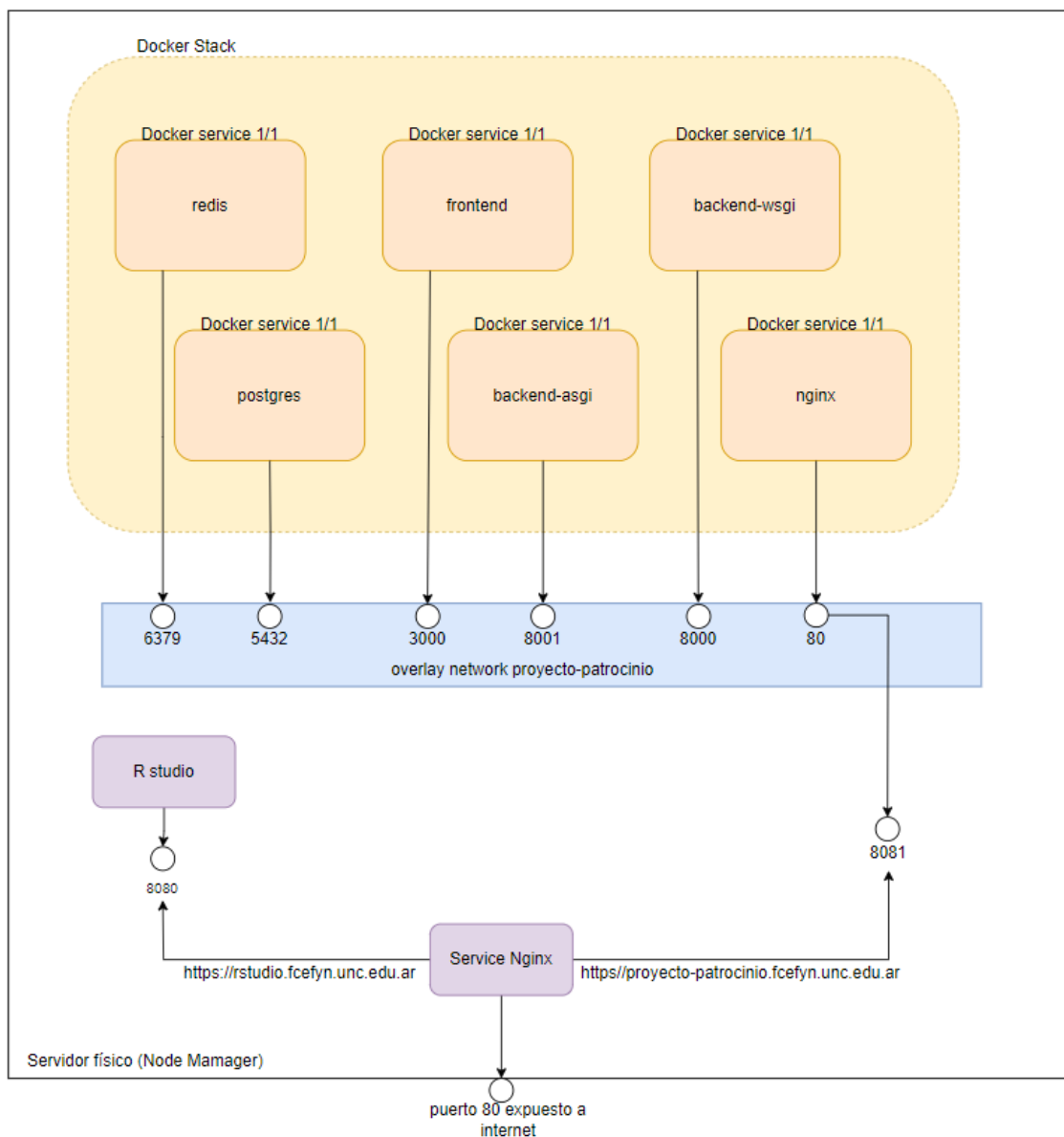


Figura 7.1: Diagrama de Despliegue

7.4. Configuración

Una vez que la plataforma esté en funcionamiento, el administrador debe acceder al panel web utilizando las credenciales proporcionadas en el archivo de configuración. A continuación, se describen los pasos para realizar la configuración inicial:

1. Inicie sesión en la plataforma como administrador.
2. Ingrese a la siguiente URL: <http://{{dominio}}/admin>.
3. Verifique la configuración del dominio en la sección de sitios (Figura 7.2). Es esencial asegurarse de que el dominio esté correctamente establecido. Si es nece-

sario realizar cambios, es **importante no crear nuevos sitios ni eliminar** el existente, sino editar la información del sitio existente.

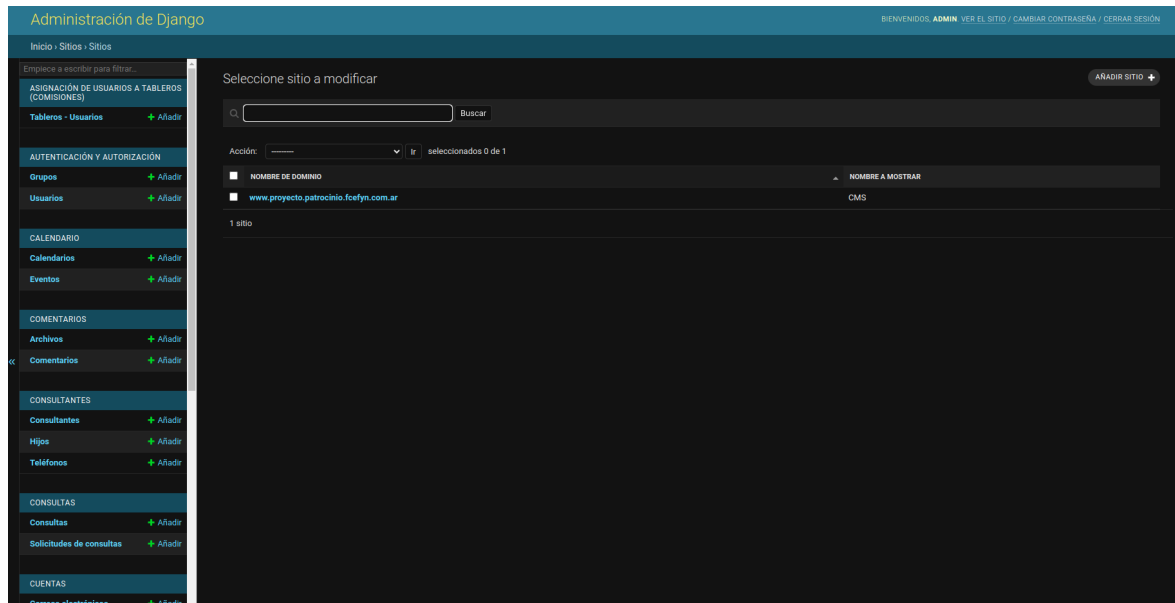


Figura 7.2: Configuración de Sitios en el Panel de Administración.

A continuación, el administrador debe crear todos los tableros de trabajo para cada comisión existente en el patrocinio a través de la sección de *Boards* (ver Figura 7.3).

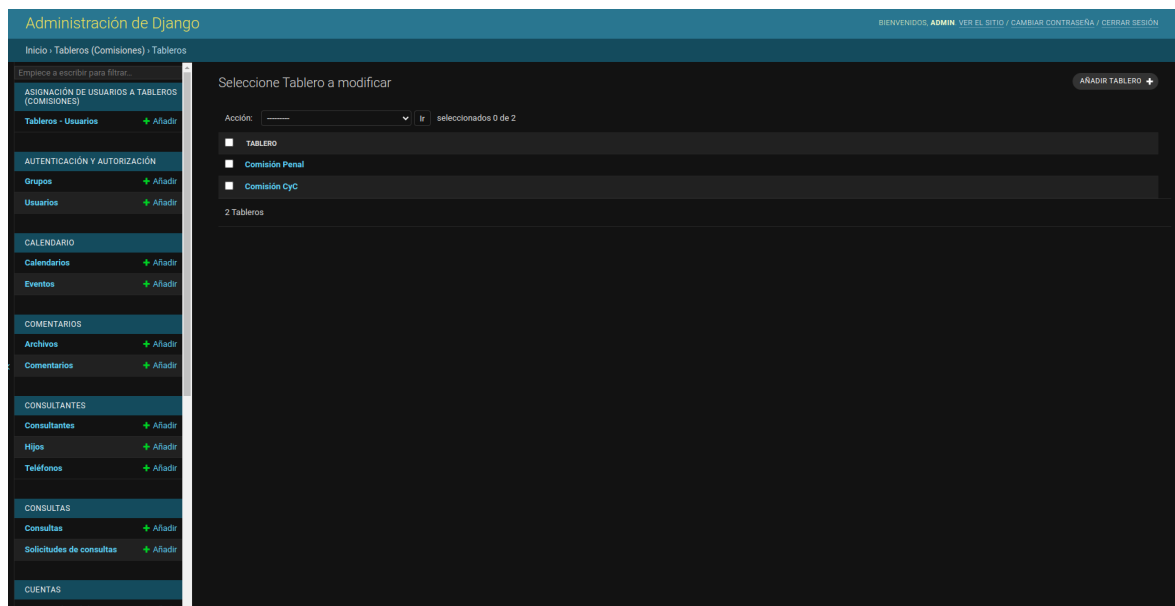


Figura 7.3: Creación de Pizarras en la Sección de Administración.


Finalmente, se deberá registrar una cuenta de usuario para la integración con

Google Forms y otorgarle permisos de “forms”. Para obtener información detallada sobre cómo realizar estos pasos, consulte la sección 7.5 para la creación de la cuenta y la sección 6.2.4 para la integración de la cuenta con Google Forms.

7.5. Registro de Usuarios

Para poder registrar un usuario, El usuario deberá seguir los siguientes pasos:

1. Ingresar a la página web y seleccionar la opción “¿No tienes una cuenta? Regístrate”.
2. Completar los datos requeridos, acepte los términos y condiciones, y haga clic en el botón “Registrarse”, como se muestra en la Figura 7.4.


Crear Cuenta

Nombre de usuario *

Correo electrónico *

Contraseña *

Repetir contraseña *

☐ He leído y acepto los [términos y condiciones](#).

REGISTRARSE

[¿Ya tienes una cuenta? Inicia sesión.](#)
[¿No recibiste el correo electrónico?](#)

[Sistema de Gestión de Casos](#) está licenciado bajo MIT © 2024

Figura 7.4: Pantalla de Registro de Usuarios.

3. Revisar la casilla de correo electrónico y seleccionar el enlace en el correo enviado (ver Figura 7.5).

Nota: El formato de este email, fue previamente configurado con los templates en la etapa de configuración de software.

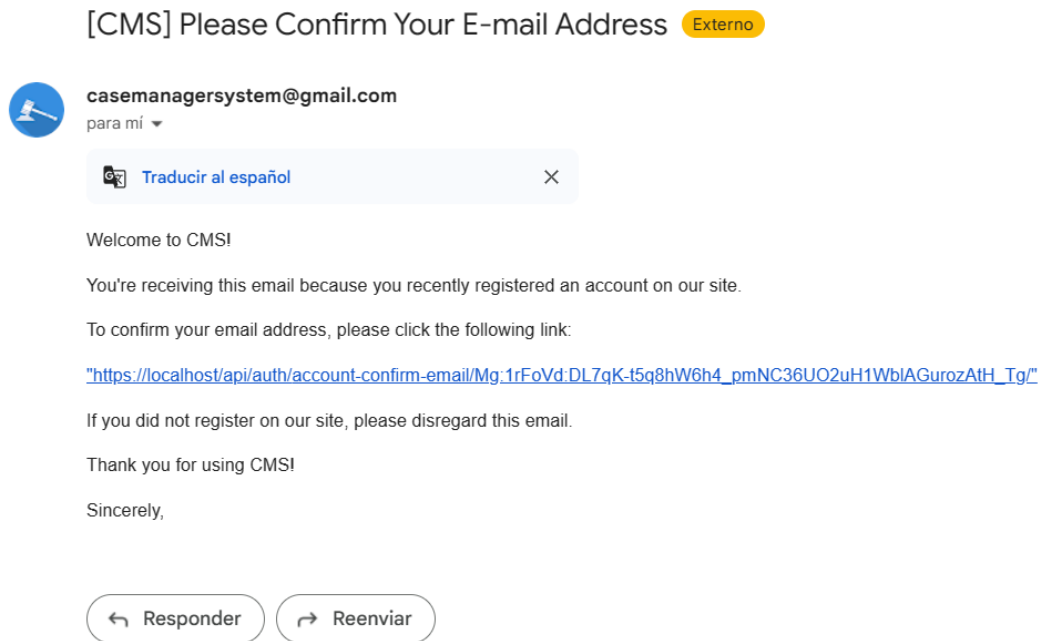


Figura 7.5: Confirmación por Correo Electrónico.

La página lo redirigirá a la principal, pero aún el usuario no podrá acceder hasta que un administrador habilite su cuenta. Para activar la cuenta, el administrador debe seguir estos pasos:

1. Ingresar a la sección de administración y dirigirse a la pestaña de usuarios.
2. Seleccionar el nuevo usuario y editar la sección de permisos, como se muestra en la figura 7.6.

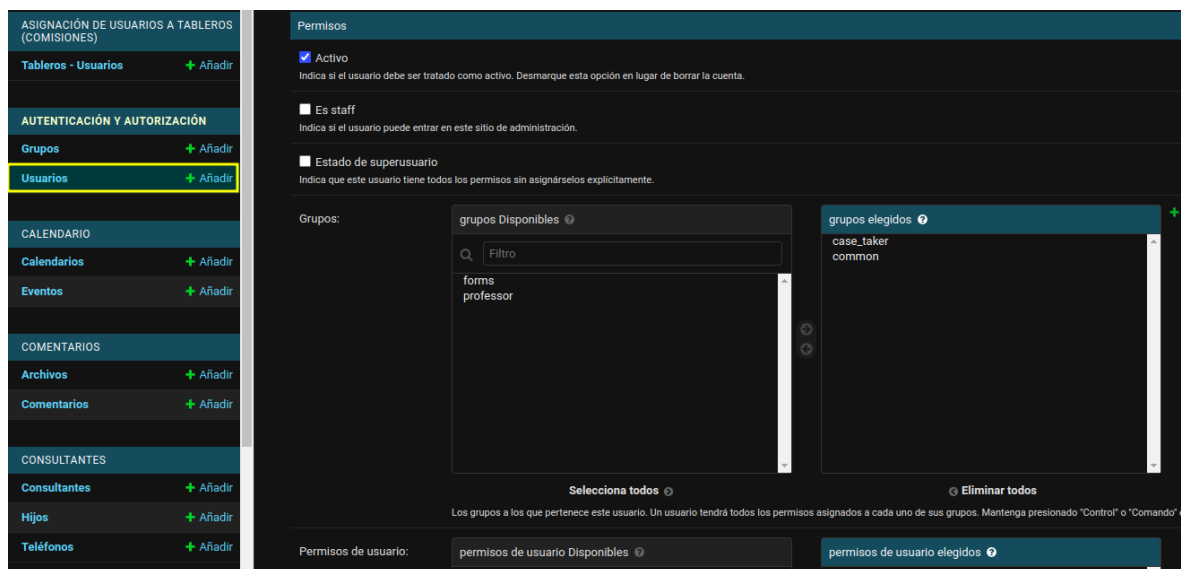


Figura 7.6: Configuración de Permisos de Usuario.

3. En la sección, marcar la opción “Activo” y otorgar los permisos necesarios (ver 7.7) antes de guardar.

Si el usuario es un miembro de una comisión, el administrador también deberá realizar los siguientes pasos adicionales.

1. Ingresar a la sección “Tableros - Usuarios”.
2. Crear la relación del usuario con la comisión correspondiente, como se observa en la figura 7.8.

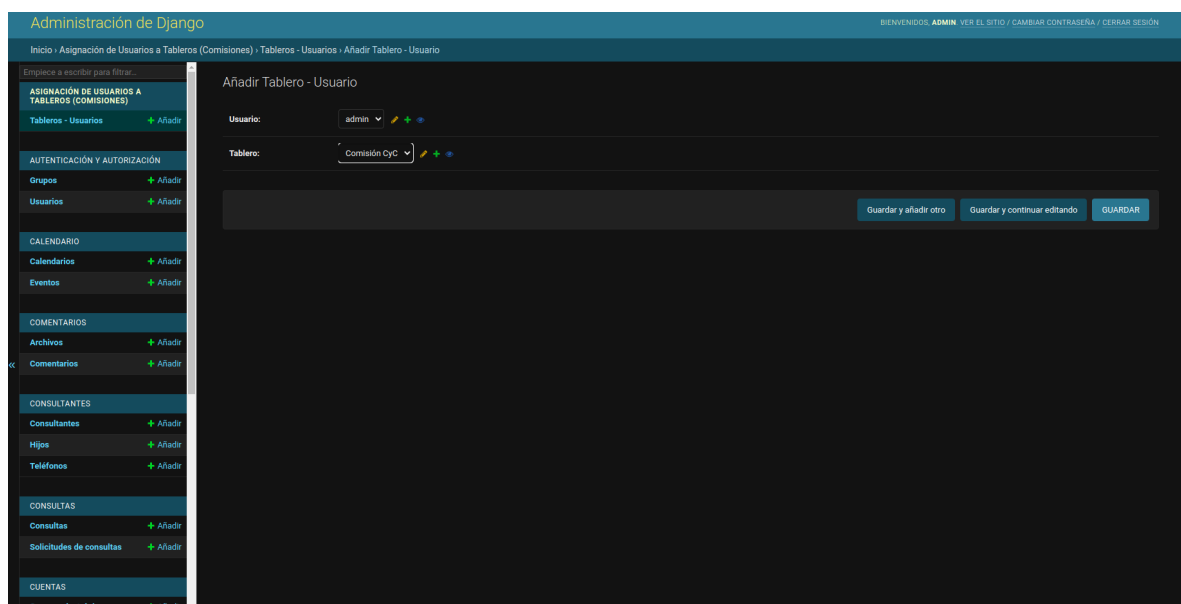


Figura 7.7: Creación de Relación Usuario-Comisión.

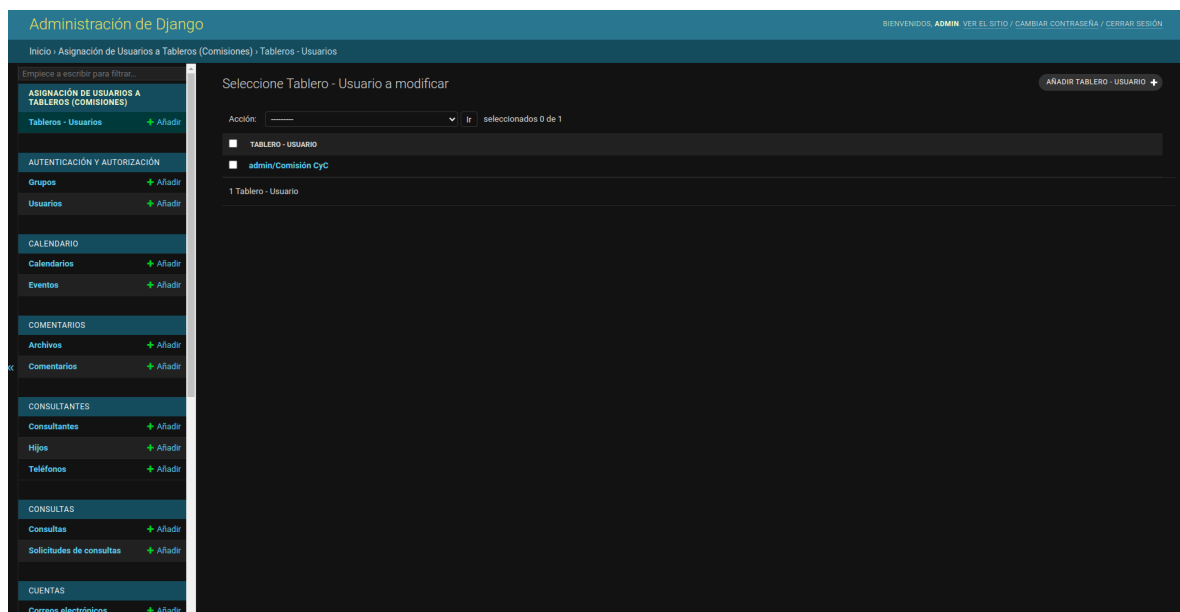



Figura 7.8: Lista de Relaciones Usuario-Comisión.

Después de completar estos pasos, el usuario podrá iniciar sesión correctamente en la web desde la página de inicio de sesión. La Figura 7.9 muestra la página de inicio de sesión.



Sistema de Gestión de Casos

[¿Olvidaste la contraseña?](#) [¿No tienes una cuenta? Regístrate](#)

Sistema de Gestión de Casos está licenciado bajo MIT © 2024

Figura 7.9: Página de Inicio de Sesión.

7.6. Permisos de Usuario

Los grupos de permisos de usuarios se encuentran en la sección "Grupos" de la página de administración y se detallan a continuación:

Grupo	Descripción
common	Grupo de permisos básicos necesarios tanto para un tomador de caso como para un profesor.
case_taker	Grupo de permisos para un tomador de caso. Necesarios para el manejo de la consultoría.
professor	Grupo de permisos para un profesor. Necesarios para el manejo de una comisión.
forms	Grupo de permisos exclusivo para la cuenta de Google Forms, que le permite registrar formularios.

Cuadro 7.5: Grupos de Permisos de Usuario.

7.7. Permisos de Usuario

Los grupos de permisos de usuarios se encuentran en la sección "Groups" de la página de administración y se detallan a continuación:

Grupo	Descripción
common	Grupo de permisos básicos necesarios tanto para un tomador de caso como para un profesor.
case_taker	Grupo de permisos para un tomador de caso. Necesarios para el manejo de la consultoría.
professor	Grupo de permisos para un profesor. Necesarios para el manejo de una comisión.
forms	Grupo de permisos exclusivo para la cuenta de Google Forms, que le permite registrar formularios.

Cuadro 7.6: Grupos de Permisos de Usuario.

Nota: El grupo de permisos *common* debe asignarse junto con *case_taker* o *professor*.

7.8. Administración

La página de administración (ver Figura 7.10) proporciona una interfaz amigable para visualizar y gestionar todos los datos del sistema. Al ingresar a la página principal, se presenta una disposición organizada de los datos agrupados en secciones. Además, a la derecha, se muestra un historial de acciones recientes.

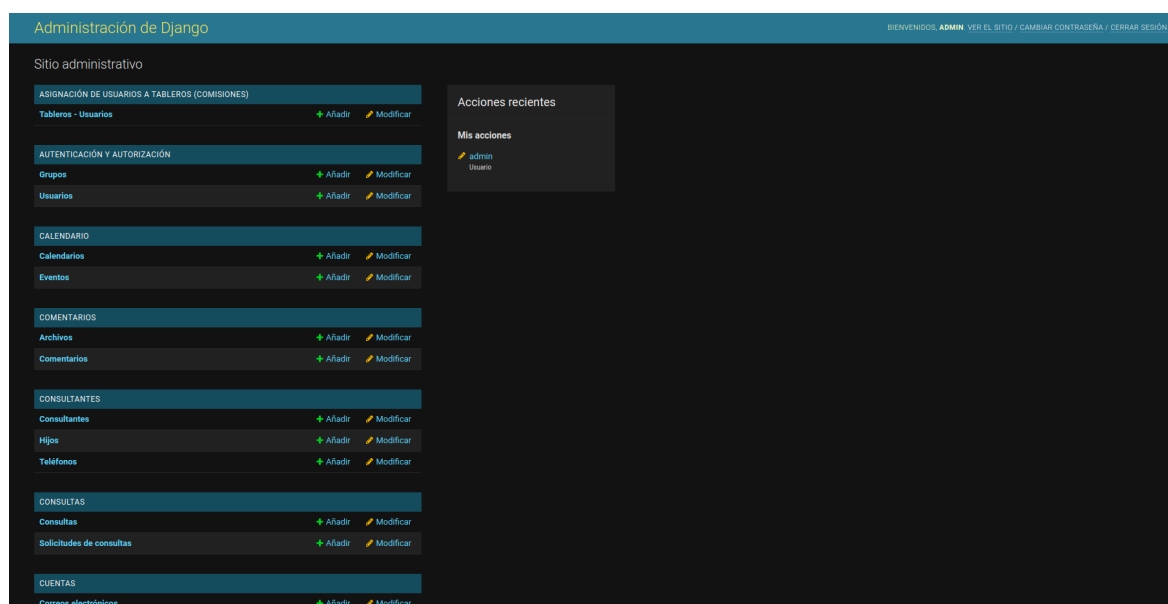


Figura 7.10: Interfaz de Administración de Django.

En esta interfaz, los administradores pueden realizar diversas acciones, como agregar, editar o eliminar registros, administrar permisos de usuario y conocer el estado del sistema.

7.9. Tablero de Trabajo de Consultoría

La página de consultoría (ver Figura 7.11) es responsable de la administración de las consultas. Aquí, los tomadores de casos pueden revisar las consultas entrantes, representadas como tickets generados a través de formularios de Google. Pueden analizarlas y verificar si cumplen con las condiciones para ser patrocinadas por la entidad. Los tomadores de casos tienen la capacidad de crear, editar y eliminar consultas o enviarlas a una comisión para solicitar patrocinio.

La página está estructurada mediante paneles. El panel izquierdo contiene las consultas sin asignar, mientras que los paneles siguientes representan comisiones, cada uno con los tickets de las solicitudes de asignación. Para asignar una consulta a una comisión, basta con arrastrar la consulta al tablero de la comisión deseada. También es posible eliminar la solicitud volviendo a arrastrar el ticket al panel izquierdo. Para eliminar el ticket, simplemente posicione el mouse sobre el mismo, seleccione el menú que aparecerá y elija la opción “eliminar”.

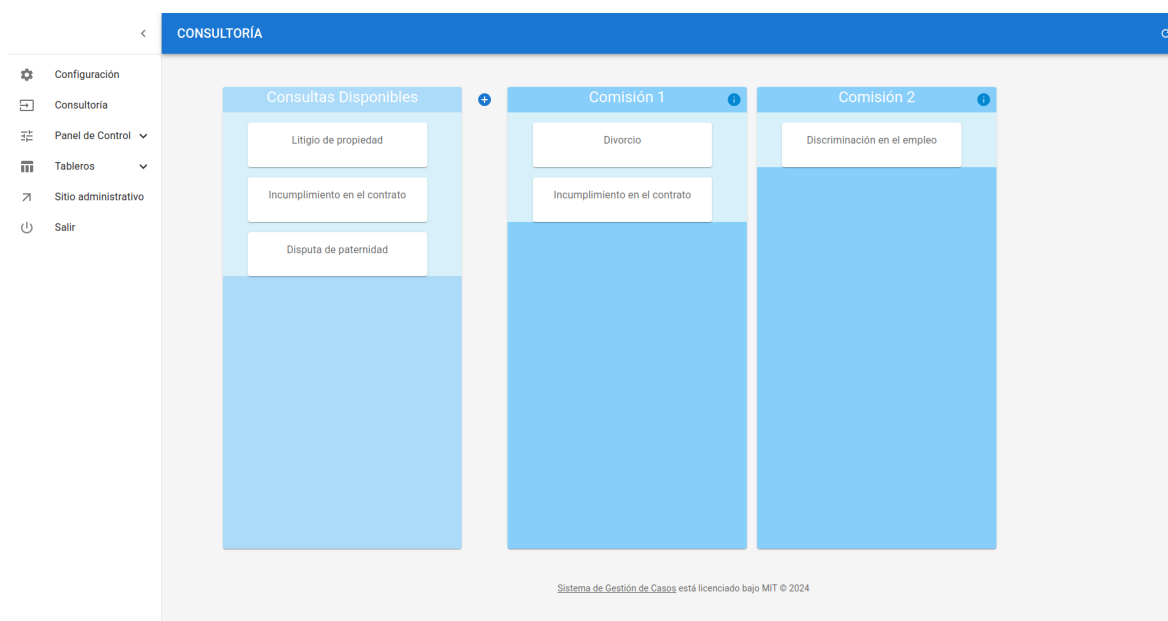


Figura 7.11: Página de Consultoría

Es importante mantener un registro del estado de cada comisión. Cada panel cuenta con un popper en el sector superior derecho que, al hacer clic, muestra la cantidad de consultas asignadas y la cantidad de consultas por estado (por hacer, en progreso o pausadas/bloqueadas). Además, como se ve en la figura 7.12, incluye un

historial de las asignaciones de los últimos 10 días, que muestra la “etiqueta” de la consulta y la fecha de asignación.

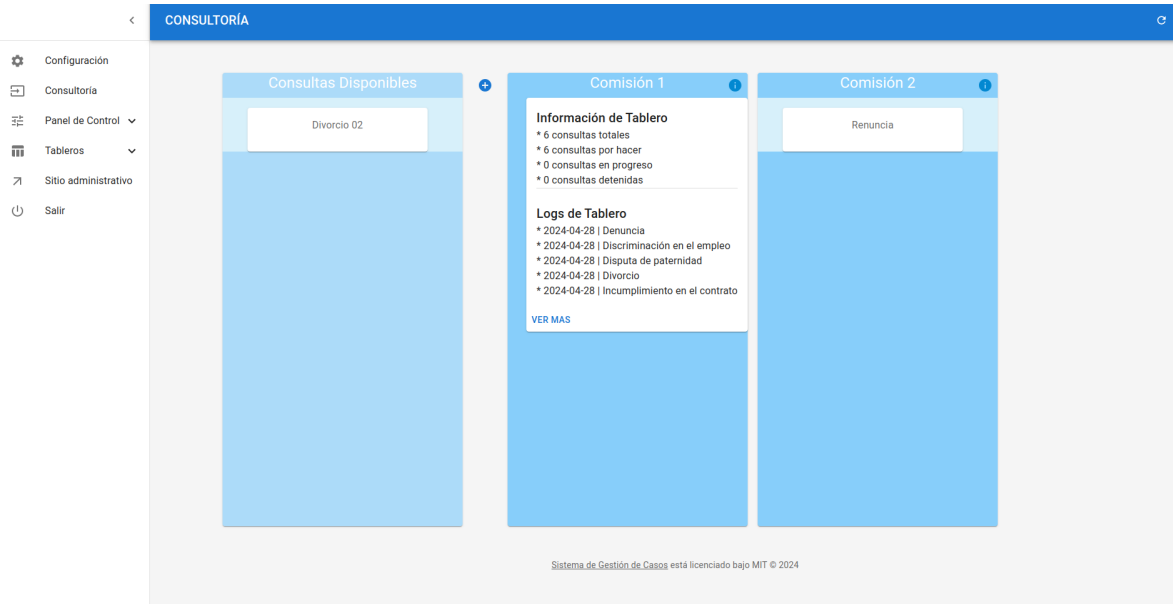


Figura 7.12: Historial de Asignaciones de una Comisión

Cuando la cantidad de consultas asignadas a la comisión en los últimos 10 días es extensa, aparecerá la opción “ver mas”, que expandirá un panel con todas las asignaciones (ver Figura 7.13).

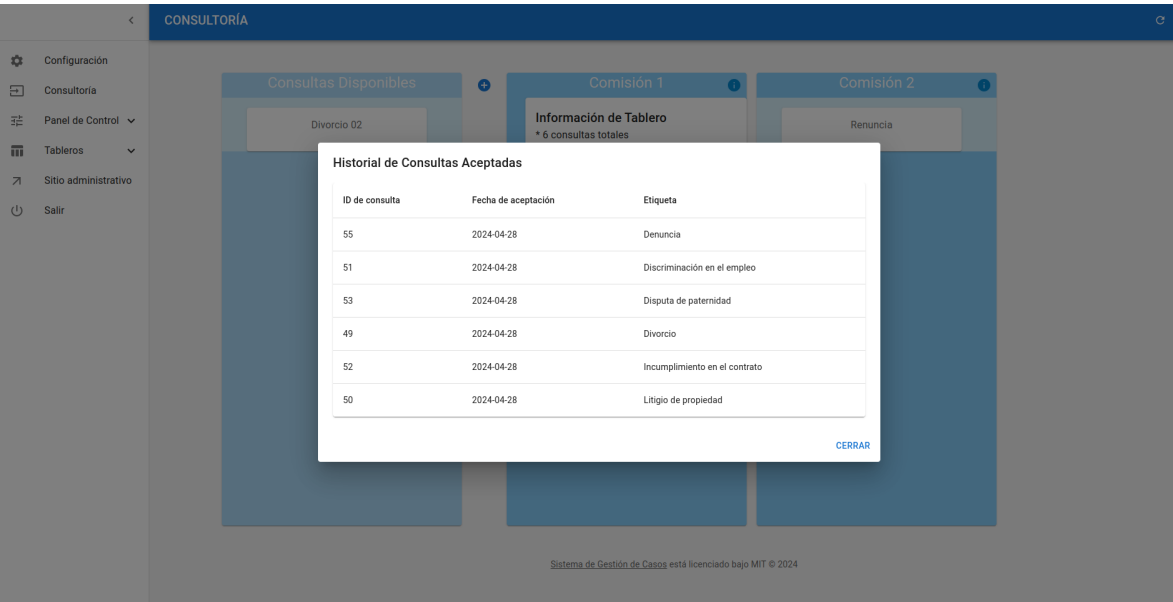


Figura 7.13: Panel Expandido de Historial de Asignaciones de una Comisión

El Usuario Tomador de Caso tiene la capacidad de crear consultas a través de la

página de consultoría. Para hacerlo, se selecciona el botón con el símbolo “+” ubicado en el panel de entrada, y luego se completa el formulario como se muestra en la figura 7.14.

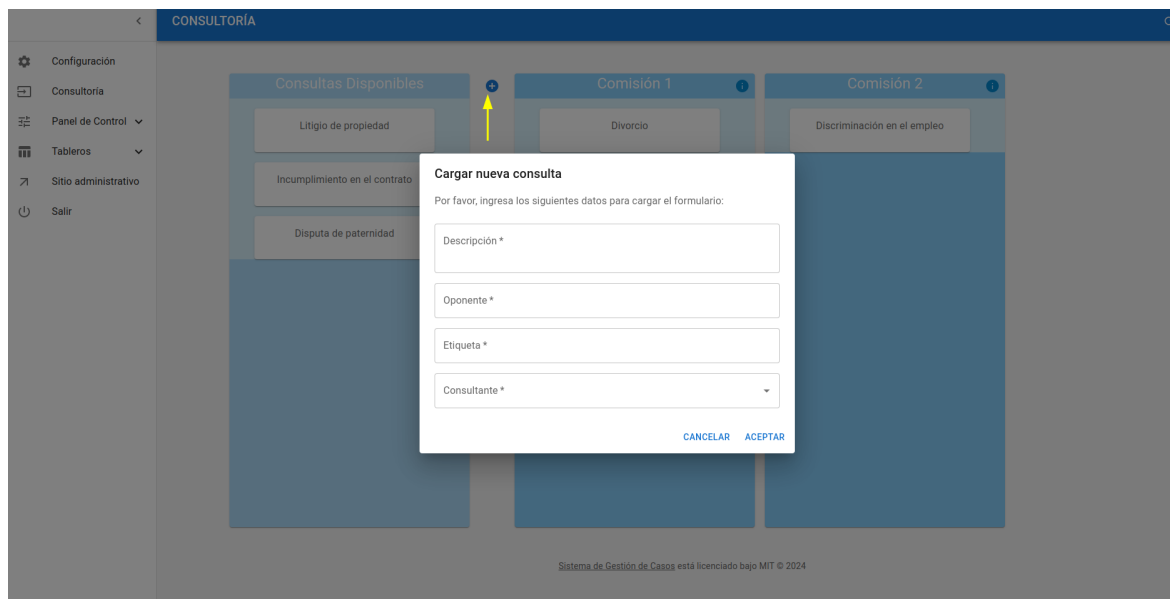


Figura 7.14: Formulario para Crear Consulta en la Página de Consultoría

7.10. Panel de Control

El panel de control (ver Figura 7.15) consta de dos pestañas principales: “Consultas” y “Consultantes”. Ambas pestañas ofrecen tablas que pueden descargarse en formato PDF o CSV. Además, proporcionan opciones de filtro y permiten la edición, creación o eliminación de registros.

A continuación, se presenta la pestaña de “Consultations”, que muestra una tabla con todas las consultas.

ID	Estado de disponibilidad	Estado de progreso	Creación	Descripción	Oponente	Etiqueta	Consultante	Acciones
7	Rechazado sin asi...	Incompleto	4/27/2024, 11:46:22 PM	Mi empresa está enfrenta...	El bar 123	Discriminación en ...	20157486	
1	Rechazado sin asi...	Incompleto	4/27/2024, 9:19:11 PM	Mi pareja no acepta el divo...	Claudio Pereyra	Divorcio 01	43601458	
4	Rechazado sin asi...	Incompleto	4/27/2024, 11:14:35 PM	La compañía de seguros e...	compañía de segu...	Lesiones personal...	43601458	
3	Rechazado sin asi...	Incompleto	4/27/2024, 11:13:50 PM	Mi vecino está disputando...	Lucas Liendo	Litigio de propiedad	20157486	
2	Solicitud de Asign...	Incompleto	4/27/2024, 9:19:58 PM	Quiero hacer un juicio a mi...	OTTO	Denuncia por Trab...	43601458	
6	Solicitud de Asign...	Incompleto	4/27/2024, 11:16:12 PM	El supuesto padre biológic...	Rogelio Perez	Disputa de paterni...	43601458	
5	Solicitud de Asign...	Incompleto	4/27/2024, 11:15:29 PM	La empresa con la que ten...	Claro	Incumplimiento de...	20157486	

Rows per page: 100 1-7 of 7

Sistema de Gestión de Casos está licenciado bajo MIT © 2024

Figura 7.15: Tabla de Consultas en el Panel de Control.

En la figura 7.16, se presenta la pestaña “Consultantes”, que muestra una tabla con todos los consultantes.

ID	Código P.	Dirección	Estado Civil	Vivienda	Estudios	Email	Tipo de doc.	Num. de doc.	Nombre	Apellido
2	2,000	Chilaver 123	Casado/a	Departamento	Secundario completo	juan@gmail.com	DNI	20157486	Juan	Cantero
1	5,012	San Martín 12	Soltero/a	Casa	Terciario incompleto	martina@gmail.com	DNI	43601458	Martina	Perez

1 row selected Rows per page: 100 1-2 of 2

Sistema de Gestión de Casos está licenciado bajo MIT © 2024

Figura 7.16: Tabla de Clientes en el Panel de Control.

Cada entrada en esta tabla puede ser eliminada o editada seleccionando los elementos en la última columna a la derecha de la entrada deseada. Además, es posible crear un nuevo registro haciendo clic en el botón “Agregar Registro” en la parte superior.

La tabla también permite aplicar filtros a algunas de sus columnas (ver Figura 7.17). Para ello, cada columna tiene un tooltip que facilita la ordenación ascendente o descendente, filtrado y ocultamiento de la columna. A continuación, se muestra un ejemplo de aplicación de un filtro a la columna “estado de progreso” de la tabla “consultas”.

ID	Estado de ...	Estado de progreso	Creación	Descripción	Oponente	Etiqueta	Consultante	Acciones
6	Asignado	Incompleto	4/27/2024, 11:16:12 PM	El supuesto padre biológic...	Rogelio Perez	Disputa de paterni...	43601458	[Edit] [Delete]

Figura 7.17: Aplicación de un Filtro en la Columna Progress State

7.11. Tablero de Trabajo para la Comisión

Cada comisión cuenta con un “Tablero” (ver Figura 7.18), el cual los usuarios con acceso podrán visualizar desde la sección “Tableros” en el menú desplegable de la página.

Este espacio está organizado en paneles, siendo el primero a la izquierda el panel de entrada de solicitudes de asignación de casos. Para aceptar una solicitud, basta con arrastrar el ticket de la consulta a alguno de los paneles internos del board. Para rechazarla, se puede seleccionar la opción “rechazado” desde el menú del ticket, el cual se hace visible acercando el ratón sobre él.

Los paneles a la izquierda son flexibles y pueden ser creados según la necesidad del usuario, ya sea como organizadores, separadores de consultas, o para cualquier otro agrupamiento conveniente. Por ejemplo, podrían ser utilizados un panel por profesor, otro por estado de progreso de las consultas, o cualquier otro criterio de agrupación preferido.

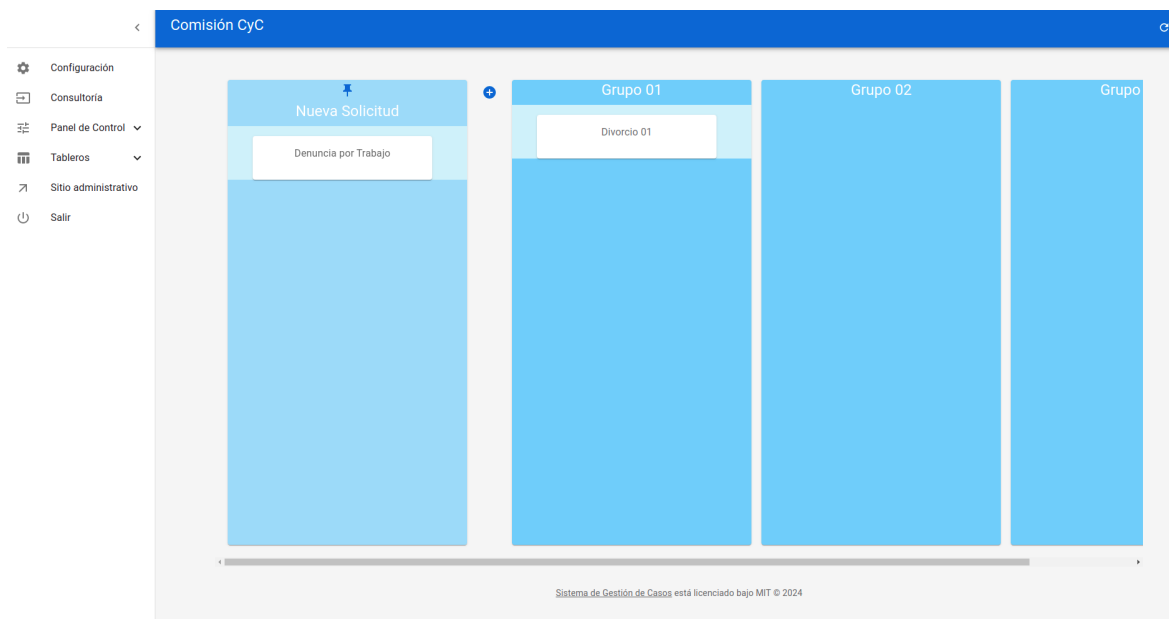


Figura 7.18: Página Board de la Comisión

Cada panel se puede crear utilizando el botón “+” e ingresando el nombre correspondiente. Asimismo, se pueden eliminar a través del menú del panel o editar el título haciendo doble clic sobre él (siempre y cuando no existan tickets en el tablero). Además, el board puede ser renombrado haciendo doble clic sobre el título.

La Figura 7.19 muestra los botones para eliminar o crear un panel del board.

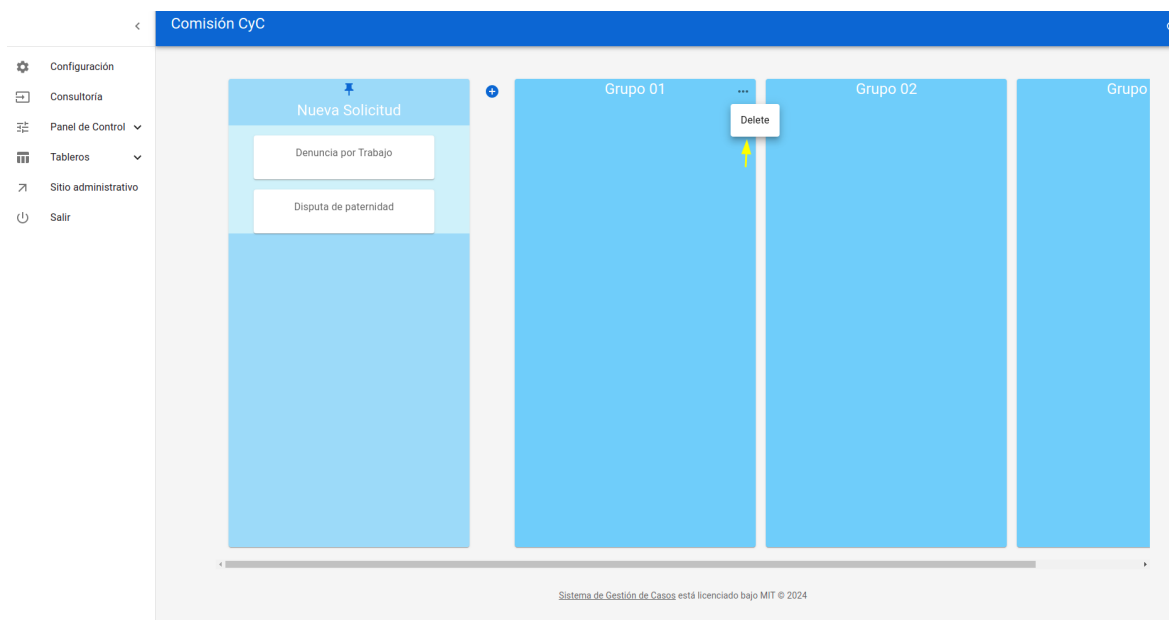


Figura 7.19: Botones para eliminar o crear un Panel del Board

7.12. Detalles de la Consulta

Para obtener más detalles sobre una consulta específica, se debe hacer click en la tarjeta correspondiente, lo que desplegará un cuadro con tres pestañas distintas, como se ve en la figura 7.20.

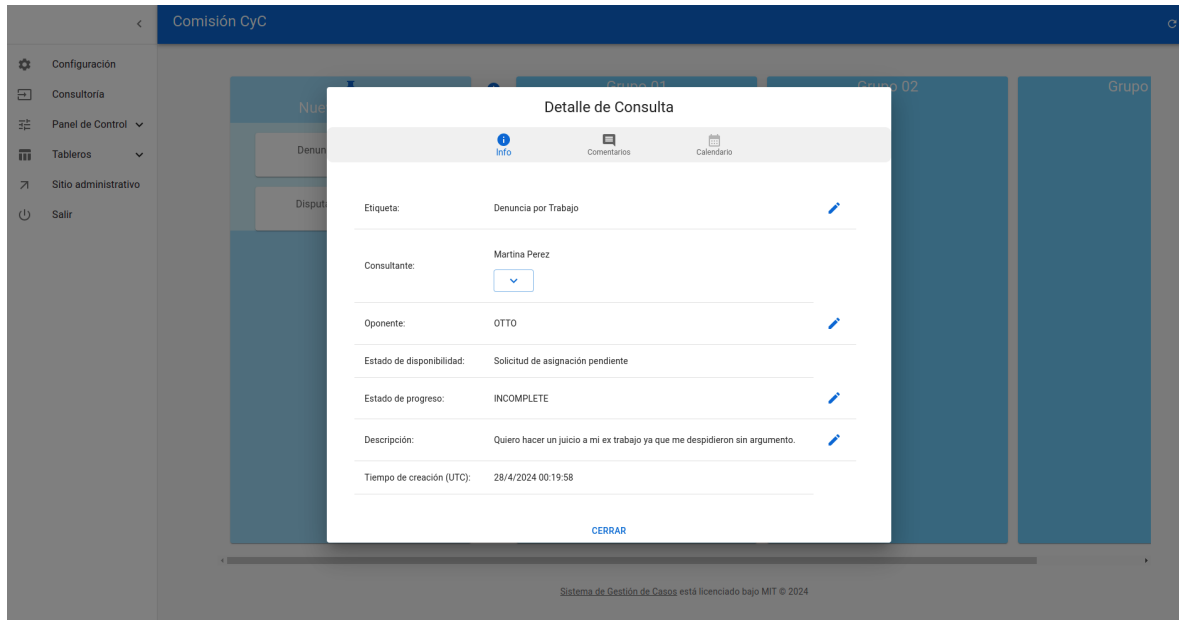


Figura 7.20: Ventana de Información de Consulta.

En la figura 7.21, muestra la primera pestaña, la cual presenta información organizada sobre el consultante y los detalles de la consulta.

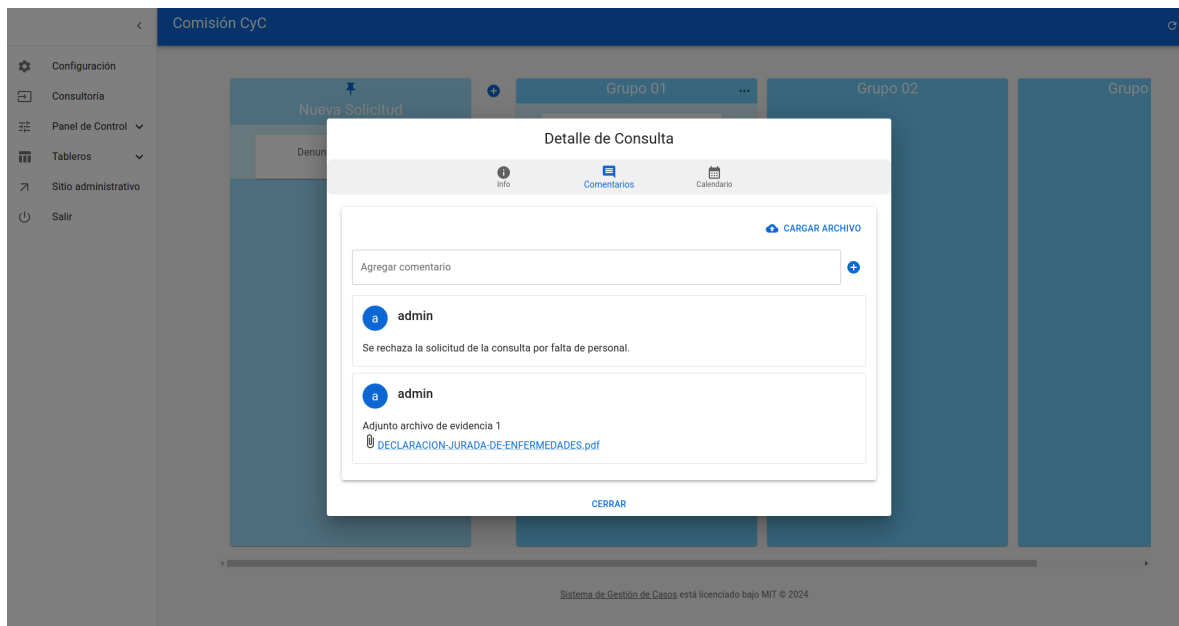


Figura 7.21: Sección de Comentarios y Archivos.

En la figura 7.22, muestra la segunda pestaña. Esta es una sección dedicada para registrar comentarios y archivos relevantes que contribuyan al seguimiento del caso.

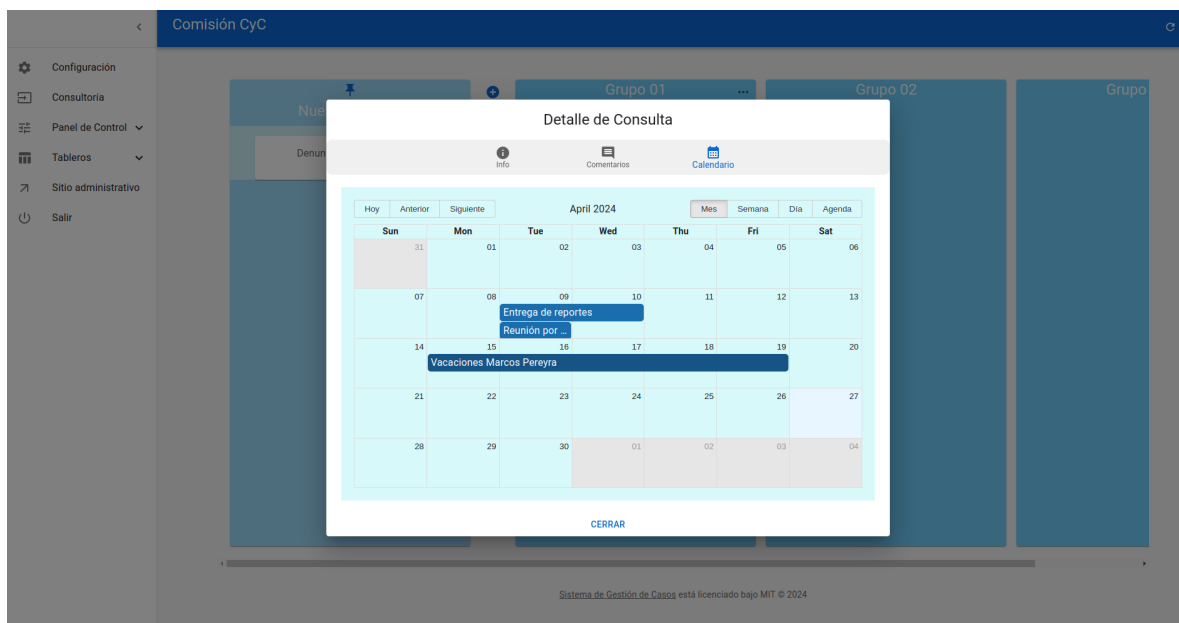


Figura 7.22: Calendario de Consulta.

En las figuras 7.23 y 7.24, se observa la tercera ventana, que es un calendario que permite registrar eventos relacionados con el caso, como reuniones o fechas importantes.

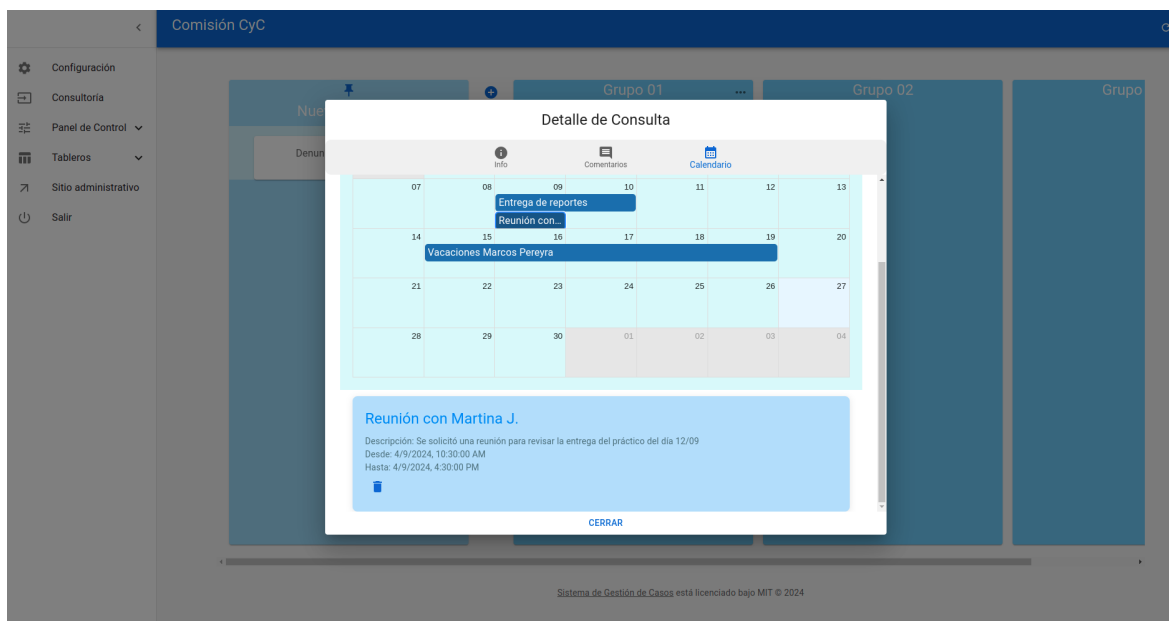


Figura 7.23: Detalle de Evento del Calendario de Consulta.

Al seleccionar un evento en el calendario, se desplegará una vista detallada con información adicional.

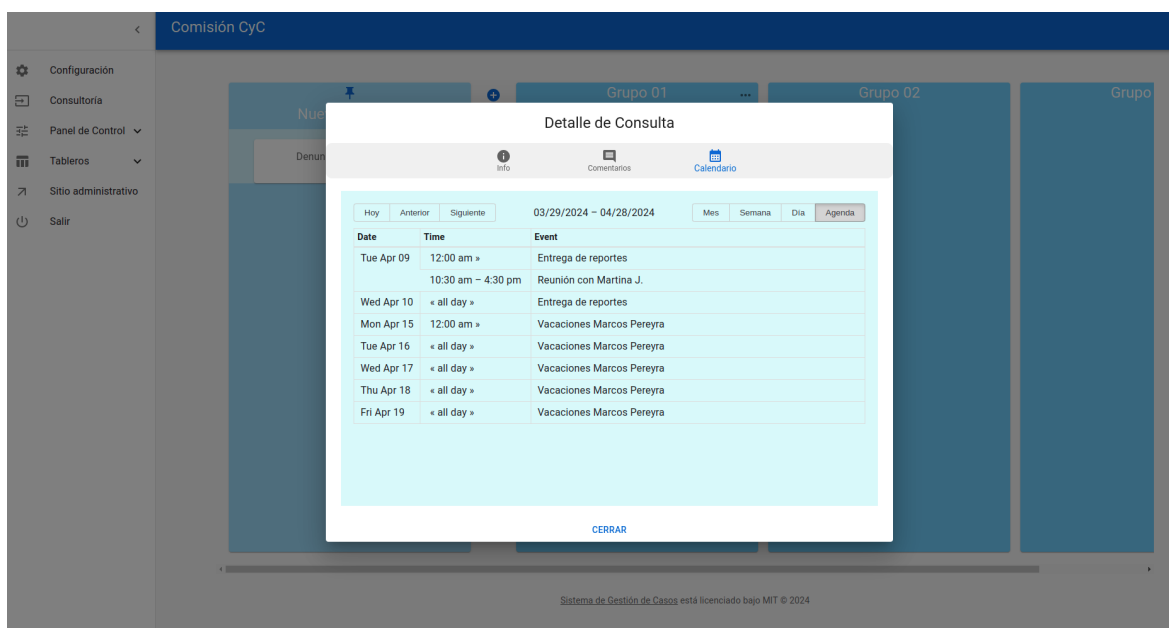


Figura 7.24: Vista Agenda de Calendario de Consulta.

Este calendario ofrece varias vistas, como mensual, semanal, diaria o en formato de agenda, según se visualiza en la imagen anterior. Es una herramienta clave para organizar y dar seguimiento a eventos relevantes asociados a la consulta.

7.13. Configuración de Cuenta

La sección de configuración de cuenta (ver Figura 7.25), accesible desde la página de “Settings”, proporciona la capacidad de visualizar y editar información personalizada, como el nombre de usuario y la contraseña.

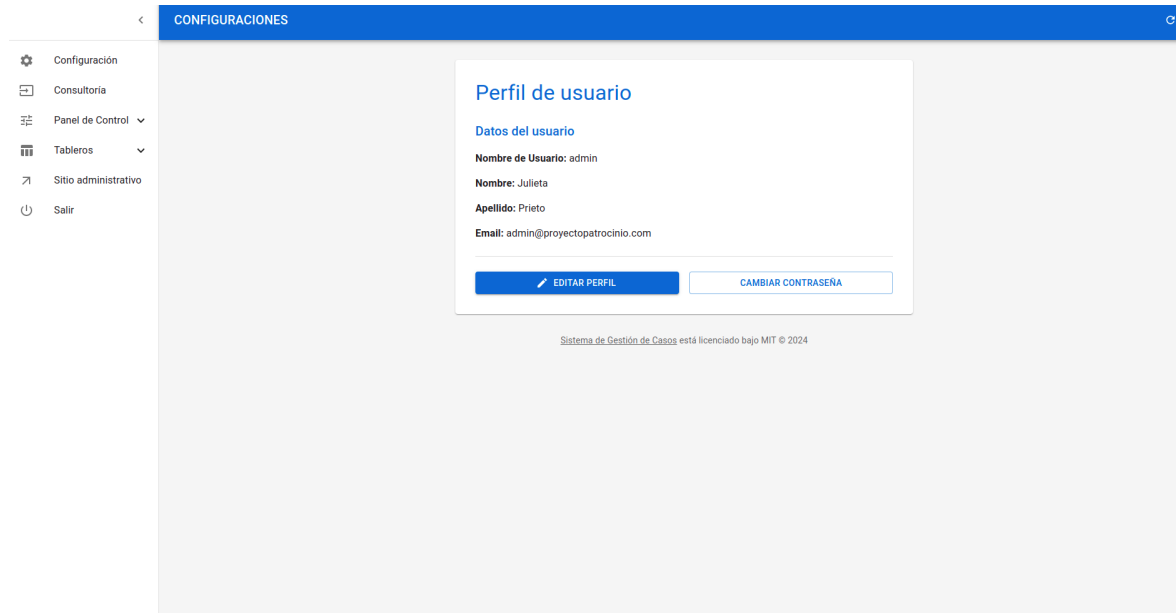


Figura 7.25: Página de Configuración de Cuenta.

8. Trabajos Futuros

En esta sección, se describen áreas que podrían explorarse en futuros desarrollos y mejoras del sistema:

8.1. Servicio Unificado de Logs

Se propone la implementación de un sistema unificado de logs para monitorizar de manera eficiente las operaciones del sistema, facilitando la identificación de problemas y el seguimiento de eventos cruciales.

8.2. Cambio en la Interfaz de Consultoría

Dado que la interfaz actual de la página de consultoría presenta limitaciones con la estructura estilo Trello, especialmente al gestionar 86 comisiones, se sugieren mejoras como:

- La creación de cupos por consultoría.
- Un sistema de actualización del orden de paneles de forma semanal para optimizar la carga distribuida de casos.
- Una barra de búsqueda, para buscar una comisión.

8.3. Tests de Performance y Estrés

Como parte de futuras iteraciones, se recomienda implementar pruebas de rendimiento y estrés. Estas pruebas permitirán evaluar el comportamiento del sistema bajo cargas significativas, identificar posibles cuellos de botella y optimizar su desempeño.

8.4. Almacenamiento Encriptado

Se sugiere la implementación de almacenamiento encriptado para los archivos almacenados en el servidor, mejorando la seguridad y evitando el almacenamiento en texto plano, lo que contribuirá a proteger la confidencialidad de los datos.

8.5. Panel de Estadísticas y Campos Adicionales

Para ofrecer un análisis más completo, se propone la implementación de un panel de estadísticas para el usuario administrador en el sistema. Este panel permitiría medir la cantidad de casos según diferentes criterios. Además, se sugiere la introducción de un campo adicional llamado “Tema” para clasificar cada caso en base a su temática.

Además, se recomienda que el panel de estadísticas permita clasificar los casos según diversos campos, incluyendo:

- Casos agrupados por Provincias Argentinas.
- Casos agrupados por Temáticas.
- Gráfico de cantidad de casos en el tiempo.

8.6. Backup Automático

Se propone la implementación de un sistema de backup automático para garantizar la seguridad y disponibilidad de los datos. Este proceso automatizado debería realizar copias periódicas de la base de datos y los archivos del sistema.

8.7. Gestión de Secretos

Se plantea la opción de incorporar un gestor de secretos, como Vault de HashiCorp [2], para manejar de manera más segura la información confidencial. La idea es evitar almacenar secretos como variables de entorno dentro de los contenedores y eliminar cualquier información sensible codificada directamente en los archivos de configuración, como contraseñas de bases de datos y credenciales del administrador de Django.

9. Conclusiones

Durante el desarrollo de este proyecto, me enfrenté a diversos desafíos que impulsaron la expansión de mis conocimientos en diferentes frameworks y lenguajes de programación. La inmersión en el paradigma de programación funcional al trabajar con React proporcionó una perspectiva declarativa que enriqueció mi capacidad para abordar problemas de manera diferente.

Este año no solo significó la adquisición de nuevas habilidades técnicas, sino también el perfeccionamiento de buenas prácticas y la exploración de nuevas tecnologías, así como la gestión de proyectos y la relación con el cliente. La experiencia en el desarrollo del Sistema de Gestión de Casos no solo amplió mi comprensión teórica, sino que también me permitió aplicar esos conocimientos participando en todas las fases del proyecto.

El desarrollo e implementación del Sistema de Gestión de Casos (Case Management System) representa un hito significativo en el ámbito del patrocinio jurídico, proporcionando una solución tecnológica adaptable a las necesidades específicas del laboratorio IALAB de la Universidad de Buenos Aires. A lo largo de este proceso, se han alcanzado varios objetivos clave que abarcan desde la automatización de la captura de consultas hasta la gestión eficiente de los casos y su asignación a comisiones especializadas.

En primer lugar, la integración con Google Forms estableció un canal eficaz para la recepción de consultas. La aplicación de estrategias y scripts personalizados en Google Forms superó desafíos como la limitación de tipos de datos, facilitando la gestión de preguntas interdependientes y proporcionando una interfaz intuitiva para la gestión de credenciales.

La arquitectura modular y la implementación de tecnologías como Django, React y Docker impulsaron la escalabilidad y el mantenimiento del sistema. La adopción de buenas prácticas en el desarrollo, como la estructuración modular del backend y frontend, el uso de contenedores para la implementación y la aplicación de patrones de diseño como el Pub/Sub, contribuyó a la solidez y flexibilidad del sistema.

La creación de un tablero de trabajo para la consultoría resultó instrumental para la toma de decisiones y el seguimiento de asignaciones. La capacidad de arrastrar y soltar consultas, la visualización detallada del historial de asignaciones y los comentarios y calendarios de las consultas optimizaron la eficiencia, trazabilidad y el control en la gestión de casos.

En cuanto a la seguridad, la implementación de roles específicos y la creación de un mecanismo de autorización fortalecieron la protección de datos y la administración de credenciales, asegurando la confidencialidad e integridad de la información sensible.

El sistema sienta las bases para un sistema modular que ofrece flexibilidad para un proceso continuo que seguirá en desarrollo. En futuras versiones, se explorarán oportunidades de mejora, como fortalecer la integración con Google Forms para permitir la

edición de datos de consultantes y consultas después de su envío inicial, realización de informes y estadísticas, como la clasificación de los casos por categorías, y la integración de filtros automáticos para asistir a los tomadores de casos en el Patrocinio Jurídico de la Universidad de Buenos Aires.

A. Diccionario de Base de Datos

Este capítulo proporciona una descripción detallada de las tablas de la base de datos y sus respectivos atributos. Es fundamental destacar que, para mantener la coherencia con el código interno, los valores de las variables de los atributos que contienen opciones se han establecido siguiendo el estándar del código y utilizando variables internas en inglés. Es importante tener en cuenta que estas variables no serán accesibles para el usuario final. En su lugar, la interfaz gráfica será responsable de interpretar y presentar valores más amigables que representen cada opción de manera comprensible para el usuario final.

A.1. Tabla Board

Un **board** representa el espacio de trabajo de una comisión en forma de tablero. Este tablero contiene paneles para agrupar tarjetas de consultas y organizar los casos y las solicitudes para el ingreso de nuevos casos a la comisión.

Atributo	Tipo de dato	Descripción	Comentarios	No nulo
id	int4	Clave Primaria.	Incremental automático.	V
title	varchar(45)	Título o nombre de la comisión.		V

Cuadro A.1: Tabla Board

A.2. Tabla Board-User

La tabla **Board-User** Registra la asociación entre usuarios y tableros, lo que permite gestionar los permisos y el acceso de cada usuario a los recursos del tablero correspondiente. Se utiliza para asignar usuarios a sus respectivas comisiones.

Atributo	Tipo de dato	Descripción	Comentarios	No nulo
id	int4	Clave Primaria.	Incremental automático.	V
created_at	timestampz	Fecha de asignación.		V
board_id	int4	PK del tablero.		V
user_id	int4	PK del usuario.		V

Cuadro A.2: Tabla Board-User

A.3. Tabla Calendar

La tabla **Calendar** representa el calendario asociado a un ticket de consulta. Se creó una tabla aparte para permitir escalabilidad y la posibilidad de agregar campos adicionales en el futuro, como la fecha de última modificación o la capacidad de crear múltiples calendarios para grupos de trabajo en una consulta.

Atributo	Tipo de dato	Descripción	Comentarios	No nulo
id	int4	Clave Primaria.	Incremental automático.	V
card_id	int4	PK de la tarjeta o ticket de consulta.		V

Cuadro A.3: Tabla Calendar

A.4. Tabla Event

La tabla **Event** representa el evento en un calendario.

Atributo	Tipo de dato	Descripción	Comentarios	No nulo
id	int4	Clave Primaria.	Incremental automático.	V
title	varchar(45)	Título del evento.		V
description	text	Descripción del evento.		F
start	timestampz	Fecha de inicio.		V
end	timestampz	Fecha de cierre.		V
calendar_id	int4	PK del calendario.		V

Cuadro A.4: Tabla Event

A.5. Tabla Card

Una **Card** se utiliza para ubicar una consulta dentro de un tablero de organización representada por una tarjeta la cual será colocada dentro de un panel.

A.6. Tabla Child

La tabla **Child** almacena los datos de un hijo de un cliente.

Atributo	Tipo de dato	Descripción	Comentarios	No nulo
consultation_id	int4	Clave primaria y PK de la consulta asociada.		V
tag	varchar(256)	Etiqueta o título del ticket.		V
panel_id	int4	PK del panel donde se encuentra ubicado el ticket..		V

Cuadro A.5: Tabla Card

Atributo	Tipo de dato	Descripción	Comentarios	No nulo
id	int4	Clave Primaria.	Incremental automático.	V
first_name	varchar(150)	Nombre.		V
last_name	varchar(150)	Apellido.		V
id.type	varchar(10)	Tipo de documento.	Valores posibles: DOCUMENT y PASSPORT.	V
id.value	varchar(20)	Valor del documento (Pasaporte o Número de documento).		V
sex	varchar(6)	Sexo.	Valores posibles: MALE y FEMALE.	V
birth_date	date	Fecha de nacimiento.		V
address	text	Dirección donde vive.		V
locality_id	int4	PK de localidad donde vive.		V
family_client_user_id	int4	PK de la familia asociada.		V

Cuadro A.6: Tabla Child

A.7. Tabla Client

Un cliente en el contexto de este sistema es un consultante que solicita patrocinio gratuito en el Patrocinio Jurídico para un caso particular.

Atributo	Tipo de dato	Descripción	Comentarios	No nulo
id	int4	Clave Primaria.	Incremental automático.	V
first_name	varchar(150)	Nombre.		V
last_name	varchar(150)	Apellido.		V
id_type	varchar(10)	Tipo de documento.	Valores posibles: DOCUMENT y PASSPORT.	V
id_value	varchar(20)	Valor del documento (Pasaporte o Número de documento).		V
sex	varchar(6)	Sexo.	Valores posibles: MALE y FEMALE.	V
birth_date	date	Fecha de nacimiento.		V
address	text	Dirección donde vive.		V
postal	int4	Código postal.		V
locality_id	int4	PK de localidad donde vive.		V
marital_status	varchar(10)	Estado civil.	Valores posibles: SINGLE, MARRIED, DIVORCED y WIDOWER.	V
housing_type	varchar(20)	Tipo de vivienda.	Valores posibles: HOUSE, DEPARTMENT, TRAILER, STREET_SITUATION.	V
studies	varchar(30)	Estudios alcanzados.		V
email	varchar(254)	Correo electrónico.		V
partner_salary	int4	Salario de la pareja del cliente.		V
employment	varchar(45)	Tabajo del cliente.		V
salary	int4	Salario del cliente.		V
amount_retirement	int4	Monto de ingreso por jubilación.		V
amount_pension	int4	Monto de ingreso por pensión.		V
other_income	int4	varchar(45)		V
amount_other_income	int4	Monto de otros ingresos del cliente.		V

Cuadro A.7: Tabla client

A.8. Tabla Tel

La tabla `Tel` almacena los números de teléfono de un cliente.

Atributo	Tipo de dato	Descripción	Comentarios	No nulo
id	int4	Clave Primaria.	Incremental automático.	V
phone_number	int4	Número de teléfono.		V
id_client	int4	PK del cliente.		V

Cuadro A.8: Tabla Family

A.9. Tabla Comment

La tabla `Comment` almacena los comentarios relacionados con las consultas realizadas. Se utiliza para realizar seguimientos de los casos (consultas). Estos son realizados por los profesores encargados de llevar a cabo dicho caso o consulta.

Atributo	Tipo de dato	Descripción	Comentarios	No nulo
id	int4	Clave Primaria.	Incremental automático.	V
time_stamp	timestampz	Momento en que se creó el comentario.		V
text	text	Texto del comentario.		V
consultation_id	int4	PK de la consulta.		V
user_id	int4	PK del usuario que realizó la consulta.		V

Cuadro A.9: Tabla Comment

A.10. Tabla File

La tabla `File` almacena la información de los archivos adjuntos en los comentarios de una consulta o caso.

Atributo	Tipo de dato	Descripción	Comentarios	No nulo
id	int4	Clave Primaria.	Incremental automático.	V
time_stamp	timestampz	Momento en que se subió el archivo.		V
filename	varchar(255)	Nombre del archivo adjunto.		V
comment_id	int4	PK del comentario al que se adjuntó el archivo.		V

Cuadro A.10: Tabla File

A.11. Tabla Consultation

Una consulta es realizada por un cliente que solicita que un caso sea patrocinado por el sistema de patrocinio.

Atributo	Tipo de dato	Descripción	Comentarios	No nulo
id	int4	Clave Primaria.	Incremental automático.	V
tag	varchar(30)	Título o código de identificación de la consulta.		V
client_id	int4	PK del cliente que realizó la consulta.		V
availability_state	varchar(12)	Estado de disponibilidad.	Valores posibles: CREATED, PENDING, ASSIGNED, REJECTED y ARCHIVED.	V
progress_state	varchar(12)	Estado de progreso.	Valores posibles: TODO, IN_PROGRESS, DONE, PAUSED, BLOCKED, INCOMPLETE.	V
time_stamp	timestampz	Fecha de creación de la consulta.		V
start_stamp	timestampz	Fecha en que se comenzó a procesar el caso.		F
description	text	Descripción detallada de la consulta realizada por el cliente.		V
opponent	varchar(500)	Oponente de la consulta.	Parte contraria a quien presenta la demanda, puede ser una persona física, una empresa, una organización, una entidad gubernamental, etc.	V

Cuadro A.11: Tabla Consultation

A.12. Tabla Request Consultation

La tabla **Request Consultation** se utiliza para registrar solicitudes de consulta dirigidas a una comisión, con el fin de que esta se haga cargo del caso o consulta correspondiente.

Atributo	Tipo de dato	Descripción	Comentarios	No nulo
consultation_id	int4	Clave Primaria y PK de la consulta asociada.		V
destiny_board_id	int4	PK del tablero de la comisión destino a la que se envía la solicitud.		V
state	varchar(12)	Estado de la solicitud.	Valores posibles: PENDING, ACCEPTED, REJECTED. Por defecto es PENDING.	V
resolution_time_stamp	timestampz	Tiempo de resolución de la solicitud, es decir, el momento en que se rechaza o acepta.	Valor por defecto es NULL.	F

Cuadro A.12: Tabla Request Consultation

Para garantizar que no haya más de una solicitud pendiente para una consulta, se estableció como única la combinación de los campos 'consultation', 'state' y 'resolution_timestamp'.

A.13. Tabla Panel

Un panel es un espacio dentro de un tablero donde se agrupan diferentes tarjetas o tickets de consultas. Cada tablero puede tener más de un panel, y estos sirven para organizar las consultas por profesor, grupo, estado u otros criterios.

Atributo	Tipo de dato	Descripción	Comentarios	No nulo
id	int4	Clave Primaria.	Incremental automático.	V
title	varchar(255)	Título del panel.		V
board_id	int4	PK del tablero al que pertenece el panel.		V

Cuadro A.13: Tabla Panel

B. Matriz de Relaciones

En esta sección, se presenta la matriz de relaciones, una herramienta utilizada para el diseño de la base de datos del sistema. La matriz de relaciones permite visualizar y comprender las interconexiones entre las diversas entidades que componen la estructura de la base de datos.

El objetivo fundamental de este análisis es identificar y descubrir las relaciones entre las entidades, proporcionando una visión clara de cómo interactúan y se relacionan unos con otros.

A lo largo de la matriz, se representan las relaciones entre entidades específicas, asignándoles significado y nombres a estas conexiones. Cada celda de la matriz detalla cómo las instancias de una entidad están relacionadas con las instancias de otra entidad, y se proporciona una descripción contextual que ayuda a comprender la naturaleza de la conexión entre ellas.

La información contenida en la matriz de relaciones se deriva de un proceso de modelado y refinamiento continuo hasta alcanzar la tercera forma normal, un estándar en la normalización de bases de datos que promueve la eficiencia y evita redundancias.

	Calendar	Event	Card	Panel	Board	Request Consultation
Calendar		Contiene	Contenido en	x	x	x
Event	Contenido en		x	x	x	x
Card	Tiene un	x		Contenida en	x	x
Panel	x	x	Contiene		Contenido en	x
Board	x	x	x	Tiene		Le llegan
RequestConsultation	x	x	x	x	Hacia un	
Consultation	x	x	Tiene una	x	x	Tiene
Client	x	x	x	x	x	x
Tel	x	x	x	x	x	x
User	x	x	x	x	Miembro de	x
Comment	x	x	Está en una	x	x	x
File	x	x	x	x	x	x
Locality	x	x	x	x	x	x
Province	x	x	x	x	x	x
Nationality	x	x	x	x	x	x
Child	x	x	x	x	x	x

Cuadro B.1: Matriz de Relaciones Primera Parte

	Consultation	Client	Tel	User	Comment	File
Calendar	x	x	x	x	x	x
Event	x	x	x	x	x	x
Card	Pertenece a	x	x	x	x	x
Panel	x	x	x	x	x	x
Board	x	x	x	Tiene Miembros	x	x
RequestConsultation	Solicita enviar una	x	x	x	x	x
Consultation		Realizada por	x	x	Tiene	x
Client	Realiza una		Tiene	x	x	x
Tel	x	Pertenece a		x		
User	x	x	x		Realiza un	x
Comment	x	x	x	Lo comenta		Tiene
File	x	x	x	x	Adjuntado en	
Locality	x	Residida por	x	x	x	x
Province	x	x	x	x	x	x
Nationality	x	x	x	x	x	x
Child	x	Es hijo de	x	x	x	x

Cuadro B.2: Matriz de Relaciones Segunda Parte

	Locality	Province	Nationality	Child
Calendar	x	x	x	x
Event	x	x	x	x
Card	x	x	x	x
Panel	x	x	x	x
Board	x	x	x	x
Request Consultation	x	x	x	x
Consultation	x	x	x	x
Client	Reside en	x	x	Tiene
Patrimony	x	x	x	x
Tel	x	x	x	
User	x	x	x	x
Comment	x	x	x	x
File	x	x	x	x
Locality		Está en	x	Residida por
Province	Tiene		Está en	x
Nationality	x	Tiene		x
Child	Reside en	x	x	

Cuadro B.3: Matriz de Relaciones Tercera Parte

C. Configuración de Nginx

C.1. Configuración del Ngnix como Reverse Proxy

El archivo de configuración Nginx utilizado como proxy reverse en el Stack de contenedores es el siguiente:

```
1 limit_conn_zone $binary_remote_addr zone=addr:10m;
2
3 upstream gunicorn{
4     server appserver:8000;
5 }
6
7 upstream daphne{
8     server appserver:8001;
9 }
10
11 upstream frontend{
12     server frontend:3000;
13 }
14
15 server {
16     client_body_timeout 5s;
17     client_header_timeout 5s; # Closing Slow Connections
18     server_tokens off;
19     listen 80;
20     listen [::]:80;
21     server_name proyecto-patrocinio.fcefyn.unc.edu.ar;
22
23     location / {
24         limit_req zone=mylimit burst=20 nodelay;
25         limit_conn addr 10;
26         try_files $uri @proxy_frontend;
27     }
28     location /api {
29         limit_req zone=mylimit burst=20 nodelay;
30         limit_conn addr 10;
31         try_files $uri @proxy_api_wsgi;
32     }
33     location /ws {
34         limit_req zone=mylimit burst=20 nodelay;
35         limit_conn addr 10;
36         try_files $uri @proxy_api_asgi;
37     }
38     location /admin {
39         limit_req zone=mylimit burst=20 nodelay;
40         limit_conn addr 10;
41         root /usr/src/app/django_static;
42         include /etc/nginx/mime.types;
43         try_files $uri @proxy_api_wsgi;
44     }
45
46     # redirect to django app: React server
```



```

47     location @proxy_frontend {
48         proxy_pass http://frontend;
49         proxy_redirect off;
50         proxy_cache_bypass $http_upgrade;
51         proxy_set_header Upgrade $http_upgrade;
52         proxy_set_header Connection "upgrade";
53         proxy_set_header Host $host;
54         proxy_set_header X-Real-IP $remote_addr;
55         proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
56         proxy_set_header X-Forwarded-Proto $scheme;
57         proxy_set_header X-Forwarded-Host $host;
58         proxy_set_header X-Forwarded-Port $server_port;
59         proxy_set_header Cookie $http_cookie;
60     }
61
62     # redirect to django app: API REST
63     location @proxy_api_wsgi {
64         proxy_pass http://gunicorn;
65         proxy_redirect off;
66         proxy_cache_bypass $http_upgrade;
67         proxy_set_header Upgrade $http_upgrade;
68         proxy_set_header Connection "upgrade";
69         proxy_set_header Host $host;
70         proxy_set_header X-Real-IP $remote_addr;
71         proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
72         proxy_set_header X-Forwarded-Proto $scheme;
73         proxy_set_header X-Forwarded-Host $host;
74         proxy_set_header X-Forwarded-Port $server_port;
75         proxy_set_header Cookie $http_cookie;
76     }
77
78     # redirect to django app with WebSocket
79     location @proxy_api_asgi {
80         proxy_pass http://daphne;
81         proxy_redirect off;
82         proxy_cache_bypass $http_upgrade;
83         proxy_set_header Upgrade $http_upgrade;
84         proxy_set_header Connection "upgrade";
85         proxy_set_header Host $host;
86         proxy_set_header X-Real-IP $remote_addr;
87         proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
88         proxy_set_header X-Forwarded-Proto $scheme;
89         proxy_set_header X-Forwarded-Host $host;
90         proxy_set_header X-Forwarded-Port $server_port;
91         proxy_set_header Cookie $http_cookie;
92     }
93
94     location /django_static/ {
95         limit_req zone=mylimit burst=20 nodelay;
96         limit_conn addr 10;
97         autoindex on;
98         alias /usr/share/nginx/staticfiles/cms/django/;
99     }
100 }

```

Extracto de código C.1: Configuración de Nginx Reverse Proxy

C.2. Configuración del Nginx como Servidor

Por otro lado, el servidor contaba con un Servidor Nginx instalado y en funcionamiento sirviendo otras aplicaciones. Para agregar el servicio Case Managment System se agregó el siguiente archivo en el directorio `etc/nginx/conf.d/proyecto-patrocinio.fcefyn.unc.edu.ar.conf`:

```
1
2 server {
3     server_name proyecto-patrocinio.fcefyn.unc.edu.ar;
4
5     location / {
6         proxy_pass http://localhost:8081;
7         proxy_redirect off;
8         proxy_cache_bypass $http_upgrade;
9         proxy_set_header Upgrade $http_upgrade;
10        proxy_set_header Connection "upgrade";
11        proxy_set_header Host $host;
12        proxy_set_header X-Real-IP $remote_addr;
13        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
14        proxy_set_header X-Forwarded-Proto $scheme;
15        proxy_set_header X-Forwarded-Host $host;
16        proxy_set_header X-Forwarded-Port $server_port;
17        proxy_set_header Cookie $http_cookie;
18    }
19
20    listen 443 ssl; # managed by Certbot
21    ssl_certificate /etc/letsencrypt/live/proyecto-patrocinio.fcefyn.unc.
22    ↪ edu.ar/fullchain.pem; # managed by Certbot
23    ssl_certificate_key /etc/letsencrypt/live/proyecto-patrocinio.fcefyn.
24    ↪ unc.edu.ar/privkey.pem; # managed by Certbot
25    include /etc/letsencrypt/options-ssl-nginx.conf; # managed by Certbot
26    ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem; # managed by Certbot
27
28 }
29
30 server {
31     if ($host = proyecto-patrocinio.fcefyn.unc.edu.ar) {
32         return 301 https://$host$request_uri;
33     } # managed by Certbot
34
35     listen 80;
36     server_name proyecto-patrocinio.fcefyn.unc.edu.ar;
37     return 404; # managed by Certbot
38
39 }
```

Extracto de código C.2: Configuración de Nginx en el Servidor

D. Endpoints de Formularios

D.1. Endpoint del Formulario Registro de Cliente

URL

`http://{ip}:{port}/api/clients/client/form/`

Método

POST

Body de Ejemplo:

A continuación se presenta un ejemplo.

```
1 {
2   "first_name": "Martina",
3   "last_name": "Gutierrez",
4   "id_type": "DOCUMENT",
5   "id_value": "42304328",
6   "birth_date": "2000-11-13",
7   "sex": "FEMALE",
8   "marital_status": "SINGLE",
9   "studies": "INCOMPLETE_UNIVERSITY",
10  "email": "martina2000@gmail.com",
11  "housing_type": "HOUSE",
12  "locality": 9,
13  "address": "Av. Patria 1234",
14  "postal": "5000",
15  "employment": "programmer",
16  "salary": "123",
17  "other_income": "No tengo otros ingresos",
18  "amount_other_income": "0",
19  "amount_retirement": "0",
20  "amount_pension": "0",
21  "vehicle": "No tengo otros vehiculos",
22  "tel": [
23    "3512254210",
24    "+54 9 25578784"
25  ],
26  "partner_salary": "0"
27 }
```

Extracto de código D.1: Body de Ejemplo

Campos del Body:

Campo	Tipo	Opciones	Descripción
first_name	String		Nombre del consultante.
last_name	String		Apellido del consultante.
id.type	String	DOCUMENT, PASSPORT	Tipo de documento de identidad.
id.value	String		Valor del documento de identidad.
birth_date	String		Fecha de nacimiento del consultante.
sex	String	MALE, FEMALE	Género del consultante.
marital_status	String	SINGLE, MARRIED, DIVORCED, WIDOWER	Estado civil del consultante.
studies	String	INCOMPLETE_PRIMARY, COMPLETE_PRIMARY, INCOMPLETE_SECONDARY, COMPLETE_SECONDARY, INCOMPLETE_TERTIARY, COMPLETE_TERTIARY, INCOMPLETE_UNIVERSITY, COMPLETE_UNIVERSITY	Nivel de estudios del consultante.
email	String		Correo electrónico del consultante.
housing_type	String	HOUSE, DEPARTMENT, TRAILER, STREET_SITUATION	Tipo de vivienda del consultante.
locality	Integer		ID de localidad del consultante.
address	String		Dirección del consultante.
postal	String		Código postal del consultante.
employment	String		Ocupación del consultante.
salary	String		Salario del consultante.
other_income	String		Otros ingresos del consultante.
amount_other_income	String		Monto de otros ingresos.
amount_retirement	String		Monto de jubilación.
amount_pension	String		Monto de pensión.
vehicle	String		Información sobre vehículos del consultante.
tel	List		Lista de números de teléfono del consultante.
partner_salary	String		Salario del cónyuge (si aplica).

Headers:

- **Authorization:** Token {{TOKEN}}
- **Content-Type:** application/json

D.2. Endpoint del Formulario Registro de Hijo

URL

http://{{ip}}:{{port}}/api/clients/son/form/

Método

POST

Campos del Body:

Campo	Tipo	Opciones	Descripción
id_consultant	String		Identificación del consultor asociado.
first_name	String		Nombre del hijo.
last_name	String		Apellido del hijo.
id_type	String	DOCUMENT, PASSPORT	Tipo de documento de identidad del hijo.
id_value	String		Valor del documento de identidad del hijo.
birth_date	String		Fecha de nacimiento del hijo.
sex	String	MALE, FEMALE	Género del hijo.
locality	Integer		ID de la localidad del hijo.
address	String		Dirección del hijo.

Cuadro D.2: Descripción de campos del Body

Body de Ejemplo:

```
1 {  
2   "id_consultant": "42304328",  
3   "first_name": "Matilda",  
4   "last_name": "Gonzales",  
5   "id_type": "PASSPORT",  
6   "id_value": "123456789",  
7   "birth_date": "2020-11-14",  
8   "sex": "FEMALE",  
9   "locality": 9,  
10  "address": "Av. Patria 1234"  
11 }
```

Extracto de código D.2: Body de Ejemplo

Headers:

- **Authorization:** Token {{TOKEN}}
- **Content-Type:** application/json

D.3. Endpoint del Formulario Consulta

URL

http://{{ip}}:{{port}}/api/consultations/consultation/form/

Método

POST

Body de Ejemplo:

```
1 {  
2   "client": "42304328",  
3   "tag": "Asesoramiento Legal",  
4   "description": "Necesito asesoramiento legal con respecto a una  
    ↳ disputa contractual con la Corporacion XYZ. El contrato  
    ↳ involucra la venta de bienes y hay desacuerdos sobre los plazos  
    ↳ de entrega y los terminos de pago.",  
5   "opponent": "Corporacion XYZ"  
6 }
```

Extracto de código D.3: Body de Ejemplo

Campos del Body:

Campo	Tipo	Opciones	Descripción
client	String		Identificación del consultante asociado a la consulta.
tag	String		Etiqueta de la consulta.
description	String		Descripción detallada de la consulta.
opponent	String		Oponente o entidad involucrada en la consulta.

Cuadro D.3: Descripción de campos del Body

Headers:

- **Authorization:** Token {{TOKEN}}
- **Content-Type:** application/json

E. Archivos de Configuración para el Despliegue de la Plataforma

E.1. Archivo de configuración .env

Archivo de configuración de variables de entorno usadas por *docker-compose.yml*.

```
1 CMS_BACKEND_IMAGE=proyectopatrocinio/backend-patrocinio:v1.3.0
2 CMS_FRONTEND_IMAGE=proyectopatrocinio/frontend-patrocinio:v1.3.0
3 CMS_PROXY_PORT=8081
4 CMS_NGINX_CONFIG_FILE=./resources/nginx.conf
5 CMS_BACKEND_ENV_FILE=./resources/backend.env
6 CMS_POSTGRES_ENV_FILE=./resources/postgres.env
7 CMS_FRONTEND_ENV_FILE=./resources/frontend.env
8 CMS_LOGO_FILE=./resources/logo.ico
9 CMS_TEMPLATES_ACCOUNT_PATH=./resources/templates/account/
10 CMS_TEMPLATES_NOTIFICATION_PATH=./resources/templates/notifications/
11 CMS_TERMS_AND_POLICIES_FILE=./resources/terms_and_policies.md
```

Extracto de código E.1: Archivo de configuración .env

E.2. Archivo de configuración backend.env

Archivo de configuración de variables de entorno del servicio backend para el deploy de Docker Swarm:

```
1 DEBUG=0
2 DJANGO_ALLOWED_HOSTS=proyecto-patrocinio.fcefyn.unc.edu.ar databaseserver
   ↳ nginxserver appserver redis_server frontend
3 SQL_ENGINE=django.db.backends.postgresql
4 SQL_DATABASE=patrocinio_prod
5 SQL_USER=patrocinio_api
6 SQL_PASSWORD=*****
7 SQL_HOST=db
8 SQL_PORT=5432
9 DATABASE=postgres
10 EMAIL_HOST_USER=*****
11 EMAIL_HOST_PASSWORD=*****
12 CORS_ALLOWED_ORIGINS=http://nginx:80 http://frontend:3000
13 HOSTNAME=proyecto-patrocinio.fcefyn.unc.edu.ar
14 CONSULTANCY_BOARD_NAME=CONSULTORIA
15 DEFAULT_HTTP_PROTOCOL=https
16 CSRF_TRUSTED_ORIGINS=http://nginx:80 http://frontend:3000 https://
   ↳ proyecto-patrocinio.fcefyn.unc.edu.ar
17 LOG_ROTATE_DAYS=10
18 SECRET_KEY=*****
19 DJANGO_SUPERUSER_USERNAME=*****
20 DJANGO_SUPERUSER_PASSWORD=*****
```



```
21 | DJANGO_SUPERUSER_EMAIL=****
```

Extracto de código E.2: Archivo de configuración backend.env

E.3. Archivo de configuración frontend.env

Archivo de configuración de variables de entorno del servicio frontend para el deploy de Docker Swarm:

```
1 # Utilizar el dominio correspondiente
2 REACT_APP_URL_BASE_API_REST_PATROCINIO=https://proyecto-patrocinio.fcefyn
   ↳ .unc.edu.ar/api/
3 REACT_APP_WS_NOTIFICATION_PATH_PATROCINIO=wss://proyecto-patrocinio.
   ↳ fcefyn.unc.edu.ar/ws/notification/
4
5 # No cambiar las siguientes variables
6 REACT_APP_PATH_TERMS=terms/
7 REACT_APP_PATH_LOGIN=auth/login/
8 REACT_APP_PATH_LOGOUT=auth/logout/
9 REACT_APP_PATH_RESET_PASSWORD=auth/password/reset/
10 REACT_APP_PATH_RESET_PASSWORD_CONFIRM=auth/password/reset-confirm/
11 REACT_APP_PATH_CHANGE_PASSWORD=auth/password/change/
12 REACT_APP_PATH_USER=auth/user/
13 REACT_APP_PATH_USER_BY_TOKEN=auth/user-by-token/?token=
14 REACT_APP_PATH_RESEND_EMAIL=auth/resend-email/
15 REACT_APP_PATH_SIGNUP=register/
16 REACT_APP_PATH_USERBOARD=boardusers/boarduser/
17 REACT_APP_PATH_USERBOARD_BY_USER=boardusers/boarduser/?user_id=
18 REACT_APP_PATH_CARDS=cards/card/
19 REACT_APP_PATH_CONSULTATIONS=consultations/consultation/
20 REACT_APP_PATH_FILTER_CONSULTATIONS_BY_AVAILABILITY=consultations/
   ↳ consultation/?availability_state=
21 REACT_APP_PATH_REQUEST_CARDS=consultations/request_consultation/
22 REACT_APP_EXTRA_PATH_ACCEPT_REQUEST_CARDS=/accepted/
23 REACT_APP_EXTRA_PATH_REJECTED_REQUEST_CARDS=/rejected/
24 REACT_APP_PATH_BOARD=boards/board/
25 REACT_APP_PATH_REQUEST_CONSULTANCY_BOARD=boards/board/consultancy_board/
26 REACT_APP_EXTRA_PATH_BOARD_LOGS=/logs/?days=
27 REACT_APP_PATH_PANELS=panels/panel/
28 REACT_APP_PATH_CLIENTS=clients/client/
29 REACT_APP_PATH_TEL=clients/tel/
30 REACT_APP_PATH_PATRIMONY=clients/patrimony/
31 REACT_APP_PATH_FAMILY=clients/family/
32 REACT_APP_PATH_CHILDREN=clients/child/
33 REACT_APP_PATH_LOCALITY=geography/locality/
34 REACT_APP_PATH_PROVINCE=geography/province/
35 REACT_APP_PATH_NATIONALITY=geography/nationality/
36 REACT_APP_PATH_COMMENTS=comments/comment/
37 REACT_APP_PATH_COMMENTS_BY_CONSULT=comments/comment/?consultation_id=
38 REACT_APP_PATH_ATTACH_COMMENT_FILE=comments/file/
39 REACT_APP_PATH_CALENDAR=calendars/calendar/
40 REACT_APP_PATH_CALENDAR_EVENT=calendars/event/
```

Extracto de código E.3: Archivo de Configuración frontend.env

E.4. Archivo de configuración portgres.env

Archivo de configuración de variables de entorno de la base de datos para el deploy de Docker Swarm:

```
1 POSTGRES_DB=patrocinio_prod
2 POSTGRES_USER=patrocinio_api
3 PGDATA=/var/lib/postgresql/data
4 POSTGRES_PASSWORD=*****
```

Extracto de código E.4: Archivo de configuración portgres.env

F. Test Unitarios

F.1. Ejemplo de Test Unitario en Backend

```
1 class BoardUserPermissionTest(APITestCase):
2
3     def setUp(self):
4         """Set up the request factory and viewset for each test."""
5         self.factory = APIRequestFactory()
6         self.viewset = BoardUserViewSet.as_view({'get': 'list', 'post': '
            ↳ create', 'put': 'update', 'delete': 'destroy'})
7         self.url = reverse('board_user-list')
8         setUpSuperUser(self)
9         self.board = Board.objects.create(title='Test Board')
10
11     def test_board_user_creation(self):
12         """Test that a BoardUser instance can be created successfully."""
13         board_user = BoardUser.objects.create(user=self.user, board=self.
            ↳ board)
14         self.assertTrue(isinstance(board_user, BoardUser))
15         self.assertEqual(board_user.user, self.user)
16         self.assertEqual(board_user.board, self.board)
17
18     def test_board_user_uniqueness(self):
19         """Test that two instances with the same user and board cannot be
            ↳ created."""
20         BoardUser.objects.create(user=self.user, board=self.board)
21         with self.assertRaises(Exception):
22             BoardUser.objects.create(user=self.user, board=self.board)
23
24     def test_board_user_str_representation(self):
25         """Test the string representation of the BoardUser object."""
26         board_user = BoardUser.objects.create(user=self.user, board=self.
            ↳ board)
27         expected_str = f'{self.user}/{self.board}'
28         self.assertEqual(str(board_user), expected_str)
29
30     def test_unauthenticated_user_cannot_access_board_users(self):
31         """Test that an unauthenticated user cannot access the board
            ↳ users endpoint."""
32         request = self.factory.get('/boardusers/')
33         response = self.viewset(request)
34         self.assertEqual(response.status_code, status.HTTP_403_FORBIDDEN)
35
36     def test_authenticated_user_can_list_board_users(self):
37         """Test that an authenticated user can list board users."""
38         request = self.factory.get('/boardusers/')
39         request.user = self.user
40         response = self.viewset(request)
41         self.assertEqual(response.status_code, status.HTTP_200_OK)
42
43     def test_authenticated_user_can_create_board_user(self):
44         """Test that an authenticated user can create a board user."""
```

```

45     request_data = {'user': self.user.id, 'board': self.board.id}
46     request = self.factory.post('/boardusers/', request_data)
47     request.user = self.user
48     response = self.viewset(request)
49     self.assertEqual(response.status_code, status.HTTP_201_CREATED)
50
51     def test_authenticated_user_cannot_delete_board_user(self):
52         """Test that an authenticated user cannot delete a board user."""
53         request = self.factory.delete('/boardusers/1/')
54         response = self.viewset(request, pk=1)
55         self.assertEqual(response.status_code, status.HTTP_403_FORBIDDEN)

```

Extracto de código F.1: Test Unitario Backend

F.2. Ejemplo de Test Unitario en Frontend

```

1 describe('ConsutationDisplay Component', () => {
2     test('renders and switches between windows', async () => {
3         // Render the component
4         render(
5             <ConsutationDisplay
6                 consultation={{ consultation: 1, id: 1 }}
7                 open={true}
8                 onClose={() => {}}
9                 updateViewTag={() => {}}
10            />
11        );
12
13        expect(screen.getByText(/Consultation Details/)).toBeInTheDocument();
14        expect(screen.getByText("Info")).toBeInTheDocument();
15        expect(screen.getByText("Comments")).toBeInTheDocument();
16        expect(screen.getByText("Calendar")).toBeInTheDocument();
17
18        // Verify that the default window is "Info"
19        expect(screen.getByTestId('info-component')).toBeInTheDocument();
20        expect(screen.queryByTestId('comment-component')).toBeNull();
21        expect(screen.queryByTestId('calendar-component')).toBeNull();
22
23        // Switch to "Comments" window
24        fireEvent.click(screen.getByText("Comments").closest('button'));
25        expect(screen.getByTestId('comment-component')).toBeInTheDocument();
26        expect(screen.queryByTestId('info-component')).toBeNull();
27        expect(screen.queryByTestId('calendar-component')).toBeNull();
28
29        // Switch to "Calendar" window
30        fireEvent.click(screen.getByText('Calendar').closest('button'));
31        expect(screen.getByTestId('calendar-component')).toBeInTheDocument();
32        expect(screen.queryByTestId('info-component')).toBeNull();
33        expect(screen.queryByTestId('comment-component')).toBeNull();
34
35    });
36 });

```

Extracto de código F.2: Test Unitario Frontend

G. Bibliografía

- [1] Conventional Changelog. Commits convencionales, 2018. URL <https://www.conventionalcommits.org/en/v1.0.0/>.
- [2] HashiCorp. Vault. URL <https://www.hashicorp.com/products/vault>.
- [3] GitHub Inc. Ci/cd: The what, why, and how, 2023. URL <https://resources.github.com/ci-cd/>.
- [4] Red Hat Inc. Que es una api rest?, 2023. URL <https://www.redhat.com/es/topics/api/what-is-a-rest-api>.
- [5] Roger S. Pressman. *Ingeniería de Software: Un enfoque práctico*. McGraw-Hill, 7 edition, 2010.
- [6] Tom Preston-Werner. Versionado semántico 2.0.0. URL <https://semver.org/lang/es/>.
- [7] Ignacio Pérez. ¿en qué consiste la vulnerabilidad cross site request forgery (csrf)?, 2015. URL <https://www.welivesecurity.com/la-es/2015/04/21/vulnerabilidad-cross-site-request-forgery-csrf/>.
- [8] Amir Rawdat. Using free let's encrypt ssl/tls certificates with nginx, 2021. URL <https://www.nginx.com/blog/using-free-ssl-tls-certificates-from-lets-encrypt-with-nginx/>.
- [9] IONOS Cloud S.L.U. Web sockets, 2023. URL <https://www.ionos.es/digitalguide/paginas-web/desarrollo-web/que-es-websocket/>.
- [10] SmartBear Software. Gherkin syntax, 2019. URL <https://cucumber.io/docs/gherkin/>.
- [11] Ian Sommerville. *Ingeniería de Software*. Pearson Education, 9 edition, 2011.
- [12] Google Workspace. Apps script, 2023. URL https://workspace.google.com/intl/es-419_ar/products/apps-script/.