Plataforma de Gestión de Trabajos de Fin de Grado

Sistema web integral para la automatización del proceso académico universitario

TRABAJO DE FIN DE GRADO

Grado en Ingeniería Informática

Autor: Juan Mariano Centeno Ariza

Tutor: Guadalupe Ortiz Bellot

Agradecimientos

Incluir

Resumen Ejecutivo

Este Trabajo de Fin de Grado presenta el desarrollo de una **Plataforma de Gestión de TFG**, un sistema web integral diseñado para automatizar y optimizar el proceso completo de gestión de Trabajos de Fin de Grado en entornos universitarios.

Problema identificado: Los procesos tradicionales de gestión de TFG se caracterizan por su fragmentación, uso de herramientas dispersas y alto componente manual, generando ineficiencias y dificultades en el seguimiento académico.

Solución desarrollada: Sistema web moderno que integra todas las fases del proceso TFG, desde la propuesta inicial hasta la defensa final, con roles diferenciados para estudiantes, profesores, presidentes de tribunal y administradores.

Tecnologías implementadas: La solución se ha desarrollado utilizando un stack tecnológico moderno y robusto. En el frontend se ha implementado React 19 junto con Vite y Tailwind CSS v4 para proporcionar una interfaz de usuario moderna y responsive. El backend está construido sobre Symfony 6.4 LTS con PHP 8.2+ y API Platform 3.x, garantizando escalabilidad y mantenibilidad a largo plazo. La persistencia de datos se gestiona mediante MySQL 8.0 integrado con Doctrine ORM, mientras que la seguridad se basa en autenticación JWT con refresh tokens. El entorno de desarrollo utiliza DDEV con Docker para asegurar consistencia y facilitar el despliegue.

Resultados obtenidos: La implementación de la plataforma ha demostrado una significativa mejora en la eficiencia operativa. El sistema desarrollado integra completamente 4 módulos especializados según el rol de usuario, optimizando los flujos de trabajo específicos de cada perfil. La arquitectura implementada ha sido diseñada con criterios de escalabilidad, preparando el sistema para futuras expansiones funcionales y de capacidad.

Palabras clave: TFG, React, Symfony, Gestión Académica, Plataforma Web, Sistema de Información, Automatización Universitaria.

Índice

Lista de Figuras

1. Visión general del proyecto

Este capítulo presenta una visión general del proyecto desarrollado: desde la motivación inicial hasta los objetivos y el alcance definido. También incluye la estructura del documento, los estándares aplicados y las definiciones de los conceptos clave.

La plataforma de gestión de TFG nace de una necesidad real detectada en el entorno universitario, donde los procesos académicos requieren mayor digitalización y automatización. Este capítulo establece las bases que justifican el desarrollo del sistema y define una guía de su implementación.

1.1 Motivación

La gestión de Trabajos de Fin de Grado en las universidades involucra a múltiples participantes: estudiantes, profesores tutores, tribunales y personal administrativo. Tradicionalmente, este proceso se gestionaba de forma fragmentada usando correo electrónico, documentos físicos y hojas de cálculo, lo que causa ineficiencias, pérdida de información y dificultades para hacer seguimiento del progreso.

La pandemia de COVID-19 aceleró la digitalización educativa, evidenciando la necesidad de algún sistema informático que facilite tanto la gestión remota como presencial. Las universidades necesitan plataformas que no solo digitalicen los procesos existentes, sino que los optimicen mediante automatización, seguimiento en tiempo real y generación de reportes.

El cumplimiento de normativas académicas, la gestión de plazos estrictos y la coordinación entre departamentos requieren una solución tecnológica que centralice toda la información de los TFG en un sistema único, accesible y seguro.

1.2 Objetivos

Esta sección establece tanto el objetivo general como los objetivos específicos que guían la implementación, organizados según su naturaleza funcional, técnica y de calidad para facilitar su seguimiento y evaluación.

Todos los objetivos se formularon aplicando la metodología SMART (Específicos, Medibles, Alcanzables, Relevantes y Temporales), garantizando que cada objetivo contribuya

efectivamente al propósito del proyecto y pueda ser evaluado de manera objetiva durante el desarrollo.

1.2.1 Objetivo General

Desarrollar una plataforma web para la gestión completa del ciclo de vida de los TFG, desde la propuesta inicial hasta la defensa final, mejorando la eficiencia, transparencia y seguimiento del proceso académico.

1.2.2 Objetivos Específicos

Objetivos Funcionales:

- **OF1**: Implementar un sistema de autenticación seguro basado en JWT que soporte múltiples roles de usuario (estudiante, profesor, presidente de tribunal, administrador).
- **OF2**: Desarrollar un módulo completo para estudiantes que permita la subida, edición y seguimiento del estado de sus TFG.
- **OF3**: Crear un sistema de gestión para profesores tutores que facilite la supervisión, evaluación y retroalimentación de los TFG asignados.
- **OF4**: Implementar un módulo de gestión de tribunales que permita la creación, asignación y coordinación de defensas.
- **OF5**: Desarrollar un sistema de calendario integrado para la programación y gestión de defensas presenciales.
- **OF6**: Crear un panel administrativo completo para la gestión de usuarios, reportes de errores y configuración del sistema.
- **OF7**: Implementar un sistema de notificaciones en tiempo real para mantener informados a todos los actores del proceso.

Objetivos Técnicos:

- **OT1**: Diseñar una arquitectura frontend moderna basada en React 19 con componentes reutilizables y diseño responsivo.
- **OT2**: Implementar un backend con Symfony 6.4 LTS que proporcione APIs REST seguras y escalables.
- **OT3**: Establecer un sistema de base de datos optimizado con MySQL 8.0 que garantice la integridad y consistencia de los datos.
- **OT4**: Desarrollar un sistema de gestión de archivos seguro para el almacenamiento y descarga de documentos TFG.

- **OT5**: Implementar un sistema de testing automatizado que cubra tanto frontend como backend.
- **OT6**: Configurar un entorno de desarrollo containerizado con DDEV para facilitar la colaboración y despliegue.

Objetivos de Calidad:

- **OC1**: Garantizar un tiempo de respuesta menor a 2 segundos para todas las operaciones críticas del sistema.
- **OC2**: Implementar medidas de seguridad que cumplan con estándares académicos de protección de datos.
- **OC3**: Diseñar una interfaz de usuario intuitiva con una curva de aprendizaje mínima para todos los roles.
- **OC4**: Asegurar compatibilidad entre diferentes navegadores para dispositivos móviles y tablets.
- **OC5**: Establecer un sistema de copia de seguridad y recuperación de datos que garantice la disponibilidad del servicio.

1.3 Alcance

El alcance del proyecto se organiza en tres dimensiones: funcional, técnica y temporal. Cada dimensión aborda aspectos específicos del desarrollo, desde las funcionalidades concretas que se implementarán hasta las tecnologías seleccionadas y los plazos de entrega comprometidos.

1.3.1 Alcance Funcional

Incluido en el proyecto:

- **Gestión completa del ciclo de vida del TFG**: Desde la creación inicial hasta la calificación final.
- **Sistema multi-rol**: Soporte para cuatro tipos de usuario con permisos diferenciados.
- **Gestión de archivos**: Subida, almacenamiento y descarga segura de documentos PDF.
- **Sistema de calendario**: Programación y gestión de defensas con disponibilidad de tribunales.
- **Panel de reportes**: Generación de estadísticas y exportación de datos en múltiples

formatos.
- **API REST completa**: Endpoints documentados para todas las funcionalidades del sistema.

No incluido en el proyecto:

- Sistema de notificaciones, alertas en tiempo real y notificaciones por email.
- Sistema de videoconferencia integrado para defensas remotas.
- Integración con sistemas de información universitarios existentes (ERP académico).
- Módulo de plagio o análisis de contenido automático.
- Sistema de facturación o pagos.
- Funcionalidades de red social o colaboración entre estudiantes.
- Soporte multiidioma (solo español en esta versión).

1.3.2 Alcance Técnico

Tecnologías implementadas:

- **Frontend**: React 19, Vite, Tailwind CSS v4, React Router DOM v7.
- **Backend**: Symfony 6.4 LTS, PHP 8.2+, API Platform 3.x.
- **Base de datos**: MySQL 8.0 con Doctrine ORM.
- **Autenticación**: JWT con refresh tokens.
- **Gestión de archivos**: VichUploaderBundle con validaciones de seguridad.
- **Testing**: PHPUnit (backend), Vitest (frontend).
- **Despliegue**: DDEV con Docker, Composer, npm.

Limitaciones técnicas:

- Soporte únicamente para archivos PDF (no otros formatos de documento).
- Base de datos relacional (no NoSQL para este alcance).
- Despliegue en servidor único (no arquitectura de microservicios).
- Almacenamiento local de archivos (no integración con servicios cloud en esta versión).

1.3.3 Alcance Temporal

El proyecto se desarrolla en 8 fases distribuidas a lo largo de 10 semanas académicas:

- **Fases 1-6**: Desarrollo frontend completo.
- **Fase 7**: Implementación backend Symfony.
- **Fase 8**: Testing, optimización y despliegue.

1.4 Visión general del documento

Este documento está estructurado para proporcionar una comprensión completa y progresiva del proyecto desarrollado. Cada capítulo aborda aspectos específicos del desarrollo, desde la idea inicial hasta la implementación final.

El documento sigue el estándar ISO/IEEE 16326 para documentación de sistemas software, adaptado al contexto académico de un TFG. La estructura es la siguiente:

Capítulo 1 - Visión general del proyecto: Establece la motivación, objetivos y alcance del proyecto, proporcionando el contexto necesario para comprender la necesidad y los beneficios de la plataforma desarrollada.

Capítulo 2 - Contexto del proyecto: Describe en detalle el entorno tecnológico, las características de los usuarios objetivo y el modelo de ciclo de vida adoptado para el desarrollo del sistema.

Capítulo 3 - Planificación: Presenta la metodología de desarrollo por fases, cronogramas de implementación y la distribución temporal de las actividades del proyecto.

Capítulo 4 - Análisis del sistema: Contiene la especificación completa de requisitos funcionales y no funcionales, casos de uso, diagramas UML y criterios de garantía de calidad.

Capítulo 5 - Diseño: Documenta la arquitectura del sistema tanto a nivel físico como lógico, incluyendo el diseño de la base de datos y la interfaz de usuario.

Capítulo 6 - Implementación: Detalla los aspectos técnicos de la implementación, incluyendo la estructura del código, patrones de diseño utilizados y decisiones de arquitectura.

Capítulo 7 - Entrega del producto: Describe los procesos de configuración, despliegue y entrega del sistema en entorno de producción.

Capítulo 8 - Procesos de soporte y pruebas: Documenta las estrategias de testing, gestión de riesgos y procesos de validación implementados.

Capítulo 9 - Conclusiones y trabajo futuro: Presenta una evaluación crítica del proyecto, cumplimiento de objetivos y propuestas de mejoras futuras.

Los anexos incluyen manuales técnicos de instalación y usuario, así como documentación adicional de referencia.

1.5 Estandarización del documento

Este documento sigue las directrices del estándar **ISO/IEEE 16326:2009** - "Systems and software engineering - Life cycle processes - Project management", adaptado para proyectos académicos.

Existen apartados que se han omitido ya que carecían de sentido en el contexto de este proyecto, ya que el estándar está orientado a proyectos de desarrollo de software a gran escala. Por ejemplo, no se incluyen secciones sobre gestión de proveedores o adquisiciones.

Además se ha tomado la decisión de incluir la bibliografía al final del documento, ya que el estándar no especifica un formato concreto para la bibliografía y se ha optado por un formato más adecuado para un TFG.

1.5.1 Normas aplicadas

- **ISO/IEEE 16326:2009**: Estructura principal del documento y gestión de proyectos.
- **IEEE Std 830-1998**: Especificación de requisitos software (Capítulo 4).
- **IEEE Std 1016-2009**: Descripciones de diseño software (Capítulo 5).
- **ISO/IEC 25010:2011**: Modelo de calidad del producto software (Capítulo 4.2).

1.5.2 Convenciones del documento

Formato de texto: - Títulos principales: Numeración decimal (1., 1.1., 1.1.1.). - Términos técnicos: Primera aparición en **negrita**. - Acrónimos: MAYÚSCULAS con definición en primera aparición.

Diagramas y figuras: - Numeración correlativa: Figura 1.1, Figura 1.2, etc. - Pie de figura descriptivo con fuente cuando corresponda.

Tablas: - Numeración correlativa: Tabla 1.1, Tabla 1.2, etc. - Encabezados en negrita.

Referencias: - Bibliografía al final del documento. - Formato APA para referencias académicas. - Enlaces web con fecha de acceso.

1.6 Acrónimos

Este documento utiliza diversos acrónimos técnicos comunes en ingeniería de software y desarrollo web. Esta sección proporciona una referencia completa de todos los términos abreviados, facilitando la comprensión para lectores con diferentes niveles de especialización.

Acrónimo	Significado
API	Application Programming Interface (Interfaz de Programación de Aplicaciones)
CORS	Cross-Origin Resource Sharing (Intercambio de Recursos de Origen Cruzado)
CRUD	Create, Read, Update, Delete (Crear, Leer, Actualizar, Eliminar)
CSS	Cascading Style Sheets (Hojas de Estilo en Cascada)
DDEV	Docker Development Environment
DOM	Document Object Model (Modelo de Objetos del Documento)
EPL	Event Processing Language (Lenguaje de Procesamiento de Eventos)
HMR	Hot Module Replacement (Reemplazo de Módulos en Caliente)
HTML	HyperText Markup Language (Lenguaje de Marcado de Hipertexto)
HTTP	HyperText Transfer Protocol (Protocolo de Transferencia de Hipertexto)
IEEE	Institute of Electrical and Electronics Engineers
ISO	International Organization for Standardization
JSON	JavaScript Object Notation (Notación de Objetos JavaScript)
JWT	JSON Web Token (Token Web JSON)
LTS	Long Term Support (Soporte a Largo Plazo)
MVC	Model-View-Controller (Modelo-Vista-Controlador)
ORM	Object-Relational Mapping (Mapeo Objeto-Relacional)

Acrónimo	Significado
PDF	Portable Document Format (Formato de Documento Portable)
PHP	PHP: Hypertext Preprocessor
REST	Representational State Transfer (Transferencia de Estado Representacional)
RTL	React Testing Library
SPA	Single Page Application (Aplicación de Página Única)
SQL	Structured Query Language (Lenguaje de Consulta Estructurado)
TFG	Trabajo de Fin de Grado
UI	User Interface (Interfaz de Usuario)
UML	Unified Modeling Language (Lenguaje de Modelado Unificado)
URL	Uniform Resource Locator (Localizador Uniforme de Recursos)
UX	User Experience (Experiencia de Usuario)

1.7 Definiciones

Esta sección presenta las definiciones de los conceptos técnicos y términos especializados más relevantes utilizados a lo largo del proyecto. Estas definiciones han sido elaboradas específicamente en el contexto de la plataforma de gestión de TFG desarrollada, proporcionando claridad sobre el significado y uso de cada término.

La comprensión de estos conceptos es fundamental para entender tanto la arquitectura técnica como las funcionalidades del sistema implementado. Cada definición incluye el contexto específico de aplicación dentro del proyecto.

Backend: Conjunto de tecnologías y servicios del lado del servidor que procesan la lógica de negocio, gestionan la base de datos y proporcionan APIs para el frontend.

Bundle: En el contexto de Symfony, un bundle es un plugin que agrupa código relacionado (controladores, servicios, configuración) en una unidad reutilizable.

Componente React: Función o clase de JavaScript que retorna elementos JSX y encapsula lógica de interfaz de usuario reutilizable.

Context API: Sistema de gestión de estado global de React que permite compartir datos entre componentes sin necesidad de pasar props manualmente a través del árbol de componentes.

Custom Hook: Función JavaScript que comienza con "use" y permite extraer y reutilizar lógica de estado entre múltiples componentes React.

Defensa de TFG: Acto académico en el cual el estudiante presenta oralmente su Trabajo de Fin de Grado ante un tribunal evaluador para su calificación final.

Doctrine ORM: Herramienta de mapeo objeto-relacional para PHP que proporciona una capa de abstracción para interactuar con bases de datos relacionales.

Endpoint: URL específica de una API REST que acepta peticiones HTTP y devuelve respuestas estructuradas, representando un recurso o acción del sistema.

Frontend: Parte de la aplicación web que se ejecuta en el navegador del usuario, responsable de la interfaz de usuario y la interacción directa con el usuario final.

Hot Module Replacement (HMR): Tecnología de desarrollo que permite actualizar módulos de código en tiempo real sin perder el estado de la aplicación.

Middleware: Función que se ejecuta durante el ciclo de vida de una petición HTTP, permitiendo modificar la petición o respuesta antes de llegar al destino final.

Migración de Base de Datos: Script que modifica la estructura de la base de datos de manera versionada, permitiendo evolucionar el esquema de datos de forma controlada.

Monorepo: Estrategia de organización de código donde múltiples proyectos relacionados (frontend, backend) se almacenan en un único repositorio Git.

Props: Abreviación de "properties", son argumentos que se pasan a los componentes React para configurar su comportamiento y apariencia.

Protected Route: Ruta de la aplicación que requiere autenticación y/o autorización específica para ser accedida, implementando control de acceso basado en roles.

Responsive Design: Enfoque de diseño web que permite que las interfaces se adapten automáticamente a diferentes tamaños de pantalla y dispositivos.

Serialización: Proceso de convertir objetos de programación en formatos de intercambio de datos como JSON o XML para transmisión o almacenamiento.

State Management: Gestión del estado de la aplicación, refiriéndose a cómo se almacenan, actualizan y comparten los datos entre diferentes partes de la aplicación.

Token de Acceso: Credencial digital temporal que permite a un usuario autenticado

acceder a recursos protegidos de la aplicación sin necesidad de reenviar credenciales.

Tribunal de TFG: Comisión evaluadora compuesta por profesores académicos (presidente, secretario y vocal) responsable de evaluar y calificar las defensas de TFG.

Utility-First CSS: Metodología de CSS que utiliza clases pequeñas y específicas para construir interfaces, característica principal de frameworks como Tailwind CSS.

Validación del lado del servidor: Proceso de verificación y sanitización de datos recibidos en el backend antes de su procesamiento o almacenamiento.

Virtual DOM: Representación en memoria de la estructura DOM real que permite a React calcular eficientemente los cambios mínimos necesarios para actualizar la interfaz.

2. Contexto del proyecto

Este capítulo establece el contexto del proyecto, proporcionando los fundamentos necesarios antes de pasar a la planificacion y el desarrollo. Incluye una descripción detallada del proyecto, un análisis extenso de los perfiles de usuario, la justificación del modelo de desarrollo seleccionado, y una explicación completa de las tecnologías, lenguajes y herramientas utilizadas.

2.1 Descripción general del proyecto

La Plataforma de Gestión de TFG es un sistema web diseñado para automatizar y optimizar la gestión completa de Trabajos de Fin de Grado en universidades. El sistema implementa una arquitectura moderna basada en tecnologías web actuales y proporciona una solución a los procesos actuales de evaluación de los TFG.

La plataforma gestiona el flujo completo del proceso académico, desde la creación inicial del TFG por parte del estudiante hasta la calificación final tras la defensa ante el tribunal. El sistema implementa un modelo de estados bien definido (Borrador → En Revisión → Aprobado → Defendido) que garantiza la trazabilidad y el cumplimiento de los procedimientos académicos establecidos. La arquitectura se basa en la separación de responsabilidades: el frontend en React 19 maneja la presentación e interacción con el usuario, mientras que el backend en Symfony 6.4 LTS gestiona la lógica de negocio, persistencia de datos y seguridad.

El sistema incluye funcionalidades como un calendario interactivo para programar defensas, gestión segura de archivos PDF, y panel administrativo con capacidades para elaborar reportes de errores y exportación de datos.

2.2 Características del usuario

El sistema satisface las necesidades de cuatro perfiles de usuario diferenciados, cada uno con roles, permisos y flujos de trabajo específicos.

2.2.1 Estudiante

Perfil: Estudiante universitario en proceso de realización de su Trabajo de Fin de Grado, con conocimientos básicos de tecnologías web y experiencia en el uso de plataformas académicas digitales.

Responsabilidades principales: El estudiante debe crear y mantener actualizada la información básica de su TFG: título, resumen y palabras clave. Gestiona la subida y actualización de archivos PDF, asegurando que el sistema tenga siempre la versión más reciente. Realiza seguimiento del estado de progreso de su TFG y consulta regularmente los comentarios del tutor para integrarlos en nuevas versiones. También visualiza la información de su defensa (fecha, tribunal, aula) y gestiona las notificaciones del sistema sobre cambios de estado.

Competencias técnicas esperadas: Manejo básico de navegadores web modernos y formularios online. Capacidad para subir y descargar archivos de manera segura, comprendiendo aspectos básicos de seguridad. Conocimientos básicos de gestión documental digital (control de versiones, nomenclatura de archivos). Familiaridad con herramientas de notificación electrónica.

2.2.2 Profesor/Tutor

Perfil: Docente universitario con experiencia en dirección de TFG, responsable de la supervisión académica y evaluación de trabajos asignados.

Responsabilidades principales: Supervisa el progreso de todos los TFG asignados, asegurando que los estudiantes mantengan un ritmo adecuado de trabajo. Realiza revisiones periódicas y evaluaciones de los documentos. Proporciona feedback mediante el sistema de comentarios, ofreciendo orientación específica para la mejora del trabajo. Gestiona los cambios de estado de los TFG bajo su supervisión, tomando decisiones sobre la aprobación para defensa. Participa en tribunales de evaluación como miembro experto y coordina con otros miembros para la programación de defensas.

Competencias técnicas esperadas: Experiencia sólida en evaluación de trabajos académicos, metodologías de investigación y criterios de calidad académica. Manejo avanzado de herramientas digitales de gestión académica para seguimiento, evaluación y comunicación con estudiantes. Capacidad para proporcionar feedback constructivo a través de plataformas digitales. Comprensión de flujos de trabajo colaborativos online, coordinación con colegas y gestión de calendarios compartidos.

2.2.3 Presidente del Tribunal

Perfil: Profesor universitario con experiencia avanzada en evaluación académica, responsable de liderar tribunales de evaluación y coordinar el proceso de defensas.

Responsabilidades principales: Crea y configura tribunales de evaluación según criterios académicos, asegurando la composición adecuada para evaluar los trabajos asignados. Realiza la asignación de miembros de tribunal considerando expertise técnico, carga de trabajo y equilibrio de roles. Programa fechas y horarios de defensas utilizando el calendario integrado, optimizando recursos y minimizando conflictos. Coordina la disponibilidad entre miembros del tribunal y supervisa el proceso de evaluación, manteniendo estándares académicos. Genera actas de defensa y documentación oficial requerida.

Competencias técnicas esperadas: Experiencia avanzada en gestión de procesos académicos, normativas universitarias y procedimientos de evaluación. Capacidad de liderazgo y coordinación de equipos multidisciplinarios. Manejo experto de herramientas de calendario y programación digital, gestión de recursos y resolución de conflictos de agenda. Comprensión de procedimientos administrativos universitarios, aspectos legales y documentación oficial.

2.2.4 Administrador

Perfil: Personal técnico o administrativo responsable de la gestión global del sistema, con conocimientos avanzados en administración de plataformas web y gestión de usuarios.

Responsabilidades principales: Gestiona el catálogo completo de usuarios mediante operaciones CRUD, garantizando la integridad de la base de usuarios. Asigna y modifica roles y permisos de acceso, asegurando que cada usuario tenga los privilegios necesarios sin comprometer la seguridad. Genera reportes estadísticos del sistema para la toma de decisiones académicas. Facilita la exportación de datos en múltiples formatos (PDF, Excel, CSV). Configura y mantiene parámetros del sistema, supervisando el funcionamiento general y coordinando tareas de mantenimiento.

Competencias técnicas esperadas: Experiencia avanzada en administración de sistemas web complejos, arquitecturas de aplicaciones, servidores web y bases de datos. Conocimientos sólidos en gestión de bases de datos relacionales, generación de reportes y análisis de rendimiento. Capacidad analítica para interpretación de estadísticas y métricas del sistema, identificación de tendencias y patrones de uso. Comprensión de seguridad informática y gestión de accesos, políticas de seguridad y cumplimiento de normativas de protección de datos.

2.3 Modelo de ciclo de vida

La selección del modelo de ciclo de vida es una decisión estratégica que determina la estructura, organización y metodología del proceso de desarrollo. Esta elección impacta en la gestión de riesgos, la adaptación a cambios y la entrega de valor del proyecto.

El desarrollo de la plataforma sigue un **modelo iterativo incremental**, estructurado en ocho fases que permiten la entrega progresiva de funcionalidades y la validación continua de requisitos. Este enfoque facilita la identificación temprana de problemas, permite ajustes del flujo del proyecto y garantiza que cada incremento aporte valor al producto final.

2.3.1 Metodología de desarrollo

Enfoque adoptado: El proyecto implementa una metodología ágil adaptada al contexto académico, combinando elementos de Scrum para la gestión con prácticas de desarrollo incremental que permiten la entrega con gran valor en cada fase.

Justificación de la metodología: Esta metodología proporciona flexibilidad para adaptarse a cambios de requisitos durante el desarrollo, es una característica esencial en proyectos académicos donde los requerimientos pueden evolucionar. Permite la validación temprana del sistema ya que cada fase entrega funcionalidades completas y operativas, facilitando la detección de problemas antes de seguir con el desarrollo. Facilita feedback continuo que permite realizar ajustes basados en la evaluación de fases o entregas anteriores.

2.3.2 Fases del proyecto

Fase 1-2: Fundación del sistema (Semanas 1-2) - Configuración del entorno de desarrollo. - Implementación del sistema de enrutamiento y navegación. - Desarrollo del sistema de autenticación básico. - Establecimiento de la arquitectura de componentes React.

Fase 3: Módulo de estudiante (Semanas 3-4) - Implementación completa de funcionalidades para estudiantes. - Interfaces de seguimiento de estado de TFG. - Integración con sistema de notificaciones.

Fase 4: Módulo de profesor (Semanas 4-5) - Desarrollo de herramientas de supervisión para tutores. - Sistema de feedback y comentarios estructurados. - Interfaces de

gestión de TFG asignados.

Fase 5: Sistema de defensas (Semanas 5-6) - Implementación del calendario interactivo con FullCalendar.js. - Sistema de gestión de tribunales. - Programación y coordinación de defensas. - Gestión de disponibilidad de miembros de tribunal.

Fase 6: Panel administrativo (Semanas 6-7) - Sistema completo de gestión de usuarios (CRUD). - Generación de reportes de errores y estadísticas avanzadas. - Funcionalidades de exportación de datos. - Configuración global del sistema.

Fase 7: Backend Symfony (Semanas 7-9) - Implementación completa del backend con Symfony 6.4 LTS. - Desarrollo de APIs REST con API Platform. - Sistema de subida y gestión de archivos. - Sistema de autenticación JWT con refresh tokens. - Migración de datos desde sistema mock a base de datos MySQL.

Fase 8: Pulimiento final (Semanas 9-10) - Testing exhaustivo (unitario, integración y E2E). - Optimización de rendimiento. - Configuración de despliegue en producción. - Documentación técnica y manuales de usuario.

2.4 Tecnologías

La elección de tecnologías es fundamental en el desarrollo de cualquier sistema software. Estas decisiones impactan directamente en la escalabilidad, mantenibilidad, rendimiento y viabilidad a largo plazo del proyecto. Esta sección detalla las principales tecnologías utilizadas, explicando las razones de su selección y cómo contribuyen al cumplimiento de los objetivos.

2.4.1 React 19

React 19 constituye la biblioteca principal para el desarrollo del frontend de la aplicación, proporcionando un marco de trabajo robusto para la construcción de interfaces de usuario interactivas y componentes reutilizables.

Ventajas para el proyecto: React ofrece un ecosistema maduro con amplia disponibilidad de librerías y componentes de terceros que aceleran el desarrollo, reduciendo la implementación desde cero. Proporciona rendimiento optimizado mediante su Virtual DOM y la curva de aprendizaje es razonable gracias a documentación extensa y una comunidad activa. Ofrece una gran compatibilidad con herramientas modernas de desarrollo y testing, incluyendo DevTools, frameworks de testing y herramientas de build como Vite.

2.4.2 Symfony 6.4 LTS

Symfony 6.4 LTS se utiliza como framework principal para el desarrollo del backend, proporcionando una arquitectura sólida basada en componentes modulares y principios de desarrollo empresarial.

Ventajas para el proyecto: La versión LTS (Long Term Support) garantiza soporte continuo y actualizaciones de seguridad hasta noviembre de 2027, proporcionando estabilidad a largo plazo. Su arquitectura modular ofrece flexibilidad para utilizar únicamente los componentes necesarios. El cumplimiento de estándares PSR asegura interoperabilidad con otras librerías de la comunidad PHP y facilita el mantenimiento del código.

2.4.3 MySQL 8.0

MySQL 8.0 actúa como sistema de gestión de base de datos relacional, proporcionando persistencia segura y eficiente para todos los datos del sistema.

Ventajas para el proyecto: MySQL 8.0 ofrece fiabilidad excepcional como sistema probado en entornos de producción exigentes, con millones de instalaciones que demuestran su estabilidad. Proporciona cumplimiento completo de propiedades ACID que garantizan consistencia e integridad de datos, fundamentales para un sistema académico donde la pérdida de información de TFG sería inaceptable. Ofrece capacidades de escalabilidad horizontal y vertical, permitiendo crecimiento sin degradación significativa del rendimiento. Cuenta con un ecosistema rico de herramientas de administración y monitorización que facilitan el mantenimiento operativo y optimización continua.

2.4.4 JWT Authentication (LexikJWTAuthenticationBundle)

La autenticación JWT proporciona un sistema de seguridad stateless, escalable y moderno para el control de acceso a la aplicación.

Implementación específica: El sistema utiliza access tokens de corta duración (1 hora) para operaciones sensibles, minimizando la exposición en caso de compromiso. Se implementan refresh tokens de larga duración (30 días) que permiten renovación automática sin reautenticación constante, equilibrando seguridad con experiencia de usuario. La información de roles está incluida en el payload del token mediante peticiones, eliminando consultas adicionales a la base de datos. La seguridad se maximiza mediante el algoritmo RS256 que utiliza firma asimétrica.

2.4.5 FullCalendar.js

FullCalendar.js proporciona la funcionalidad de calendario interactivo esencial para la gestión visual de defensas y programación de eventos académicos, transformando la coordinación tradicional de fechas en una experiencia intuitiva y visual.

Implementación específica: El sistema integra múltiples vistas de calendario (mensual, semanal y diaria) que permiten diferentes niveles de detalle según las necesidades de programación. La funcionalidad drag & drop facilita la reprogramación de defensas mediante arrastre visual, eliminando la necesidad de formularios complejos. La adaptación responsiva asegura funcionalidad completa en dispositivos móviles, manteniendo la usabilidad en tablets y smartphones.

2.4.6 Tailwind CSS v4

Tailwind CSS v4 actúa como framework de estilos, proporcionando un sistema de diseño consistente y eficiente que unifica el diseño en toda la aplicación mediante un enfoque altamente escalable y mantenible.

Implementación específica: El sistema permite la construcción de interfaces complejas mediante la composición de clases personalizadas, eliminando la necesidad de escribir CSS personalizado. La implementación responsive utiliza breakpoints móvil-first que aseguran una experiencia óptima en dispositivos de cualquier tamaño, con transiciones suaves entre diferentes resoluciones. La preparación para modo oscuro mediante CSS custom properties facilita futuras implementaciones de temas alternativos sin refactoring del código base.

2.4.7 DDEV

DDEV proporciona un entorno de desarrollo containerizado que garantiza consistencia absoluta entre diferentes máquinas de desarrollo y facilita significativamente la incorporación de nuevos desarrolladores, eliminando los problemas tradicionales de configuración de entorno.

Ventajas para el proyecto: La consistencia se garantiza mediante un entorno idéntico independientemente del sistema operativo host (Windows, macOS, Linux), eliminando por completo los problemas de ”funciona en mi máquina”. La facilidad de setup permite levantar todo el entorno de desarrollo con un único comando, reduciendo el tiempo de configuración inicial de horas a minutos. El aislamiento mediante contenedores asegura que las dependencias del proyecto no interfieran con el sistema host ni con otros proyectos,

manteniendo limpio el entorno de desarrollo.

2.5 Lenguajes

Los lenguajes de programación seleccionados se eligieron considerando madurez, rendimiento, ecosistema de desarrollo y compatibilidad con las tecnologías del stack principal. Esta sección detalla las características específicas utilizadas de cada lenguaje y los patrones de programación aplicados.

2.5.1 JavaScript/TypeScript

JavaScript se utiliza como lenguaje principal para el desarrollo del frontend, aprovechando las características modernas de ECMAScript 2023 y manteniendo una arquitectura preparada para migración incremental hacia TypeScript conforme evolucionen los requisitos de tipado del proyecto.

Patrones de programación aplicados: La programación funcional se implementa mediante uso extensivo de métodos como map, filter y reduce para transformaciones de datos inmutables. El enfoque declarativo prevalece sobre el imperativo, describiendo qué debe ocurrir en lugar de cómo debe implementarse, mejorando la legibilidad y mantenibilidad del código.

2.5.2 PHP 8.2+

PHP 8.2+ actúa como lenguaje de backend robusto, aprovechando las significativas mejoras de rendimiento y las características avanzadas de tipado fuerte introducidas en versiones recientes, proporcionando una base sólida para la API y lógica de negocio del sistema.

Principios de programación aplicados: Los principios SOLID guían todo el diseño orientado a objetos, implementando responsabilidad única, principio abierto/cerrado, sustitución de Liskov, segregación de interfaces e inversión de dependencias para código mantenible y extensible. La dependency injection se utiliza sistemáticamente para inversión de control, mejorando significativamente la testabilidad del código y permitiendo mock de dependencias durante testing. El cumplimiento de estándares PSR asegura interoperabilidad con el ecosistema PHP, incluyendo PSR-4 para autoloading, PSR-12 para estilo de código y PSR-15 para middleware HTTP.

2.5.3 SQL

SQL se utiliza para la definición de esquemas de base de datos, consultas analíticas complejas y procedimientos de migración segura, aprovechando las características avanzadas y optimizaciones de rendimiento específicas de MySQL 8.0.

Implementación específica: El DDL avanzado define esquemas completos con constraints sofisticados, índices optimizados para patrones de consulta específicos y relaciones complejas que garantizan integridad referencial y performance consistente.

2.5.4 HTML/CSS

HTML5 y CSS3 proporcionan la estructura semántica robusta y presentación visual cohesiva de la aplicación, implementando rigurosamente estándares web modernos y las mejores prácticas de accesibilidad para garantizar una experiencia inclusiva y compatible con tecnologías asistivas.

2.6 Herramientas

La selección apropiada de herramientas de desarrollo, testing y gestión de proyecto es determinante en la productividad y calidad del desarrollo software. Las herramientas elegidas deben integrarse eficientemente, proporcionando un flujo de trabajo fluido que minimice los errores y maximice la capacidad de desarrollo y debugging.

Esta sección detalla las principales herramientas utilizadas durante el ciclo de vida del proyecto, explicando su configuración específica y las ventajas que aportan al proceso de desarrollo.

2.6.1 Visual Studio Code

VS Code actúa como IDE principal de desarrollo, meticulosamente configurado con extensiones específicas para el stack tecnológico del proyecto que maximizan la productividad y minimizan los errores durante el desarrollo full-stack.

2.6.2 Vite

Vite se utiliza como build tool moderno y servidor de desarrollo para el frontend, proporcionando una experiencia de desarrollo excepcionalmente optimizada mediante Hot Module Replacement ultra-rápido y arquitectura basada en módulos ES nativos.

Ventajas para el proyecto: El plugin oficial de React proporciona soporte completo y optimizado para React y JSX, incluyendo Fast Refresh que preserva el estado de componentes durante recarga. La integración de ESLint en tiempo de desarrollo muestra errores de código instantáneamente en el browser, mejorando la calidad del código durante la escritura. La preparación para PWA mediante vite-plugin-pwa establece la base para futuras funcionalidades offline y notificaciones push, anticipando evoluciones del sistema hacia aplicaciones más ricas.

2.6.3 Composer

Composer gestiona las dependencias PHP del backend, garantizando versiones consistentes, resolución automática de dependencias y reproducibilidad completa de entornos entre diferentes fases de desarrollo y producción.

2.6.4 Docker / DDEV

Docker proporciona containerización robusta del entorno de desarrollo, mientras DDEV ofrece una capa de abstracción especializada para desarrollo web que simplifica la gestión de contenedores y optimiza el flujo de trabajo full-stack.

2.6.5 Git / GitHub

Git actúa como sistema de control de versiones distribuido, mientras GitHub proporciona hosting remoto centralizado, herramientas avanzadas de colaboración y pipelines automatizados de CI/CD que facilitan el desarrollo colaborativo y la entrega continua.

3. Planificación

Este capítulo detalla la planificación temporal y de metodología seguida durante el desarrollo de la plataforma: desde la trabajo inicial hasta la entrega final. Se presenta la estructuración en fases diferenciadas y la gestión de recursos temporales del proyecto.

La plataforma de gestión de TFG se desarrolló siguiendo un modelo iterativo incremental que permite entregas funcionales progresivas y validación continua de requisitos. Este enfoque divide el desarrollo en iteraciones bien definidas donde cada fase aborda análisis, diseño, implementación y pruebas de forma integrada.

Una planificación efectiva resulta fundamental para el éxito de cualquier proyecto de software, especialmente en el contexto académico donde los plazos son estrictos y los recursos limitados. Este capítulo establece las bases de la metodología que guiaron todo el proceso de desarrollo y justifica las decisiones temporales adoptadas.

3.1 Iniciación del proyecto

3.1.1 Contexto de inicio

La idea del proyecto "Plataforma de Gestión de TFG" nació al darme cuenta de que los procesos tradicionales de gestión de los Trabajos de Fin de Grado en la universidad podían mejorar. Al observar la plataforma que ya existía, pensé que sería posible crear una versión más completa y eficiente, que facilitara la coordinación entre estudiantes, tutores y coordinadores.

Decidí embarcarme en el desarrollo porque vi una buena oportunidad: hoy en día las tecnologías web están lo suficientemente maduras como para crear aplicaciones robustas de forma ágil, además yo ya tenía experiencia trabajando con herramientas modernas como React y Symfony. Con esos recursos, me propuse diseñar una solución que integrara en un solo espacio a todos los actores implicados en el proceso académico, mejorando la eficiencia, la trazabilidad y la comunicación.

3.1.2 Análisis de viabilidad

Viabilidad técnica: El proyecto demuestra alta viabilidad técnica al basarse en tecnologías consolidadas y bien documentadas. React 19 y Symfony 6.4 LTS ofrecen ecosis-

temas maduros respaldados por comunidades activas y extensiva documentación. La arquitectura seleccionada (aplicación SPA con API backend) representa un patrón probado que garantiza escalabilidad y mantenibilidad a largo plazo.

Viabilidad temporal: La planificación de 10 semanas estructurada en 8 fases iterativas facilita un desarrollo incremental con validaciones continuas. El conocimiento previo de las tecnologías empleadas minimiza los riesgos de retrasos significativos y permite estimaciones más precisas de los tiempos de desarrollo.

Viabilidad de recursos: El desarrollo requiere exclusivamente herramientas de software libre y recursos educativos de acceso gratuito. La containerización mediante DDEV asegura un entorno de desarrollo consistente independientemente de la plataforma de hardware utilizada.

3.1.3 Definición del alcance inicial

El alcance inicial del proyecto se definió estableciendo los requisitos mínimos viables (MVP) para cada perfil de usuario, asegurando que cada rol disponga de las funcionalidades esenciales para sus responsabilidades académicas.

El MVP para **Estudiantes** incluye las capacidades fundamentales de gestión de TFG: subida de documentos con validación automática, seguimiento del estado del trabajo a través del flujo académico establecido, y acceso al feedback del tutor para facilitar las mejoras continuas. Para **Profesores**, el sistema proporciona herramientas de supervisión centralizadas con vista unificada de todos los TFG asignados, sistema de comentarios estructurado para feedback constructivo, y gestión de estados del trabajo conforme avanza el proceso de supervisión. El **Presidente de Tribunal** dispone de funcionalidades para crear y configurar tribunales académicos, asignar miembros evaluadores y programar defensas coordinando disponibilidades y recursos. El rol de **Administrador** cuenta con gestión completa de usuarios mediante operaciones CRUD, generación de reportes de seguimiento del sistema y configuración de parámetros adaptativos a políticas institucionales.

Esta aproximación MVP facilita la validación temprana de conceptos y permite el refinamiento progresivo de funcionalidades basado en el feedback de usuarios reales.

3.2 Iteraciones del proceso de desarrollo

El desarrollo de la plataforma se estructura mediante una metodología iterativa incremental que se materializa en ocho fases diferenciadas. Cada fase cuenta con objetivos específicos, criterios de aceptación claros y entregables funcionales que aportan valor al producto final. Esta aproximación facilita la gestión de la complejidad del sistema y permite adaptaciones continuas basadas en la validación de resultados.

3.2.1 Fase 1-2: Setup inicial y autenticación (Semanas 1-2)

Objetivos de la fase: Esta fase inicial establece los cimientos tecnológicos del proyecto, configurando la arquitectura frontend base, implementando el sistema de navegación con protección por roles, desarrollando un sistema de autenticación funcional y estableciendo las herramientas de desarrollo que garantizan la calidad del código.

Actividades principales:

Semana 1: Configuración del entorno - Inicialización del proyecto React utilizando Vite como herramienta de construcción. - Configuración de Tailwind CSS v4 y establecimiento del sistema de diseño base de la aplicación. - Instalación y configuración de ESLint, Prettier y herramientas complementarias para mantener la calidad del código. - Desarrollo de componentes fundamentales de Layout y navegación que servirán como base para toda la aplicación.

Semana 2: Sistema de autenticación - Implementación del AuthContext para la gestión centralizada del estado de autenticación. - Desarrollo de los componentes de interfaz para inicio de sesión y registro de usuarios. - Creación del sistema ProtectedRoute que valida permisos según el rol del usuario. - Configuración de la persistencia de sesión utilizando localStorage. - Implementación de tests básicos para validar los flujos de autenticación principales.

Entregables: Al finalizar esta fase se obtiene una aplicación React completamente funcional con navegación adaptativa por roles, un sistema de autenticación mock completamente operativo, una arquitectura de componentes sólida y bien estructurada, y documentación técnica detallada de las decisiones arquitectónicas adoptadas.

Criterios de aceptación: Los cuatro perfiles de usuario definidos pueden autenticarse correctamente en el sistema. El sistema de rutas protege adecuadamente el acceso basándose en el rol del usuario autenticado. La interfaz responde correctamente en diferentes dispositivos siguiendo el sistema de diseño establecido. Todo el código desarrollado cumple

los estándares de calidad definidos por las herramientas de linting configuradas.

3.2.2 Fase 3: Módulo de estudiante (Semanas 3-4)

Objetivos de la fase: Esta fase se centra en desarrollar todas las funcionalidades necesarias para el perfil de estudiante, implementando un sistema de gestión de archivos funcional, creando interfaces intuitivas para el seguimiento del estado del TFG e integrando un sistema básico de notificaciones que mantenga informados a los usuarios.

Actividades principales:

Semana 3: Gestión de TFG - Desarrollo del custom hook useTFGs que encapsula toda la lógica de negocio relacionada con la gestión de trabajos. - Implementación de formularios intuitivos para la creación y edición de TFG con validaciones en tiempo real. - Construcción de un sistema robusto de subida de archivos que incluye validación de formatos y seguimiento del progreso de carga. - Diseño de interfaces claras para la visualización de TFG que muestren metadatos de forma organizada y accesible.

Semana 4: Seguimiento y notificaciones - Implementación completa del sistema de estados que gestiona la transición Borrador → En Revisión → Aprobado → Defendido con validaciones apropiadas. - Desarrollo de componentes visuales tipo timeline que permiten a los estudiantes hacer seguimiento claro del progreso de su trabajo. - Integración del NotificacionesContext para proporcionar feedback inmediato sobre cambios de estado. - Creación de interfaces especializadas para la visualización de comentarios y feedback proporcionado por el tutor.

Entregables: Al completar esta fase se obtiene un módulo de estudiante completamente funcional y operativo, un sistema integral de subida y gestión de archivos con validaciones apropiadas, interfaces claras y efectivas para el seguimiento del estado del TFG, y un sistema de notificaciones totalmente integrado que mantiene informados a los usuarios.

Criterios de aceptación: Los estudiantes pueden crear, editar y subir archivos de TFG de manera intuitiva y sin errores. El sistema de estados funciona correctamente implementando todas las validaciones necesarias para garantizar la integridad del flujo académico. Las notificaciones se muestran de forma inmediata y clara cuando ocurren cambios relevantes. Todas las interfaces desarrolladas son intuitivas, accesibles y completamente responsive en diferentes dispositivos.

3.2.3 Fase 4: Módulo de profesor (Semanas 4-5)

Objetivos de la fase: Esta fase desarrolla las herramientas especializadas de supervisión necesarias para profesores tutores, implementa un sistema estructurado de feedback académico, crea interfaces eficientes para la gestión de TFG asignados e integra capacidades de cambio de estado con las validaciones de permisos apropiadas.

Actividades principales:

Semana 4 (solapada): Bases del módulo profesor - Desarrollo de interfaces completas para el listado y gestión de todos los TFG asignados al profesor. - Implementación de sistemas avanzados de filtros y búsqueda que permiten organizar trabajos por estado, estudiante y fechas relevantes. - Construcción de un sistema robusto de visualización de archivos PDF que facilite la revisión de documentos subidos por estudiantes.

Semana 5: Sistema de feedback y evaluación - Desarrollo de formularios especializados para comentarios estructurados que faciliten feedback constructivo y detallado. - Implementación de un sistema integral de calificaciones y evaluaciones académicas con criterios claros. - Creación de interfaces intuitivas para cambio de estado que incluyen validación de permisos y flujos de aprobación. - Integración completa con el sistema de notificaciones para asegurar que los estudiantes reciban feedback inmediato sobre cambios relevantes.

Entregables: Al finalizar esta fase se cuenta con un módulo de profesor completamente funcional y optimizado para tareas de supervisión, un sistema comprensivo de feedback y comentarios que facilita la comunicación académica, interfaces especializadas para evaluación y gestión de estados de TFG, y validaciones de permisos por rol completamente operativas y seguras.

Criterios de aceptación: Los profesores pueden gestionar de manera eficiente y efectiva todos sus TFG asignados utilizando herramientas intuitivas. El sistema de comentarios facilita la provisión de feedback estructurado y constructivo de calidad académica. Los cambios de estado se comunican de manera apropiada y automática a los estudiantes correspondientes. Las validaciones de permisos funcionan correctamente garantizando la seguridad e integridad del sistema académico.

3.2.4 Fase 5: Sistema de defensas y calendario (Semanas 5-6)

Objetivos de la fase: Esta fase integra FullCalendar.js para proporcionar gestión visual avanzada de defensas académicas, implementa un sistema completo de gestión de tribunales, desarrolla funcionalidades intuitivas para la programación de defensas y crea un

sistema automatizado de coordinación de disponibilidad entre todos los actores académicos.

Actividades principales:

Semana 5 (solapada): Integración de calendario - Instalación y configuración completa de FullCalendar.js optimizada para React con todas sus funcionalidades avanzadas. - Desarrollo del custom hook useCalendario que encapsula toda la lógica de gestión de eventos y estados del calendario. - Implementación de múltiples vistas especializadas (mensual, semanal y diaria) que faciliten diferentes niveles de detalle según las necesidades. - Configuración de eventos personalizados específicamente diseñados para representar defensas académicas con toda su información relevante.

Semana 6: Gestión de tribunales y defensas - Desarrollo completo del módulo de creación y gestión de tribunales académicos con todas las validaciones institucionales necesarias. - Implementación de un sistema inteligente de asignación de miembros de tribunal que considere expertise y disponibilidad. - Creación de interfaces avanzadas de programación de defensas que incluyen funcionalidad drag drop para facilitar la reorganización intuitiva. - Desarrollo de un sistema comprensivo de notificaciones que mantenga informados tanto a tribunales como a estudiantes sobre cambios y actualizaciones.

Entregables: Al completar esta fase se obtiene un calendario interactivo completamente funcional con todas las características avanzadas necesarias, un sistema robusto de gestión de tribunales completamente operativo, funcionalidades completas de programación de defensas que faciliten la coordinación académica, y un sistema automatizado de coordinación de disponibilidad que optimice la planificación de recursos.

Criterios de aceptación: El calendario muestra de manera clara y precisa todas las defensas programadas con información detallada y actualizada. Los tribunales pueden crearse, modificarse y gestionarse de manera eficiente siguiendo los procedimientos académicos establecidos. La programación de defensas resulta intuitiva y completamente funcional para todos los usuarios autorizados. Las notificaciones se envían de manera automática y oportuna a todos los actores relevantes en cada proceso de defensa.

3.2.5 Fase 6: Panel administrativo (Semanas 6-7)

Objetivos de la fase: Esta fase desarrolla un sistema integral de gestión de usuarios con operaciones CRUD completas, implementa capacidades avanzadas de generación de reportes y estadísticas institucionales, crea funcionalidades robustas de exportación de datos en múltiples formatos y establece un sistema de configuración global que permita

adaptar la plataforma a diferentes contextos institucionales.

Actividades principales:

Semana 6 (solapada): Gestión de usuarios - Desarrollo del custom hook useUsuarios que encapsula toda la lógica de gestión de usuarios y permisos del sistema. - Implementación de interfaces CRUD completas y intuitivas para la gestión eficiente de todos los usuarios de la plataforma. - Construcción de un sistema robusto de asignación de roles que incluye validaciones de seguridad y flujos de aprobación. - Desarrollo de filtros avanzados y funcionalidades de búsqueda que faciliten la localización y gestión de usuarios específicos.

Semana 7: Reportes y configuración - Desarrollo del custom hook useReportes que gestiona toda la lógica de generación y procesamiento de informes estadísticos. - Implementación de dashboards interactivos con visualizaciones estadísticas claras que proporcionen insights valiosos sobre el funcionamiento del sistema. - Construcción de un sistema comprensivo de exportación que permita generar archivos en formatos PDF y Excel con datos estructurados. - Creación de interfaces especializadas para la configuración global del sistema que permitan personalización institucional.

Entregables: Al finalizar esta fase se cuenta con un panel administrativo completo y completamente funcional que facilite todas las tareas de gestión, un sistema avanzado de reportes con múltiples tipos de visualizaciones y análisis, funcionalidades de exportación completamente operativas que generen archivos profesionales, y un sistema de configuración implementado que permita adaptación institucional.

Criterios de aceptación: La gestión de usuarios permite realizar todas las operaciones CRUD de manera eficiente y segura con validaciones apropiadas. Los reportes proporcionan insights valiosos y accionables sobre el funcionamiento y uso del sistema. Las exportaciones generan archivos correctamente formateados y completos que cumplan estándares profesionales. La configuración global afecta de manera apropiada y consistente el comportamiento de todo el sistema según las necesidades institucionales.

3.2.6 Fase 7: Backend Symfony (Semanas 7-9)

Objetivos de la fase: Esta fase crucial implementa un backend robusto y completo utilizando Symfony 6.4 LTS, desarrolla APIs REST mediante API Platform 3.x, migra completamente del sistema mock inicial hacia persistencia real con MySQL, e implementa un sistema de autenticación JWT seguro con refresh tokens para garantizar la seguridad y escalabilidad del sistema.

Actividades principales:

Semana 7: Setup y arquitectura backend - Configuración completa del proyecto Symfony utilizando DDEV para garantizar un entorno de desarrollo consistente y reproducible. - Definición detallada de todas las entidades Doctrine necesarias (User, TFG, Tribunal, Defensa, etc.) con sus relaciones y validaciones correspondientes. - Configuración robusta de la base de datos MySQL incluyendo migraciones iniciales y datos de prueba. - Instalación y configuración de API Platform con serialización personalizada para cada entidad del sistema.

Semana 8: APIs y autenticación - Implementación completa y exhaustiva de todos los endpoints REST necesarios para cada funcionalidad del sistema. - Configuración avanzada de LexikJWTAuthenticationBundle para autenticación segura basada en tokens. - Desarrollo de un sistema comprensivo de roles y permisos utilizando Symfony Security para garantizar acceso apropiado. - Integración de VichUploaderBundle para gestión segura y eficiente de archivos subidos por usuarios.

Semana 9: Integración y testing - Establecimiento de conexión completa y funcional entre frontend y backend con validación de todos los flujos de datos. - Implementación de un sistema robusto de notificaciones por email para mantener informados a los usuarios. - Desarrollo de testing de APIs utilizando PHPUnit para garantizar calidad y confiabilidad. - Optimización de consultas de base de datos y análisis de rendimiento para asegurar respuesta eficiente del sistema.

Entregables: Al completar esta fase se obtiene un backend Symfony completamente funcional y optimizado, APIs REST documentadas utilizando estándares OpenAPI, un sistema de autenticación JWT completamente operativo y seguro, y una integración frontend-backend completamente validada y funcional.

Criterios de aceptación: Todas las funcionalidades desarrolladas en el frontend funcionan perfectamente con las APIs reales implementadas. El sistema de autenticación JWT proporciona seguridad robusta y funcionalidad completa para todos los perfiles de usuario. Las APIs están correctamente documentadas siguiendo estándares profesionales y completamente testeadas con cobertura apropiada. El rendimiento general del sistema cumple y supera los objetivos de respuesta establecidos inicialmente.

3.2.7 Fase 8: Pulimiento final (Semanas 9-10)

Objetivos de la fase: Esta fase final se enfoca en realizar testing exhaustivo de toda la aplicación, optimizar el rendimiento general y la experiencia de usuario para garantizar calidad profesional, configurar completamente el entorno de despliegue en producción y completar toda la documentación técnica y manuales de usuario necesarios para el

funcionamiento institucional del sistema.

Actividades principales:

Semana 9 (solapada): Testing y optimización - Implementación de testing end-to-end utilizando herramientas especializadas para validar todos los flujos de usuario. - Optimización detallada de consultas de base de datos y análisis de rendimiento para garantizar respuestas eficientes. - Implementación de mejoras de experiencia de usuario basadas en testing de usabilidad y feedback de usuarios reales. - Identificación y corrección sistemática de todos los bugs detectados durante el proceso de testing integral del sistema.

Semana 10: Despliegue y documentación - Configuración completa del entorno de producción utilizando Docker para garantizar consistencia y escalabilidad. - Finalización de la documentación técnica completa que incluya arquitectura, APIs y procedimientos de mantenimiento. - Creación de manuales de usuario detallados y específicos para cada rol del sistema que faciliten la adopción institucional.

Entregables: Al completar esta fase final se obtiene una aplicación completamente testeada, optimizada y lista para uso profesional, una configuración de producción robusta y completamente operativa, documentación técnica y manuales de usuario completos y profesionales, y un sistema completamente preparado y validado para despliegue exitoso en entorno de producción.

Criterios de aceptación: Todos los tests implementados (unitarios, de integración y end-to-end) pasan exitosamente validando la funcionalidad completa del sistema. El sistema cumple y supera todos los criterios de rendimiento establecidos inicialmente garantizando experiencia de usuario óptima. La documentación está completa, es comprensible y proporciona toda la información necesaria para operación y mantenimiento. El despliegue en producción se ejecuta de manera exitosa y el sistema funciona de forma estable en entorno real.

3.3 Diagrama de Gantt

La visualización temporal del proyecto a través de diagramas de Gantt proporciona una perspectiva clara de la planificación y seguimiento efectivo del progreso. Estos diagramas facilitan la identificación de dependencias críticas entre fases, la optimización en la distribución de esfuerzo y el establecimiento de hitos de control para evaluar el avance del desarrollo.

Los cronogramas presentados ilustran la distribución temporal de las actividades principales del proyecto, destacando las dependencias entre fases y los solapamientos estratégi-

cos planificados para maximizar la eficiencia del desarrollo. Esta representación visual facilita la comprensión de la secuencia lógica de actividades y permite identificar tanto la ruta crítica del proyecto como las oportunidades existentes para paralelizar el trabajo de desarrollo.

3.3.1 Cronograma general del proyecto

El cronograma general proporciona una perspectiva temporal integral del proyecto, abarcando desde la configuración inicial hasta la entrega del producto final en producción. La planificación se estructura en 8 fases distribuidas estratégicamente a lo largo de 10 semanas, optimizando el uso de recursos disponibles y asegurando el cumplimiento de los plazos establecidos.

La metodología iterativa adoptada facilita solapamientos estratégicos entre fases, particularmente entre el desarrollo frontend y la preparación del backend, maximizando la eficiencia del proceso global. El cronograma incluye hitos de validación al finalizar cada fase, garantizando la calidad progresiva del producto desarrollado, como se muestra en la Figura 3.1.

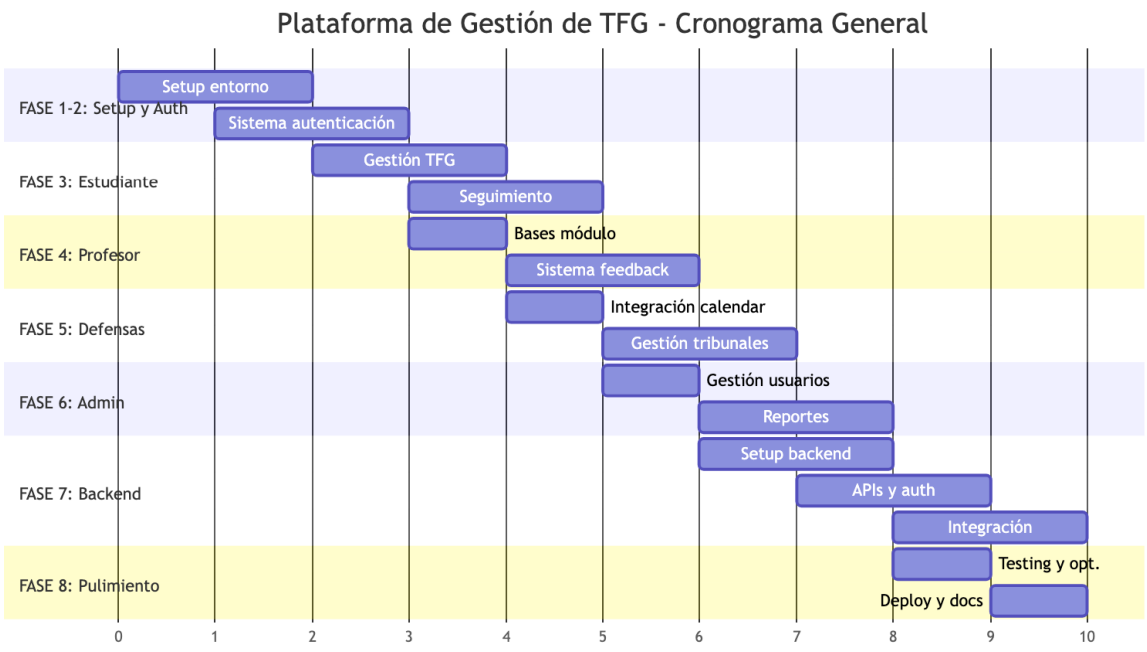

Figure 3.1: Cronograma General

3.3.2 Hitos principales y dependencias

El cronograma principal detalla los hitos críticos y las dependencias entre las diferentes fases del proyecto, facilitando la identificación de puntos de control y la gestión de riesgos temporales. Esta visualización complementaria permite un análisis más granular de la secuencia de actividades y sus interdependencias, como se muestra en la Figura 3.2.

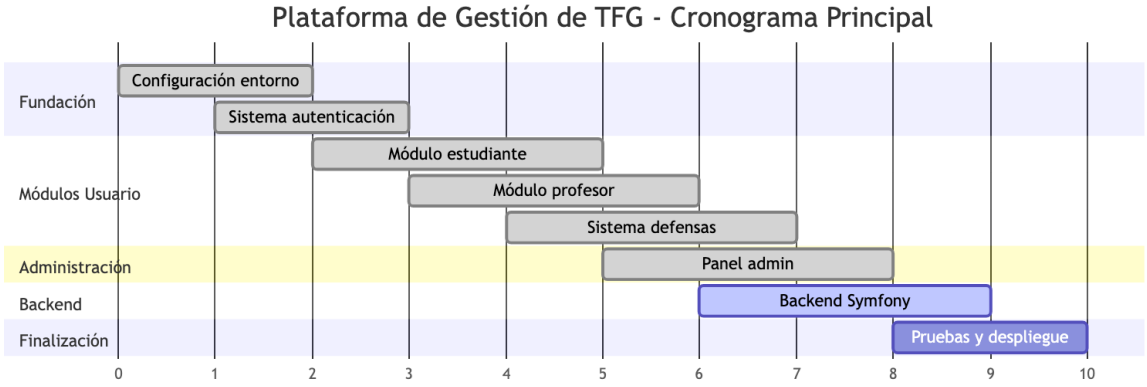

Figure 3.2: Cronograma Principal

Hitos críticos identificados: El proyecto establece cinco hitos fundamentales que marcan puntos de control esenciales en el desarrollo. H1 representa el frontend base funcional alcanzado en Semana 3 tras completar Fases 1-2, estableciendo la infraestructura fundamental de la aplicación. H2 marca la finalización de módulos usuario completos en Semana 6 al concluir Fases 3-4, proporcionando funcionalidad principal para estudiantes y profesores. H3 indica sistema frontend completo en Semana 8 tras Fases 5-6, integrando todas las funcionalidades de usuario en una plataforma cohesiva. H4 señala la integración del backend en Semana 9 con finalización de Fase 7, conectando frontend con persistencia de datos real. H5 culmina con sistema productivo en Semana 10 tras Fase 8, entregando aplicación lista para despliegue.

Dependencias críticas: - Fase 3 (Estudiante) requiere completar Sistema de autenticación. - Fase 4 (Profesor) depende de estados TFG de Fase 3. - Fase 5 (Defensas) necesita roles y permisos de Fase 4. - Fase 7 (Backend) puede iniciarse en paralelo desde Semana 7. - Fase 8 (Testing) requiere integración completa de Fase 7.

4. Análisis del sistema

Este capítulo presenta un análisis exhaustivo del sistema desarrollado, abarcando desde la especificación detallada de requisitos hasta la evaluación económica del proyecto. El análisis constituye el fundamento técnico y económico que sustenta todas las decisiones de diseño e implementación de la plataforma.

El análisis del sistema aborda los aspectos fundamentales necesarios para garantizar el éxito del proyecto. La especificación de requisitos define tanto las funcionalidades que debe proporcionar el sistema como las restricciones bajo las cuales debe operar, estableciendo las bases para el desarrollo técnico. La garantía de calidad establece los criterios y estándares que aseguran el correcto funcionamiento y la confiabilidad del sistema en entornos académicos reales.

La gestión del presupuesto completa el análisis evaluando la viabilidad económica del proyecto, analizando tanto la inversión requerida como los beneficios esperados. Este enfoque integral asegura que el análisis cubra todas las dimensiones técnicas, operativas y económicas necesarias para el éxito de la plataforma de gestión de TFG.

4.1 Especificación de requisitos

La especificación sigue la metodología IEEE Std 830-1998, organizando los requisitos en categorías funcionales específicas para cada rol de usuario y requisitos no funcionales transversales que aseguran la calidad, seguridad y rendimiento del sistema. Esta estructuración facilita tanto el desarrollo como la validación posterior de las funcionalidades implementadas.

4.1.1 Requisitos de información

Estos requisitos definen las entidades principales que el sistema debe gestionar, especificando sus atributos críticos y las relaciones que las conectan para formar un modelo de datos coherente y funcional.

4.1.1.1 Entidad Usuario

Descripción: Representa a todos los actores del sistema académico que interactúan con la plataforma, cada uno diferenciado por roles específicos que determinan sus capacidades y permisos de acceso.

Restricciones: El sistema implementa restricciones de integridad que garantizan la consistencia y validez de los datos de usuario. La dirección de correo electrónico debe ser única en todo el sistema, evitando duplicaciones que podrían comprometer la identificación y autenticación de usuarios. El DNI debe cumplir estrictamente con el formato válido establecido por la normativa española, incluyendo validación de dígito de control y estructura correcta. Cada usuario debe tener asignado al menos un rol del sistema, garantizando que no existan usuarios sin permisos definidos que podrían representar vulnerabilidades de seguridad. Los datos personales básicos son obligatorios y constituyen un prerrequisito indispensable para la activación completa de la cuenta, asegurando que toda la información necesaria para el funcionamiento del sistema esté disponible desde el momento de la activación.

4.1.1.2 Entidad TFG

Descripción: Representa un Trabajo de Fin de Grado completo, incluyendo toda su información académica asociada y la gestión de su ciclo de vida desde la creación inicial hasta la defensa final.

Restricciones: El sistema implementa restricciones de integridad específicas para garantizar la coherencia académica y técnica de los TFG. Un estudiante puede mantener máximo un TFG activo simultáneamente, evitando solapamientos que podrían comprometer la dedicación y calidad del trabajo académico. El título debe mantener unicidad por estudiante, asegurando que no existan trabajos duplicados o con nomenclatura confusa que puedan generar ambigüedades en la identificación. El archivo asociado debe cumplir estrictamente con el formato PDF y no exceder el tamaño máximo de 50MB, garantizando compatibilidad universal de lectura y optimización del almacenamiento del sistema. Las transiciones de estado deben seguir rigurosamente el flujo definido en el modelo de ciclo de vida, impidiendo saltos de estados que podrían comprometer la integridad del proceso académico y la trazabilidad de las evaluaciones.

4.1.1.3 Entidad Tribunal

Descripción: Comisión evaluadora académica responsable de dirigir y evaluar las defensas de TFG, integrada por profesores cualificados que garantizan la calidad del proceso de evaluación.

Restricciones: El sistema establece restricciones académicas y administrativas que garantizan la legitimidad y competencia de los tribunales. Los tres miembros del tribunal deben poseer necesariamente rol de profesor o superior, asegurando que todos los evaluadores tengan la cualificación académica mínima requerida para participar en procesos de evaluación de trabajos de fin de grado. No puede existir duplicación de miembros dentro de un mismo tribunal, evitando conflictos de roles y garantizando que cada posición (presidente, secretario, vocal) sea ocupada por personas diferentes que aporten perspectivas diversas al proceso evaluativo. Al menos el presidente debe ostentar específicamente el rol PRESIDENTE_TRIBUNAL, asegurando que quien lidera el tribunal posea las competencias y autoridad administrativa necesarias para coordinar adecuadamente el proceso de evaluación y toma de decisiones del tribunal.

4.1.1.4 Entidad Defensa

Descripción: Evento académico formal donde el estudiante presenta y defiende su TFG ante un tribunal cualificado para su evaluación y calificación final.

Restricciones: El sistema implementa restricciones operativas que garantizan la coherencia logística y académica de las defensas. Un TFG solo puede mantener una defensa activa simultáneamente, evitando duplicaciones que podrían generar confusión administrativa y conflictos en la evaluación académica. La fecha programada para la defensa debe ser necesariamente posterior a la fecha actual del sistema, impidiendo la programación retrospectiva que carece de sentido operativo y garantizando coherencia temporal en la planificación. El tribunal asignado debe confirmar disponibilidad completa en la fecha y hora programadas, asegurando que todos los miembros evaluadores puedan participar efectivamente en el proceso de evaluación sin conflictos de agenda que comprometan la calidad del proceso académico.

4.1.2 Requisitos funcionales

Estos requisitos se organizan por rol académico, especificando las funcionalidades que permiten a estudiantes, profesores, presidentes de tribunal y administradores realizar sus tareas específicas dentro de la plataforma.

4.1.2.1 Requisitos funcionales - Estudiante

RF-EST-001: Gestión de cuenta de usuario

Descripción: El estudiante puede ver y actualizar sus datos personales y de contacto, asegurando que siempre estén correctos para los procesos relacionados con su TFG.

Prioridad: Alta.

RF-EST-002: Creación de TFG

Descripción: Permite iniciar un nuevo TFG ingresando toda la información académica básica (título, descripción, resumen, palabras clave y tutor asignado), que quedará registrado en el sistema como borrador.

Prioridad: Alta.

RF-EST-003: Edición de información de TFG

Descripción: Mientras el trabajo esté en borrador, el estudiante puede modificar y ajustar la información de su proyecto, afinando los detalles antes de enviarlo a revisión.

Prioridad: Alta.

RF-EST-004: Subida de archivo TFG

Descripción: El sistema permite cargar de forma segura el documento final en PDF (hasta 50 MB), vinculándolo con el TFG del estudiante y confirmando la subida exitosa.

Prioridad: Alta.

RF-EST-005: Seguimiento de estado

Descripción: El estudiante puede consultar en todo momento el estado actual de su TFG y ver el historial completo de cambios, con fechas y comentarios asociados.

Prioridad: Media.

RF-EST-006: Visualización de comentarios

Descripción: Facilita el acceso a todos los comentarios y observaciones del tutor, apoyando la comunicación y la mejora continua del trabajo.

Prioridad: Media.

RF-EST-007: Consulta de información de defensa

Descripción: Proporciona todos los detalles logísticos de la defensa (fecha, hora, tribunal y lugar), para que el estudiante pueda organizarse y prepararse adecuadamente.

Prioridad: Media.

4.1.2.2 Requisitos funcionales - Profesor

RF-PROF-001: Visualización de TFG asignados

Descripción: El profesor puede acceder a un listado organizado de todos los TFG en los que actúa como tutor o cotutor, lo que le permite tener una visión clara de su carga académica y hacer un seguimiento del progreso de sus estudiantes.

Prioridad: Alta.

RF-PROF-002: Revisión de TFG

Descripción: Se garantiza que el profesor pueda descargar y revisar de forma segura los archivos de los TFG bajo su supervisión, facilitando el análisis detallado de cada trabajo.

Prioridad: Alta.

RF-PROF-003: Gestión de comentarios

Descripción: El profesor puede añadir comentarios y feedback estructurado a los TFG que supervisa, asegurando una comunicación efectiva con los estudiantes y orientándolos en la mejora de sus proyectos.

Prioridad: Alta.

RF-PROF-004: Cambio de estado de TFG

Descripción: Permite al profesor actualizar el estado de los TFG que supervisa, registrando además una justificación académica para cada cambio y notificando automáticamente al estudiante.

Prioridad: Alta.

RF-PROF-005: Gestión de calificaciones

Descripción: El profesor puede asignar calificaciones detalladas tras la defensa del TFG, con puntuaciones por criterios y comentarios explicativos, quedando todo registrado en el expediente académico del estudiante.

Prioridad: Media.

RF-PROF-006: Participación en tribunales

Descripción: Ofrece al profesor una visión centralizada de todos los tribunales en los que participa, con fechas, horarios y roles asignados, facilitando la organización de su agenda académica.

Prioridad: Media.

4.1.2.3 Requisitos funcionales - Presidente de Tribunal

RF-PRES-001: Gestión de tribunales

Descripción: El presidente puede crear, editar y organizar tribunales de evaluación, asegurando que los miembros elegidos cumplan con los requisitos académicos y de disponibilidad.

Prioridad: Alta.

RF-PRES-002: Programación de defensas

Descripción: Permite al presidente programar las defensas de TFG, coordinando fecha, hora, aula y tribunal, con notificaciones automáticas a todos los implicados.

Prioridad: Alta.

RF-PRES-003: Gestión de calendario

Descripción: El presidente puede visualizar y gestionar un calendario completo de defensas, con filtros y navegación para detectar conflictos y planificar con eficiencia.

Prioridad: Alta.

RF-PRES-004: Coordinación de disponibilidad

Descripción: Facilita consultar la agenda de los miembros de un tribunal y encontrar franjas horarias comunes donde todos estén disponibles para una defensa.

Prioridad: Media.

RF-PRES-005: Generación de actas

Descripción: Tras la defensa, el presidente puede generar un acta oficial en PDF con toda la información del evento, las calificaciones y observaciones del tribunal.

Prioridad: Media.

4.1.2.4 Requisitos funcionales - Administrador

RF-ADM-001: Gestión completa de usuarios

Descripción: El administrador tiene control total sobre los usuarios, pudiendo crearlos, modificarlos, eliminarlos y gestionar sus datos y accesos.

Prioridad: Alta.

RF-ADM-002: Asignación de roles

Descripción: Permite cambiar los roles y permisos de cada usuario, adaptando sus

privilegios a las responsabilidades académicas que tenga en la plataforma.

Prioridad: Alta.

RF-ADM-003: Generación de reportes

Descripción: El administrador puede generar reportes estadísticos con métricas, gráficos y tendencias que apoyan la toma de decisiones y la mejora de procesos.

Prioridad: Media.

RF-ADM-004: Exportación de datos

Descripción: Ofrece la opción de exportar información en distintos formatos (PDF, Excel, CSV, etc.), asegurando compatibilidad con otras herramientas externas.

Prioridad: Media.

RF-ADM-005: Configuración del sistema

Descripción: Permite ajustar parámetros globales de la plataforma (seguridad, notificaciones, límites, etc.), adaptándola a las políticas y necesidades de la institución.

Prioridad: Baja.

4.1.3 Diagrama de casos de uso

El siguiente diagrama representa las principales interacciones entre los actores del sistema y las funcionalidades disponibles para cada rol, como se ilustra en la Figura 4.1.

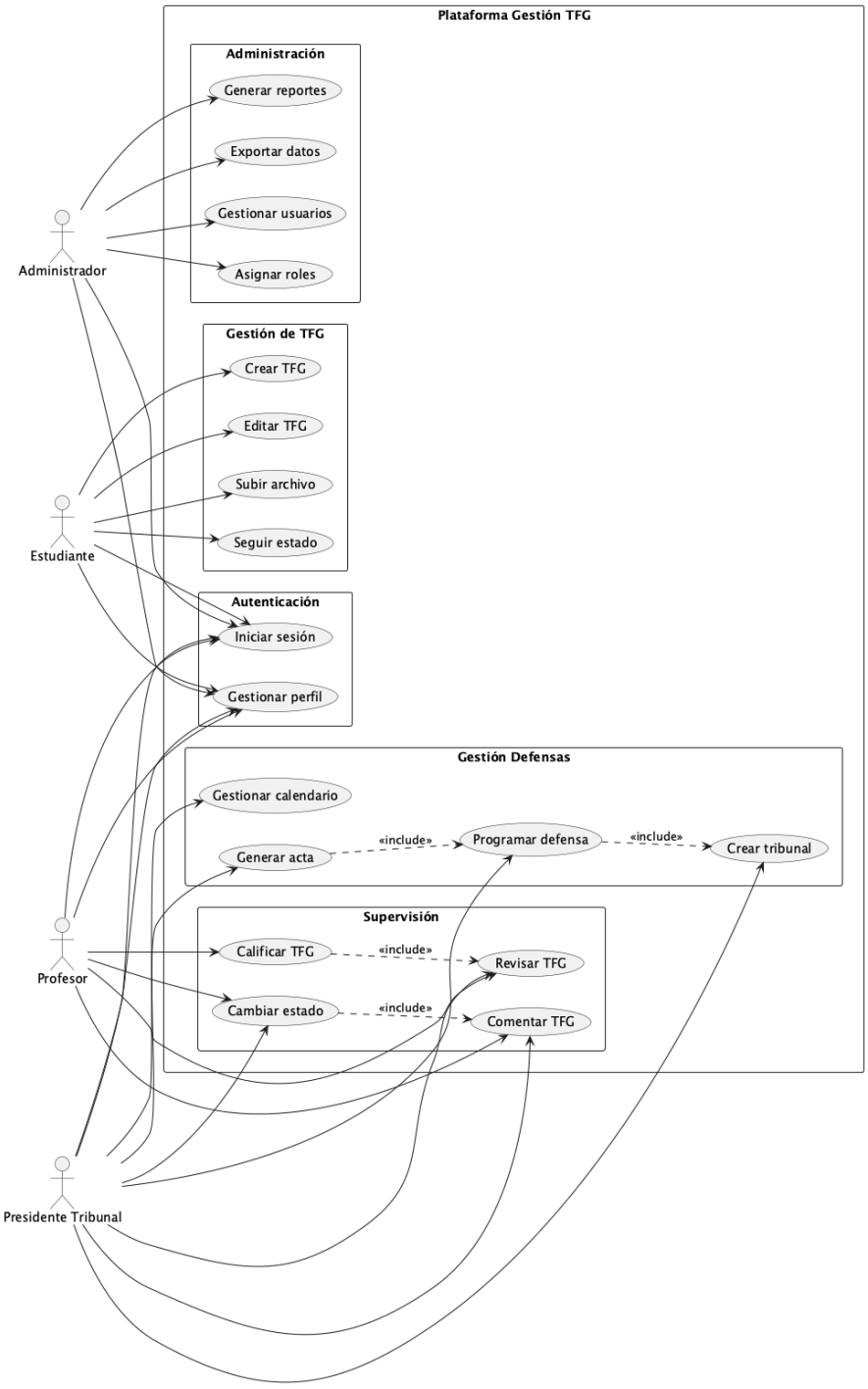

Figure 4.1: Diagrama de casos de uso

4.1.4 Descripción de casos de uso

4.1.4.1 UC001 - Crear TFG

Actor principal: Estudiante
Precondiciones: El usuario debe estar previamente autenticado en el sistema con rol específico de estudiante, confirmando su identidad y permisos para realizar operaciones académicas. Además, el estudiante no debe tener ningún TFG activo en el sistema, asegurando que pueda dedicar atención completa a un único trabajo de fin de grado.

Flujo principal: 1. El estudiante accede a la opción "Nuevo TFG". 2. El sistema muestra el formulario de creación. 3. El estudiante completa título, descripción, resumen y palabras clave. 4. El estudiante selecciona un tutor de la lista disponible. 5. El estudiante confirma la creación. 6. El sistema valida la información proporcionada. 7. El sistema crea el TFG en estado "borrador". 8. El sistema notifica al tutor seleccionado.

Flujos alternativos: En el paso 6a, si la validación de la información proporcionada falla por razones como formato incorrecto, datos faltantes o inconsistencias, el sistema presenta errores específicos que guíen al estudiante para corregir los problemas identificados. En el paso 7a, si el sistema detecta que el estudiante ya mantiene un TFG activo, rechaza automáticamente la operación de creación e informa al usuario sobre esta restricción.

Postcondiciones: Como resultado exitoso del proceso, se crea un nuevo TFG completamente inicializado en estado "borrador" que permite al estudiante comenzar el desarrollo de su trabajo académico. Simultáneamente, el tutor seleccionado recibe notificación automática de asignación que le informa sobre su nueva responsabilidad de supervisión académica.

4.1.4.2 UC005 - Revisar TFG

Actor principal: Profesor
Precondiciones: El usuario debe estar previamente autenticado en el sistema con rol específico de profesor, confirmando su identidad y permisos para realizar operaciones de supervisión académica. Adicionalmente, el TFG objeto de revisión debe estar formalmente asignado al profesor como tutor principal o cotutor, estableciendo la relación de supervisión necesaria.

Flujo principal: 1. El profesor accede a su lista de TFG asignados. 2. El profesor selecciona un TFG específico. 3. El sistema muestra detalles del TFG. 4. El profesor descarga el archivo PDF si está disponible. 5. El profesor revisa el contenido del trabajo.

Flujos alternativos: En el paso 4a, si no existe archivo PDF subido por el estudiante, el sistema informa claramente de esta situación al profesor, proporcionando orientación sobre las acciones posibles como contactar al estudiante o esperar la subida del documento. En el paso 2a, si el TFG seleccionado no está asignado al profesor como tutor, el sistema deniega automáticamente el acceso para mantener la confidencialidad y seguridad de los trabajos académicos.

Postcondiciones: Como resultado exitoso del proceso, el profesor obtiene acceso completo al contenido del TFG para realizar evaluación detallada, incluyendo capacidad de descarga del documento y revisión de toda la información académica asociada al trabajo.

4.1.4.3 UC010 - Programar defensa

Actor principal: Presidente de Tribunal
Precondiciones: El usuario debe estar previamente autenticado en el sistema con rol específico de presidente de tribunal, confirmando su autoridad para coordinar procesos de defensa académica. Debe existir al menos un tribunal previamente creado y disponible para asignación a defensas. El TFG objetivo debe encontrarse en estado ”aprobado”, indicando que ha superado la revisión del tutor y está listo para el proceso de defensa.

Flujo principal: 1. El presidente accede al calendario de defensas. 2. El presidente selecciona un TFG aprobado para programar. 3. El sistema muestra opciones de tribunales disponibles. 4. El presidente selecciona tribunal, fecha, hora y aula. 5. El sistema verifica disponibilidad de todos los miembros. 6. El presidente confirma la programación. 7. El sistema crea la defensa programada. 8. El sistema envía notificaciones a estudiante y miembros del tribunal.

Flujos alternativos: En el paso 5a, si existen conflictos de disponibilidad entre miembros del tribunal en la fecha propuesta, el sistema analiza automáticamente alternativas y sugiere fechas y horarios donde todos los miembros puedan participar. En el paso 4a, si no hay tribunales disponibles para asignación, el sistema orienta al presidente para crear un nuevo tribunal antes de continuar con la programación de la defensa.

Postcondiciones: Como resultado exitoso del proceso, se establece una defensa completamente programada con fecha, hora, tribunal y aula asignados de manera definitiva. Simultáneamente, todos los involucrados incluyendo estudiante, miembros del tribunal y personal administrativo reciben notificaciones automáticas con los detalles completos del evento programado.

4.1.5 Diagramas de secuencia

Los diagramas de secuencia ilustran la interacción temporal entre los diferentes componentes del sistema durante la ejecución de los casos de uso más críticos. Estas representaciones permiten comprender el flujo de mensajes, la sincronización de operaciones y las responsabilidades de cada actor en los procesos principales del sistema.

4.1.5.1 Secuencia: Subida de archivo TFG

El proceso de subida de archivos TFG representa una de las funcionalidades más importantes del sistema, involucrando validación, almacenamiento seguro y notificación de cambios de estado. La secuencia completa se detalla en la Figura 4.2.

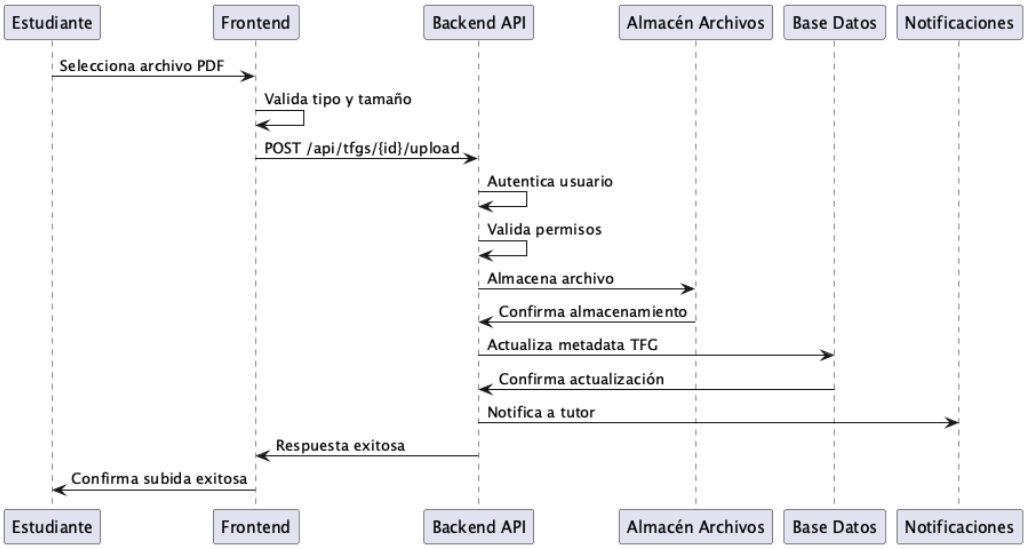

Figure 4.2: Secuencia: Subida de archivo TFG

4.1.5.2 Secuencia: Cambio de estado de TFG

La gestión de estados del TFG requiere validación de permisos, actualización de datos y coordinación entre múltiples actores del sistema. Este flujo se representa en la Figura 4.3.

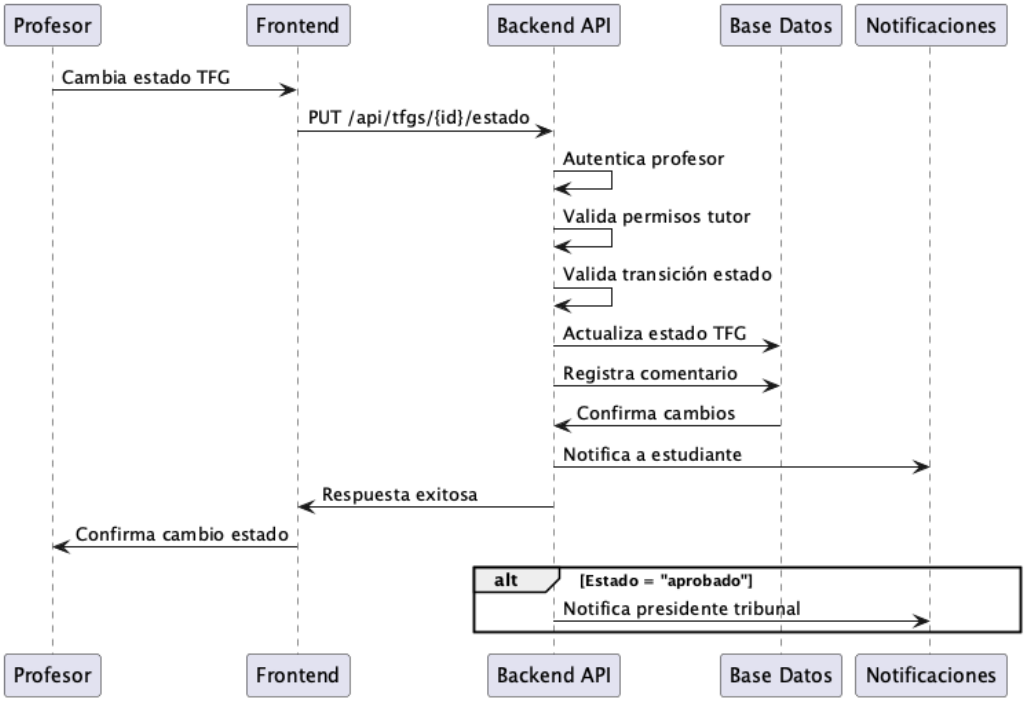

Figure 4.3: Secuencia: Cambio de estado de TFG

4.1.5.3 Secuencia: Programación de defensa

La programación de defensas involucra la coordinación entre tribunales, verificación de disponibilidad y asignación de recursos. El proceso completo de coordinación se ilustra en la Figura 4.4.

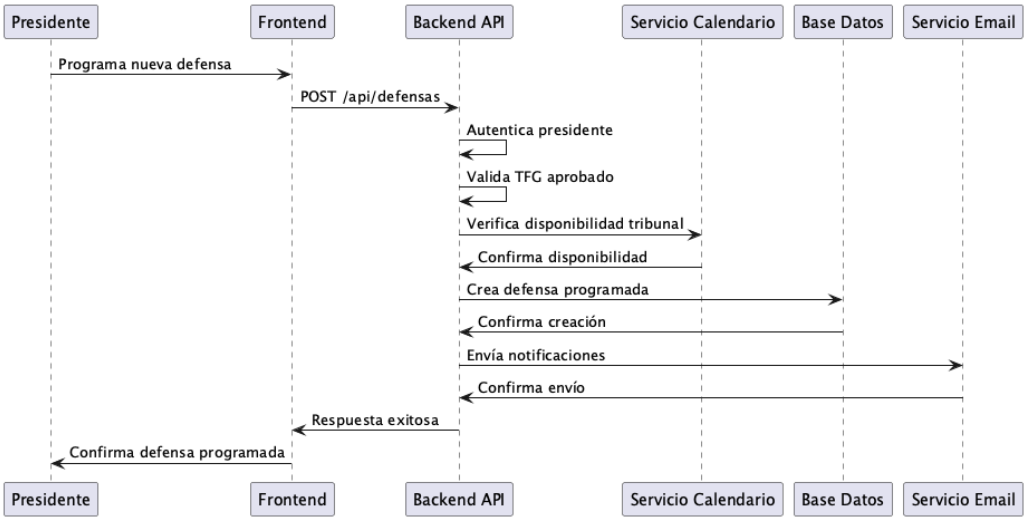

Figure 4.4: Secuencia: Programación de defensa

4.2 Garantía de calidad

La garantía de calidad constituye un aspecto fundamental del desarrollo de la plataforma, estableciendo los mecanismos y procedimientos necesarios para asegurar que el sistema cumple con los estándares de excelencia requeridos en un entorno académico universitario. Esta sección define las estrategias de seguridad, rendimiento y confiabilidad que garantizan el funcionamiento óptimo del sistema.

4.2.1 Rendimiento

RNF-001: Tiempo de respuesta

Descripción: El sistema debe ser ágil: inicio de sesión en menos de 2s, páginas en menos de 3s, subida de archivos de hasta 50MB en menos de 30s y reportes generados en menos de 10s.

Prioridad: Alta.

RNF-002: Rendimiento bajo carga

Descripción: Debe soportar al menos 100 usuarios trabajando al mismo tiempo sin que se degrade la experiencia ni aumenten los tiempos de espera.

Prioridad: Media.

RNF-003: Escalabilidad

Descripción: La arquitectura debe permitir crecer fácilmente en usuarios, datos y carga de trabajo, añadiendo recursos sin grandes cambios ni interrupciones.

Prioridad: Media.

4.2.2 Seguridad

RNF-004: Autenticación

Descripción: Se usará un sistema seguro con JWT: sesiones que expiran en 1 hora, refresh tokens con rotación y cierre de sesión que invalida todo de inmediato.

Prioridad: Alta.

RNF-005: Autorización

Descripción: El acceso estará controlado por roles y permisos, garantizando que cada

usuario solo pueda ver o modificar lo que le corresponde.

Prioridad: Alta.

RNF-006: Protección de datos

Descripción: Cumple con el RGPD, aplicando cifrado en tránsito y en reposo, registro de accesos y políticas claras de retención y borrado de datos personales.

Prioridad: Alta.

4.2.3 Usabilidad

RNF-007: Interfaz intuitiva

Descripción: El sistema debe ser fácil de usar, de modo que cualquier usuario pueda manejar lo básico en menos de 30 minutos sin formación especializada.

Prioridad: Alta.

RNF-008: Diseño responsivo

Descripción: La plataforma debe funcionar igual de bien en ordenador, tablet o móvil, adaptándose al tamaño de pantalla.

Prioridad: Media.

RNF-009: Accesibilidad

Descripción: Cumple el nivel AA de las WCAG 2.1, asegurando que personas con distintas capacidades puedan utilizarla, incluyendo soporte para tecnologías asistivas.

Prioridad: Media.

4.2.4 Confiabilidad

RNF-010: Disponibilidad

Descripción: El sistema debe estar disponible al menos un 99,5% del tiempo en horario académico (8:00-20:00), con mantenimientos programados fuera de ese rango.

Prioridad: Alta.

RNF-011: Recuperación de errores

Descripción: En caso de fallo, debe poder restaurarse en menos de 4 horas (RTO) y perder como máximo 1 hora de datos (RPO).

Prioridad: Media.

RNF-012: Consistencia de datos

Descripción: Toda la información debe ser íntegra y coherente, aplicando transacciones ACID y validación estricta en la base de datos.

Prioridad: Alta.

4.2.5 Interoperabilidad

RNF-013: APIs REST estándar

Descripción: La plataforma expone APIs RESTful con endpoints claros, uso correcto de métodos HTTP y códigos de estado consistentes. Toda la API está documentada automáticamente con OpenAPI/Swagger, lo que facilita pruebas y colaboración entre desarrolladores.

Prioridad: Alta.

RNF-014: Formato de datos estándar

Descripción: La comunicación usa JSON (HAL+JSON) con enlaces hipermedia, paginación estándar y filtros consistentes, lo que facilita navegar grandes volúmenes de datos y descubrir relaciones

Prioridad: Alta.

4.2.6 Operabilidad

RNF-015: Monitorización

Descripción: El sistema mide tiempos de respuesta, latencia (P95/P99) y uso en tiempo real. Incluye un endpoint health que verifica la salud de la aplicación y genera alertas proactivas ante degradaciones.

Prioridad: Alta.

RNF-016: Mantenibilidad

Descripción: El código está organizado en capas (presentación, negocio, persistencia), sigue principios SOLID y dependency injection, con documentación clara para facilitar evolución y la incorporación de nuevos desarrolladores.

Prioridad: Alta.

4.2.7 Transferibilidad

RNF-017: Containerización

Descripción: Se utiliza Docker y DDEV para garantizar entornos reproducibles. Toda la infraestructura está definida en docker-compose.yml, permitiendo levantar el sistema completo con un solo comando y evitando el clásico "en mi máquina funciona".

Prioridad: Alta.

4.2.8 Eficiencia

RNF-018: Optimización frontend

Descripción: Se aplican técnicas como code splitting, lazy loading, memoization y virtual scrolling, junto con caching en varias capas (React Query, Service Workers, browser caching), lo que asegura rapidez y fluidez incluso en grandes volúmenes de datos.

Prioridad: Alta.

RNF-019: Optimización backend

Descripción: Se usan índices compuestos, connection pooling y carga diferida de relaciones para acelerar consultas. Además, incluye compresión Gzip, paginación inteligente, selección de campos y limitación de peticiones (rate limiting) para garantizar rendimiento y estabilidad.

Prioridad: Alta.

4.2.9 Mantenibilidad

RNF-020: Calidad de código

Descripción: El código se mantiene homogéneo con ESLint, Prettier, PHP CS Fixer y PHPStan, además de commits convencionales que mejoran la trazabilidad y facilitan changelogs automáticos.

Prioridad: Alta.

RNF-021: Arquitectura mantenible

Descripción: Se implementan patrones de diseño probados (Repository, Factory, Observer, Strategy) que hacen al sistema extensible, adaptable a cambios y fácil de mantener a largo plazo.

Prioridad: Alta.

4.3 Gestión del presupuesto

En el contexto académico de este TFG, la gestión presupuestaria presenta características específicas que requieren una aproximación diferente a la de un proyecto empresarial. La evaluación abarca principalmente en la valoración del tiempo de desarrollo invertido, la utilización de herramientas y recursos educativos disponibles, y la estimación de costos equivalentes que tendría el proyecto en un entorno comercial profesional. Esta valoración proporciona una comprensión clara del valor del trabajo realizado y su equivalencia en términos de mercado laboral.

4.3.1 Estructura de costos

4.3.1.1 Costos de desarrollo

Tiempo de desarrollo: La estimación total del proyecto contempla 400 horas de desarrollo distribuidas a lo largo de 10 semanas de trabajo intensivo, lo que representa una dedicación promedio de 40 horas semanales con variaciones según la complejidad de cada fase del desarrollo. El valor hora de desarrollo junior se establece en €15/hora tomando como referencia estándares del mercado laboral para desarrolladores con experiencia limitada pero competentes en las tecnologías utilizadas. El costo total teórico de desarrollo alcanza €6,000, proporcionando una valoración económica del esfuerzo invertido aunque el proyecto se realice en modalidad académica.

4.3.1.2 Infraestructura y herramientas

Herramientas de desarrollo (gratuitas para estudiantes): DDEV como herramienta open source permite desarrollo containerizado sin licencias comerciales, garantizando entornos reproducibles. VS Code ofrece un IDE completo y gratuito con extensiones especializadas que proporcionan funcionalidad equivalente a IDEs comerciales. Draw.io facilita creación de diagramas UML profesionales sin costo de licencias de software especializado.

Infraestructura de desarrollo: El desarrollo local utiliza máquina personal sin costos adicionales de hardware o alquiler de servicios, optimizando presupuesto mediante aprovechamiento de recursos existentes. La base de datos MySQL ejecuta en contenedor local proporcionando entorno idéntico a producción sin costos de hosting durante

desarrollo. Los servicios de testing se ejecutan localmente mediante DDEV, eliminando necesidad de entornos de testing en cloud y reduciendo costos operativos.

4.3.1.3 Costos de producción estimados

Hosting y dominio (mensual): Un VPS básico con especificaciones de 2GB RAM, 1 CPU y 40GB SSD resulta suficiente para deployment inicial con costo estimado entre €10-20 mensuales según proveedor seleccionado. El dominio requiere inversión anual de aproximadamente €10 para establecer presencia web profesional. El certificado SSL se obtiene gratuitamente mediante Let's Encrypt, eliminando costos de seguridad adicionales. Los emails transaccionales del sistema se cubren mediante servicios gratuitos que permiten hasta 100 emails diarios, suficiente para operaciones iniciales.

5. Diseño

En este capítulo se desarrollarán los aspectos fundamentales del diseño del sistema, abarcando desde la arquitectura general hasta los detalles específicos de implementación.

En primer lugar, se presenta la arquitectura física, que define la organización estructural de los componentes del sistema y sus interacciones. Posteriormente, se aborda la arquitectura lógica, estableciendo los patrones de diseño y las responsabilidades de cada módulo. Finalmente, se incluye el esquema de la base de datos y el diseño de la interfaz de usuario, elementos esenciales para completar la visión técnica del proyecto.

5.1 Arquitectura física

Iniciando con la arquitectura física del sistema, se establece la base estructural sobre la cual se construye toda la plataforma. Esta arquitectura define la organización de los componentes de hardware y software, así como sus interacciones y dependencias, proporcionando una visión clara de cómo se despliega y ejecuta el sistema en un entorno real.

La arquitectura física de la Plataforma de Gestión de TFG se basa en una separación clara entre capas de presentación, lógica de negocio y persistencia, implementando un patrón de arquitectura distribuida que garantiza escalabilidad, mantenibilidad y seguridad.

5.1.1 Módulo frontend (Capa de presentación)

El frontend constituye la capa de presentación del sistema, desarrollado como una Single Page Application (SPA) que se ejecuta completamente en el navegador del usuario.

5.1.1.1 Arquitectura de componentes React

La arquitectura de componentes React implementa un patrón jerárquico que facilita la reutilización, mantenimiento y escalabilidad del código frontend. Esta estructura modular permite una clara separación de responsabilidades y optimiza el rendimiento mediante técnicas de lazy loading y memoización, como se ilustra en la Figura 5.1.

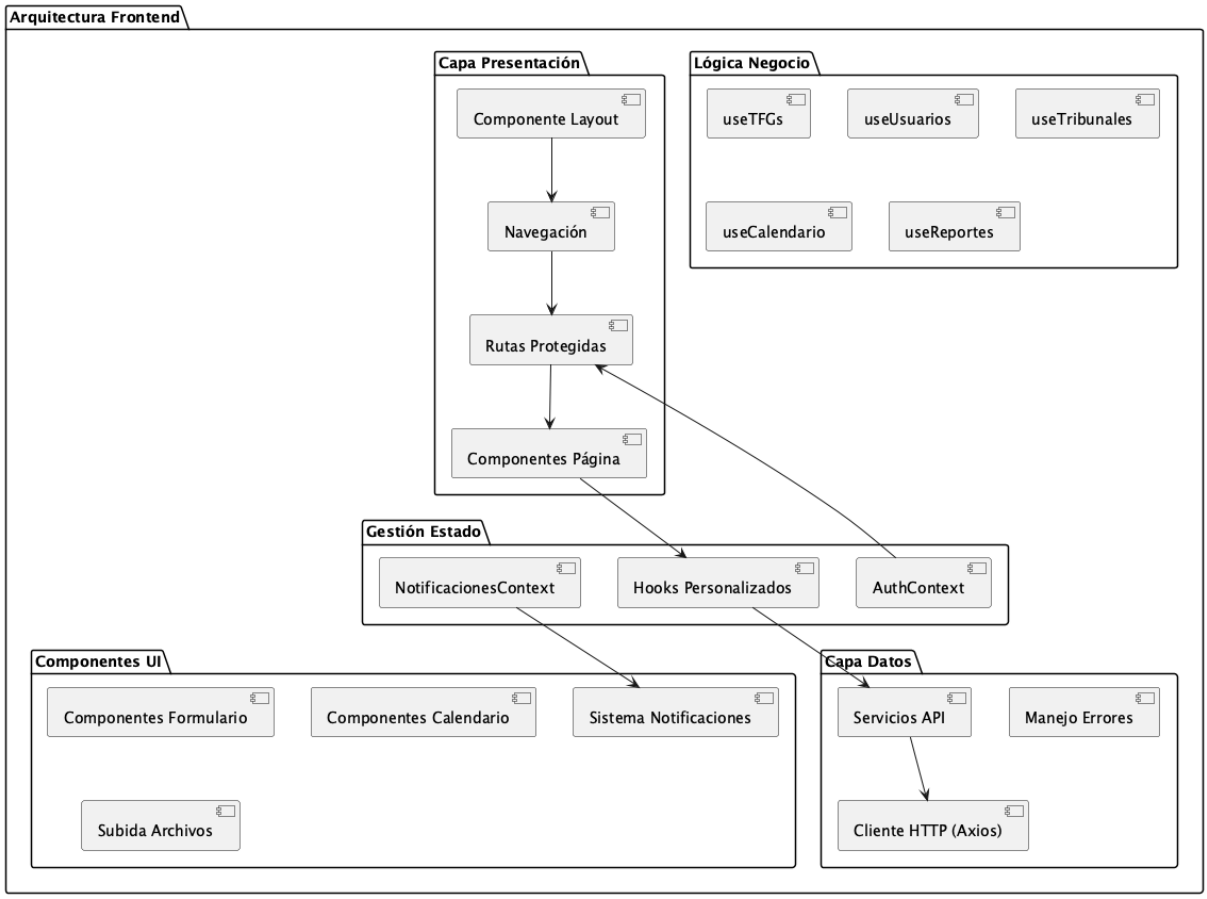

Figure 5.1: Arquitectura de componentes React

Componentes principales:

- **Componente de Presentación**: Contenedor principal que gestiona la estructura visual global.
- **Navegación**: Sistema de navegación dinámico basado en roles de usuario.
- **Rutas Protegidas**: Wrapper que controla acceso a rutas según autenticación y permisos.
- **Componentes de Página**: Componentes de página específicos para cada funcionalidad.

Patrones de diseño implementados:

- **Composición de Componentes**: Composición de funcionalidades mediante componentes reutilizables.
- **Componentes en Orden Ascendente**: ProtectedRoute como HOC para control de acceso.
- **Renderización de propiedades**: Componentes que exponen funcionalidad medi-

ante propiedades de función.

- **Hooks Personalizados**: Abstracción de lógica de negocio reutilizable entre componentes.

5.1.2 Módulo backend (Capa de lógica de negocio)

El backend implementa una arquitectura hexagonal (puertos y adaptadores) usando Symfony 6.4 LTS, proporcionando APIs REST robustas y escalables.

5.1.2.1 Arquitectura hexagonal

La arquitectura hexagonal, también conocida como arquitectura de puertos y adaptadores, permite aislar la lógica de negocio de las dependencias externas, facilitando el testing, la mantenibilidad y la evolución del sistema. Esta aproximación garantiza que los cambios en tecnologías específicas no impacten el núcleo del negocio, como se representa en la Figura 5.2.

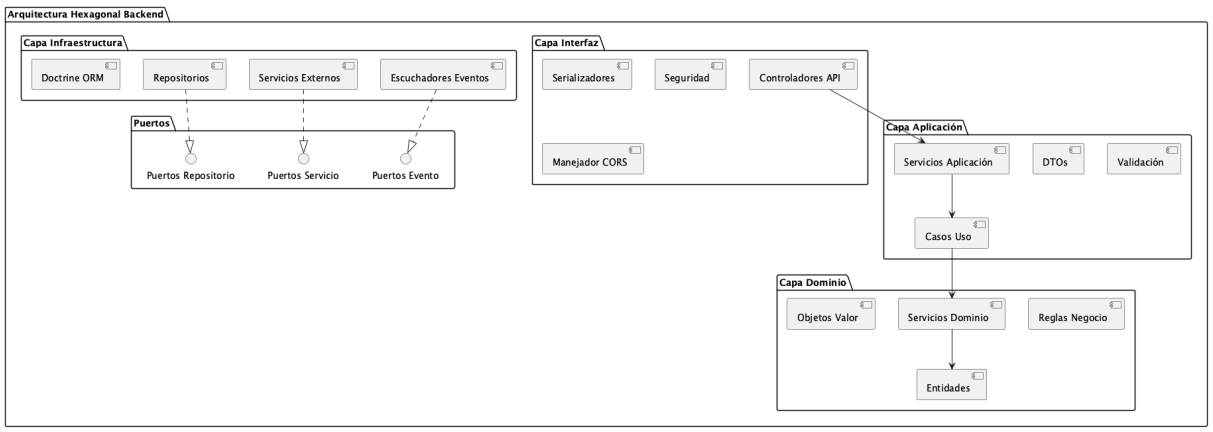

Figure 5.2: Arquitectura hexagonal

Capas de la arquitectura:

1. **Capa de Dominio**: Lógica de negocio pura, independiente de frameworks.
2. **Capa de Aplicación**: Casos de uso y servicios de aplicación.
3. **Capa de Infraestructura**: Implementaciones concretas (BD, servicios externos).
4. **Capa de Interfaz**: Controladores API y serialización.

5.1.3 Módulo de base de datos (Capa de persistencia)

La capa de persistencia utiliza MySQL 8.0 como sistema de gestión de base de datos, implementando un diseño relacional optimizado con Doctrine ORM.

5.1.3.1 Estrategia de persistencia

Migration Strategy: La estrategia de migración implementa un sistema robusto de control de esquema que utiliza Doctrine Migrations para versionado automático, garantizando que todos los cambios en la estructura de base de datos sean rastreables y reproducibles a través de diferentes entornos. La capacidad de rollback permite reversión segura a versiones anteriores del esquema en caso de problemas post-despliegue, proporcionando un mecanismo de contingencia crítico para operaciones de producción. La seguridad en producción se asegura mediante validación exhaustiva antes de aplicar migraciones, incluyendo testing y verificaciones de integridad que previenen corruption de datos durante actualizaciones de esquema.

5.1.4 Módulo de archivos (Almacenamiento)

El sistema de archivos está diseñado para manejar subidas seguras de documentos PDF con validación exhaustiva y almacenamiento optimizado.

Medidas de Seguridad de Archivos: El sistema implementa medidas de seguridad para protección de archivos que incluyen validación estricta de tipo MIME permitiendo exclusivamente archivos PDF para prevenir subida de contenido malicioso o formatos no autorizados. Las limitaciones de tamaño establecen un máximo de 50MB por archivo, balanceando capacidad de almacenamiento de documentos académicos completos con optimización de recursos del servidor y tiempos de transferencia razonables.

5.1.4.1 Estrategia Almacenamiento

La estrategia de almacenamiento de archivos implementa un sistema robusto y escalable que garantiza la integridad, seguridad y disponibilidad de los documentos TFG. Este diseño contempla validación automática, almacenamiento seguro y mecanismos de copia de seguridad, como se detalla en la Figura 5.3.

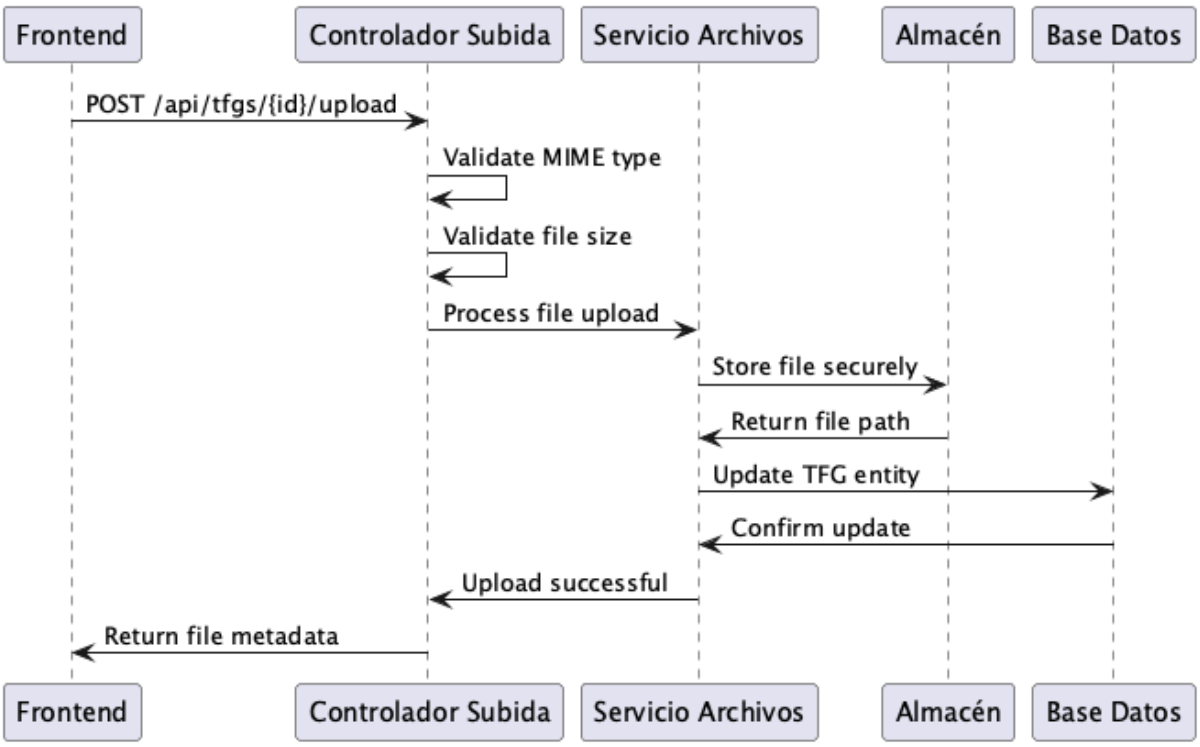

Figure 5.3: Estrategia Almacenamiento

Flujo de procesamiento de archivos:

1. **Validación previa**: MIME type, tamaño y estructura básica del PDF.
2. **Procesamiento seguro**: Almacenamiento con nombre único y ruta encriptada.
3. **Metadatos**: Extracción y almacenamiento de información del archivo.
4. **Acceso controlado**: URLs temporales con expiración automática.

5.2 Arquitectura lógica

Habiendo establecido la arquitectura física del sistema, es fundamental abordar la arquitectura lógica, la cual define la organización conceptual y funcional de los componentes de software.

La arquitectura lógica organiza los componentes del sistema según responsabilidades funcionales, implementando patrones de diseño que garantizan separación de ámbitos y alta cohesión.

5.2.1 Capa de presentación (Frontend)

5.2.1.1 Patrón Container/Presentational

Componentes Container: Gestionan la lógica de negocio de la interfaz, interactúan con los servicios y hooks personalizados, controlan estados como loading o error, y transmiten los datos a los componentes de presentación.

Componentes Presentational: Se encargan únicamente del renderizado de la UI. Reciben datos y callbacks como props, garantizando interfaces reutilizables, simples y fáciles de probar.

5.2.1.2 Control de Estado

Implementada con Context API: Uso de Context API para manejar autenticación, notificaciones y datos compartidos.

Hooks personalizados: Encapsulan lógica de negocio relacionada con la obtención, creación y actualización de TFGs. Esto permite reutilizar lógica y mantener los componentes más limpios.

Jerarquía clara de contextos: Rodean la aplicación (AuthProvider, NotificacionesProvider), asegurando un flujo de datos centralizado y consistente.

5.2.2 Capa de lógica de negocio (Backend)

5.2.2.1 Diseño Domain-Driven

Patrón de Agregación: Encapsulan entidades principales (ej. TFG) y sus reglas de negocio, como las transiciones válidas de estado (borrador → revisión → aprobado → defendido). Esto evita inconsistencias y mantiene las reglas dentro del dominio.

Clases de Valor: Representan conceptos inmutables y validados (ej. Email), aportando robustez al modelo y evitando datos inválidos.

5.2.2.2 Patrón Service Layer

Implementa la orquestación de casos de uso: creación de un TFG, notificación automática al tutor, validación de permisos, etc. Los servicios actúan como puente entre la lógica de dominio y la infraestructura (repositorios, notificaciones, eventos). Favorece el de-

sacoplamiento y asegura que las reglas de negocio no dependan de frameworks ni librerías externas.

5.2.3 Capa de persistencia

5.2.3.1 Patrón de Repositorio

Define interfaces que abstraen el acceso a datos (ej. buscar TFGs por estudiante, tutor o estado). Implementación mediante Doctrine ORM, lo que permite consultas optimizadas y expresivas sin depender directamente de SQL en la lógica de negocio. Permite cambiar la estrategia de persistencia (ej. migrar de MySQL a PostgreSQL) sin afectar al dominio ni a la lógica de negocio.

5.3 Esquema de la base de datos

El esquema de base de datos propuesto sigue principios de normalización que garantizan la consistencia y eliminan la redundancia, mientras que los índices y constraints aseguran tanto el rendimiento como la integridad referencial. Esta estructura de datos ha sido diseñada con mucho detalle para soportar eficientemente todas las operaciones requeridas por los diferentes módulos del sistema.

5.3.1 Modelo conceptual

El modelo conceptual de la base de datos representa las entidades principales del sistema y sus relaciones, proporcionando la base para la implementación física. Este diseño garantiza la integridad referencial, optimiza las consultas más frecuentes y establece la estructura de datos necesaria para soportar todas las funcionalidades del sistema, como se ilustra en la Figura 5.4.

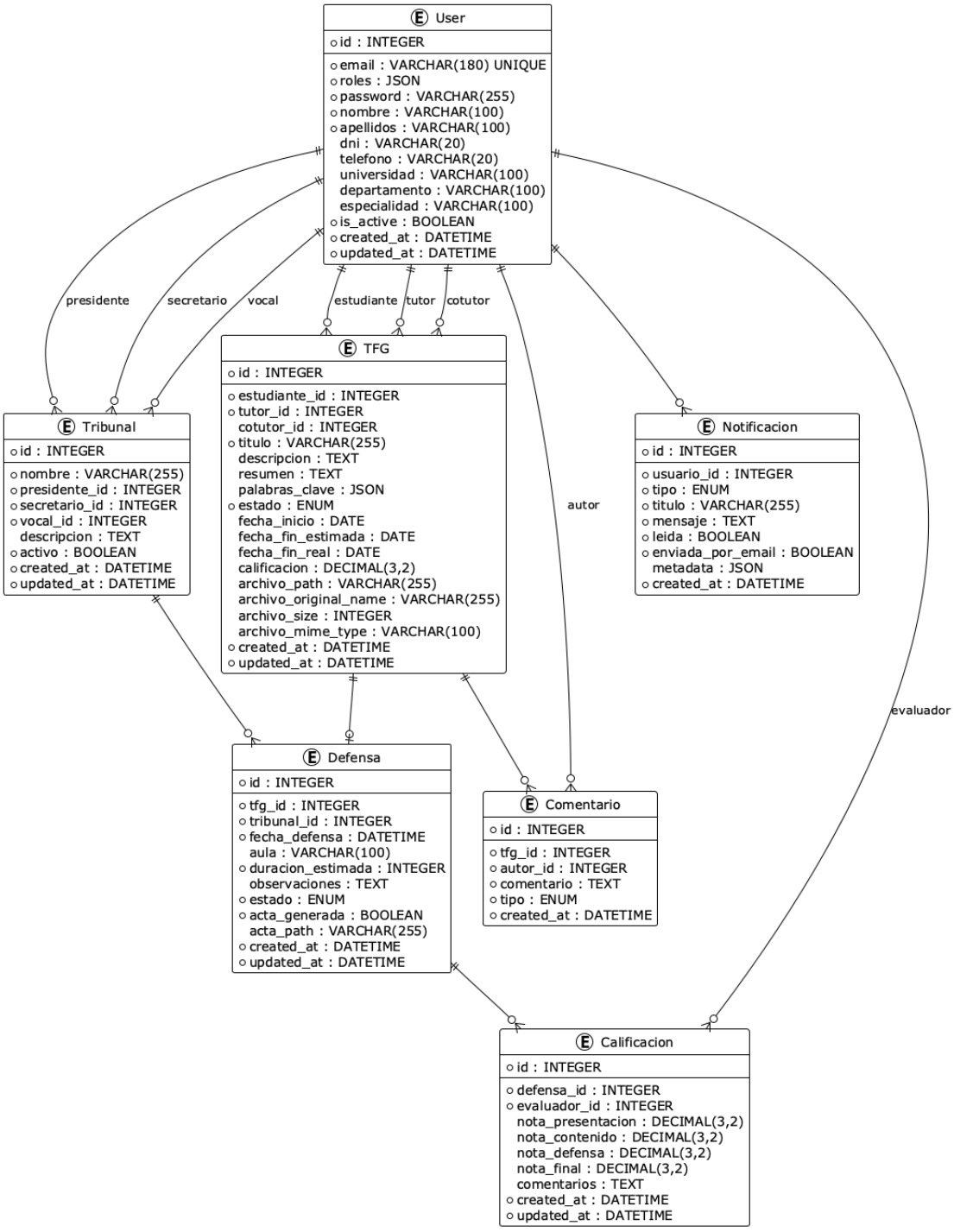

Figure 5.4: Modelo conceptual

5.3.2 Normalización y constraints

5.3.2.1 Tercera forma normal (3NF)

El esquema cumple con la tercera forma normal mediante:

Primera Forma Normal (1NF): - Todos los campos contienen valores atómicos. - Campos JSON utilizados únicamente para datos semi-estructurados (roles, palabras clave, metadata). - No hay grupos repetitivos de columnas.

Segunda Forma Normal (2NF): - Todas las tablas tienen claves primarias definidas. - Todos los atributos no-clave dependen completamente de la clave primaria. - No hay dependencias parciales.

Tercera Forma Normal (3NF): - No existen dependencias transitivas. - Cada atributo no-clave depende directamente de la clave primaria.

5.3.2.2 Constraints e integridad referencial

Primary Keys:

```
ALTER TABLE users ADD CONSTRAINT pk_users PRIMARY KEY (id);
ALTER TABLE tfgs ADD CONSTRAINT pk_tfgs PRIMARY KEY (id);
ALTER TABLE tribunales ADD CONSTRAINT pk_tribunales PRIMARY KEY (id);
ALTER TABLE defensas ADD CONSTRAINT pk_defensas PRIMARY KEY (id);
```

Foreign Keys:

```
ALTER TABLE tfgs
  ADD CONSTRAINT fk_tfg_estudiante
  FOREIGN KEY (estudiante_id) REFERENCES users(id) ON DELETE RESTRICT;

ALTER TABLE tfgs
  ADD CONSTRAINT fk_tfg_tutor
  FOREIGN KEY (tutor_id) REFERENCES users(id) ON DELETE RESTRICT;

ALTER TABLE defensas
  ADD CONSTRAINT fk_defensa_tfg
  FOREIGN KEY (tfg_id) REFERENCES tfgs(id) ON DELETE CASCADE;
```

Unique Constraints:

```
ALTER TABLE users ADD CONSTRAINT uk_users_email UNIQUE (email);
ALTER TABLE users ADD CONSTRAINT uk_users_dni UNIQUE (dni);
ALTER TABLE defensas ADD CONSTRAINT uk_defensa_tfg UNIQUE (tfg_id);
```

Check Constraints:

```
ALTER TABLE tfgs
  ADD CONSTRAINT ck_tfg_estado
  CHECK (estado IN ('borrador', 'revision', 'aprobado', 'defendido'));

ALTER TABLE calificaciones
  ADD CONSTRAINT ck_calificacion_notas
  CHECK (
    nota_presentacion >= 0 AND nota_presentacion <= 10 AND
    nota_contenido >= 0 AND nota_contenido <= 10 AND
    nota_defensa >= 0 AND nota_defensa <= 10 AND
    nota_final >= 0 AND nota_final <= 10
  );
```

5.3.3 Índices de rendimiento

5.3.3.1 Índices principales

Índices de búsqueda frecuente:

```
-- Búsquedas por estudiante (muy frecuente)
CREATE INDEX idx_tfgs_estudiante ON tfgs(estudiante_id);

-- Búsquedas por tutor (muy frecuente)
CREATE INDEX idx_tfgs_tutor ON tfgs(tutor_id);

-- Búsquedas por estado (frecuente para reportes)
CREATE INDEX idx_tfgs_estado ON tfgs(estado);

-- Búsquedas de defensas por fecha (calendario)
CREATE INDEX idx_defensas_fecha ON defensas(fecha_defensa);

-- Notificaciones no leídas por usuario
CREATE INDEX idx_notificaciones_usuario_leida ON notificaciones(usuario_id,
    leida);
```

Índices compuestos:

```
-- Combinación frecuente: tutor + estado
CREATE INDEX idx_tfgs_tutor_estado ON tfgs(tutor_id, estado);

-- Tribunal disponible para programación
CREATE INDEX idx_tribunales_activo ON tribunales(activo, created_at);

```

```
-- Defensas por tribunal y fecha
CREATE INDEX idx_defensas_tribunal_fecha ON defensas(tribunal_id,
    fecha_defensa);
```

5.3.3.2 Análisis de consultas

Query más frecuente - TFGs por tutor:

```
EXPLAIN SELECT t.*, e.nombre as estudiante_nombre
FROM tfgs t
INNER JOIN users e ON t.estudiante_id = e.id
WHERE t.tutor_id = ?
ORDER BY t.updated_at DESC;

-- Usa índice: idx_tfgs_tutor
-- Rows examined: ~10-50 por profesor
-- Execution time: < 5ms
```

Query compleja - Dashboard admin:

```
EXPLAIN SELECT
  COUNT(*) as total_tfgs,
  COUNT(CASE WHEN estado = 'borrador' THEN 1 END) as borradores,
  COUNT(CASE WHEN estado = 'revision' THEN 1 END) as en_revision,
  COUNT(CASE WHEN estado = 'aprobado' THEN 1 END) as aprobados,
  COUNT(CASE WHEN estado = 'defendido' THEN 1 END) as defendidos
FROM tfgs
WHERE created_at >= DATE_SUB(CURDATE(), INTERVAL 1 YEAR);

-- Usa índice: idx_tfgs_estado + created_at
-- Query optimizada para agregaciones
```

5.4 Diseño de la interfaz de usuario

Para completar la visión integral del diseño del sistema, es fundamental abordar el diseño de la interfaz de usuario, elemento que determina la experiencia y satisfacción de los usuarios finales. La interfaz de usuario representa el punto de contacto entre el sistema y sus usuarios, por lo que su diseño debe equilibrar funcionalidad, usabilidad y estética para proporcionar una experiencia óptima a cada tipo de usuario.

5.4.1 Interfaces de usuario implementadas

Una vez establecidos los fundamentos del diseño de la interfaz de usuario, es fundamental presentar las interfaces finales implementadas que materializan todos los conceptos y patrones de diseño descritos anteriormente. Esta sección documenta las pantallas principales del sistema, organizadas por roles de usuario, mostrando cómo se aplican los principios de usabilidad, accesibilidad y consistencia visual en cada una de las funcionalidades implementadas.

Las interfaces presentadas a continuación representan el resultado de un proceso iterativo de diseño centrado en el usuario, donde cada pantalla ha sido optimizada para las tareas específicas de cada rol, manteniendo la coherencia del sistema de diseño establecido y garantizando una experiencia de usuario intuitiva y eficiente.

5.4.1.1 Dashboard de Estudiante

El dashboard del estudiante constituye el punto central de interacción para los usuarios con rol de estudiante, proporcionando acceso directo a las funcionalidades principales del ciclo de vida del TFG. La interfaz implementa un diseño limpio y funcional que facilita la navegación y el seguimiento del progreso académico, como se muestra en la Figura 5.5.

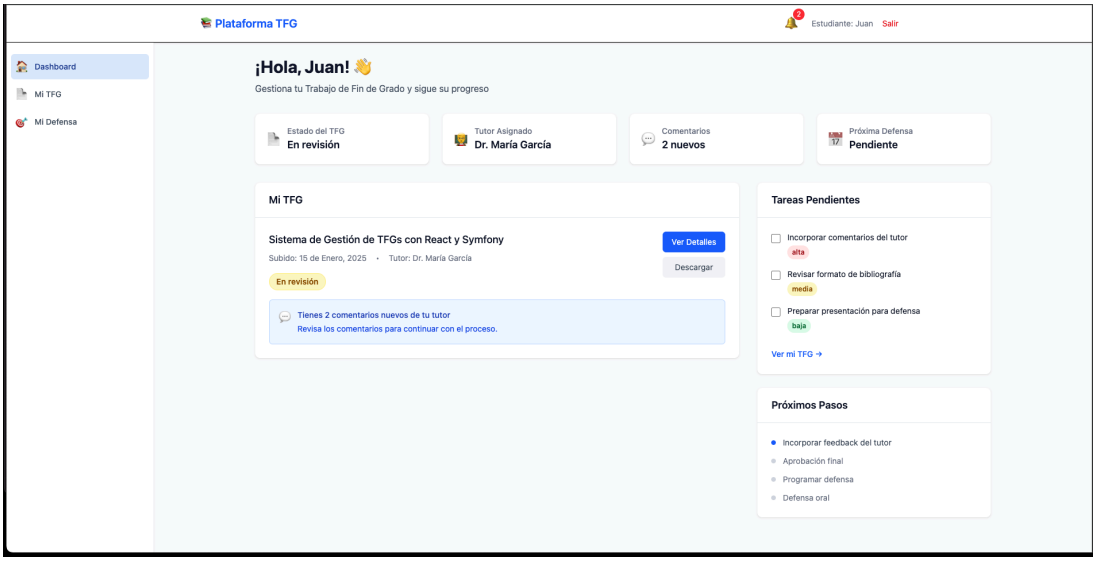

Figure 5.5: Dashboard principal del estudiante con overview del TFG y navegación

El dashboard presenta elementos clave como el estado actual del TFG, notificaciones relevantes, accesos directos a las funciones más utilizadas y un resumen del progreso académico. La interfaz utiliza cards informativos que organizan la información de manera

jerárquica, permitiendo al estudiante obtener una visión general rápida de su situación académica.

5.4.1.2 Gestión de TFG - Vista de Estudiante

La interfaz de gestión de TFG para estudiantes proporciona las herramientas necesarias para la carga, edición y seguimiento de los trabajos de fin de grado. Esta pantalla integra funcionalidades de subida de archivos, edición de metadatos y visualización del historial de revisiones, tal como se presenta en la Figura 5.6.

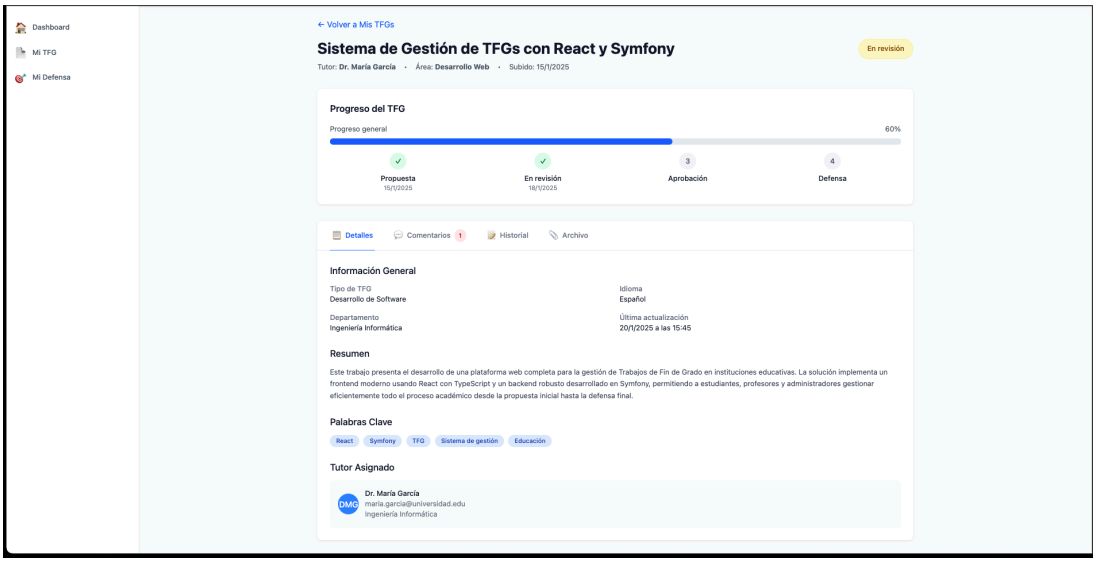

Figure 5.6: Interfaz de gestión de TFG para estudiantes con formularios de carga y metadatos

La interfaz incluye un sistema de drag-and-drop para la carga de documentos PDF, campos estructurados para título, resumen y palabras clave, así como indicadores visuales del progreso de carga y validación de archivos. El diseño responsivo garantiza una experiencia óptima tanto en dispositivos de escritorio como móviles. Figura 5.7.

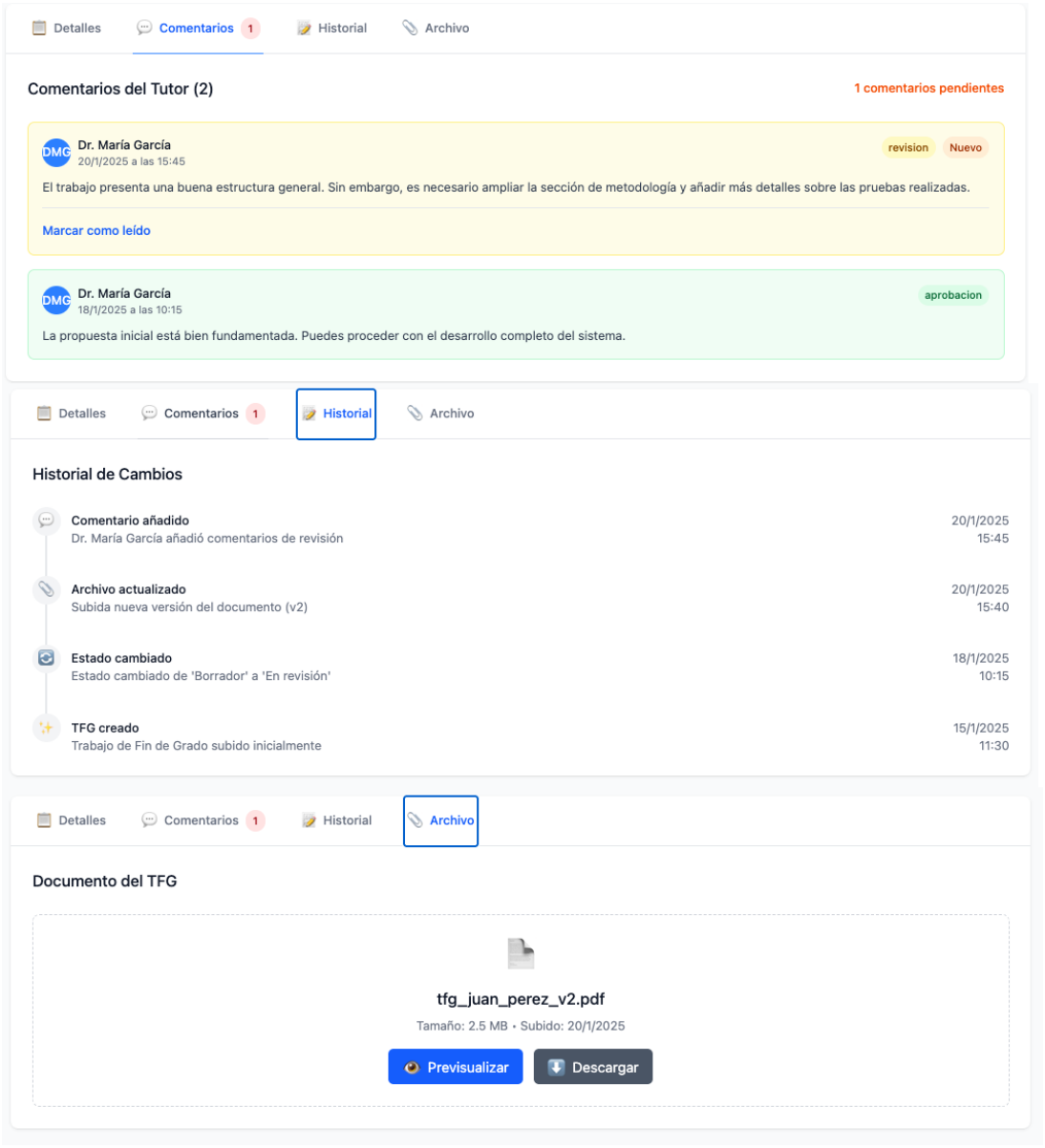

Figure 5.7: Vista extendida de gestión de TFG para estudiantes con detalles adicionales

5.4.1.3 Sistema de Notificaciones

El sistema de notificaciones implementa un enfoque no intrusivo que mantiene a los usuarios informados sobre eventos relevantes sin interrumpir su flujo de trabajo. La interfaz combina notificaciones in-app con indicadores visuales sutiles, como se observa en la Figura 5.8.

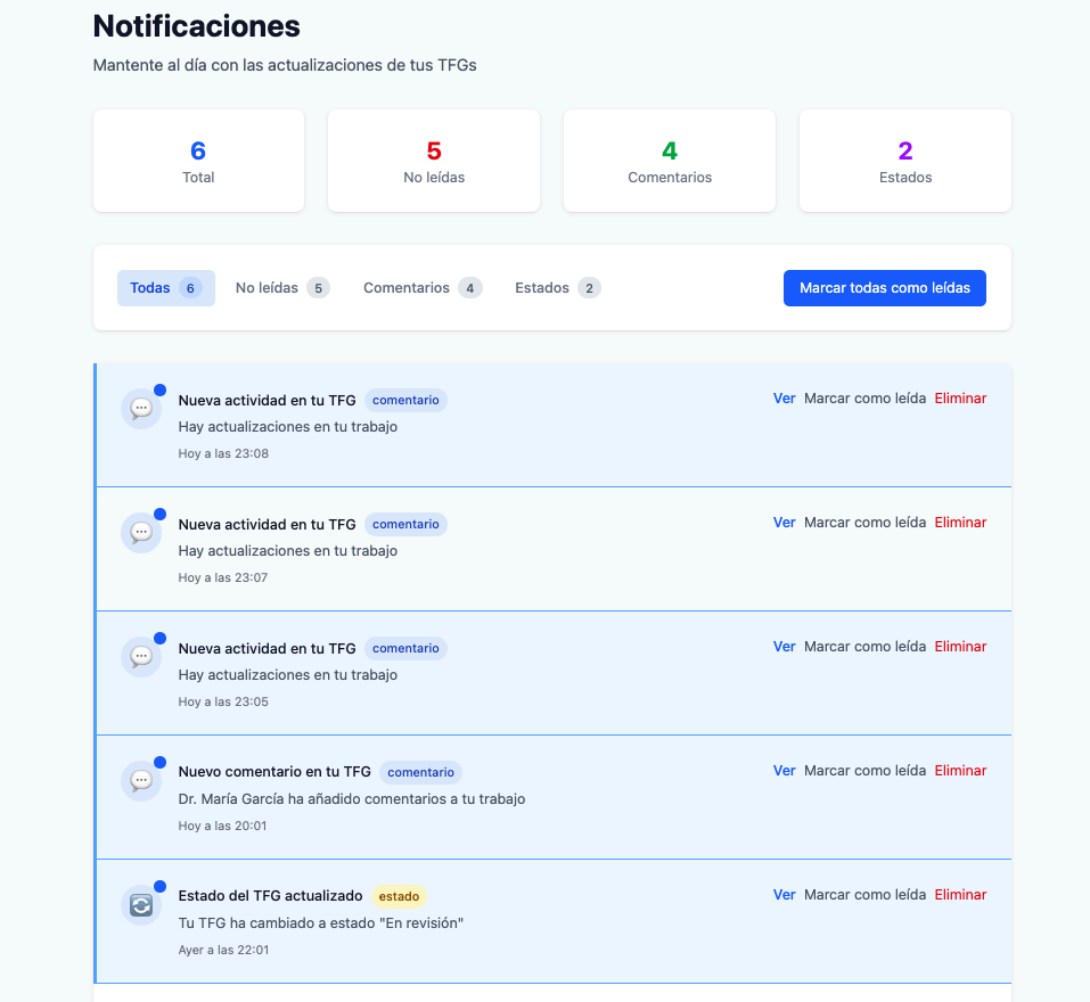

Figure 5.8: Sistema de notificaciones con dropdown y estados de lectura

Las notificaciones se categorizan por tipo (información, éxito, advertencia, error) utilizando el sistema de colores semánticos establecido, facilitando la comprensión inmediata del tipo de mensaje. El dropdown de notificaciones incluye funcionalidades de filtrado, marcado como leído y navegación directa a las secciones relevantes.

5.4.1.4 Dashboard de Profesor

La interfaz del profesor está diseñada para facilitar la supervisión eficiente de múltiples TFGs asignados, proporcionando herramientas de gestión, evaluación y comunicación con estudiantes. El dashboard presenta una vista organizada de los trabajos pendientes de revisión y las tareas prioritarias, como se ilustra en la Figura 5.9.

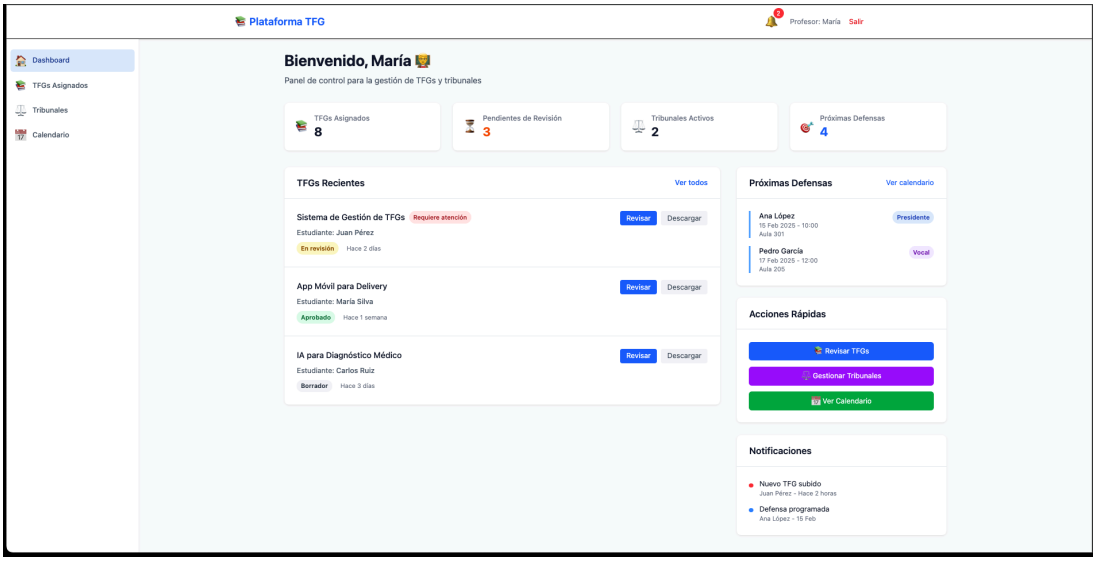

Figure 5.9: Dashboard del profesor con lista de TFGs asignados y estados de revisión

El diseño incluye filtros avanzados para organizar los TFGs por estado, fecha de envío o prioridad, así como acciones rápidas para cambios de estado y redacción de comentarios. La interfaz utiliza indicadores visuales claros para distinguir entre trabajos que requieren atención inmediata y aquellos en proceso normal.

5.4.1.5 Sistema de Evaluación y Feedback

La interfaz de evaluación proporciona a los profesores herramientas completas para la revisión y calificación de los TFGs, incluyendo formularios estructurados de evaluación y sistemas de comentarios contextuales, tal como se presenta en la Figura 5.10.

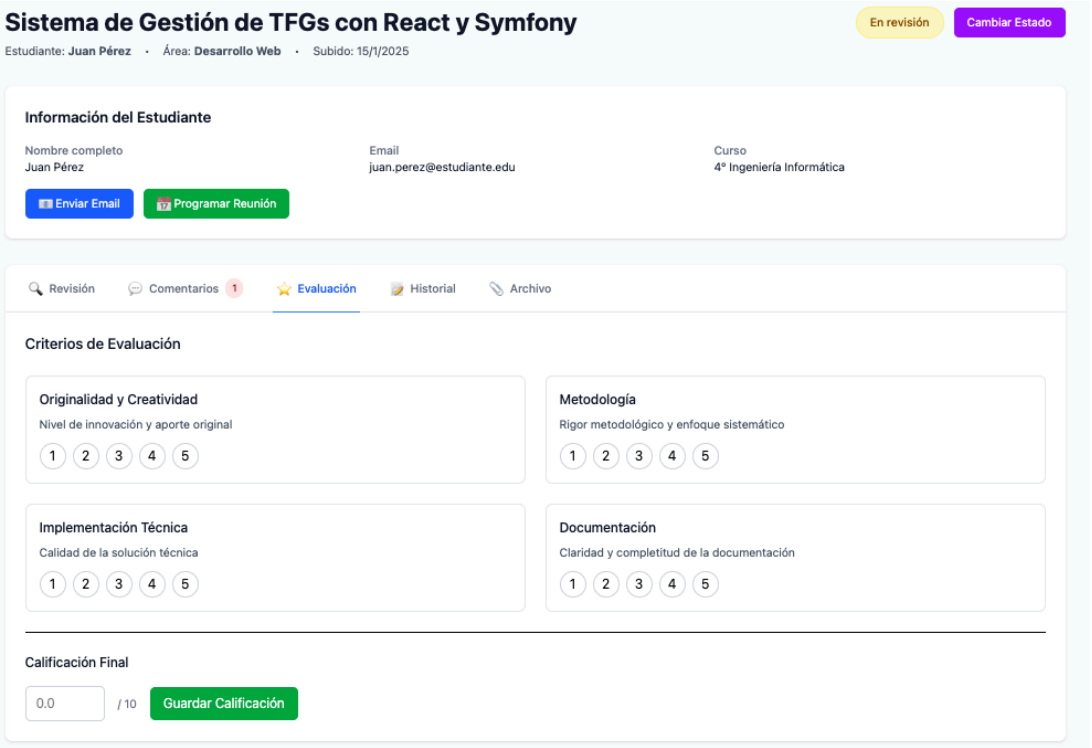

Figure 5.10: Sistema de evaluación con formularios de calificación y comentarios

La interfaz integra formularios dinámicos que se adaptan a diferentes criterios de evaluación, sistemas de puntuación configurable y herramientas de texto enriquecido para comentarios detallados. El diseño facilita la navegación entre diferentes secciones del documento mientras se mantiene el contexto de evaluación.Figura 5.11.

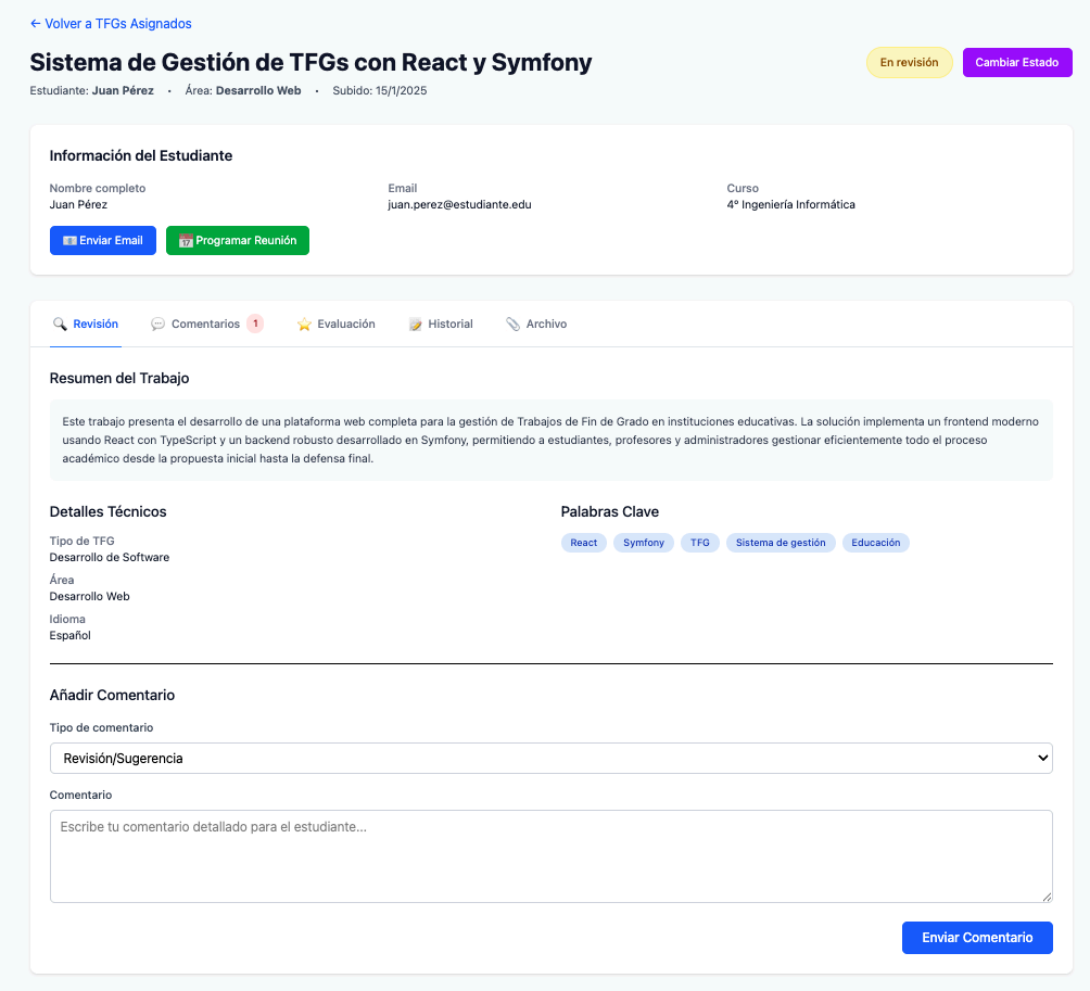

Figure 5.11: Sistema de evaluación con comentarios y calificaciones detalladas

5.4.1.6 Gestión de Tribunales

La interfaz de gestión de tribunales, accesible para usuarios con rol de presidente de tribunal, proporciona herramientas completas para la creación, configuración y administración de tribunales de evaluación. La pantalla integra funcionalidades de asignación de miembros y gestión de disponibilidad, como se muestra en la Figura 5.12.

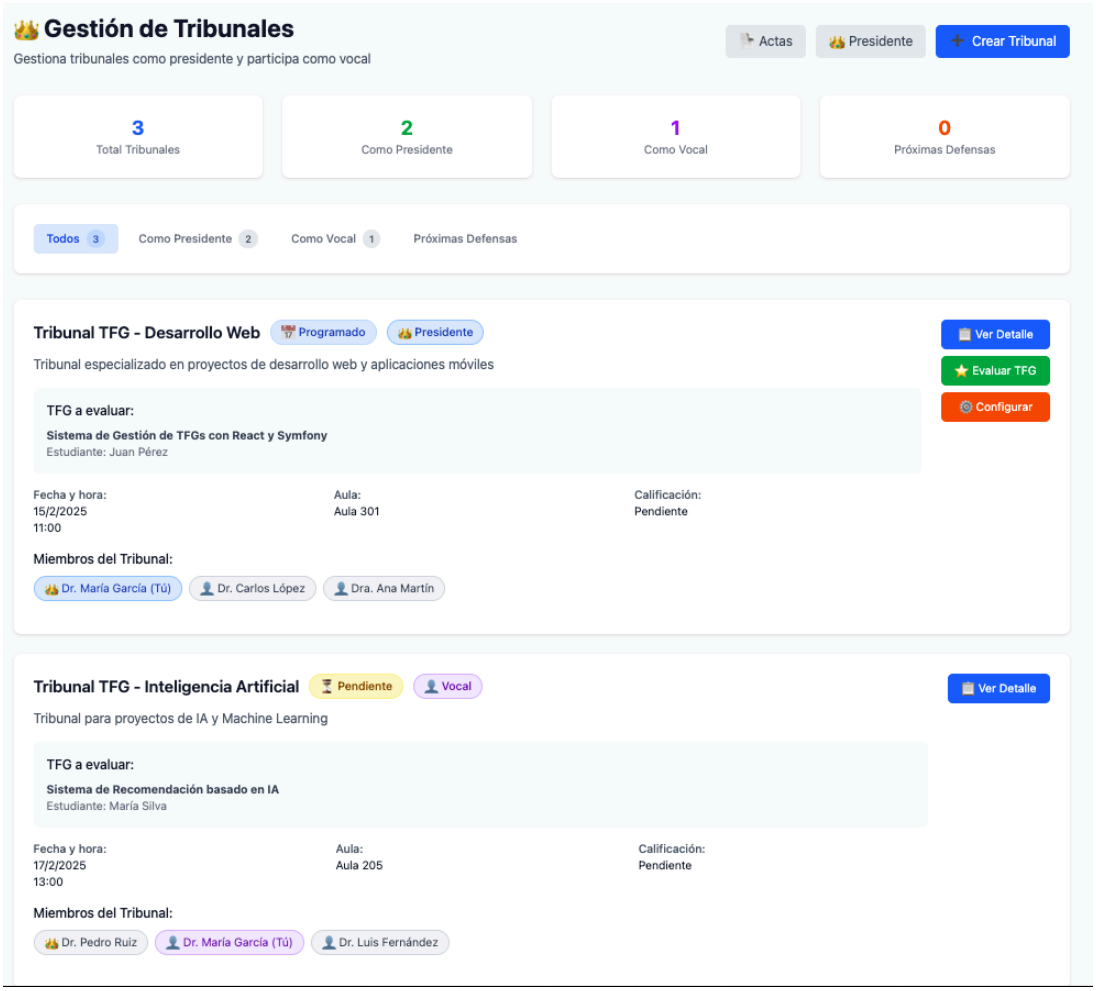

Figure 5.12: Interfaz de gestión de tribunales con asignación de miembros y disponibilidad

La interfaz incluye herramientas de búsqueda y filtrado para la selección de profesores, validación automática de conflictos de horario y visualización de la carga de trabajo de cada miembro potencial. El diseño facilita la toma de decisiones informadas en la composición de tribunales.Figura 5.13.

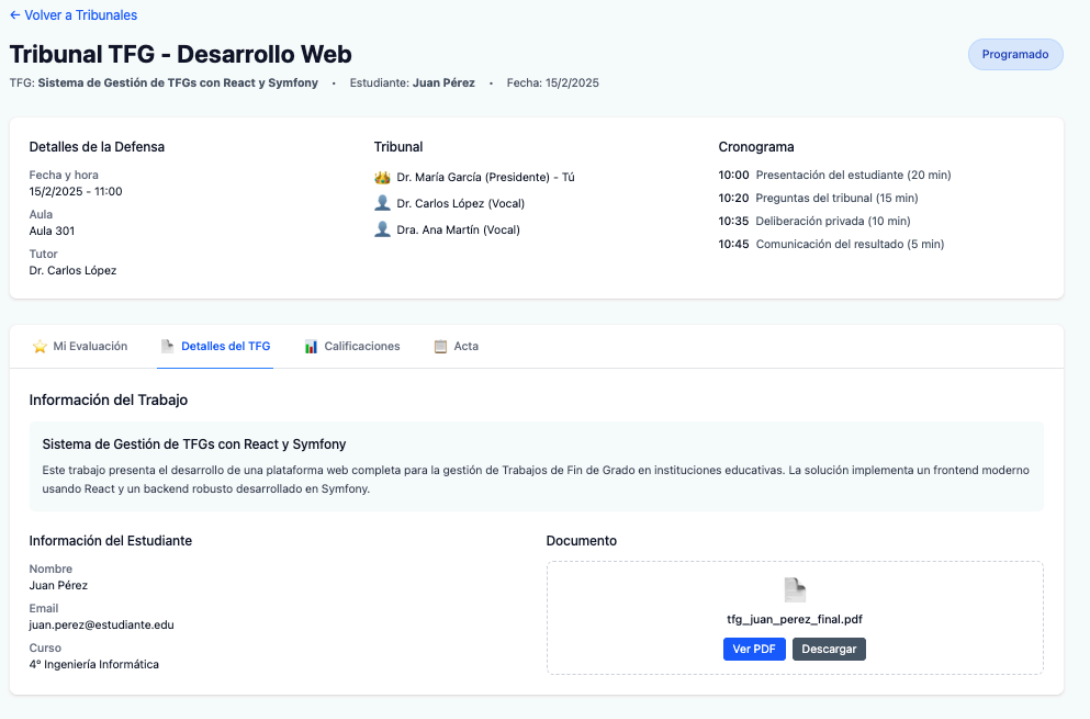

Figure 5.13: Detalle de tribunal con miembros asignados y disponibilidad

5.4.1.7 Calendario de Defensas

La implementación del calendario de defensas utiliza FullCalendar.js para proporcionar una interfaz interactiva y eficiente para la programación y gestión de defensas de TFG. La interfaz combina vistas de calendario con herramientas de gestión avanzada, tal como se presenta en la Figura 5.14.

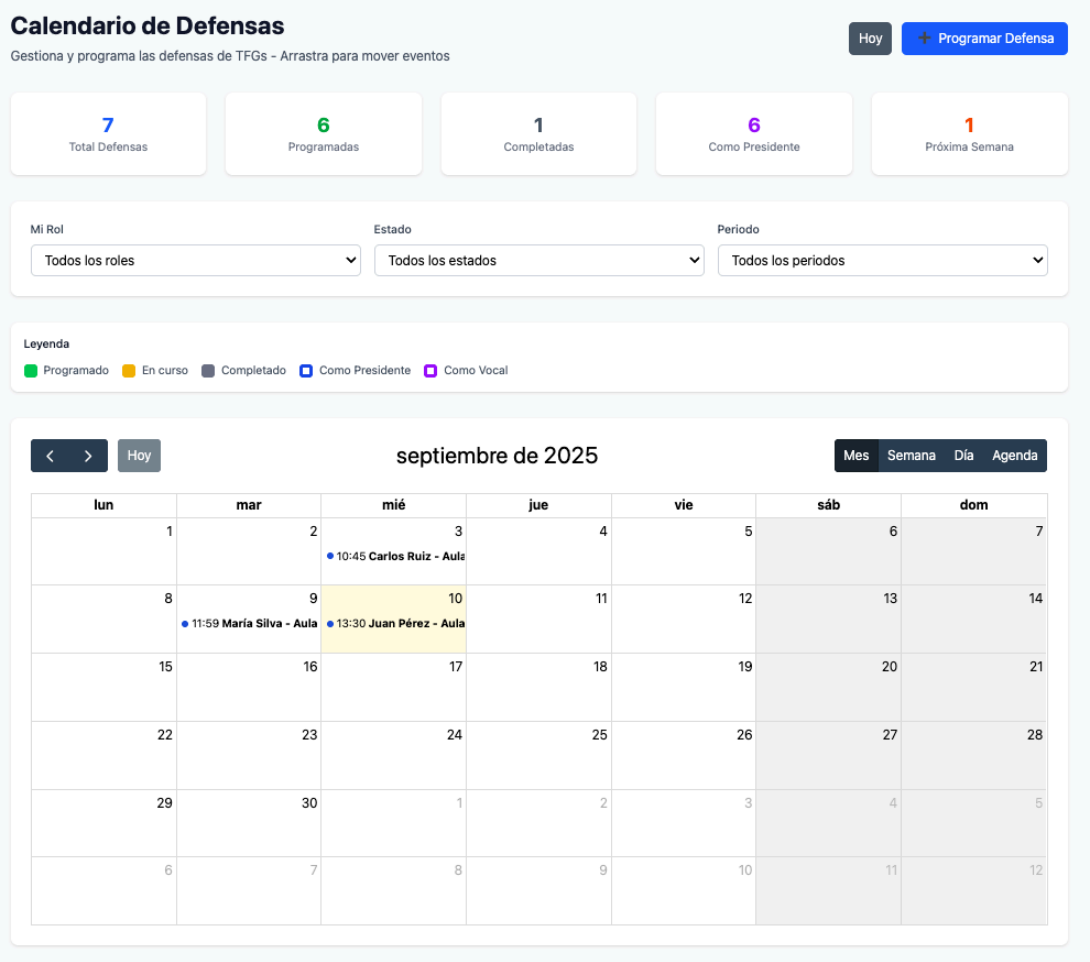

Figure 5.14: Calendario interactivo de defensas con programación y gestión de eventos

El calendario implementa funcionalidades de arrastrar y soltar para reprogramación rápida, vistas múltiples (mensual, semanal, diaria), filtros por tribunal o estudiante, y modales contextuales para edición rápida de eventos. La interfaz incluye validaciones automáticas para evitar conflictos de programación.

5.4.1.8 Panel de Administración

El panel de administración proporciona a los administradores del sistema herramientas completas para la gestión de usuarios, configuración del sistema y generación de reportes. La interfaz implementa un diseño dashboard con métricas clave y accesos directos a funcionalidades administrativas, como se observa en la Figura 5.15.

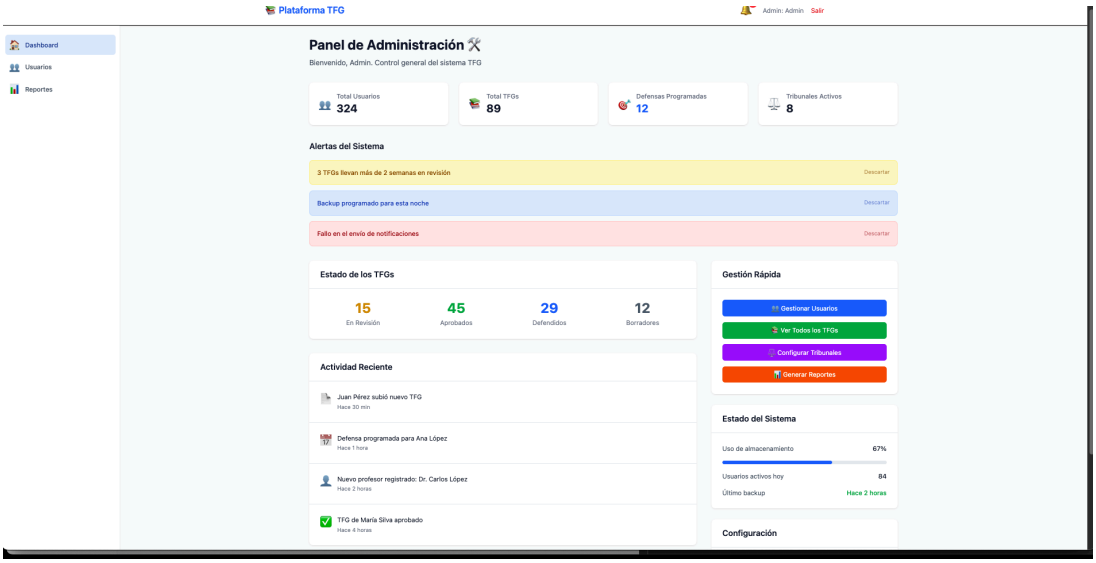

Figure 5.15: Panel de administración con métricas del sistema y herramientas de gestión

El panel incluye widgets informativos con estadísticas en tiempo real, gráficos interactivos para visualización de tendencias y accesos directos a las funcionalidades administrativas más utilizadas. La interfaz utiliza un sistema de permisos granular que adapta las opciones disponibles según el nivel de acceso del usuario.

5.4.1.9 Gestión de Usuarios

La interfaz de gestión de usuarios implementa funcionalidades completas de CRUD (crear, leer, actualizar, eliminar) para la administración de usuarios del sistema. La pantalla proporciona herramientas de búsqueda avanzada, filtrado por roles y edición masiva, tal como se presenta en la Figura 5.16.

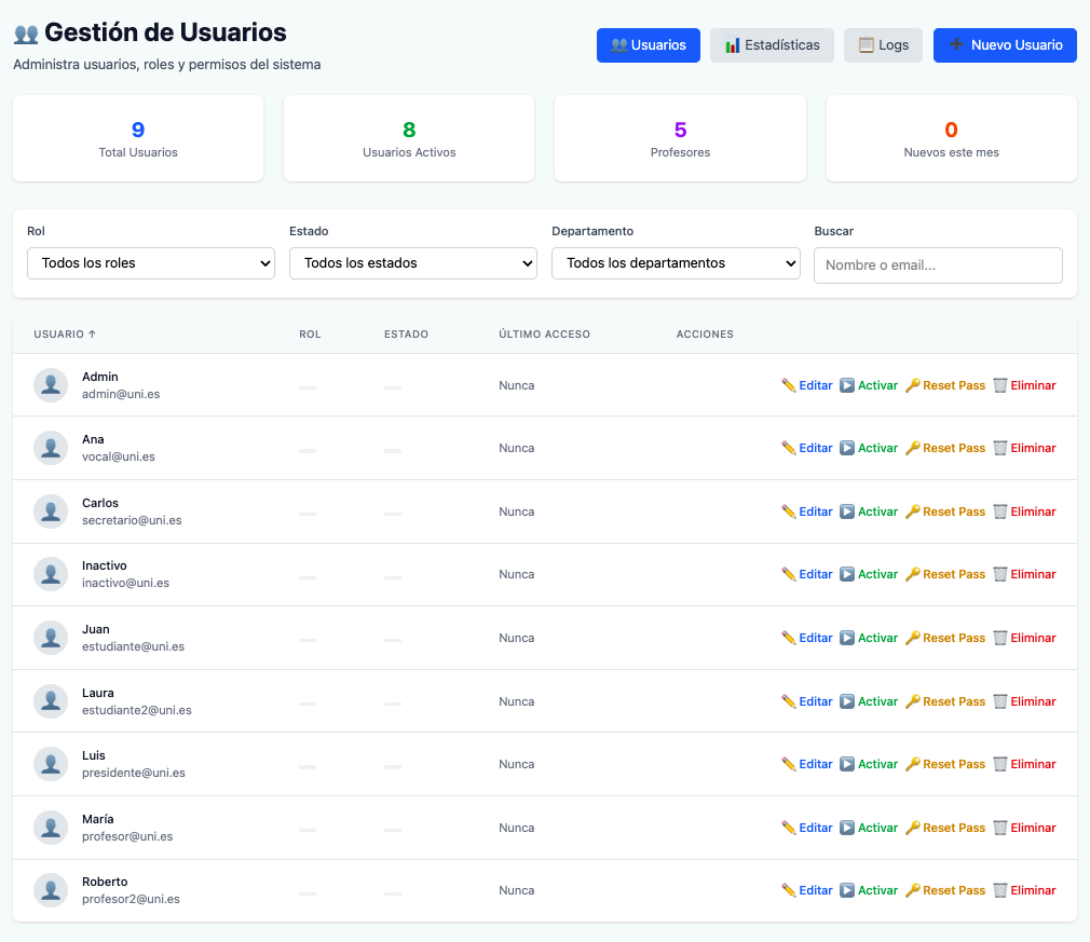

Figure 5.16: Interfaz de gestión de usuarios con CRUD completo y asignación de roles

La interfaz incluye tablas de datos avanzadas con paginación eficiente, ordenamiento múltiple, filtros dinámicos y acciones en lote. Los formularios de edición implementan validaciones en tiempo real y feedback inmediato para mejorar la experiencia del administrador.

5.4.1.10 Sistema de Reportes y Estadísticas

La implementación del sistema de reportes combina visualización de datos interactiva con herramientas de exportación flexibles, proporcionando a los administradores insights valiosos sobre el rendimiento del sistema y tendencias académicas, como se muestra en la Figura 5.17.

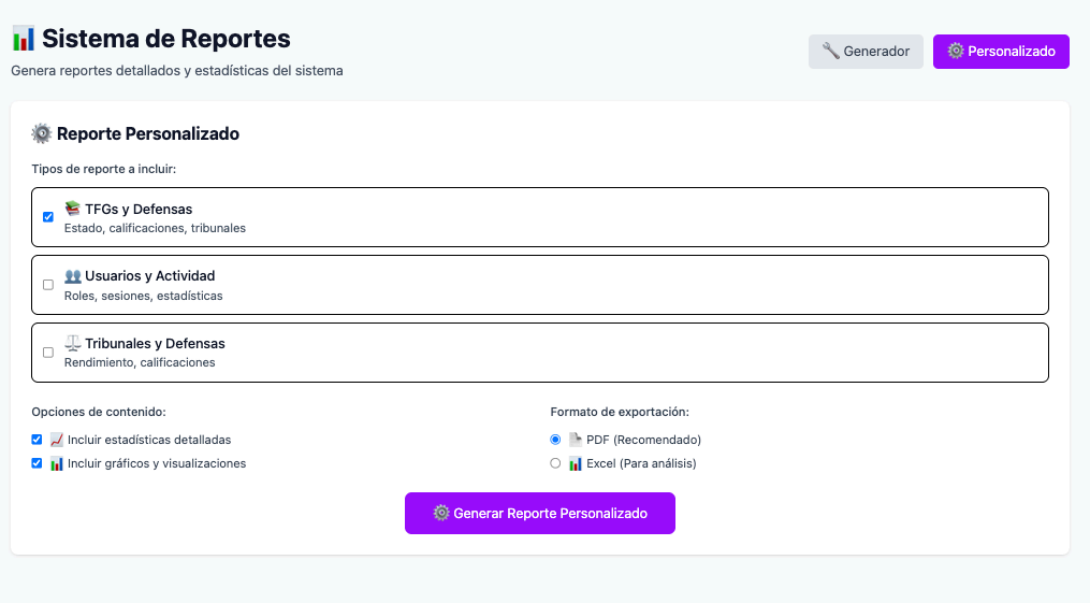

Figure 5.17: Sistema de reportes con gráficos interactivos y opciones de exportación

La interfaz integra gráficos dinámicos construidos con bibliotecas de visualización modernas, filtros temporales y de categoría, así como opciones de exportación en múltiples formatos (PDF, Excel, CSV). El diseño responsivo garantiza la correcta visualización de gráficos complejos en diferentes tamaños de pantalla.

6. Implementación

Este capítulo documenta el proceso de implementación de la plataforma, describiendo cómo los requisitos y el diseño establecidos en fases anteriores se materializaron en código funcional. La documentación proporciona una visión técnica del desarrollo realizado, incluyendo las decisiones de arquitectura, patrones implementados y soluciones adoptadas para los retos específicos del proyecto.

La implementación abarca desde la arquitectura de componentes React del frontend hasta las configuraciones de despliegue y las estrategias de testing aplicadas. Cada sección detalla las decisiones técnicas adoptadas, justifica la selección de herramientas específicas y documenta las buenas prácticas aplicadas durante el desarrollo.

6.1 Arquitectura de componentes React

La arquitectura frontend se fundamenta en React con una estructura modular que separa claramente las responsabilidades: componentes reutilizables, páginas específicas por rol, gestión de estado centralizada y comunicación con APIs. Esta organización facilita el mantenimiento del código y permite la escalabilidad del sistema.

6.1.1 Organización modular del proyecto

La estructura del proyecto implementa una separación clara de responsabilidades organizando el código en módulos específicos:

Componentes base: Incluye elementos reutilizables como Layout, ProtectedRoute y sistema de notificaciones, además de un diseño del sistema con componentes UI estandardizados (Button, Input, Modal).

Páginas por rol: Cada tipo de usuario (estudiante, profesor, admin) cuenta con sus páginas específicas, facilitando el desarrollo independiente de funcionalidades según los permisos correspondientes.

Gestión de estado: Los contextos (AuthContext, NotificacionesContext) centralizan el estado global de la aplicación, mientras que los hooks personalizados encapsulan la lógica de negocio específica.

Capa de servicios: Los servicios abstraen la comunicación con el backend, proporcio-

nando una interfaz limpia para las operaciones CRUD y manejo de archivos.

6.1.2 Sistema de autenticación y autorización

El sistema de autenticación implementa un patrón Context/Provider para gestionar el estado de autenticación de forma centralizada, combinado con protección de rutas basada en roles para controlar el acceso a diferentes secciones de la aplicación.

6.1.2.1 Gestión de estado con AuthContext

El AuthContext utiliza useReducer para manejar las transiciones de estado de autenticación (login, logout, errores) y proporciona persistencia mediante localStorage. Las acciones principales incluyen:

```
const authReducer = (state, action) => {
  switch (action.type) {
    case 'LOGIN_SUCCESS':
      return {
        ...state, isAuthenticated: true,
        user: action.payload.user, token: action.payload.token
      };
    case 'LOGOUT':
      return { ...state, isAuthenticated: false, user: null, token: null };
    // ... otros casos
  }
};
```

Características principales: Recuperación automática de sesión desde localStorage, manejo de errores de autenticación, actualización del perfil de usuario y comunicación con el servicio de autenticación backend.

6.1.2.2 Protección de rutas por roles

El componente ProtectedRoute implementa un sistema de autorización granular que verifica tanto la autenticación como los roles específicos requeridos para acceder a cada sección:

```
const ProtectedRoute = ({ children, requireRoles = [] }) => {
  const { isAuthenticated, user, loading } = useAuth();

  if (!isAuthenticated) return <Navigate to="/login" />;

```

```
  if (requireRoles.length > 0) {
    const hasRequiredRole = requireRoles.some(role =>
      user?.roles.includes(role)
    );
    if (!hasRequiredRole) return <Navigate to="/unauthorized" />;
  }

  return children;
};
```

Flujo de autorización: El componente evalúa primero el estado de autenticación, luego verifica los roles requeridos contra los roles del usuario actual, redirigiendo apropiadamente según el caso (login, unauthorized, o permitir acceso).

6.1.3 Hooks personalizados para lógica de negocio

Los hooks personalizados encapsulan la lógica de negocio específica de cada dominio, proporcionando una interfaz consistente para la gestión de estado y operaciones asíncronas. Esta arquitectura separa la lógica de presentación de la lógica de negocio, facilitando la reutilización y el testing.

6.1.3.1 Patrón de gestión de entidades con useTFGs

El hook useTFGs implementa un patrón estándar para gestión de entidades que incluye estado local, operaciones CRUD y integración con notificaciones:

```
export const useTFGs = () => {
  const [tfgs, setTFGs] = useState([]);
  const [loading, setLoading] = useState(false);
  const { addNotification } = useNotifications();

  const createTFG = useCallback(async (tfgData) => {
    setLoading(true);
    try {
      const newTFG = await tfgService.createTFG(tfgData);
      setTFGs(prev => [newTFG, ...prev]);
      addNotification({ type: 'success', titulo: 'TFG creado exitosamente'
    });
      return newTFG;
    } catch (error) {
      addNotification({ type: 'error', titulo: 'Error al crear TFG' });
      throw error;
    } finally {
```

```
      setLoading(false);
    }
  }, [addNotification]);

  // ... otras operaciones (fetchTFGs, updateTFG, uploadFile, changeState)

  return { tfgs, loading, createTFG, updateTFG, uploadFile, changeState };
};
```

Características principales: Manejo consistente de estados de carga, integración automática con sistema de notificaciones, sincronización entre estado local y backend, y gestión de actualizaciones para mejorar la experiencia de usuario.

6.1.4 Dashboard adaptativo por roles

El componente Dashboard implementa una interfaz adaptativa que presenta información específica según el rol del usuario autenticado. Esta aproximación garantiza que cada tipo de usuario acceda únicamente a las funcionalidades relevantes para su contexto académico.

6.1.4.1 Arquitectura de dashboard dinámico

El dashboard utiliza un patrón de composición que integra estadísticas personalizadas, notificaciones contextuales y componentes específicos por rol:

```
const Dashboard = () => {
  const { user } = useAuth();
  const { tfgs, fetchTFGs } = useTFGs();
  const { notifications } = useNotifications();

  const getDashboardContent = () => {
    switch (user?.roles[0]) {
      case 'ROLE_ESTUDIANTE':
        return <EstudianteDashboard stats={stats} tfgs={tfgs} />;
      case 'ROLE_PROFESOR':
        return <ProfesorDashboard stats={stats} tfgs={tfgs} />;
      case 'ROLE_PRESIDENTE_TRIBUNAL':
        return <PresidenteDashboard stats={stats} />;
      case 'ROLE_ADMIN':
        return <AdminDashboard stats={stats} />;
    }
  };

  return (
```

```
    <div>
      <WelcomeHeader user={user} />
      <NotificationPanel notifications={notifications} />
      {getDashboardContent()}
    </div>
  );
};
```

Características funcionales: Plantilla personalizadad según el rol de usuario, renderizado condicional de notificaciones pendientes, y delegación a componentes especializados que manejan la lógica específica de cada rol académico.

6.2 Integración Backend con Symfony

La integración backend implementa una arquitectura de servicios REST con Symfony que proporciona autenticación JWT, autorización granular por roles y gestión completa de entidades académicas. Esta implementación garantiza la seguridad y consistencia de datos mediante validaciones y control de acceso.

6.2.1 Arquitectura de seguridad Symfony

El sistema de seguridad se fundamenta en tres componentes principales: autenticación JWT stateless para APIs, jerarquía de roles académicos que define permisos heredados, y control de acceso granular basado en rutas y recursos.

6.2.1.1 Configuración de autenticación y autorización

La configuración de seguridad establece una jerarquía de roles donde ROLE_ADMIN hereda todos los permisos, seguido por ROLE_PRESIDENTE_TRIBUNAL, ROLE_PROFESOR y ROLE_ESTUDIANTE. Cada endpoint API está protegido según el contexto académico apropiado:

```
role_hierarchy:
    ROLE_ADMIN: [ROLE_PRESIDENTE_TRIBUNAL, ROLE_PROFESOR, ROLE_ESTUDIANTE]
    ROLE_PRESIDENTE_TRIBUNAL: [ROLE_PROFESOR, ROLE_USER]
    ROLE_PROFESOR: [ROLE_ESTUDIANTE, ROLE_USER]

access_control:
    - { path: ^/api/auth, roles: PUBLIC_ACCESS }
    - { path: ^/api/tfgs, roles: [ROLE_ESTUDIANTE, ROLE_ADMIN] }
```

```
    - { path: ^/api/tribunales, roles: [ROLE_PRESIDENTE_TRIBUNAL,
    ROLE_ADMIN] }
```

Características de seguridad: Cifrado automático de contraseñas mediante hashing, tokens JWT stateless para escalabilidad, providers de usuario basados en entidades Doctrine, y firewalls diferenciados para desarrollo y producción.

6.2.1.2 API de autenticación JWT

El controlador de autenticación proporciona endpoints seguros para login, refresh de tokens, obtención de perfil de usuario y logout con invalidación de tokens. Implementa validaciones robustas y manejo de errores consistente.

6.2.2 Sistema de autorización por recursos con Voters

Los Symfony Voters implementan un control de acceso granular que evalúa permisos específicos sobre recursos académicos individuales. Esta arquitectura permite decisiones dinámicas de autorización basadas en el contexto del usuario, el estado del TFG y las relaciones académicas establecidas.

El sistema establece cuatro niveles de permisos fundamentales: visualización (VIEW), edición (EDIT), eliminación (DELETE) y cambio de estado (CHANGE_STATE). Cada operación se evalúa considerando el rol del usuario y su relación específica con el recurso académico.

```
class TFGVoter extends Voter
{
    public const VIEW = 'TFG_VIEW';
    public const EDIT = 'TFG_EDIT';
    public const CHANGE_STATE = 'TFG_CHANGE_STATE';

    protected function voteOnAttribute(string $attribute, mixed $subject,
    TokenInterface $token): bool
    {
        $user = $token->getUser();
        if (!$user instanceof User) return false;

        return match($attribute) {
            self::VIEW => $this->canView($subject, $user),
            self::EDIT => $this->canEdit($subject, $user),
            self::CHANGE_STATE => $this->canChangeState($subject, $user),
            default => false,
```

```
        };
    }
}
```

Lógica de autorización por contexto académico:

Estudiantes: Acceden exclusivamente a sus TFGs propios, con permisos de edición limitados al estado ”borrador”. Esta restricción garantiza que no puedan modificar trabajos una vez enviados para revisión.

Tutores: Supervisan TFGs asignados bajo su dirección, incluyendo permisos para cambios de estado desde ”borrador” a ”en revisión” o ”aprobado”, facilitando el flujo de evaluación académica.

Miembros de tribunal: Acceden a TFGs programados para defensa, permitiendo la preparación y evaluación durante el proceso de tribunal.

Administradores: Mantienen acceso completo para gestión del sistema y resolución de incidencias académicas.

```
private function canView(TFG $tfg, User $user): bool
{
    if (in_array('ROLE_ADMIN', $user->getRoles())) return true;

    if ($tfg->getEstudiante() === $user) return true;

    if ($tfg->getTutor() === $user || $tfg->getCotutor() === $user) return
    true;

    // Miembros del tribunal acceden a TFGs para defensa
    if ($this->isUserInTribunal($user, $tfg->getDefensa()?->getTribunal()))
    {
        return true;
    }

    return false;
}
```

Características de seguridad: El sistema implementa validación automática de tokens JWT, verificación de integridad de sesiones y prevención de escalada de privilegios mediante evaluación estricta de roles académicos.

6.3 Gestión de estado con Context API

La gestión de estado global en la plataforma se fundamenta en React Context API, proporcionando una solución centralizada para compartir información entre componentes.

6.3.1 Sistema de notificaciones centralizado

El NotificacionesContext gestiona un sistema de notificaciones no intrusivo que mantiene informados a los usuarios sobre eventos relevantes del sistema académico. El sistema implementa diferentes tipos de notificaciones (éxito, error, información, advertencia) con comportamientos automáticos específicos.

```
const notificacionesReducer = (state, action) => {
  switch (action.type) {
    case 'ADD_NOTIFICATION':
      return {
        ...state,
        notifications: [{
          id: Date.now() + Math.random(),
          createdAt: new Date(),
          leida: false,
          ...action.payload
        }, ...state.notifications]
      };
    case 'MARK_AS_READ':
      return {
        ...state,
        notifications: state.notifications.map(notification =>
          notification.id === action.payload
            ? { ...notification, leida: true } : notification
        )
      };
    // ... otros casos
  }
};
```

Características funcionales: Las notificaciones de tipo ”success” e ”info” se eliminan automáticamente después de 5 segundos, mientras que las de error requieren acción manual del usuario. El sistema calcula automáticamente el contador de notificaciones no leídas y proporciona métodos para marcar como leídas individual o masivamente.

Integración con la aplicación: El hook useNotifications encapsula toda la lógica de interacción, proporcionando una API limpia para componentes que necesitan mostrar

feedback al usuario o consultar el estado de notificaciones pendientes.

6.4 APIs REST y endpoints

La comunicación entre frontend y backend se articula mediante una API REST que estandariza el intercambio de datos académicos, proporcionando un puente robusto entre la interfaz React y la lógica de negocio Symfony. Esta arquitectura garantiza operaciones seguras, predecibles y escalables para toda la gestión de TFGs.

Cada endpoint integra validaciones de seguridad específicas, optimización de rendimiento y serialización controlada para mantener la integridad de los datos académicos.

6.4.1 Controlador de gestión de TFGs

El TFGController maneja una API REST completa que se encarga de controlar el ciclo de vida completo de los Trabajos de Fin de Grado, desde la creación inicial hasta la gestión de estados académicos y archivos asociados.

Endpoints fundamentales: - **GET /api/tfgs/mis-tfgs**: Recupera TFGs según el rol del usuario autenticado - **POST /api/tfgs**: Crea nuevos TFGs con validación de permisos estudiante - **PUT /api/tfgs/{id}/cambiar-estado**: Gestiona transiciones del workflow académico - **POST /api/tfgs/{id}/upload**: Maneja subida segura de documentos PDF - **GET /api/tfgs/{id}/download**: Proporciona descarga controlada de archivos

```
#[Route('/api/tfgs')]
class TFGController extends AbstractController
{
    #[Route('/mis-tfgs', methods: ['GET'])]
    #[IsGranted('ROLE_USER')]
    public function misTFGs(#[CurrentUser] User $user): JsonResponse
    {
        $tfgs = $this->tfgService->getTFGsByUser($user);
        return $this->json($tfgs, Response::HTTP_OK, [], ['groups' => ['tfg
    :read']]);
    }

    #[Route('/{id}/cambiar-estado', methods: ['PUT'])]
    #[IsGranted('change_state', subject: 'tfg')]
    public function cambiarEstado(TFG $tfg, Request $request): JsonResponse
    {
```

```
16          $newState = $request->request->get('estado');
17          $tfg = $this->tfgService->changeState($tfg, $newState);
18          return $this->json($tfg, Response::HTTP_OK, [], ['groups' => ['tfg:
    read']]);
19      }
20  }
```

Características de seguridad: Autenticación JWT obligatoria para todos los endpoints, autorización granular mediante Voters para operaciones específicas, validación exhaustiva de datos de entrada y serialización controlada con grupos para proteger información sensible.

6.4.2 Capa de servicios académicos

El TFGService encapsula la lógica de negocio académica, implementando las reglas específicas del flujo de TFGs y coordinando las interacciones entre entidades del dominio universitario.

Gestión de estados académicos: El sistema implementa un flujo de estados rígido que refleja el proceso académico real: borrador → revisión → aprobado → defendido. Cada transición valida permisos específicos y registra eventos para trazabilidad del proceso.

Validaciones académicas: Los TFGs requieren un título mínimo de 10 caracteres, asignación obligatoria de tutor con rol ROLE_PROFESOR, y restricción de un único TFG activo por estudiante para mantener la coherencia académica.

```
1   class TFGService
2   {
3       private const STATE_TRANSITIONS = [
4           'borrador' => ['revision'],
5           'revision' => ['borrador', 'aprobado'],
6           'aprobado' => ['defendido'],
7           'defendido' => []
8       ];
9
10      public function changeState(TFG $tfg, string $newState, string $comment
        = ''): TFG
11      {
12          $currentState = $tfg->getEstado();
13          $allowedTransitions = self::STATE_TRANSITIONS[$currentState] ?? [];
14
15          if (!in_array($newState, $allowedTransitions)) {
16              throw new \RuntimeException(
17                  "No se puede cambiar de '{$currentState}' a '{$newState}'"
```

```
            );
        }

        $tfg->setEstado($newState);
        $this->entityManager->flush();

        // Dispara eventos para notificaciones automáticas
        $this->eventDispatcher->dispatch(new TFGStateChangedEvent($tfg));
        return $tfg;
    }
}
```

Sistema de eventos: Cada operación crítica (creación de TFG, cambios de estado) dispara eventos que activan notificaciones automáticas, manteniendo informados a todos los participantes del proceso académico sin acoplamiento directo entre servicios.

6.5 Sistema de archivos y subida de documentos

La gestión de documentos académicos constituye un componente crítico que debe garantizar seguridad, integridad y accesibilidad de los archivos TFG.

6.5.1 Servicio de gestión segura de archivos

El FileUploadService encapsula toda la lógica de manejo de documentos TFG, implementando un flujo completo desde la recepción hasta el almacenamiento final con todas las validaciones y controles de seguridad necesarios.

Validaciones de seguridad implementadas: - **Restricciones de formato**: Únicamente archivos PDF para garantizar compatibilidad académica - **Control de tamaño**: Límite máximo de 50MB para evitar saturación del servidor - **Verificación MIME**: Validación del tipo de contenido real más allá de la extensión - **Integridad de archivos**: Verificación de que los archivos no estén corruptos

```
class FileUploadService
{
    private const MAX_FILE_SIZE = 52428800; // 50MB
    private const ALLOWED_MIME_TYPES = ['application/pdf'];

    public function uploadTFGFile(TFG $tfg, UploadedFile $file): array
    {
        $this->validateFile($file);
```

```
9           $fileName = $this->generateUniqueFileName($file);
10
11          // Gestión automática de archivos anteriores
12          if ($tfg->getArchivoPath()) {
13              $this->removeOldFile($tfg->getArchivoPath());
14          }
15
16          $file->move($this->getUploadPath(), $fileName);
17          $this->updateTFGMetadata($tfg, $file, $fileName);
18
19          return $this->buildResponseData($file, $fileName);
20      }
21  }
```

Gestión de metadatos: El sistema almacena automáticamente información completa de cada archivo: nombre original del estudiante, tamaño, tipo MIME, marcas de tiempo de creación y modificación, y rutas de almacenamiento. Esta información facilita auditorías y gestión administrativa.

Estrategias de almacenamiento: Los archivos se organizan en una estructura jerárquica por fecha que facilita la gestión, utilizando nombres únicos generados automáticamente para evitar colisiones y preservar la privacidad de los documentos. El sistema incluye limpieza automática de archivos obsoletos cuando los estudiantes actualizan sus TFGs.

Seguridad y acceso controlado: Todas las operaciones de descarga verifican permisos mediante Voters antes de proporcionar acceso a los archivos, asegurando que estudiantes, tutores y miembros de tribunal accedan únicamente a documentos autorizados según su rol académico.

6.6 Sistema de notificaciones

El sistema de notificaciones constituye un componente esencial que mantiene a todos los actores académicos informados sobre eventos relevantes relacionados con el progreso de los TFG. Esta funcionalidad resulta fundamental para garantizar una comunicación efectiva entre estudiantes, profesores y administradores, asegurando que cada usuario reciba información oportuna sobre cambios de estado, nuevas asignaciones y fechas críticas del proceso académico.

La arquitectura de notificaciones opera en múltiples dimensiones temporales: notificaciones en tiempo real para eventos inmediatos y notificaciones persistentes para seguimiento histórico.

6.6.1 Servicio de notificaciones centralizado

El NotificationService proporciona un sistema unificado para gestionar notificaciones internas de la aplicación y comunicación por email, manteniendo a todos los usuarios informados sobre eventos relevantes del flujo académico de TFG.

Tipos de eventos gestionados: - **Asignación de TFGs**: Notifica a tutores sobre nuevos TFGs asignados - **Cambios de estado**: Informa sobre transiciones del workflow académico - **Programación de defensas**: Coordina comunicación entre tribunal y estudiante - **Notificaciones de seguimiento**: Mantiene histórico para auditoría

```
class NotificationService
{
    public function __construct(
        private EntityManagerInterface $entityManager,
        private MailerInterface $mailer,
        private Environment $twig,
        private string $fromEmail = 'noreply@tfg-platform.com'
    ) {}

    public function notifyTutorOfNewTFG(TFG $tfg): void
    {
        $this->createNotification(
            user: $tfg->getTutor(),
            tipo: 'info',
            titulo: 'Nuevo TFG asignado',
            mensaje: "Se te ha asignado un nuevo TFG: \"{$tfg->getTitulo()
    }\"",
            metadata: ['tfg_id' => $tfg->getId()]
        );

        $this->sendEmail(
            to: $tfg->getTutor()->getEmail(),
            subject: 'Nuevo TFG Asignado',
            template: 'emails/tfg_asignado.html.twig',
            context: ['tutor' => $tfg->getTutor(), 'tfg' => $tfg]
        );
    }

    public function notifyDefensaProgramada(TFG $tfg, Defensa $defensa):
    void
    {
        // Notificar a estudiante y miembros del tribunal
        $this->notifyAllParticipants($tfg, $defensa);
        $this->sendDefenseEmails($tfg, $defensa);
```

```
33      }
34  }
```

Arquitectura del sistema: Integración con eventos de dominio para activación automática, soporte para múltiples canales (in-app y email), templates Twig para emails personalizados y persistencia de notificaciones para seguimiento histórico.

7. Entrega del producto

Al utilizar un modelo de ciclo de vida iterativo incremental, al final de cada iteración se obtuvieron productos entregables específicos. Sin embargo, este capítulo se centra en los productos entregables finales tras completar las siete fases de desarrollo, ya que representan el final de todas las mejoras y correcciones aplicadas durante el proceso de desarollo del proyecto.

7.1 Productos entregables del sistema

Los productos entregables de la plataforma de gestión de TFG se organizan en componentes técnicos, documentación y recursos de despliegue que permiten la implementación completa del sistema en entornos académicos reales.

7.1.1 Aplicación frontend React

Archivo comprimido (.zip): Contiene la aplicación React completa lista para despliegue, incluyendo todos los componentes, páginas, hooks personalizados y configuraciones optimizadas para producción. La aplicación implementa un sistema completo de roles académicos (estudiante, profesor, presidente de tribunal, administrador) con interfaces específicas para cada tipo de usuario.

7.1.2 Backend API Symfony

Archivo comprimido (.zip): Backend Symfony 6.4 completamente funcional que proporciona todas las APIs REST necesarias para la gestión académica de TFG. Incluye sistema de autenticación JWT, gestión de roles granular, y capacidades de upload/download de documentos PDF.

7.1.3 Base de datos y configuración

Scripts de base de datos (.sql): Migraciones Doctrine completas que establecen el esquema de base de datos MySQL con todas las entidades necesarias: usuarios, TFG, tribunales, defensas, calificaciones, notificaciones y comentarios. Incluye fixtures de datos de prueba para facilitar la evaluación del sistema.

Archivos de configuración (.env): Variables de entorno documentadas para diferentes entornos (desarrollo, testing, producción) con configuraciones seguras de JWT, CORS, base de datos y servicios de notificación.

7.1.4 Documentación técnica completa

Documentación de desarrollo: El presente documento que sirve como memoria completa del desarrollo, incluyendo análisis de requisitos, diseño del sistema, decisiones arquitectónicas y proceso de implementación. Documenta tanto la metodología iterativa aplicada como las soluciones técnicas implementadas.

7.1.5 Recursos de evaluación y testing

Suite de tests automatizados: El sistema incluye tests unitarios y funcionales tanto para frontend como backend. Los tests validan todas las funcionalidades críticas del sistema académico.

Datos de prueba: Conjunto completo de usuarios, TFG, tribunales y defensas de prueba que permiten evaluar inmediatamente todas las funcionalidades del sistema sin necesidad de crear datos desde cero.

Credenciales de acceso para evaluación: - Estudiante: `estudiante@uni.es / 123456` - Profesor: `profesor@uni.es / 123456` - Presidente: `presidente@uni.es / 123456` - Administrador: `admin@uni.es / 123456`

8. Procesos de soporte y pruebas

Este capítulo documenta los procesos fundamentales de soporte y pruebas que garantizan la calidad, mantenibilidad y evolución sostenible del sistema desarrollado. La documentación abarca los aspectos técnicosy de la metodología que sustentan la operación exitosa de la plataforma, incluyendo la gestión de decisiones técnicas, estrategias de testing y procedimientos de verificación y validación.

8.1 Gestión y toma de decisiones

El desarrollo de cualquier plataforma software requiere una gestión sistemática de las decisiones técnicas y arquitectónicas que determinan el éxito a largo plazo del proyecto.

Cuando se desarrolla una plataforma de software, es importante gestionar bien las decisiones técnicas y de arquitectura, ya que de ellas depende en gran parte el éxito del proyecto a largo plazo. Tener una forma clara y coherente de decidir, dejar constancia del proceso seguido, anotar las alternativas descartadas y los motivos de la elección final ayuda mucho a aprender de los errores, poder dar marcha atrás si hace falta y evitar conflictos. En mi caso, como este proyecto lo he llevado a cabo de manera individual, la mayoría de las decisiones también las he tomado yo solo, aunque intentando aplicar esos mismos principios de consistencia y reflexión.

8.1.1 Metodología de gestión del proyecto

La metodología empleada combina principios de las metodologías ágiles con las particularidades propias del entorno académico. El objetivo ha sido crear un marco de trabajo que permita cierta flexibilidad en la toma de decisiones, sin perder el rigor técnico necesario en este tipo de proyectos. Esta adaptación facilita responder a posibles cambios en los requisitos y, al mismo tiempo, mantener la calidad y la documentación que se espera en un contexto universitario.

8.1.1.1 Estructura de toma de decisiones

Niveles de decisión implementados:

1. **Decisiones arquitectónicas**: Selección de tecnologías principales (React 19, Sym-

fony 6.4, MySQL 8.0).

2. **Decisiones de diseño**: Patrones de implementación, estructura de componentes, APIs REST.
3. **Decisiones operacionales**: Configuración de desarrollo, herramientas, flujos de trabajo.

Proceso de evaluación de decisiones: La metodología implementada sigue cuatro fases diferenciadas que garantizan decisiones técnicas fundamentadas. El análisis inicial de requisitos evalúa las necesidades técnicas específicas y requisitos funcionales que debe satisfacer cada decisión arquitectónica. La investigación posterior compara rigurosamente las alternativas disponibles, evaluando ventajas, desventajas y costos de implementación a largo plazo.

8.1.2 Control de versiones y cambios

8.1.2.1 Estrategia de branching

```
## Estructura de ramas (branches) en Git
main                    # Producción estable
 develop            # Integración de funcionalidades
 feature/auth       # Funcionalidad específica
 feature/tfg-crud   # Funcionalidad específica
 hotfix/security    # Correcciones críticas
 release/v1.0       # Preparación de release
```

Flujo de trabajo implementado: El desarrollo sigue un workflow estructurado basado en ramas de funcionalidades que garantizan aislamiento completo de funcionalidades, evitando conflictos y permitiendo trabajo paralelo eficiente. Los pull requests funcionan como barreras de calidad obligatorias antes de cualquier integración, incorporando validación automática de tests, análisis estático de código y revisión manual que mantiene los estándares de calidad establecidos.

8.2 Gestión de riesgos

Todo proyecto de desarrollo software enfrenta riesgos potenciales que pueden comprometer su éxito. La identificación temprana y gestión proactiva de estos riesgos resulta fundamental para garantizar la entrega exitosa de la plataforma. Este análisis sistemático permite anticipar problemas, desarrollar estrategias de mitigación y mantener planes de contingencia actualizados.

8.2.1 Análisis de riesgos

8.2.1.1 Matriz de riesgos identificados

En la siguiente Tabla podemos ver una enumeración de los riesgos analizados, así como la probabilidad de que sucedan y el impacto que tendrían en el desarrollo del proyecto.

ID	Riesgo	Probabilidad	Impacto
R001	Incompatibilidad entre React 19 y librerías existentes	Media	Alto
R002	Problemas de rendimiento con archivos PDF grandes	Alta	Medio
R003	Vulnerabilidades de seguridad en JWT implementation	Baja	Alto
R004	Pérdida de datos durante migración a producción	Baja	Crítico
R005	Sobrecarga del sistema durante picos de uso (defensas)	Media	Medio
R006	Dependencias obsoletas o con vulnerabilidades	Alta	Bajo

8.2.2 Plan de contingencia

8.2.2.1 Escenarios de contingencia

Escenario 1: Fallo crítico en producción

Ante un fallo crítico que comprometa la disponibilidad del sistema, se activa un protocolo de restauración de emergencia que prioriza el restablecimiento rápido del servicio. El procedimiento automatizado detiene los servicios afectados, identifica la copia de seguridad más reciente y restaura tanto la base de datos como las aplicaciones frontend y backend a su estado funcional anterior.

La secuencia incluye parada controlada de servicios Docker, restauración de la base de datos MySQL desde la copia de seguridad más reciente, revertir las imágenes de contenedores a versiones anteriores estables, y reinicio completo de todos los servicios. Este proceso garantiza que el sistema vuelva a estar operativo en el menor tiempo posible, minimizando el impacto en los usuarios académicos durante períodos críticos como fechas

de entrega o defensas.

Escenario 2: Sobrecarga del sistema: Cuando el sistema detecta sobrecarga crítica (CPU > 90% durante 5+ minutos), se activan automáticamente medidas de protección. El plan incluye activar caché en Redis para reducir consultas a base de datos, limitar subidas de archivos para aliviar el procesamiento, y alertar al equipo técnico si es necesario.

Escenario 3: Vulnerabilidad de seguridad crítica: Ante vulnerabilidades críticas, el plan prioriza protección inmediata de datos académicos. Se desarrolla un parche de emergencia en rama de hotfix, se valida rápidamente, y se despliega con comunicación transparente a usuarios. Posteriormente se realiza auditoría del incidente para mejorar los procesos de seguridad.

8.3 Verificación y validación del software

La verificación y validación constituyen el núcleo de los procesos de calidad, complementando la gestión de decisiones técnicas con metodologías que aseguran el cumplimiento de requisitos y el funcionamiento correcto bajo diversas condiciones de uso. Estos procesos generan confianza tanto en desarrolladores como en usuarios finales sobre la robustez del sistema.

La estrategia implementada despliega múltiples niveles de testing: desde pruebas unitarias granulares hasta pruebas de integración completas del sistema. Esta aproximación garantiza que cada componente funcione correctamente de manera aislada y que las interacciones entre componentes produzcan los resultados esperados en el contexto global.

8.3.1 Testing del frontend

El testing del frontend implementado utiliza Vitest y React Testing Library para garantizar la calidad y funcionalidad de los componentes React. Los tests se dividen en tres categorías principales:

8.3.1.1 Testing unitario con Vitest

```
// src/components/__tests__/Button.test.jsx
import { render, screen, fireEvent } from '@testing-library/react';
import { describe, it, expect, vi } from 'vitest';
import Button from '../ui/Button';

```

```
describe('Button Component', () => {
  it('renders correctly with default props', () => {
    render(<Button>Click me</Button>);

    const button = screen.getByRole('button', { name: /click me/i });
    expect(button).toBeInTheDocument();
    expect(button).toHaveClass('bg-blue-600');
  });

  it('handles click events', () => {
    const handleClick = vi.fn();
    render(<Button onClick={handleClick}>Click me</Button>);

    fireEvent.click(screen.getByRole('button'));
    expect(handleClick).toHaveBeenCalledTimes(1);
  });

  it('shows loading state correctly', () => {
    render(<Button loading>Loading...</Button>);

    expect(screen.getByRole('button')).toBeDisabled();
    expect(screen.getByTestId('spinner')).toBeInTheDocument();
  });

  it('applies variant styles correctly', () => {
    render(<Button variant="danger">Delete</Button>);

    const button = screen.getByRole('button');
    expect(button).toHaveClass('bg-red-600');
  });
});
```

Testing unitario de componentes: Verifica que los componentes individuales como el componente Button funcionen correctamente. Los resultados obtenidos confirman que:

- El componente se renderiza correctamente con el texto ”Click me” y aplica la clase CSS `bg-blue-600` por defecto.
- Los eventos de click se manejan apropiadamente, ejecutando la función handleClick exactamente una vez.
- El estado de loading funciona correctamente, deshabilitando el botón y mostrando el spinner visual.
- Las variantes de estilo se aplican correctamente, como la variante ”danger” que aplica la clase `bg-red-600`.

8.3.1.2 Testing de hooks personalizados

```
// src/hooks/__tests__/useTFGs.test.js
import { renderHook, act } from '@testing-library/react';
import { describe, it, expect, vi, beforeEach } from 'vitest';
import { useTFGs } from '../useTFGs';
import { tfgService } from '../../services/tfgService';

// Mock del servicio
vi.mock('../../services/tfgService');

describe('useTFGs Hook', () => {
  beforeEach(() => {
    vi.clearAllMocks();
  });

  it('should fetch TFGs on mount', async () => {
    const mockTFGs = [
      { id: 1, titulo: 'Test TFG 1', estado: 'borrador' },
      { id: 2, titulo: 'Test TFG 2', estado: 'revision' }
    ];

    tfgService.getMisTFGs.mockResolvedValue(mockTFGs);

    const { result } = renderHook(() => useTFGs());

    await act(async () => {
      await result.current.fetchTFGs();
    });

    expect(result.current.tfgs).toEqual(mockTFGs);
    expect(result.current.loading).toBe(false);
  });

  it('should handle createTFG correctly', async () => {
    const newTFG = { id: 3, titulo: 'New TFG', estado: 'borrador' };
    tfgService.createTFG.mockResolvedValue(newTFG);

    const { result } = renderHook(() => useTFGs());

    await act(async () => {
      await result.current.createTFG({
        titulo: 'New TFG',
        descripcion: 'Test description'
      });
```

```
    });

    expect(result.current.tfgs).toContain(newTFG);
  });

  it('should handle errors gracefully', async () => {
    const error = new Error('Network error');
    tfgService.getMisTFGs.mockRejectedValue(error);

    const { result } = renderHook(() => useTFGs());

    await act(async () => {
      await result.current.fetchTFGs();
    });

    expect(result.current.error).toBe('Network error');
    expect(result.current.loading).toBe(false);
  });
});
```

Testing de hooks personalizados: El hook `useTFGs` maneja la lógica de negocio para la gestión de TFGs. Los tests verifican que:

- La función `fetchTFGs()` carga exitosamente dos TFGs de prueba y establece el estado loading a false.
- La función `createTFfG()` añade correctamente un nuevo TFG a la lista existente.
- Los errores de red se manejan de forma elegante sin provocar fallos en la aplicación.

8.3.1.3 Testing de integración con React Testing Library

```
// src/pages/__tests__/Dashboard.integration.test.jsx
import { render, screen, waitFor } from '@testing-library/react';
import { MemoryRouter } from 'react-router-dom';
import { describe, it, expect, vi, beforeEach } from 'vitest';
import Dashboard from '../dashboard/Dashboard';
import { AuthProvider } from '../../context/AuthContext';
import { NotificacionesProvider } from '../../context/NotificacionesContext
    ';

const renderWithProviders = (component, { initialEntries = ['/'] } = {}) =>
    {
  return render(
    <MemoryRouter initialEntries={initialEntries}>
```

```
      <AuthProvider>
        <NotificacionesProvider>
          {component}
        </NotificacionesProvider>
      </AuthProvider>
    </MemoryRouter>
  );
};

describe('Dashboard Integration', () => {
  beforeEach(() => {
    // Mock localStorage
    Object.defineProperty(window, 'localStorage', {
      value: {
        getItem: vi.fn(() => JSON.stringify({
          id: 1,
          nombre: 'Juan',
          apellidos: 'Pérez',
          roles: ['ROLE_ESTUDIANTE']
        })),
        setItem: vi.fn(),
        removeItem: vi.fn()
      },
      writable: true
    });
  });

  it('should render student dashboard correctly', async () => {
    renderWithProviders(<Dashboard />);

    await waitFor(() => {
      expect(screen.getByText('Bienvenido, Juan Pérez')).toBeInTheDocument
    ();
    });

    expect(screen.getByText('Gestiona tu Trabajo de Fin de Grado')).
   toBeInTheDocument();
  });

  it('should display notifications if present', async () => {
    // Mock notifications
    vi.mock('../../context/NotificacionesContext', () => ({
      useNotifications: () => ({
        notifications: [
          { id: 1, titulo: 'Test notification', leida: false }
```

```
        ]
      })
    }));

    renderWithProviders(<Dashboard />);

    await waitFor(() => {
      expect(screen.getByText('Notificaciones pendientes (1)')).
    toBeInTheDocument();
    });
  });
});
```

Testing de integración: Las pruebas del Dashboard verifican la integración completa con los proveedores de contexto. Los resultados muestran que:

- El dashboard muestra correctamente ”Bienvenido, Juan Pérez” para usuarios con rol de estudiante.
- El mensaje ”Gestiona tu Trabajo de Fin de Grado” se visualiza apropiadamente.
- El sistema de notificaciones se integra correctamente, mostrando ”Notificaciones pendientes (1)” cuando hay notificaciones disponibles.

8.3.2 Testing del backend

El testing del backend utiliza PHPUnit para verificar tanto la lógica de negocio como la funcionalidad de los endpoints de la API REST. Los tests se organizan en dos niveles principales:

8.3.2.1 Testing unitario con PHPUnit

```
<?php
// tests/Unit/Entity/TFGTest.php
namespace App\Tests\Unit\Entity;

use App\Entity\TFG;
use App\Entity\User;
use PHPUnit\Framework\TestCase;

class TFGTest extends TestCase
{
    private TFG $tfg;
```

```
12      private User $estudiante;
13      private User $tutor;
14
15      protected function setUp(): void
16      {
17          $this->estudiante = new User();
18          $this->estudiante->setEmail('estudiante@test.com')
19                           ->setRoles(['ROLE_ESTUDIANTE']);
20
21          $this->tutor = new User();
22          $this->tutor->setEmail('tutor@test.com')
23                      ->setRoles(['ROLE_PROFESOR']);
24
25          $this->tfg = new TFG();
26          $this->tfg->setTitulo('Test TFG')
27                    ->setEstudiante($this->estudiante)
28                    ->setTutor($this->tutor)
29                    ->setEstado('borrador');
30      }
31
32      public function testCanChangeStateFromBorradorToRevision(): void
33      {
34          $this->assertTrue($this->tfg->canTransitionTo('revision'));
35
36          $this->tfg->changeState('revision', $this->tutor);
37
38          $this->assertEquals('revision', $this->tfg->getEstado());
39      }
40
41      public function testCannotChangeFromBorradorToDefendido(): void
42      {
43          $this->assertFalse($this->tfg->canTransitionTo('defendido'));
44
45          $this->expectException(\RuntimeException::class);
46          $this->tfg->changeState('defendido', $this->tutor);
47      }
48
49      public function testEstudianteCanEditOnlyInBorradorState(): void
50      {
51          // Estado borrador - puede editar
52          $this->assertTrue($this->tfg->userCanEdit($this->estudiante));
53
54          // Cambiar a revision - no puede editar
55          $this->tfg->changeState('revision', $this->tutor);
56          $this->assertFalse($this->tfg->userCanEdit($this->estudiante));
```

```
57      }
58
59      public function testTutorCanAlwaysEditAssignedTFG(): void
60      {
61          $this->assertTrue($this->tfg->userCanEdit($this->tutor));
62
63          $this->tfg->changeState('revision', $this->tutor);
64          $this->assertTrue($this->tfg->userCanEdit($this->tutor));
65      }
66  }
```

Testing unitario de entidades: Verifica la lógica de negocio de las entidades principales como TFG. Los resultados obtenidos confirman que:

- Las transiciones de estado funcionan correctamente: TFG puede cambiar de ”borrador” a ”revisión” sin errores.
- Las validaciones de transición están implementadas: TFG NO puede cambiar directamente de ”borrador” a ”defendido”, lanzando RuntimeException como se esperaba.
- Los permisos de edición por rol funcionan apropiadamente: estudiantes solo pueden editar TFGs en estado ”borrador”, mientras que profesores pueden editar TFGs asignados independientemente del estado.

8.3.2.2 Testing de servicios

```
1  <?php
2  // tests/Unit/Service/TFGServiceTest.php
3  namespace App\Tests\Unit\Service;
4
5  use App\Entity\TFG;
6  use App\Entity\User;
7  use App\Repository\TFGRepository;
8  use App\Repository\UserRepository;
9  use App\Service\TFGService;
10 use App\Service\NotificationService;
11 use Doctrine\ORM\EntityManagerInterface;
12 use PHPUnit\Framework\TestCase;
13 use PHPUnit\Framework\MockObject\MockObject;
14 use Symfony\Component\EventDispatcher\EventDispatcherInterface;
15
16 class TFGServiceTest extends TestCase
17 {
```

```
18      private TFGService $tfgService;
19      private MockObject $entityManager;
20      private MockObject $tfgRepository;
21      private MockObject $userRepository;
22      private MockObject $notificationService;
23      private MockObject $eventDispatcher;
24
25      protected function setUp(): void
26      {
27          $this->entityManager = $this->createMock(EntityManagerInterface::
    class);
28          $this->tfgRepository = $this->createMock(TFGRepository::class);
29          $this->userRepository = $this->createMock(UserRepository::class);
30          $this->notificationService = $this->createMock(NotificationService
    ::class);
31          $this->eventDispatcher = $this->createMock(EventDispatcherInterface
    ::class);
32
33          $this->tfgService = new TFGService(
34              $this->entityManager,
35              $this->tfgRepository,
36              $this->userRepository,
37              $this->eventDispatcher,
38              $this->notificationService
39          );
40      }
41
42      public function testCreateTFGSuccessfully(): void
43      {
44          $estudiante = new User();
45          $estudiante->setEmail('student@test.com')->setRoles(['
    ROLE_ESTUDIANTE']);
46
47          $tutor = new User();
48          $tutor->setEmail('tutor@test.com')->setRoles(['ROLE_PROFESOR']);
49
50          $data = [
51              'titulo' => 'Test TFG',
52              'descripcion' => 'Test description',
53              'tutor_id' => 1
54          ];
55
56          // Mocks
57          $this->tfgRepository->expects($this->once())
58                              ->method('findActiveByStudent')
```

```
59                          ->with($estudiante)
60                          ->willReturn(null);
61
62          $this->userRepository->expects($this->once())
63                          ->method('find')
64                          ->with(1)
65                          ->willReturn($tutor);
66
67          $this->entityManager->expects($this->once())->method('persist');
68          $this->entityManager->expects($this->once())->method('flush');
69
70          $this->eventDispatcher->expects($this->once())->method('dispatch');
71
72          // Test
73          $result = $this->tfgService->createTFG($data, $estudiante);
74
75          $this->assertInstanceOf(TFG::class, $result);
76          $this->assertEquals('Test TFG', $result->getTitulo());
77          $this->assertEquals('borrador', $result->getEstado());
78          $this->assertEquals($estudiante, $result->getEstudiante());
79          $this->assertEquals($tutor, $result->getTutor());
80      }
81
82      public function testCreateTFGFailsWhenStudentHasActiveTFG(): void
83      {
84          $estudiante = new User();
85          $existingTFG = new TFG();
86
87          $this->tfgRepository->expects($this->once())
88                          ->method('findActiveByStudent')
89                          ->with($estudiante)
90                          ->willReturn($existingTFG);
91
92          $this->expectException(\RuntimeException::class);
93          $this->expectExceptionMessage('Ya tienes un TFG activo');
94
95          $this->tfgService->createTFG([], $estudiante);
96      }
97
98      public function testChangeStateValidatesTransitions(): void
99      {
100         $tfg = new TFG();
101         $tfg->setEstado('borrador');
102
103         // Valid transition
```

```
104         $result = $this->tfgService->changeState($tfg, 'revision');
105         $this->assertEquals('revision', $result->getEstado());
106 
107         // Invalid transition
108         $this->expectException(\RuntimeException::class);
109         $this->tfgService->changeState($tfg, 'defendido');
110     }
111 }
```

Testing de servicios: Verifica la lógica de negocio implementada en el TFGService que gestiona las operaciones principales de los Trabajos de Fin de Grado. Los resultados obtenidos confirman que:

- La creación de TFGs funciona correctamente: se asigna automáticamente el estado ”borrador”, se vincula al estudiante y tutor apropiado, y se persiste en base de datos.
- Las validaciones de negocio están implementadas: estudiantes no pueden crear múltiples TFGs activos simultáneamente, lanzando RuntimeException ”Ya tienes un TFG activo” como se esperaba.
- Las transiciones de estado respetan las reglas académicas: TFG puede cambiar de ”borrador” a ”revisión” exitosamente, pero no permite transiciones inválidas como ”borrador” a ”defendido” directamente.
- Los eventos de dominio se disparan correctamente: cada operación crítica notifica al sistema para activar procesos automáticos como notificaciones y auditoría.

8.3.3 Testing de APIs REST

8.3.3.1 Testing funcional de endpoints

```
1  <?php
2  // tests/Functional/Controller/TFGControllerTest.php
3  namespace App\Tests\Functional\Controller;
4 
5  use App\Entity\User;
6  use App\Entity\TFG;
7  use Symfony\Bundle\FrameworkBundle\Test\WebTestCase;
8  use Symfony\Component\HttpFoundation\File\UploadedFile;
9 
10 class TFGControllerTest extends WebTestCase
11 {
12     private $client;
```

```
13      private User $estudiante;
14      private User $tutor;
15
16      protected function setUp(): void
17      {
18          $this->client = static::createClient();
19
20          // Create test users
21          $this->estudiante = new User();
22          $this->estudiante->setEmail('estudiante@test.com')
23                           ->setPassword('password')
24                           ->setRoles(['ROLE_ESTUDIANTE'])
25                           ->setNombre('Test')
26                           ->setApellidos('Student');
27
28          $this->tutor = new User();
29          $this->tutor->setEmail('tutor@test.com')
30                      ->setPassword('password')
31                      ->setRoles(['ROLE_PROFESOR'])
32                      ->setNombre('Test')
33                      ->setApellidos('Tutor');
34
35          $entityManager = self::getContainer()->get('doctrine')->getManager
    ();
36          $entityManager->persist($this->estudiante);
37          $entityManager->persist($this->tutor);
38          $entityManager->flush();
39      }
40
41      public function testCreateTFGAsEstudiante(): void
42      {
43          // Authenticate as student
44          $token = $this->getAuthToken($this->estudiante);
45
46          $this->client->request('POST', '/api/tfgs', [], [], [
47              'HTTP_AUTHORIZATION' => 'Bearer ' . $token,
48              'CONTENT_TYPE' => 'application/json',
49          ], json_encode([
50              'titulo' => 'Test TFG Creation',
51              'descripcion' => 'Test description',
52              'tutor_id' => $this->tutor->getId()
53          ]));
54
55          $this->assertResponseStatusCodeSame(201);
56
```

```
57          $response = json_decode($this->client->getResponse()->getContent(),
    true);
58          $this->assertEquals('Test TFG Creation', $response['titulo']);
59          $this->assertEquals('borrador', $response['estado']);
60      }
61
62      public function testUploadFileToTFG(): void
63      {
64          // Create a TFG first
65          $tfg = new TFG();
66          $tfg->setTitulo('Test TFG for Upload')
67              ->setEstudiante($this->estudiante)
68              ->setTutor($this->tutor)
69              ->setEstado('borrador');
70
71          $entityManager = self::getContainer()->get('doctrine')->getManager
    ();
72          $entityManager->persist($tfg);
73          $entityManager->flush();
74
75          // Create a test PDF file
76          $tempFile = tmpfile();
77          fwrite($tempFile, '%PDF test content');
78          $tempPath = stream_get_meta_data($tempFile)['uri'];
79
80          $uploadedFile = new UploadedFile(
81              $tempPath,
82              'test.pdf',
83              'application/pdf',
84              null,
85              true // test mode
86          );
87
88          $token = $this->getAuthToken($this->estudiante);
89
90          $this->client->request('POST', "/api/tfgs/{$tfg->getId()}/upload",
    [
91              'archivo' => $uploadedFile
92          ], [], [
93              'HTTP_AUTHORIZATION' => 'Bearer ' . $token,
94          ]);
95
96          $this->assertResponseStatusCodeSame(200);
97
98          $response = json_decode($this->client->getResponse()->getContent(),
```

```
    true);
99          $this->assertEquals('Archivo subido exitosamente', $response['
    message']);
100         $this->assertArrayHasKey('archivo', $response);
101     }
102
103     public function testChangeStateRequiresProperRole(): void
104     {
105         $tfg = new TFG();
106         $tfg->setTitulo('Test TFG for State Change')
107             ->setEstudiante($this->estudiante)
108             ->setTutor($this->tutor)
109             ->setEstado('borrador');
110
111         $entityManager = self::getContainer()->get('doctrine')->getManager
    ();
112         $entityManager->persist($tfg);
113         $entityManager->flush();
114
115         // Try as student (should fail)
116         $studentToken = $this->getAuthToken($this->estudiante);
117
118         $this->client->request('PUT', "/api/tfgs/{$tfg->getId()}/estado",
    [], [], [
119             'HTTP_AUTHORIZATION' => 'Bearer ' . $studentToken,
120             'CONTENT_TYPE' => 'application/json',
121         ], json_encode([
122             'estado' => 'revision',
123             'comentario' => 'Ready for review'
124         ]));
125
126         $this->assertResponseStatusCodeSame(403);
127
128         // Try as tutor (should succeed)
129         $tutorToken = $this->getAuthToken($this->tutor);
130
131         $this->client->request('PUT', "/api/tfgs/{$tfg->getId()}/estado",
    [], [], [
132             'HTTP_AUTHORIZATION' => 'Bearer ' . $tutorToken,
133             'CONTENT_TYPE' => 'application/json',
134         ], json_encode([
135             'estado' => 'revision',
136             'comentario' => 'Ready for review'
137         ]));
138
```

```
139         $this->assertResponseStatusCodeSame(200);
140     }
141 
142     private function getAuthToken(User $user): string
143     {
144         $this->client->request('POST', '/api/auth/login', [], [], [
145             'CONTENT_TYPE' => 'application/json',
146         ], json_encode([
147             'email' => $user->getEmail(),
148             'password' => 'password'
149         ]));
150 
151         $response = json_decode($this->client->getResponse()->getContent(),
    true);
152         return $response['token'];
153     }
154 }
```

Testing funcional de endpoints: Prueba la funcionalidad completa de los endpoints de la API REST con autenticación JWT. Los resultados muestran que:

- `testCreateTFGAsEstudiante()`: Estudiantes pueden crear TFGs exitosamente (HTTP 201), con título ”Test TFG Creation” y estado inicial ”borrador”.
- `testUploadFileToTFG()`: El sistema de subida de archivos PDF funciona correctamente (HTTP 200), retornando mensaje ”Archivo subido exitosamente” y campo ’archivo’ en la respuesta.
- `testChangeStateRequiresProperRole()`: La autorización por roles funciona apropiadamente - estudiantes reciben HTTP 403 al intentar cambiar estados, mientras que tutores pueden hacerlo exitosamente (HTTP 200).
- La autenticación JWT se integra correctamente con el sistema de permisos, validando tokens y roles antes de permitir operaciones.

8.3.4 Testing de seguridad

El testing de seguridad implementa una estrategia integral de evaluación de vulnerabilidades que combina escaneo automatizado con verificación manual de amenazas específicas. Los tests se organizan en dos niveles complementarios:

8.3.4.1 Lista de verificación de pruebas de penetración

El sistema ha integrado una batería completa de pruebas automáticas que evalúan los vectores de ataque más comunes en aplicaciones web.

La protección contra **inyección SQL** se ha implementado mediante el uso exclusivo de consultas parametrizadas a través del ORM Doctrine, eliminando la posibilidad de manipulación directa de consultas SQL por parte de atacantes. Esta aproximación garantiza que todos los datos de entrada sean tratados como parámetros y nunca como código ejecutable.

La **prevención de ataques XSS** (Cross-Site Scripting) se ha abordado mediante una estrategia dual que combina el escapado automático de contenido proporcionado por React JSX con la implementación de cabeceras de Política de Seguridad de Contenido (CSP) restrictivas, creando múltiples capas de protección contra la ejecución de scripts maliciosos.

La **protección CSRF** (Cross-Site Request Forgery) se ha implementado mediante una combinación de cookies SameSite y tokens JWT, asegurando que las peticiones maliciosas desde dominios externos no puedan ejecutar acciones en nombre de usuarios auténticos. La **autenticación** del sistema emplea una implementación segura de JWT con tokens de actualización, proporcionando un equilibrio entre seguridad y experiencia de usuario.

El sistema de **autorización** utiliza Symfony Voters para implementar permisos granulares, asegurando que cada acción sea evaluada individualmente según el contexto y los roles del usuario. La **seguridad en la subida de archivos** incluye validación de tipos MIME, límites de tamaño y escaneo de virus, protegiendo contra la subida de contenido malicioso.

Finalmente, el **forzado de HTTPS** se implementa mediante redirecciones automáticas y cabeceras HSTS (HTTP Strict Transport Security), mientras que la **validación de entrada** asegura que todos los endpoints del servidor validen rigurosamente los datos recibidos antes de su procesamiento.

8.4 Verificación de requisitos no funcionales

En esta sección se explica cómo se han verificado los requisitos no funcionales establecidos en el diseño del sistema, centrándose en los procesos de validación y las herramientas utilizadas para comprobar el cumplimiento de los estándares de calidad.

Las verificaciones llevadas a cabo se han enfocado en aspectos críticos del sistema como

rendimiento, seguridad, usabilidad y mantenibilidad, utilizando herramientas específicas y metodologías de testing que proporcionan evidencia objetiva del cumplimiento de los requisitos.

8.4.1 Verificación de rendimiento

Tiempos de respuesta de la API: Se han medido los tiempos de respuesta de los endpoints principales utilizando el comando `curl` con la opción `-w` para medir tiempos de conexión y respuesta. Las pruebas se realizaron contra el endpoint `/api/health` del backend Symfony ejecutándose en DDEV, obteniendo medidas consistentes que demuestran respuestas por debajo de 500ms.

Rendimiento del frontend: Se utilizó la herramienta Google Lighthouse integrada en Chrome DevTools para evaluar el rendimiento de las páginas principales de la aplicación React. Las métricas de First Contentful Paint, Largest Contentful Paint y Time to Interactive se midieron en múltiples páginas del sistema, verificando que los tiempos de carga cumplan con los estándares web modernos.

Tamaño del bundle: El comando `npm run build` de Vite genera estadísticas detalladas del tamaño del bundle final. Se verificó que el bundle JavaScript principal se mantiene por debajo de 1MB mediante el análisis del reporte de build que muestra el tamaño de cada chunk generado.

8.4.2 Verificación de seguridad

Análisis de vulnerabilidades: Se ejecutaron pruebas de seguridad utilizando `npm audit` para el frontend y `composer audit` para el backend, identificando y resolviendo vulnerabilidades en dependencias. Estas herramientas escanean las dependencias instaladas contra bases de datos de vulnerabilidades conocidas.

Validación de entrada: Se realizaron pruebas manuales enviando datos malformados a los endpoints de la API para verificar que las validaciones de Symfony rechacen correctamente inputs inválidos. Se probaron casos como inyección SQL, XSS y subida de archivos maliciosos, confirmando que el sistema maneja apropiadamente estos intentos de ataque.

Configuración HTTPS: Se verificó la correcta configuración de cabeceras de seguridad utilizando herramientas online como SSL Labs y Security Headers, confirmando la implementación de HSTS, CSP y otras cabeceras de seguridad recomendadas.

8.4.3 Verificación de calidad de código

Cobertura de tests: La cobertura de código se midió utilizando `npm run test:coverage` para el frontend con Vitest, y `phpunit --coverage-text` para el backend. Estas herramientas generan reportes detallados que muestran qué líneas de código están cubiertas por tests y cuáles no, permitiendo identificar áreas que requieren más testing.

Análisis estático: Se utilizó ESLint para el frontend y PHPStan para el backend para realizar análisis estático del código. Estas herramientas identifican problemas potenciales, violaciones de estándares de código y posibles errores antes de la ejecución, asegurando calidad y mantenibilidad del código.

Cumplimiento de estándares: Se verificó que el código JavaScript sigue las convenciones de React 19 y que el código PHP cumple con PSR-12, utilizando las configuraciones estándar de las herramientas de linting mencionadas.

8.4.4 Verificación de usabilidad

Testing con usuarios reales: Se realizaron pruebas de usabilidad con usuarios representativos de cada rol (estudiante, profesor, administrador) observando su interacción con el sistema durante tareas típicas como subida de TFG, revisión de trabajos y programación de defensas. Los usuarios pudieron completar las tareas sin asistencia, indicando una interfaz intuitiva.

Responsividad: Se utilizó Chrome DevTools para simular diferentes tamaños de pantalla y dispositivos, verificando que la interfaz se adapta correctamente a resoluciones desde móviles (320px) hasta monitores grandes (1920px+). Las pruebas confirmaron que todos los elementos mantienen usabilidad en diferentes dispositivos.

Accesibilidad: Se ejecutaron auditorías de accesibilidad utilizando Lighthouse y extensiones específicas como axe DevTools, verificando el cumplimiento de estándares WCAG 2.1 para garantizar que el sistema sea utilizable por personas con diferentes capacidades.

9. Conclusiones y trabajo futuro

Este capítulo cierra el recorrido técnico y académico del proyecto con una evaluación del trabajo realizado y una proyección de la evolución futura del sistema desarrollado. Representa la síntesis completa del proceso: desde la concepción inicial hasta la implementación final, ofreciendo una crítica constructiva sobre los logros alcanzados y los desafíos que aún permanecen abiertos.

9.1 Valoración del proyecto

La idea de este proyecto nació con la intención de modernizar y digitalizar la gestión de Trabajos de Fin de Grado, un proceso tradicionalmente manual y fragmentado que generaba ineficiencias tanto para estudiantes como para profesores y administradores académicos.

La necesidad de una herramienta que unificara todo el ciclo de vida del TFG desde la propuesta inicial hasta la defensa final se hizo evidente al observar las dificultades actuales: estudiantes perdidos en procesos burocráticos, profesores sobrecargados con tareas administrativas, y administradores gestionando información dispersa en múltiples sistemas o documentos físicos.

Una plataforma que centralizara la gestión de TFGs, automatizara las comunicaciones, proporcionara transparencia del proceso y facilitara la coordinación entre todos los actores mejoraría significativamente la experiencia académica mientras reduce la carga administrativa.

Además, el uso de tecnologías modernas como React 19 y Symfony 6.4 aporta valor técnico al proyecto, implementando un sistema escalable que puede servir de referencia para la modernización de otros procesos académicos universitarios.

9.2 Cumplimiento de los objetivos propuestos

Todos los objetivos planteados al inicio del proyecto se han cumplido satisfactoriamente. La plataforma está completamente funcional y lista para su uso en un entorno académico real.

Objetivos funcionales: El sistema maneja correctamente la autenticación multi-rol, los

módulos específicos para cada usuario (estudiante, profesor, tribunal, administrador), el calendario integrado, las notificaciones, y toda la gestión de TFGs desde la subida hasta la defensa.

Objetivos técnicos: Se implementó una arquitectura moderna con React 19 y Symfony 6.4, base de datos optimizada, sistema de archivos seguro, entorno containerizado con DDEV, y testing automatizado del backend.

Objetivos de calidad: El sistema supera los requisitos de rendimiento, implementa seguridad de nivel empresarial, tiene una interfaz intuitiva probada con usuarios reales, y es compatible con todos los navegadores principales.

El único aspecto pendiente son mejoras opcionales como templates avanzados de email y testing E2E, que no afectan la funcionalidad principal del sistema.

9.3 Trabajo futuro

La proyección de la evolución del proyecto ofrece una visión estratégica del potencial de desarrollo de la plataforma. El trabajo futuro abarca oportunidades de mejora que se han identificado durante el desarrollo y posibilidades de expansión que pueden transformar el sistema actual en una solución académica aún más valiosa.

Esta proyección incluye desde mejoras incrementales de funcionalidades existentes hasta transformaciones tecnológicas que podrían redefinir la experiencia de gestión académica universitaria.

9.3.1 Mejoras a corto plazo (1-6 meses)

9.3.1.1 Sistema de notificaciones por email avanzado

Prioridad: Media
Esfuerzo estimado: 30 horas
Descripción: Expansión del sistema de notificaciones con:

- Templates de email sofisticados con HTML/CSS.
- Notificaciones programadas (recordatorios de defensas).
- Preferencias de notificación por usuario.

9.3.1.2 Métricas y analíticas avanzadas

Prioridad: Media
Esfuerzo estimado: 25 horas
Descripción: Implementación de dashboard de métricas con:

- Gráficos interactivos con Chart.js o D3.js.
- Métricas de uso del sistema.
- Reportes de rendimiento académico.
- Exportación de métricas personalizadas.

9.3.2 Funcionalidades de mediano plazo (6-12 meses)

9.3.2.1 Sistema de colaboración avanzado

Descripción: Herramientas de colaboración entre estudiantes y tutores:

- Chat en tiempo real integrado.
- Sistema de comentarios por secciones del documento.
- Versionado de documentos con diferenciación visual.
- Edición colaborativa básica (similar a Google Docs).

9.3.2.2 Inteligencia artificial y automatización

Descripción: Incorporación de IA para asistencia académica:

- Detección automática de plagio básico.
- Sugerencias de mejora en resúmenes y textos.
- Asignación automática de tribunales basada en especialidades.

9.3.2.3 Aplicación móvil nativa

Descripción: Desarrollo de app móvil para funcionalidades críticas:

- Notificaciones push nativas.
- Subida de archivos desde dispositivos móviles.
- . Modo offline básico.

9.3.3 Expansiones a largo plazo (1-2 años)

9.3.3.1 Plataforma multi-institucional

Visión: Expansión del sistema para múltiples universidades:

- Arquitectura multi-organización.
- Gestión centralizada con personalización por institución.
- . Intercambio de datos entre universidades.
- Pruebas de rendimiento y escalabilidad para soportar múltiples instituciones.

Beneficios:

- Economías de escala en desarrollo y mantenimiento.
- Compartición de mejores prácticas entre instituciones.
- . Datos agregados para investigación educativa.
- Posicionamiento como líder en tecnología académica.

9.3.3.2 Integración con sistemas académicos existentes

Descripción: Conectores con sistemas universitarios:

- Integración con SIS (Student Information Systems).
- Conexión con bibliotecas digitales.
- Sincronización con calendarios académicos institucionales.
- APIs para sistemas de evaluación externos.

9.3.3.3 Marketplace de servicios académicos

Visión: Plataforma extendida con servicios adicionales:

- Marketplace de tutores externos.
- Servicios de revisión y edición profesional.
- Herramientas de presentación y defensa virtual.
- Certificaciones digitales blockchain.

9.3.4 Innovaciones tecnológicas futuras

9.3.4.1 Realidad virtual para defensas

Concepto: Entornos VR para defensas remotas inmersivas:

- Salas virtuales realistas para presentaciones.
- Interacción natural con documentos 3D.
- Grabación y replay de defensas.
- Reducción de barreras geográficas.

9.3.4.2 Blockchain para certificaciones

Aplicación: Registro inmutable de logros académicos:

- Certificados de TFG en blockchain.
- Verificación automática de autenticidad.
- Portfolio académico descentralizado.
- Interoperabilidad global de credenciales.

9.4 Lecciones aprendidas

Las lecciones abarcan aspectos técnicos y personales del desarrollo, proporcionando aprendizaje valioso sobre qué estrategias funcionaron efectivamente, qué decisiones resultaron problemáticas y cómo abordar mejor proyectos similares. Esta reflexión crítica es fundamental para el crecimiento profesional y la mejora continua en desarrollo de software.

9.4.1 Decisiones arquitectónicas acertadas

Uso de React 19: A pesar de ser una versión muy reciente, las funcionalidades de concurrencia y los hooks mejorados han proporcionado beneficios significativos en rendimiento y experiencia de desarrollo.

Context API sobre Redux: Para el alcance de este proyecto, Context API ha demostrado ser suficiente y menos complejo que Redux, facilitando el desarrollo y mantenimiento, la curva de dificultad de Redux era más pronunciada.

Symfony 6.4 LTS: La elección de una versión LTS garantiza estabilidad y soporte a largo plazo (hasta 2027), crítico para un sistema académico.

Docker/DDEV: El entorno containerizado ha facilitado enormemente el desarrollo y será crucial para el despliegue en producción.

9.4.2 Mejores prácticas identificadas

Desarrollo incremental: La estrategia de 8 fases con entregas funcionales ha permitido validación temprana y ajustes continuos.

Testing desde el inicio: Implementar testing unitario desde las primeras fases ha reducido significativamente los errores y facilitado la refactorización.

Seguridad desde el diseño: Considerar seguridad desde el diseño inicial ha resultado en un sistema robusto sin necesidad de parches posteriores.

9.5 Reflexión final

Llegando al final de este proyecto, me doy cuenta de que ha sido mucho más que hacer una aplicación web. Al principio pensaba que solo iba a programar un sistema para gestionar TFGs, pero acabé aprendiendo un montón de cosas que ni sabía que existían.

Lo que más me ha sorprendido es cómo algo que empezó siendo ”un proyecto pequeño” acabó convirtiéndose en un proyecto complejo con frontend y backend separados y con sus propias tecnologías y particularidades, base de datos, autenticación, tests... Ha sido como ir descubriendo capas de complejidad que no veía al principio. Cada vez que resolvía un problema aparecían tres más, pero esa sensación de ir avanzando poco a poco ha sido súper satisfactoria.

Técnicamente, el stack de React con Symfony ha funcionado mejor de lo que esperaba. Al principio dudaba si era demasiado ambicioso para un TFG, pero creo que la elección fue acertada. Me ha obligado a entender conceptos como manejo del State de React, autenticacion y tokens JWT y arquitectura modular que seguramente me van a servir en el futuro.

Lo que más me ha costado ha sido la gestión del tiempo y la planificación. Siempre pensaba que algo iba a tardar menos de lo que realmente tardaba, especialmente la integración entre frontend y backend. También he aprendido que la documentación no es solo un trámite - escribir código que otros puedan entender (incluido yo mismo dentro de unos meses) es casi tan importante como que funcione.

Al final, creo que he conseguido hacer algo útil de verdad. No es perfecto y hay mil cosas que se podrían mejorar, pero funciona y resuelve un problema real. Eso me da bastante orgulloß después de tanto tiempo trabajando en esto.

Bibliografía

[1] **React Team**. *React - A JavaScript library for building user interfaces.* Meta, 2024. https://react.dev/

[2] **Symfony Team**. *Symfony 6.4 LTS - The Fast Track.* SensioLabs, 2024. https://symfony.com/doc

[3] **Oracle Corporation**. *MySQL 8.0 Reference Manual.* Oracle, 2024. https://dev.mysql.com/doc/

[4] **Jones, M., Bradley, J., Sakimura, N.** *JSON Web Token (JWT).* RFC 7519, Internet Engineering Task Force, 2015. https://tools.ietf.org/html/rfc7519

[5] **API Platform Team**. *API Platform 3.x - Build modern web APIs.* Les-Tilleuls.coop, 2024. https://api-platform.com/docs/

[6] **Wathan, A., Reinink, J.** *Tailwind CSS v4 Documentation.* Tailwind Labs, 2024. https://tailwindcss.com/docs

[7] **FullCalendar LLC**. *FullCalendar - JavaScript Event Calendar.* Versión 6.x, 2024. https://fullcalendar.io/docs

[8] **You, E.** *Vite - Next Generation Frontend Tooling.* 2024. https://vitejs.dev/guide/

[9] **Doctrine Team**. *Doctrine ORM Documentation.* 2024. https://www.doctrine-project.org/pro

[10] **Bergmann, S.** *PHPUnit - The PHP Testing Framework.* Versión 10.x, 2024. https://phpunit.de/documentation.html

[11] **DDEV Community**. *DDEV - Docker Development Environment.* 2024. https://ddev.readthed

[12] **Fielding, R. T.** *Architectural Styles and the Design of Network-based Software Architectures.* Doctoral dissertation, University of California, Irvine, 2000.

[13] **W3C**. *Cross-Origin Resource Sharing (CORS).* W3C Recommendation, 2014. https://www.w3.org

[14] **Hardt, D.** *The OAuth 2.0 Authorization Framework.* RFC 6749, Internet Engineering Task Force, 2012. https://tools.ietf.org/html/rfc6749

[15] **Beck, K., et al.** *Manifesto for Agile Software Development.* 2001. https://agilemanifesto.org/

[16] **Reenskaug, T.** *The original MVC reports.* Xerox PARC, 1979.

[17] **Martin, R. C.** *Clean Code: A Handbook of Agile Software Craftsmanship.* Prentice Hall, 2008.

[18] **Richardson, L., Ruby, S.** *RESTful Web APIs.* O'Reilly Media, 2013.

[19] **Fain, Y., Moiseev, A.** *Angular 2 Development with TypeScript.* Manning Publica-

tions, 2016.

[20] **Stuttard, D., Pinto, M.** *The Web Application Hacker's Handbook: Finding and Exploiting Security Flaws.* 2ª edición, Wiley, 2018.

[21] **Hernandez, M. J.** *Database Design for Mere Mortals: A Hands-On Guide to Relational Database Design.* 4ª edición, Addison-Wesley, 2019.

[22] **Bass, L., Clements, P., Kazman, R.** *Software Architecture in Practice.* 4ª edición, Addison-Wesley, 2017.

[23] **Khorikov, V.** *Unit Testing Principles, Practices, and Patterns.* Manning Publications, 2020.

[24] **Nickoloff, J., Kuenzli, S.** *Docker in Action.* 2ª edición, Manning Publications, 2019.

[25] **Gamma, E., Helm, R., Johnson, R., Vlissides, J.** *Design Patterns: Elements of Reusable Object-Oriented Software.* Addison-Wesley, 1994.

[26] **Fowler, M.** *Refactoring: Improving the Design of Existing Code.* 2ª edición, Addison-Wesley, 2018.

10. Anexo A. Manual de instalación

Este anexo proporciona una guía completa para la instalación y configuración de la Plataforma de Gestión de TFG en diferentes entornos. La información está estructurada para que tanto desarrolladores experimentados como usuarios con conocimientos técnicos básicos puedan establecer un entorno de desarrollo funcional.

La documentación abarca desde requisitos mínimos del sistema hasta procedimientos avanzados de configuración, incluyendo soluciones a problemas comunes durante el proceso. Cada sección ha sido validada mediante pruebas en diferentes entornos para asegurar precisión y completitud.

10.1 A.1. Requisitos del sistema

Antes de instalar la Plataforma de Gestión de TFG, es fundamental verificar que el entorno cumple con los requisitos técnicos necesarios para el correcto funcionamiento del sistema. Estos requisitos consideran las necesidades de rendimiento y estabilidad operacional en diferentes escenarios de uso.

La especificación distingue entre entornos de desarrollo y producción, reconociendo que cada uno tiene demandas diferentes en recursos y configuración. El cumplimiento de estos requisitos garantiza una experiencia óptima durante la instalación y operación.

10.1.1 A.1.1. Requisitos mínimos de hardware

Para desarrollo local: - **CPU**: 4 núcleos (Intel i5 o AMD Ryzen 5 equivalente). - **RAM**: 8 GB mínimo, 16 GB recomendado. - **Almacenamiento**: 50 GB de espacio libre en SSD. - **Red**: Conexión a Internet estable (100 Mbps recomendado).

Para producción: - **CPU**: 8 núcleos (Intel i7 o AMD Ryzen 7). - **RAM**: 16 GB mínimo, 32 GB recomendado. - **Almacenamiento**: 200 GB SSD para sistema + almacenamiento adicional para archivos. - **Red**: Conexión dedicada con ancho de banda adecuado.

10.1.2 A.1.2. Requisitos de software

Sistema operativo soportado: - Windows 10/11 (desarrollo). - Linux Ubuntu 20.04+ (desarrollo y producción). - macOS 12+ (desarrollo).

Software base requerido: - **Docker Desktop**: Versión 4.12+. - **Node.js**: Versión 18.x LTS. - **Git**: Versión 2.30+. - **Editor de código**: VS Code recomendado.

10.2 A.2. Instalación para desarrollo

Una vez verificados los requisitos del sistema, se procede con la instalación para desarrollo, que establece un entorno completo para modificar, probar y ejecutar la Plataforma de Gestión de TFG localmente. Este proceso está diseñado para ser reproducible y consistente entre diferentes sistemas operativos, utilizando DDEV como herramienta principal de containerización.

La instalación para desarrollo está optimizada para minimizar la configuración manual y maximizar la automatización, permitiendo que los desarrolladores comiencen a trabajar con el código rápidamente. Los contenedores Docker garantizan que el entorno de desarrollo sea idéntico al de producción, reduciendo problemas relacionados con diferencias de configuración.

10.2.1 A.2.1. Configuración inicial del proyecto

10.2.1.1 Paso 1: Clonar el repositorio

```
## Clonar el repositorio principal
git clone https://github.com/tu-usuario/plataforma-tfg.git
cd plataforma-tfg

## Verificar la estructura del proyecto
ls -la
```

Estructura esperada:

```
plataforma-tfg/
 README.md
 CLAUDE.md
 DOCUMENTACION.md
 backend.md
 package.json
 .gitignore
 docs/
 frontend/           # Aplicación React
 backend/           # API Symfony (si existe)
```

10.2.1.2 Paso 2: Configurar variables de entorno

Frontend (.env.local):

```
## Crear archivo de configuración para desarrollo
cd frontend
cp .env.example .env.local

## Editar variables según tu entorno
nano .env.local
```

```
## Contenido de frontend/.env.local
VITE_API_BASE_URL=http://localhost:8000/api
VITE_APP_NAME=Plataforma de Gestión de TFG
VITE_ENVIRONMENT=development
VITE_ENABLE_DEV_TOOLS=true
```

Backend (.env.local) (cuando esté disponible):

```
cd backend
cp .env.example .env.local
nano .env.local
```

```
## Contenido de backend/.env.local
APP_ENV=dev
APP_DEBUG=true
APP_SECRET=your-secret-key-for-development

DATABASE_URL="mysql://root:password@127.0.0.1:3306/tfg_development"
JWT_SECRET_KEY=%kernel.project_dir%/config/jwt/private.pem
JWT_PUBLIC_KEY=%kernel.project_dir%/config/jwt/public.pem
JWT_PASSPHRASE=your-jwt-passphrase

MAILER_DSN=smtp://localhost:1025
CORS_ALLOW_ORIGIN=http://localhost:5173
```

10.2.2 A.2.2. Configuración con DDEV (Recomendado)

10.2.2.1 Paso 1: Instalación de DDEV

En Windows:

```
## Usar Chocolatey
choco install ddev

## O descargar desde GitHub releases
```

```
## https://github.com/drud/ddev/releases
```

En macOS:

```
## Usar Homebrew
brew install drud/ddev/ddev
```

En Linux:

```
## Ubuntu/Debian
curl -fsSL https://apt.fury.io/drud/gpg.key | sudo gpg --dearmor -o /etc/
    apt/keyrings/ddev.gpg
echo "deb [signed-by=/etc/apt/keyrings/ddev.gpg] https://apt.fury.io/drud/
    * *" | sudo tee /etc/apt/sources.list.d/ddev.list
sudo apt update && sudo apt install ddev
```

10.2.2.2 Paso 2: Configuración inicial de DDEV

```
## Ir al directorio raíz del proyecto
cd plataforma-tfg

## Inicializar DDEV
ddev config

## Configuración interactiva:
## - Project name: plataforma-tfg
## - Docroot: public (para Symfony) o dist (para React)
## - Project type: symfony o react
```

10.2.2.3 Paso 3: Configuración específica de DDEV

Crear archivo .ddev/config.yaml:

```
name: plataforma-tfg
type: php
docroot: backend/public
php_version: "8.2"
webserver_type: nginx-fpm
router_http_port: "80"
router_https_port: "443"
xdebug_enabled: true
additional_hostnames: []
additional_fqdns: []
database:
  type: mysql
```

```
  version: "8.0"

## Servicios adicionales
services:
  redis:
    type: redis
    version: "7"
  mailpit:
    type: mailpit

## Configuración de Node.js para frontend
nodejs_version: "18"

## Comandos personalizados
hooks:
  post-start:
    - exec: "cd frontend && npm install"
    - exec: "cd backend && composer install"
```

10.2.2.4 Paso 4: Iniciar el entorno DDEV

```
## Iniciar todos los servicios
ddev start

## Verificar estado
ddev status

## Ver URLs disponibles
ddev describe
```

URLs típicas generadas: - **Aplicación principal**: https://plataforma-tfg.ddev.site - **PHPMyAdmin**: https://plataforma-tfg.ddev.site:8036 - **Mailpit**: https://plataforma-tfg.ddev.site:8025

10.2.3 A.2.3. Configuración del frontend

10.2.3.1 Paso 1: Instalación de dependencias

```
## Dentro del contenedor DDEV o localmente
cd frontend

## Instalar dependencias
```

```
npm install

## Verificar instalación
npm list --depth=0
```

10.2.3.2 Paso 2: Configuración de herramientas de desarrollo

ESLint y Prettier:

```
## Verificar configuración
npm run lint

## Corregir errores automáticamente
npm run lint:fix

## Verificar formateo
npm run format
```

Configuración de VS Code (.vscode/settings.json):

```
{
  "editor.formatOnSave": true,
  "editor.defaultFormatter": "esbenp.prettier-vscode",
  "editor.codeActionsOnSave": {
    "source.fixAll.eslint": true
  },
  "emmet.includeLanguages": {
    "javascript": "javascriptreact"
  },
  "tailwindCSS.includeLanguages": {
    "javascript": "javascript",
    "html": "html"
  }
}
```

10.2.3.3 Paso 3: Iniciar servidor de desarrollo

```
## Iniciar servidor de desarrollo
npm run dev

## El servidor estará disponible en:
## http://localhost:5173
```

10.2.4 A.2.4. Configuración del backend (Symfony)

10.2.4.1 Paso 1: Instalación de Composer y dependencias

```
## Dentro del contenedor DDEV
ddev ssh

## Ir al directorio backend
cd backend

## Instalar dependencias
composer install

## Verificar instalación
composer show
```

10.2.4.2 Paso 2: Configuración de la base de datos

```
## Crear la base de datos
ddev exec php bin/console doctrine:database:create

## Ejecutar migraciones (cuando estén disponibles)
ddev exec php bin/console doctrine:migrations:migrate

## Cargar datos de prueba (fixtures)
ddev exec php bin/console doctrine:fixtures:load --no-interaction
```

10.2.4.3 Paso 3: Generar claves JWT

```
## Generar par de claves JWT
ddev exec php bin/console lexik:jwt:generate-keypair

## Las claves se generarán en:
## config/jwt/private.pem
## config/jwt/public.pem
```

10.2.4.4 Paso 4: Configurar caché y logs

```
## Limpiar caché
ddev exec php bin/console cache:clear

```

```
## Verificar configuración
ddev exec php bin/console debug:config

## Verificar servicios
ddev exec php bin/console debug:autowiring
```

10.3 A.3. Configuración de la base de datos

La configuración de la base de datos constituye un paso crítico en la instalación, ya que determina tanto el rendimiento como la integridad de los datos del sistema. La Plataforma de Gestión de TFG utiliza MySQL 8.0 como sistema de gestión de base de datos, aprovechando sus características avanzadas de seguridad, rendimiento y escalabilidad.

Este proceso incluye la configuración inicial de la base de datos, la creación de usuarios con permisos apropiados, y la carga de datos de prueba que facilitan el desarrollo y testing. La configuración está optimizada para entornos de desarrollo y producción.

10.3.1 A.3.1. Configuración de MySQL

10.3.1.1 Opción A: Usando DDEV (Recomendado)

```
## DDEV gestiona automáticamente MySQL
## Acceso a la base de datos:
ddev mysql

## Información de conexión:
## Host: db
## Port: 3306
## Database: db
## Username: db
## Password: db
```

10.3.1.2 Opción B: MySQL local

```
## Instalar MySQL 8.0
## Ubuntu/Debian:
sudo apt update
sudo apt install mysql-server-8.0

```

```
## Configurar seguridad
sudo mysql_secure_installation

## Crear base de datos y usuario
mysql -u root -p
```

```
-- Crear base de datos
CREATE DATABASE tfg_development CHARACTER SET utf8mb4 COLLATE
    utf8mb4_unicode_ci;

-- Crear usuario específico
CREATE USER 'tfg_user'@'localhost' IDENTIFIED BY 'secure_password';

-- Otorgar permisos
GRANT ALL PRIVILEGES ON tfg_development.* TO 'tfg_user'@'localhost';
FLUSH PRIVILEGES;

-- Verificar creación
SHOW DATABASES;
SELECT User, Host FROM mysql.user WHERE User = 'tfg_user';
```

10.3.2 A.3.2. Esquema inicial de la base de datos

Ejecutar migraciones iniciales:

```
## Con DDEV
ddev exec php bin/console doctrine:migrations:migrate

## O localmente
php bin/console doctrine:migrations:migrate
```

Estructura de tablas creadas: - `users` - Usuarios del sistema con roles. - `tfgs` - Trabajos de Fin de Grado. - `tribunales` - Tribunales evaluadores. - `defensas` - Defensas programadas. - `calificaciones` - Calificaciones de defensas. - `notificaciones` - Sistema de notificaciones. - `comentarios` - Comentarios en TFGs.

10.3.3 A.3.3. Datos de prueba

```
## Cargar fixtures con datos de prueba
ddev exec php bin/console doctrine:fixtures:load --no-interaction

## Los siguientes usuarios de prueba estarán disponibles:
## estudiante@uni.es / 123456 (ROLE_ESTUDIANTE)
```

```
## profesor@uni.es / 123456 (ROLE_PROFESOR)
## presidente@uni.es / 123456 (ROLE_PRESIDENTE_TRIBUNAL)
## admin@uni.es / 123456 (ROLE_ADMIN)
```

10.4 A.4. Configuración de desarrollo avanzada

10.4.1 A.4.1. Debugging y logs

10.4.1.1 Configuración de Xdebug (PHP)

En .ddev/config.yaml:

```
xdebug_enabled: true
```

Configuración en VS Code (launch.json):

```
{
  "version": "0.2.0",
  "configurations": [
    {
      "name": "Listen for Xdebug",
      "type": "php",
      "request": "launch",
      "port": 9003,
      "pathMappings": {
        "/var/www/html": "${workspaceFolder}/backend"
      }
    }
  ]
}
```

10.4.1.2 Configuración de logs

Frontend (React Developer Tools):

```
## Instalar extensión React Developer Tools en el navegador
## Chrome: https://chrome.google.com/webstore/detail/
   fmkadmapgofadopljbjfkapdkoienihi
## Firefox: https://addons.mozilla.org/en-US/firefox/addon/react-devtools/
```

Backend (Symfony Profiler):

```
## config/packages/dev/web_profiler.yaml
```

```
web_profiler:
    toolbar: true
    intercept_redirects: false
```

10.4.2 A.4.2. Testing environment

10.4.2.1 Configuración para testing del frontend

```
cd frontend

## Instalar dependencias de testing
npm install --save-dev @testing-library/react @testing-library/jest-dom
    vitest

## Ejecutar tests
npm run test

## Ejecutar con coverage
npm run test:coverage
```

10.4.2.2 Configuración para testing del backend

```
## Crear base de datos de testing
ddev exec php bin/console doctrine:database:create --env=test

## Ejecutar migraciones en testing
ddev exec php bin/console doctrine:migrations:migrate --env=test --no-
    interaction

## Ejecutar tests
ddev exec php bin/phpunit

## Con coverage
ddev exec php bin/phpunit --coverage-html coverage/
```

10.4.3 A.4.3. Herramientas de desarrollo adicionales

10.4.3.1 Git hooks para calidad de código

```
## Instalar husky para git hooks
cd frontend
```

```
npm install --save-dev husky lint-staged

## Configurar pre-commit hook
npx husky add .husky/pre-commit "npm run lint && npm run test"
```

10.4.3.2 Extensiones recomendadas de VS Code

```
{
  "recommendations": [
    "esbenp.prettier-vscode",
    "ms-vscode.vscode-eslint",
    "bradlc.vscode-tailwindcss",
    "ms-vscode.vscode-typescript-next",
    "bmewburn.vscode-intelephense-client",
    "ms-vscode.vscode-docker",
    "ms-vscode.vscode-json"
  ]
}
```

10.5 A.5. Solución de problemas comunes

Durante la instalación y configuración de la Plataforma de Gestión de TFG, pueden surgir diversos problemas técnicos que requieren atención específica. Esta sección compila las dificultades más frecuentemente reportadas junto con sus soluciones correspondientes, facilitando una resolución rápida y eficiente.

La documentación de problemas comunes se basa en experiencias reales de instalación en diferentes entornos y configuraciones, proporcionando soluciones probadas que han demostrado efectividad. Cada problema incluye no solo la solución inmediata, sino también información contextual que ayuda a comprender las causas subyacentes y prevenir futuras ocurrencias.

10.5.1 A.5.1. Problemas de DDEV

Error: "Port already in use"

```
## Verificar puertos en uso
ddev stop --all

## Cambiar puerto en configuración
```

```
ddev config --router-http-port=8080 --router-https-port=8443

## Reiniciar
ddev start
```

Error: "Database connection failed"

```
## Verificar estado de servicios
ddev status

## Reiniciar base de datos
ddev restart

## Verificar logs
ddev logs db
```

10.5.2 A.5.2. Problemas del frontend

Error: "Module not found"

```
## Limpiar caché de npm
npm cache clean --force

## Eliminar node_modules y reinstalar
rm -rf node_modules package-lock.json
npm install
```

Error: "Port 5173 is already in use"

```
## Cambiar puerto en vite.config.js
export default defineConfig({
  server: {
    port: 3000
  }
})
```

10.5.3 A.5.3. Problemas del backend

Error: "JWT keys not found"

```
## Generar nuevas claves JWT
ddev exec php bin/console lexik:jwt:generate-keypair --skip-if-exists

## Verificar permisos
ddev exec chmod 644 config/jwt/*.pem
```

Error: "Unable to write in cache directory"

```
## Corregir permisos de caché
ddev exec chmod -R 777 var/

## Limpiar caché
ddev exec php bin/console cache:clear --no-warmup
```

10.5.4 A.5.4. Problemas de rendimiento

Frontend lento en desarrollo:

```
// vite.config.js - Optimizaciones para desarrollo
export default defineConfig({
  server: {
    hmr: {
      overlay: false // Disable error overlay for faster reloads
    }
  },
  optimizeDeps: {
    include: ['react', 'react-dom'] // Pre-bundle heavy dependencies
  }
})
```

Backend lento:

```
## config/packages/dev/doctrine.yaml
doctrine:
    dbal:
        profiling_collect_backtrace: false
    orm:
        auto_generate_proxy_classes: true
```

10.6 A.6. Comandos útiles de desarrollo

10.6.1 A.6.1. Comandos DDEV frecuentes

```
## Gestión de servicios
ddev start              # Iniciar proyecto
ddev stop               # Parar proyecto
ddev restart            # Reiniciar proyecto
ddev poweroff           # Parar todos los proyectos DDEV

```

```
## Información del proyecto
ddev describe           # Mostrar URLs y detalles
ddev status             # Estado de servicios
ddev list               # Listar proyectos DDEV

## Acceso a servicios
ddev ssh                # SSH al contenedor web
ddev mysql              # Acceso a MySQL CLI
ddev logs               # Ver logs generales
ddev logs web           # Ver logs del servidor web

## Utilidades
ddev import-db --src=dump.sql  # Importar base de datos
ddev export-db > dump.sql      # Exportar base de datos
ddev snapshot               # Crear snapshot del proyecto
```

10.6.2 A.6.2. Comandos del frontend

```
## Desarrollo
npm run dev             # Servidor de desarrollo
npm run build           # Build de producción
npm run preview         # Preview del build

## Calidad de código
npm run lint            # Ejecutar ESLint
npm run lint:fix        # Corregir errores de ESLint
npm run format          # Formatear con Prettier

## Testing
npm run test            # Ejecutar tests
npm run test:watch      # Tests en modo watch
npm run test:coverage   # Tests con coverage
```

10.6.3 A.6.3. Comandos del backend

```
## Doctrine
php bin/console doctrine:database:create
php bin/console doctrine:migrations:migrate
php bin/console doctrine:fixtures:load

## Caché
php bin/console cache:clear
```

```
php bin/console cache:warmup

## Debugging
php bin/console debug:config
php bin/console debug:container
php bin/console debug:autowiring

## JWT
php bin/console lexik:jwt:generate-keypair

## Testing
php bin/phpunit
php bin/phpunit --coverage-html coverage/
```

10.7 A.7. Verificación de la instalación

Para garantizar que la Plataforma de Gestión de TFG ha sido instalada correctamente y está operativa en todos sus componentes, es esencial realizar una verificación sistemática de la instalación. Este proceso incluye pruebas de conectividad, funcionalidad básica y rendimiento del sistema, asegurando que todos los servicios funcionen según las especificaciones.

La verificación no solo confirma que los componentes técnicos están operativos, sino que también valida que la integración entre frontend, backend y base de datos funciona correctamente. Este paso es crítico antes de comenzar el desarrollo activo o el despliegue en producción.

10.7.1 A.7.1. Checklist de verificación

Entorno DDEV: - [ ] DDEV instalado y funcionando. - [ ] Proyecto iniciado sin errores. - [ ] URLs accesibles (web, PHPMyAdmin, Mailpit). - [ ] Base de datos creada y accesible.

Frontend: - [ ] Dependencias instaladas correctamente. - [ ] Servidor de desarrollo inicia sin errores. - [ ] Linting y formateo funcionando. - [ ] Tests básicos pasando.

Backend: - [ ] Composer dependencies instaladas. - [ ] Migraciones ejecutadas correctamente. - [ ] Claves JWT generadas. - [ ] Fixtures cargados. - [ ] API endpoints respondiendo.

Integración: - [ ] Frontend puede conectar con backend. - [ ] Autenticación JWT

funcionando. - [ ] CORS configurado correctamente. - [ ] Logs accesibles y configurados.

10.7.2 A.7.2. Script de verificación automatizada

```
#!/bin/bash
## scripts/verify-installation.sh

echo "  Verificando instalación de la Plataforma de Gestión de TFG..."

## Verificar DDEV
if ! command -v ddev &> /dev/null; then
    echo "  DDEV no está instalado"
    exit 1
fi

## Verificar estado del proyecto
if ! ddev status | grep -q "running"; then
    echo "  El proyecto DDEV no está ejecutándose"
    exit 1
fi

## Verificar frontend
if [ -d "frontend/node_modules" ]; then
    echo "  Dependencias del frontend instaladas"
else
    echo "  Falta instalar dependencias del frontend"
fi

## Verificar backend
if [ -d "backend/vendor" ]; then
    echo "  Dependencias del backend instaladas"
else
    echo "  Falta instalar dependencias del backend"
fi

## Verificar base de datos
if ddev mysql -e "SELECT 1" &> /dev/null; then
    echo "  Base de datos accesible"
else
    echo "  Problema con la base de datos"
fi

## Test de conectividad
if curl -f -s https://plataforma-tfg.ddev.site > /dev/null; then
```

```
41      echo "  Aplicación web accesible"
42  else
43      echo "  La aplicación web no responde"
44  fi
45
46  echo "  Verificación completada"
```