
This is the **published version** of the bachelor thesis:

Lozada Pérez, Verónica. *Aplicación web para gestionar la creación y la asignación de propuestas asociadas al Trabajo de Final de Grado*. Treball de Final de Grau (Universitat Autònoma de Barcelona), 2026 (Enginyeria Informàtica)

This version is available at <https://ddd.uab.cat/record/334436>

under the terms of the  license.

Aplicación web para gestionar la creación y la asignación de propuestas asociadas al Trabajo de Final de Grado

Verónica Lozada Pérez

Resumen — La gestión del proceso de creación y asignación de TFGs en la Facultad de Ingeniería Informática de la UAB se realiza de forma descentralizada, con documentación dispersa y sin una fuente única de verdad. Esto genera exceso de correos electrónicos, información desactualizada y dificultad para conocer el estado real del proceso en cada momento. El objetivo de este TFG consistía en desarrollar una aplicación web que centraliza todo el ciclo de vida de las propuestas de TFG, desde su creación y selección hasta la asignación final de tutor y alumno. La plataforma gestiona tres perfiles de usuario: Alumno, Profesor y Coordinador, con funcionalidades específicas para cada uno. EL software desarrollado cuenta con un frontend en React, un backend con NodeJS y una base de datos que utiliza PrismaORM y MySQL. El resultado es una aplicación completamente funcional que permite replicar los flujos reales del proceso de principio a fin, desde el registro de usuarios hasta la exportación de las asignaciones finales. El proyecto cumple todos los objetivos planteados y establece una base desplegable que abre la vía a ser utilizada en el próximo curso académico.

Paraules clau — API REST, desarrollo web, escola d'enginyeria, Express 5, ingeniería del software, MySQL, NodeJS, Prisma ORM, React18, TFG

Abstract — The process of creating and assigning final degree projects (TFGs) at the UAB's School of Engineering is managed in a decentralized manner, with scattered documentation and no single source of truth. This leads to an excessive volume of emails, outdated information, and difficulty in determining the actual status of the process at any given time. The objective of this TFG was to develop a web application that centralizes the entire lifecycle of TFG proposals, from their creation and selection to the final assignment of a advisor and student. The platform manages three user profiles: Student, Professor, and Coordinator, each with specific functionalities. The developed software features a React frontend, a Node.js backend, and a database that uses PrismaORM and MySQL. The result is a fully functional application that replicates the actual workflows of the process from start to finish, from user registration to the export of final assignments. The project meets all the stated objectives and establishes a deployable foundation that paves the way for its use in the next academic year.

Index Terms — engineering school, Express 5, Final Project, MySQL, Node.js, Prisma ORM, React 18, REST API, software engineering, web development,



1 INTRODUCCIÓN

El Trabajo de Final de Grado es un proyecto de carácter obligatorio que deben realizar todos los estudiantes durante el último año de su carrera universitaria. Es un trabajo de investigación donde han de reflejarse las competencias y conocimientos adquiridos durante la titulación.

Para su realización, los estudiantes deben escoger un tema a abordar, ya sea idea propia o propuesta desde el profesorado. Sin embargo, el proceso actual para proponer y elegir Trabajos de Final de Grado en la Facultad de Ingeniería Informática de la Universidad Autónoma de Barcelona es de difícil gestión para aquellos involucrados.

Todo lo que respecta a la creación de propuestas y la asignación de estas, a alumnos y tutores por iguales, depende en gran medida de documentación interna (excels, pdfs...), provocando exceso de emails, falta de comunicación, información desfasada y dificultad para valorar el estado actual del sistema.

La investigación que se lleva a cabo en este TFG es con el objetivo de proporcionar una página web que gestione el proceso de creación y asignación de Trabajos de Final de Grado de forma centralizada, intuitiva y práctica para profesores, alumnos y coordinadores de l'Escola d'Enginyeria.

2 OBJETIVOS DEL PROYECTO

El objetivo general de este proyecto es el desarrollo de una plataforma web para gestionar los procesos relacionados con la selección y asignación de TFGs. Como objetivos

- E-mail de contacte: veronica.lozada@autonoma.cat
- Menció realitzada: Enginyeria del Software
- Treball tutoritzat per: Jorge Bernal del Nozal (departament de Ciències de la Computació)
- Curs 2025/26

secundarios se han desarrollado:

1. **Diseño de una interfaz web profesional e intuitiva** acorde con el estilo de una plataforma de la propia universidad y para que los múltiples usuarios encuentren la información que necesitan de manera rápida y cómoda.
2. **Diseño de una base de datos relacional con el objetivo de centralizar la información del proceso.** En esta base de datos se almacenan los diferentes usuarios, las propuestas y las solicitudes de evaluación que garantiza un procesamiento estructurado y unificado de los datos.
3. **Definición e implementación de distintos roles de usuario** para alumnos, profesores y coordinadores donde existen las funcionalidades comunes y las específicas a cada uno. Estas últimas acorde a la responsabilidad que cumplen dentro del sistema.
4. **Generación de una página resumen del estado del sistema** que permite al coordinador un acceso rápido a los datos más relevantes del sistema, cómo las solicitudes pendientes, los trabajos sin tutores, los profesores sin menciones asignadas, los alumnos sin ninguna solicitud.
5. **Readaptación del proceso de selección de TFGs** para que los alumnos añadan de forma más sencilla los trabajos que quieran para su solicitud, que puedan modificar la prioridad al escogerlos antes de grabar la solicitud final.

3 ESTADO DEL ARTE

La funcionalidad principal de la plataforma a desarrollar es ser capaz de gestionar tanto el flujo de propuestas entre estudiantes, profesores y coordinadores, como la asignación de estas. Que toda la información esté centralizada y unificada y que se pueda hacer un seguimiento de cada tarea y del estado del sistema.

Actualmente, existen plataformas de gestión de incidencias, como pueden ser Jira Service Management [1]. En la Figura 1 del anexo A1 se encuentra un dashboard de Jira. El flujo de Jira para la creación de tickets sería, en nuestro contexto, lo equivalente a la generación de solicitudes. También se podría realizar un seguimiento de estados y ver las personas asignadas a cada TFG.

En el caso de ZenDesk [2], plataforma de gestión de incidencias, el flujo de procesamiento se asemeja más al de atención al cliente y mientras que también hay un procesamiento de solicitudes, se aleja del formato que deseamos para nuestro proyecto concreto.

En ambos casos se observa que son herramientas generales adaptadas a procesar proyectos o gestiones concretas, y carecen de características que se ajusten a las necesidades de la propia facultad y que puedan incorporar los procesos de la UAB para los trabajos de finales de grado. Se

necesitan establecer vistas y funcionalidades concretas para diferentes usuarios, ya que no todos tienen el mismo rol dentro del proceso de asignación de propuestas.

Desde el punto de vista estrictamente universitario, hay múltiples herramientas desarrolladas desde distintas universidades para la gestión de TFGs. La Universidad Rey Juan Carlos de Madrid cuenta con un proyecto para la Gestión Integral Digital de los Trabajos de Fin de Titulación [3]. Esta plataforma ha permitido automatizar y mejorar todo el ciclo de vida de los TFGs y TFM. La Universidad de Alcalá cuenta con GestTFx, Figura 2 del anexo A1, su propia plataforma de gestión también del ciclo de vida completo de estos trabajos.

En la Universidad Autónoma de Madrid, concretamente en la Escuela Politécnica Superior, se desarrolló la aplicación web "TFGTool" [4], que también gestiona que los alumnos y profesores compartan y automaticen tareas. Que se visualicen los trabajos accesibles y que se pueda visualizar los estados de desarrollo, además de gestión de la defensa, la nota y el cierre de actas.

En la Universidad de Granada, la Facultad de Ciencias también desarrolló su propio espacio de gestión de la asignatura de Trabajo de Fin de Grado [5]. La pantalla principal permite la visualización de las propuestas del Grado con múltiples posibilidades de filtrado e incluso de exportar el listado en Excel y PDF. (Figura 3 del anexo A1)

Además, también cuentan con el apartado correspondiente a la solicitud de plazas.



Figura 4. Solicitud de plazas de la plataforma de la Universidad de Granada.

Mientras que la Universidad Autónoma de Barcelona también cuenta con un espacio de seguimiento de TFGs [6], carece de un medio que gestione el proceso previo, que facilite la generación de propuestas y el poder escoger las mismas. Los proyectos anteriormente mencionados son fuentes de información relevantes a considerar durante el proceso de desarrollo.

Sin embargo, la propuesta de valor de este proyecto es la de dotar a la Facultad de Ingeniería Informática de la UAB de un portal capaz de hacer la gestión de creación, solicitud y asignación de TFGs de manera orgánica y cómoda para los usuarios y que pueda tener funcionalidades fácilmente compatibles con los sistemas existentes. El aspecto diferenciador radica en poder hacer uso de las

herramientas existentes, aspecto que no cubren los ejemplos mencionados, y ser capaz de simplificar el procedimiento de creación y asignación mientras se sigue haciendo uso del flujo actual de trabajo.

4 REQUISITOS DEL SISTEMA

Para definir con claridad las funcionalidades de la plataforma, se han identificado los requisitos funcionales (Figura 5 anexo A2) y no funcionales del sistema, organizados por perfiles de usuario. Esta documentación crea la base del proyecto y define de manera clara y sin ambigüedad todo lo que se espera del sistema.

4.1 Requisitos de Hardware

En cuanto a los requisitos de hardware, se han establecido como especificaciones actuales las que se utilizaron durante el desarrollo del TFG, con la consideración de que existe un mínimo con el que el proyecto también podría desarrollarse:

Hardware	Actual	Mínimo
Procesador	Intel(R) Core(TM) i7	Intel(R) Core(TM) i5
RAM	16 GB	8 GB
Almacenamiento	11 GB	8 GB
Sistema Operativo	Ubuntu 24.04	Windows 11

4.2 Stack tecnológico

Para el desarrollo del software se ha seleccionado el siguiente stack tecnológico, detallando la herramienta principal, la alternativa investigada y la justificación de la elección:

Capa	Tecnología	Justificación
Frontend	React19 [9] + Vite [10] + React Router 7 [11]	Familiaridad con el framework. Permite construir vistas modulares diferenciadas por rol sin duplicar lógica. La internacionalización se implementa con react-i18next.
	Alternativa	
	Vue [12] + Next.js [13]	
Backend	Node.js [14] + Express 5 [15]	Arquitectura REST con rutas y controladores por recurso. Express 5 incluye manejo de errores asíncronos.
	Alternativa	
	Deno[16] + NestJS[17]	
ORM - BD	Prisma [18] + MySQL [19]	Prisma facilita la gestión del esquema y las migraciones desde código, con tipado fuerte que reduce errores en desarrollo. MySQL encaja bien con la naturaleza relacional del modelo, donde la integridad referencial es crítica.
	Alternativa	
	TypeORM [20] + PostgreSQL [21]	

Auth	JWT [22]	Los tokens son autocontenidos y permiten verificar la identidad y el rol del usuario en cada petición sin consultar la base de datos.
	Alternativa	
	Sesiones en el servidor	
IDE – Control de versiones	VSCode [23] + GitHub [24]	Herramientas ampliamente adoptadas con soporte nativo de extensiones y flujo de trabajo Git.
	Alternativa	
	WebStorm [25] + Bitbucket [26]	

5 RIESGOS

Riesgo	Explicación	Inciden- cia	Proba- bilidad	Contingencia
No hay datos reales	No se deja usar datos reales de estudiantes	Baja	Alta	Generar estudiantes virtuales
No se cumple con el ritmo de desarrollo estipulado	Se infravalora el peso de ciertas tareas que tardan más	Alta	Media	Dejar margen y reevaluar las tareas de cada sprint en las reuniones
Alto número de funcionalidades sugeridas	A medida que avanza el desarrollo pueden surgir nuevas funcionalidades que los usuarios finales pidan	Media	Media	Priorizar primero el set de funcionalidades básicas y fundamentales
Problemas para montar un backend sólido	La poca experiencia con las tecnologías utilizadas	Media	Baja	Priorizar un desarrollo robusto y simplificar tecnologías en caso de que estancuen el desarrollo
Implementar incorrectamente el sistema de permisos	No todos los usuarios del sistema tienen permisos para todas las tareas	Alta	Media	Priorizar los roles y los permisos de cada uno a la hora de crear la base de datos
No llegar a uno de los entregables	Problemas de solapamientos con exámenes	Medio	Media	Permitir la entrega de un entregable dentro de la misma fase, aunque se haya de mover de sprint

6 METODOLOGIA

Para el desarrollo de la plataforma nos hemos basado en la metodología Scrum [7]. Al estar realizando una plataforma web, es fundamental escoger un marco de trabajo ágil que permita la mejora continua e iterar sobre un problema. Que permita integrar funcionalidades nuevas y reformular objetivos durante el proceso sin sacrificar la integridad del desarrollo.

Para ello se realizó un sprint de dos semanas, siguiendo el principio de tener ciclos donde realizar suficientes tareas para ser revisado en la siguiente sesión. La idea fue ir obteniendo feedback, haciendo correcciones e iterando sobre las tareas. A fecha de este informe se han completado siete sprints y se han mantenido las respectivas reuniones de seguimiento con el tutor, distribuidas a lo largo de las fases de planificación, diseño e implementación.

7 PLANIFICACIÓN

En cuanto a la planificación de este proyecto siguió la metodología Scrum donde se iteró sobre tareas en los sprints y con las reuniones se validó el cumplimiento de estas. En general, el proyecto se estructuró en cinco fases principales:

- **Fase de Planificación (febrero-marzo):** Definición del alcance, investigación del estado del arte y concreción de objetivos.
- **Fase de Diseño (marzo):** Definición de perfiles de usuario, identificación de casos de uso, elaboración del diagrama de clases y del modelo entidad-relación, y selección del stack tecnológico. Ceró con una primera implementación de la base de datos y vistas mínimas en frontend.
- **Fase de Implementación (marzo-abril-mayo):** Desarrollo iterativo en sprints de dos semanas, abordando autenticación y control de acceso por rol, el CRUD de propuestas, la validación por el coordinador, la tutorización y selección del alumno, la asignación final y el cierre automático, y el dashboard del coordinador.
- **Fase de Pruebas (mayo-junio):** Verificación de extremo a extremo de los flujos completos para los tres roles, comprobación de los casos límite y corrección de defectos.
- **Fase de Documentación (junio):** Redacción de la memoria final, preparación de la presentación de defensa y elaboración del dossier completo.

7.1 Ajustes de la planificación

Los cambios más relevantes respecto a la planificación inicial fueron:

1. El modelo entidad-relación y el esquema SQL

se adelantaron al primer sprint para derivarlos directamente del diagrama de clases validado.

2. Se descartó un maquetado extenso en Figma por la simplicidad de las vistas.
3. Durante la implementación emergieron reglas de negocio no previstas inicialmente que fueron incorporadas al diseño y al código.

El detalle de la planificación prevista frente a la ejecutada se presenta en el diagrama de Gantt en los apéndices (Figura 6 anexo A3).

8 ANÁLISIS Y DISEÑO

8.1 Perfiles, Casos de Uso y Reglas de Negocio

El sistema cuenta con tres perfiles diferenciados: Alumno, Profesor y Coordinador.

Todos parten de las funcionalidades comunes del Usuario. Luego el perfil Coordinador hereda todas las funcionalidades del Profesor y añade las tareas de gestión del sistema, mientras que el perfil Alumno tiene las funciones básicas y un conjunto de operaciones sobre las propuestas (selección priorizada, creación y seguimiento). Esta herencia se modela a nivel de aplicación: en el backend se aplica mediante un middleware que admite listas de roles autorizados por endpoint.

Las reglas de negocio principales del sistema siguen las ya establecidas por la Escola d'Enginyeria para el TFG de Enginyeria Informàtica:

1. Un alumno puede seleccionar como máximo 6 propuestas, ordenadas por prioridad.
2. Solo puede crear una propuesta propia; si tiene una propuesta propia activa (ya sea aprobada, rechazada o con solicitud de cambios), no puede seleccionar ni crear otras.
3. Las selecciones son provisionales hasta que el alumno ejecuta la operación "Grabar solicitud", a partir de la cual la lista queda bloqueada.
4. El sistema filtra automáticamente por mención, mostrando a cada usuario únicamente las propuestas compatibles con su área de especialización.

8.2 Modelo de datos

El modelo de datos de la plataforma se estructura en torno a cinco entidades principales: **User**, **Mention**, **Proposal**, **TfgSelection** y **ProfessorMention**.

La entidad **User** aplica una estrategia de single-table inheritance, es decir, alumnos, profesores y coordinadores comparten la misma tabla y se diferencian por un campo de rol. Los campos que solo son específicos de un alumno se dejan como nulos en los demás roles.

La entidad **Mention** es una tabla sencilla con el nombre

de cada mención y su ID.

La entidad **Proposal** es aquella que define una propuesta y reúne todos campos necesarios para su definición. Contiene los datos académicos y administrativos de un TFG, incluyendo el estado en el que se encuentra y las dos tablas intermedias (**TfgSelection** y **ProfessorMention**) que resuelven las relaciones N:N entre alumnos y propuestas y entre profesores y menciones, respectivamente.

La entidad **TfgSelection**, además de resolver la relación, guarda con la prioridad y la motivación de cada propuesta escogida. [Figuras 7 anexo A3]

8.3 Diagramas UML

Se han definido diagramas de casos de uso que muestran a grandes rasgos las funcionalidades de cada. [Figuras 8 – 11 anexo A4]

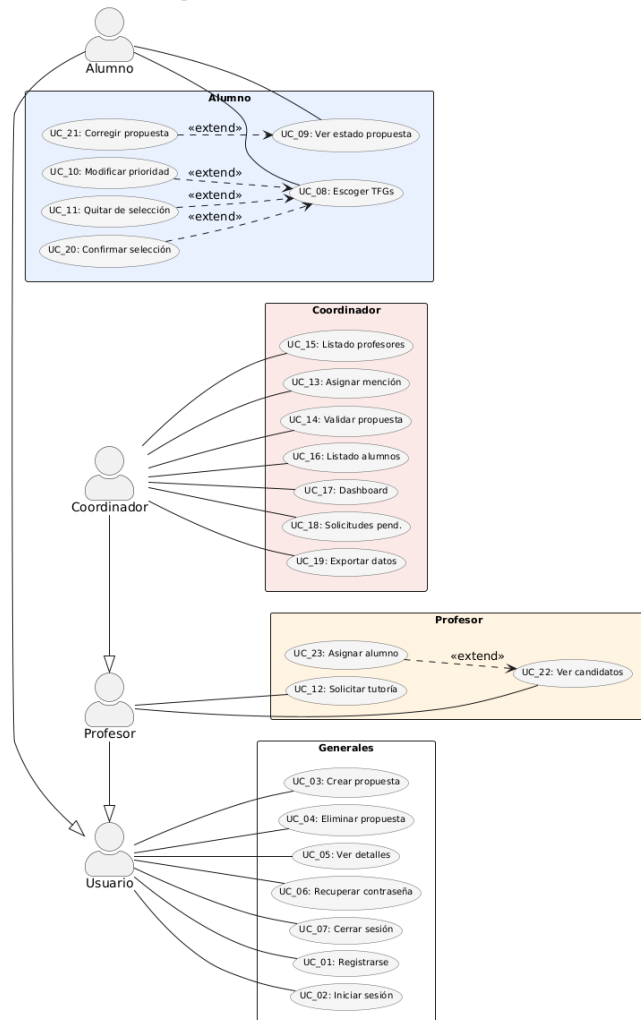


Figura 12. Diagrama de casos de uso del sistema simplificado.

Asimismo, se ha elaborado un diagrama de clases UML completo que refleja todas las entidades del sistema y sus relaciones, y el esquema relacional final de la base de datos.

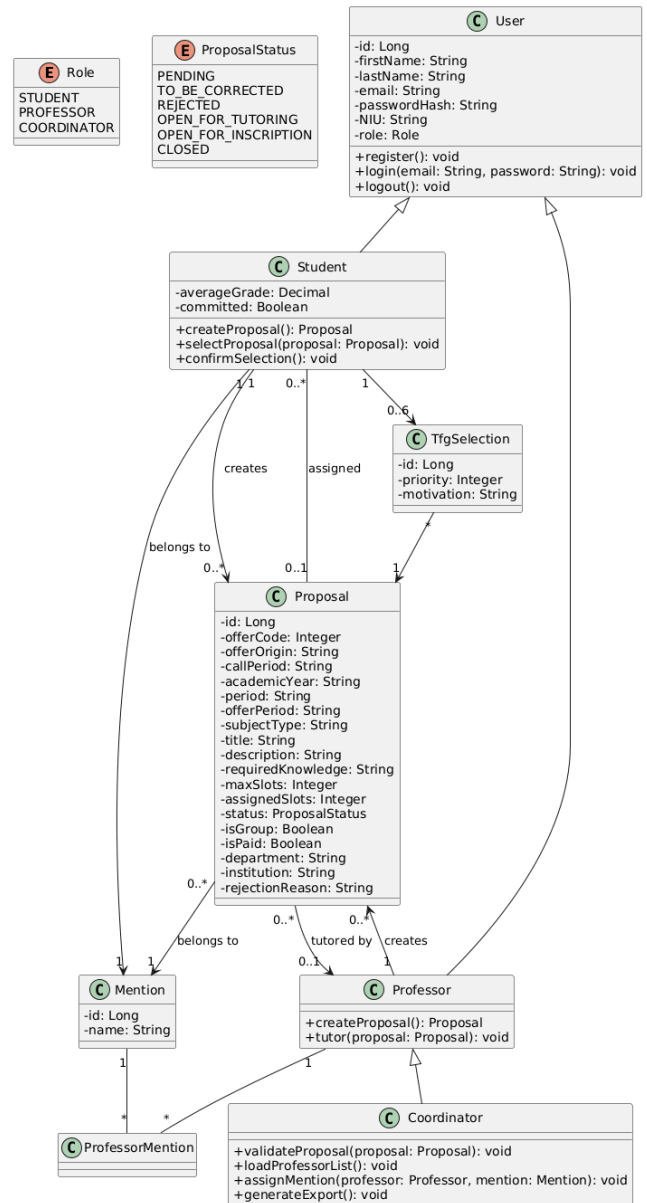


Figura 13. Diagrama de clases del sistema.

9 DESARROLLO

El proyecto está compuesto principalmente por tres componentes: Frontend, Backend y Base de Datos.

El Frontend se encarga de realizar las peticiones al Backend, mientras que el Backend procesa estas peticiones y recupera los datos desde la base de datos, que en este caso es MySQL. Es un repositorio monolítico donde Frontend y Backend conviven, pero en diferentes directorios. Se consideró la organización más coherente y manejable teniendo en cuenta el alcance del proyecto.

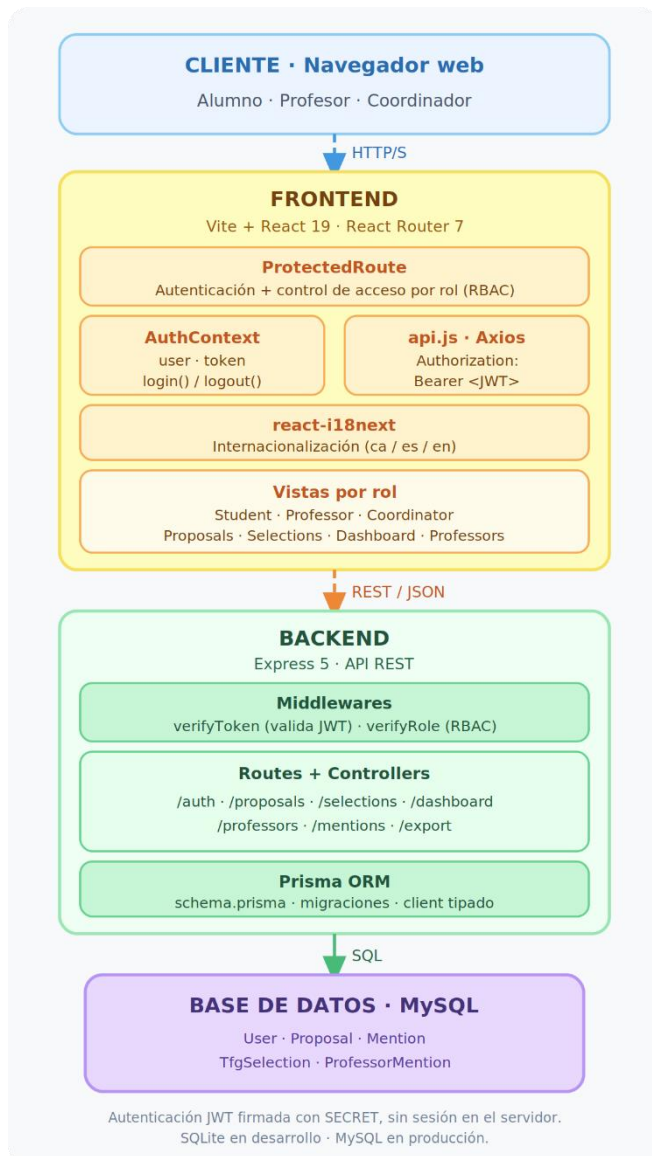


Figura 14. Esquema de la arquitectura del sistema. (Realizado con asistencia de Claude.ai)

9.1 Frontend

El frontend del proyecto se ha implementado utilizando React 19 junto con Vite como herramienta de construcción, además de usar React Router 7 para la navegación entre vistas.

Las vistas se agrupan por rol y se conectan a la API mediante una instancia central de Axios que añade automáticamente en el header el token de autorización.

El AuthContext mantiene el estado global de sesión y el componente ProtectedRoute envuelve las rutas que requieren autenticación o rol específico. La interfaz está traducida al catalán mediante react-i18next, con la estructura preparada para añadir más idiomas.

9.1.1 Vistas por rol

Cada perfil de usuario dispone de sus propias vistas. Los alumnos acceden al listado de propuestas disponibles filtradas por su mención, a su lista de selecciones

priorizadas o al estado de su propia propuesta creada. Los profesores gestionan sus propuestas creadas y la asignación de alumnos candidatos a las mismas. Los coordinadores acceden a un panel de control (dashboard), a los TFGs pendientes por validar, a un listado de alumnos y profesores, además de a la gestión de estos últimos y de sus menciones.

9.1.2 Gestion del estado y la autenticación

El AuthContext y el ProtectedRoute mencionados anteriormente, validan el acceso por rol antes de renderizar cada vista, redirigiendo al usuario en caso de acceso no autorizado. Esta separación garantiza que ningún usuario pueda acceder a vistas o funcionalidades de otro perfil, sin necesidad de duplicar comprobaciones de seguridad en cada componente.

No obstante, las validaciones del frontend tienen un propósito principalmente de experiencia de usuario: la seguridad real reside en el backend, donde cada endpoint verifica de forma independiente la identidad y el rol del usuario antes de procesar cualquier petición.

9.2 Backend

El backend del sistema se ha implementado con Node.js, empleando el framework Express 5.

Para englobar todas las peticiones posibles entre el cliente y el servidor, se ha optado por una arquitectura REST API. Este enfoque ha permitido definir un conjunto de endpoints claros mediante métodos estándar de HTTP (GET, POST, PUT, PATCH, DELETE), que proporcionan acceso a los datos y recursos del sistema.

9.2.1 Estructura general

El backend se estructura en tres capas:

- **Rutas (/routes):** Aquí están los archivos que conectan los métodos HTTP con los controladores correspondientes. Cada recurso tiene su propio archivo de rutas donde se aplican los middlewares de seguridad.
- **Controladores (/controllers):** Son los que contienen toda la lógica de negocio. Gestionan las peticiones entrantes, ejecutan las operaciones sobre Prisma y devuelven las respuestas al cliente.
- **Middlewares (/middlewares):** Aquí se encuentran las funciones de verifyToken que valida el JWT en cada petición protegida, y verifyRole que comprueba que el rol del usuario autenticado esté autorizado a acceder al endpoint concreto.

9.2.2 Listado de recursos

La API REST del sistema expone múltiples endpoints distribuidos en cinco recursos principales.

El recurso de autenticación cubre el registro y el inicio

de sesión. El de propuestas agrupa todas las operaciones sobre los TFGs: creación, consulta, edición, eliminación y las acciones específicas de validación por el coordinador, solicitud de tutorización por el profesor y asignación definitiva del alumno. El recurso de selecciones gestiona la lista provisional del alumno, incluyendo la adición y eliminación de propuestas, la reordenación por prioridad y la operación de grabación atómica. El recurso de dashboard expone las métricas clave del sistema para el coordinador. Por último, el recurso de profesores y menciones permite consultar el listado de profesores y gestionar la asignación de menciones y el recurso de exportación se dedica a la generación del fichero final.

9.2.3 Decisiones de implementación relevantes

Aquí se especifican las decisiones...

- La reordenación por arrastrar-y-soltar recibe el orden completo y reasigna prioridades preservando la limitación única de (studentId, priority). La eliminación de una propuesta reordena automáticamente las restantes.
- La confirmación definitiva con "Grabar" activa un único campo committed a nivel de alumno, lo que convierte el bloqueo de la lista de propuestas en una operación atómica y simplifica las comprobaciones de acceso en los endpoints de manipulación de selecciones.
- El endpoint correspondiente a la validación del coordinador admite tres acciones diferenciadas: aceptar, solicitar corrección y rechazar. El motivo es obligatorio en las dos últimas, garantizando que el alumno siempre reciba información accionable.
- En la asignación de un alumno se realizan tres operaciones en la misma transacción: se registra la asignación, se eliminan las demás selecciones activas del alumno en otras propuestas, y se incrementa el contador de plazas ocupadas. Si se alcanzan las plazas máximas, la propuesta transita inmediatamente a CLOSED sin requerir una acción adicional.

9.3 Base de datos

Se ha empleado una base de datos relacional, MySQL, gestionada mediante Prisma ORM.

La elección de MySQL responde a la naturaleza fuertemente relacional del modelo de datos: las entidades del sistema están interrelacionadas entre sí y la integridad referencial es crítica para garantizar la consistencia del flujo de TFGs. Prisma ha facilitado significativamente el proceso de desarrollo gracias a su sistema de migraciones, que

permite modificar la estructura de la base de datos directamente desde el esquema sin intervención manual, propagando cualquier cambio de forma controlada y trazable.

Además, la funcionalidad de seed ha permitido poblar la base de datos con datos de prueba de forma reproducible, lo que ha agilizado tanto el desarrollo iterativo como la validación de los flujos end-to-end.

10 PRUEBAS Y VALIDACIÓN

10.1 Validación de endpoints

Durante el desarrollo se ha llevado a cabo una validación manual de los endpoints de la API REST al cierre de cada sprint, verificando que las peticiones HTTP retornan los datos esperados y que las reglas de negocio se aplican correctamente.

También se ha desarrollado una suite de tests automatizados con Jest, ejecutados sobre una base de datos de prueba aislada de la de desarrollo, que cubre los flujos más críticos del sistema.

Para cada uno de estos flujos se ha comprobado tanto el caso de éxito como los principales casos de error: acceso no autorizado, rol no permitido, datos inválidos y violaciones de las restricciones de negocio, además de edge cases propios de cada flujo.

En concreto se ha validado: el flujo completo de creación, validación, rechazo y corrección de propuestas, comprobando que los estados transicionan correctamente y que el motivo es obligatorio en rechazo y solicitud de corrección; el acceso cruzado entre roles; el reordenamiento de selecciones y la preservación de la prioridad única de cada una; y el bloqueo de la lista de selecciones una vez ejecutada la operación de grabación definitiva. La asignación final del alumno y el cierre automático de la propuesta al alcanzar el número máximo de plazas se han validado también con tests de integración.

10.2 Pruebas de flujo end-to-end

Se ha realizado una verificación de extremo a extremo de los flujos principales del sistema, usando Playwright para automatizar el navegador y simular la interacción real de un usuario con la interfaz, para los tres roles. Se ha comprobado el inicio de sesión y el acceso diferenciado según el rol, la consulta y creación de propuestas, la selección de TFGs y la grabación de la solicitud por parte del alumno, la aceptación de tutorización por parte del profesor, y la validación de propuestas y gestión del panel por parte del coordinador. Se han identificado casos límite, como el intento de rechazar una propuesta sin indicar motivo o el acceso a una pantalla no permitida para el rol del usuario, y se han corregido los defectos detectados.

11 RESULTADOS

Tras el desarrollo de este proyecto se ha conseguido

implementar una plataforma que cubre el ciclo completo de gestión de TFGs, desde el registro de usuarios hasta la asignación definitiva del trabajo. El sistema queda operativo con 27 endpoints REST distribuidos en 6 recursos, 3 perfiles de usuario diferenciados y más de 20 vistas de interfaz. A continuación, se encuentra una [demo](#) que muestra el funcionamiento de la plataforma y se detallan las características funcionales más relevantes del sistema entregado.

11.1 Listado de propuestas filtrado por mención y acceso diferenciado por rol

El sistema filtra automáticamente las propuestas visibles según la mención del usuario autenticado, de forma que cada alumno y cada profesor solo accede a las propuestas compatibles con su área de especialización. Esta lógica de filtrado, aplicada tanto en las vistas del frontend como en el endpoint correspondiente, evita exponer información irrelevante al usuario y reduce el riesgo de asignaciones fuera de mención.

11.2 Lista priorizada con reordenación y grabación atómica

El alumno puede seleccionar del listado, las propuestas que más le interesen y, si lo considera pertinente, puede añadir la motivación que tiene por la oferta. De esta manera, construye una lista de hasta 6 propuestas con motivación opcional, que puede reordenar mediante drag-and-drop (arrastrar y ordenar) sin perder el orden ni duplicar prioridades.

La integridad de la lista es garantizada por la limitación existente en la BD y la llamada al backend que guarda el estado de la lista a cada cambio que se hace. La confirmación definitiva mediante "Grabar solicitud" activa el campo committed del alumno, bloqueando la lista completa en una única operación atómica y simplificando las comprobaciones de acceso en el resto de endpoints de selección. Además, impide que se borren propuestas ya grabadas y se hagan cambios que afecten al resto de usuarios candidatos.

11.3 Ciclo de validación del coordinador

El sistema cuenta con un endpoint que permite al coordinador aceptar, rechazar o solicitar corrección sobre cada propuesta, exigiendo un motivo obligatorio en estos dos últimos casos. El estado TO_BE_CORRECTED indica al alumno que ha de corregir su propuesta y le permite editarla y reenviarla. Al reenviarla, la propuesta pasa automáticamente al estado de PENDING.

11.4 Asignación final y cierre automático

La asignación de un alumno candidato ejecuta en una única transacción el registro de la asignación, la eliminación de las demás selecciones activas del alumno en otras propuestas y el incremento del contador de plazas

ocupadas. Al cubrirse el total de plazas, la propuesta transita automáticamente a CLOSED, sin requerir ninguna acción adicional por parte del profesor o del coordinador.

11.5 Dashboard del coordinador

El coordinador dispone de un panel de control con las métricas clave del sistema y acceso directo a cada sección de gestión (validaciones pendientes, alumnos, profesores y menciones), centralizando el seguimiento global del proceso sin necesidad de navegar entre vistas independientes.

12 CONCLUSIÓN

12.1 Conclusiones generales de proyecto

El propósito de este TFG era poder proporcionar una página web que gestione el proceso de creación y asignación de Trabajos de Final de Grado de forma centralizada, intuitiva y práctica para todos los usuarios. Según los objetivos planteados al inicio podemos afirmar que este proyecto se ha concluido de manera satisfactoria. Se ha desarrollado una plataforma cuya interfaz permite el filtrado automático por mención y tener un acceso diferenciado por rol. A través de una base de datos MySQL gestionado con Prisma ORM, se gestiona de todo el ciclo de propuestas, desde la creación hasta la asignación definitiva, con tablas que garantizan la integridad referencial en todo momento. Los estudiantes pueden modificar su listado de propuesta de la forma que consideren pertinente y grabarla cuando lo consideren necesario.

Además, se ha conseguido centralizar toda la información, lo que hace tener una única fuente de verdad para profesores, alumnos y coordinadores. El estado del sistema va a ser siempre el mismo para todos. Los profesores no necesitan consultar la bandeja de entrada de su correo para ver el interés que muestran los alumnos por sus propuestas. La propia plataforma gestiona la motivación y los interesados. Los coordinadores tienen siempre a su disposición un listado de estudiantes, profesores y propuestas, actualizados y gestionables.

Aún quedan aspectos por mejorar que serán comentado en el apartado de trabajo futuro.

12.2 Aprendizaje personal

El desarrollo de este proyecto ha supuesto un crecimiento significativo tanto a nivel técnico como metodológico. Aunque ya tenía experiencia previa con algunas de las tecnologías utilizadas, este TFG me exigió profundizar en ellas de una forma que no había hecho hasta ahora. Con React aprendí a entender de verdad conceptos como el contexto, los hooks y la composición de componentes, y a modular el código de forma que fuera manejable y escalable a medida que el proyecto crecía. A nivel de arquitectura, seguir el patrón MVC me ayudó a separar responsabilidades de forma clara entre directorios, y trabajar con la comunicación entre frontend y backend gestionando peticiones,

respuestas y el manejo de errores con try-catch, lo que me dio una visión práctica de como gestionar una API REST real. También aprendí a integrar herramientas que no conocía, lo que amplía el conjunto de recursos que tendré a mi disposición en proyectos futuros.

Más allá del aprendizaje técnico, este proyecto me ha dado una visión mucho más global de lo que implica construir software. Se hizo palpable que una buena toma de requisitos y un diseño cuidadoso son fases cruciales y que condicionan directamente la calidad y la mantenibilidad de todo lo que viene después. Tienen muchísima importancia tenerlas en cuenta antes de cualquier implementación. Plantearme cada vez cómo quería desarrollar una nueva funcionalidad no solo qué quería hacer, me obligó a pensar siempre en el sistema en su totalidad. Aunque no siempre se consiguió y quedan aspectos a mejorar, fue muy enriquecedora la experiencia de tener que analizar cómo una decisión de modelado afecta a los endpoints, cómo una regla de negocio se traduce a nivel de base de datos, componentes o vistas, cómo un cambio en el flujo de una parte impacta en la vista del otras. Esa forma de pensar es lo que me permitió entender el alcance que puede llegar a tener la ingeniería del software y lo más valioso que me llevo de este trabajo

12.3 Trabajo futuro

Dado que el tiempo de desarrollo de este trabajo ha sido limitado cabe destacar que aún existen funcionalidades y mejoras contempladas que no se han podido implementar dentro del alcance actual. Concretamente, se han identificado las siguientes:

- Enviar un correo de confirmación una vez creada la cuenta en la plataforma.
- Restablecimiento de contraseña mediante enlace enviado por correo (RF_).
- Mejoras en la internacionalización, ya que existe la librería quedaría implementar el soporte completo para español.
- Implementación de un pipeline de integración y despliegue continuos (CI/CD).
- Validación con usuarios reales se encuentra pendiente por las restricciones de acceso a datos académicos reales de alumnado y profesorado (riesgo en la sección 5). Una vez resuelta, permitiría recoger feedback real y aplicar mejoras de UI/UX basadas en el uso real de la plataforma.
- Despliegue en producción: la plataforma está preparada para ejecutarse sobre MySQL y puede desplegarse en cualquier proveedor cloud que soporte Node.js. Como objetivo futuro natural quedaría containerizarla con Docker, configurar las variables de entorno por entorno (desarrollo, preproducción, producción) e integrarla con la infraestructura existente de la UAB, incluyendo la

autenticación corporativa mediante SSO.

USO DE LA IA GENERATIVA

Actualmente, ser un ingeniero informático conlleva tener conocimiento y dominio de las herramientas de inteligencia artificial disponibles en el mercado, ya que es la única manera de usarlas con coherencia sin permitir nunca que asuman total autoría del trabajo realizado ni que se encarguen del desarrollo del propio proyecto. Durante la realización de este TFG se han utilizado las siguientes herramientas de IA generativa, con los usos que se detallan a continuación.

Claude Code (Anthropic) - uso frecuente. Asistencia técnica puntual en desarrollo (clarificación de mensajes de error, explicaciones de APIs de Prisma, Express o React no familiares, revisión de fragmentos de código escritos por la autora), apoyo en la estructura y revisión de los textos de los informes y, redacción de checklist de tareas basadas en planificaciones previamente redactadas y formateo a tabla de contenido previamente elaborado por la autora (casos de uso, requisitos funcionales, modelo de datos y estado de implementación). La autora ha revisado, ajustado y validado todo el contenido antes de la entrega.

Stitch - uso puntual. Inspiración y referencia visual después de tener todo el frontend diseñado, pero con la intención de mejorar las vistas de cara al futuro.

El diseño del sistema, la identificación y especificación de los casos de uso y de los requisitos funcionales, el modelo de datos con sus decisiones de modelado, la arquitectura, las reglas de negocio del flujo de TFG, la implementación del código y las conclusiones del trabajo son responsabilidad íntegra de la autora. La IA generativa se ha empleado como herramienta de asistencia para tareas mecánicas, de soporte o de depuración, pero nunca como sustituto de las decisiones técnicas, de diseño o de análisis del proyecto.

AGRADECIMIENTOS

Quisiera agradecer a mi tutor su apoyo, sus comentarios y la confianza depositada en mí para llevar a cabo este proyecto. A mis padres, por estar a mi lado en todo momento. A mis amigos, por darme fuerza y acompañarme durante este camino, y a mi pareja, por creer siempre en mí.

BIBLIOGRAFÍA

- [1] A. Torres, "Qué es y cómo funciona Jira Service Management," excentia. [En línea]. Disponible: <https://www.excentia.es/que-es-como-funciona-jira-service-management>
- [2] InboundCycle, "Zendesk: qué es, funciones y ventajas," InboundCycle Blog. [En línea]. Disponible: <https://www.inboundcycle.com/blog-de-inbound-marketing/que-es-zendesk>
- [3] Asociación @aslan, "Gestión integral digital de los trabajos de fin de titulación - XVI convocatoria premios AAPP." [En línea]. Disponible: <https://aslan.es/gestion-integral-digital-de-lostrabajos-de-fin-de-titulacion-candidatura2024/>

- [4] Ó. Ruiz Rodríguez, "Aplicación web para la gestión completa de los TFG 'TFGTool'," Trabajo de grado, Universidad Autónoma de Madrid, Madrid, España, 2020. [En línea]. Disponible: <https://repositorio.uam.es/entities/publication/95beb4df-12c7-4963-a4d6-316691ad1fa7>
- [5] "Estudiantes – plataforma para la gestión del trabajo fin de grado," Universidad de Granada. [En línea]. Disponible: <https://wpd.ugr.es/~fcienciastfg/estudiantes/>
- [6] "Eina de seguiment de treballs de final d'estudis (TFE)," Suport del Campus Virtual UAB. [En línea]. Disponible: <https://cv.uab.cat/ajuda/ca/2021/09/03/eina-de-seguiment-de-treballs-fi-destudis-tfe/>
- [7] Slack, "Las metodologías Scrum en la gestión de proyectos," Slack Blog. [En línea]. Disponible: <https://slack.com/intl/es-es/blog/productivity/metodologias-scrum>
- [8] Amazon Web Services, Inc., "¿Qué es el SDLC? Explicación del ciclo de vida del desarrollo de software," AWS. [En línea]. Disponible: <https://aws.amazon.com/es/what-is/sdlc/>
- [9] "React v19 – react". React. [En línea]. Disponible: <https://react.dev/blog/2024/12/05/react-19>
- [10] "Vite". vitejs. [En línea]. Disponible: <https://vite.dev/>
- [11] "React router official documentation". React Router Official Documentation. [En línea]. Disponible: <https://reactrouter.com/>
- [12] "Vue.js". Vue.js - The Progressive JavaScript Framework | Vue.js. [En línea]. Disponible: <https://vuejs.org/>
- [13] "Next.js by vercel - the react framework". Next.js by Vercel - The React Framework. [En línea]. Disponible: <https://nextjs.org/>
- [14] "Node.js — Run JavaScript Everywhere". Node.js — Run JavaScript Everywhere. [En línea]. Disponible: <https://nodejs.org/en>
- [15] "Introducing express v5: A new era for the node.js framework · express.js". Express.js. [En línea]. Disponible: <https://expressjs.com/en/blog/2024-10-15-v5-release/>
- [16] "Deno, the next-generation JavaScript runtime". Deno. [En línea]. Disponible: <https://deno.com/>
- [17] "NestJS - A progressive Node.js framework". NestJS - A progressive Node.js framework. [En línea]. Disponible: <https://nestjs.com/>
- [18] Prisma, "Introduction to Prisma Client," Prisma Documentation. [En línea]. Disponible: <https://www.prisma.io/docs/orm/prisma-client/setup-and-configuration/introduction#3-importing-prisma-client>
- [19] "MySQL en detalle: Guía para dominar esta potente base de datos". fw_error_www. [En línea]. Disponible: <https://www.oracle.com/es/mysql/what-is-mysql/>
- [20] "TypeORM - Code with Confidence. Query with Power. | TypeORM". TypeORM - Code with Confidence. Query with Power. | TypeORM. [En línea]. Disponible: <https://typeorm.io/>
- [21] "PostgreSQL". PostgreSQL. [En línea]. Disponible: <https://www.postgresql.org/>
- [22] "JSON web token introduction - jwt.io". JSON Web Tokens - jwt.io. [En línea]. Disponible: <https://www.jwt.io/introduction#what-is-json-web-token>
- [23] Microsoft. "Visual Studio Code - The open source AI code editor | Your home for multi-agent development". Visual Studio Code - The open source AI code editor | Your home for multi-agent development. [En línea]. Disponible: <https://code.visualstudio.com/>
- [24] "GitHub · change is constant. github keeps you ahead." GitHub. [En línea]. Disponible: <https://github.com/>
- [25] JetBrains. "WebStorm: The JavaScript and TypeScript IDE, by JetBrains". JetBrains. [En línea]. Disponible: <https://www.jetbrains.com/webstorm/>
- [26] "Bitbucket | Git solution for teams using Jira". Bitbucket. [En línea]. Disponible: <https://bitbucket.org/product/>
- [27] Prisma, "Transactions and batch queries (Reference)," Prisma Documentation. [En línea]. Disponible: <https://www.prisma.io/docs/orm/prisma-client/queries/transactions#nested-writes-for-dependent-operations>
- [28] Prisma, "Filtering and sorting," Prisma Documentation. [En línea]. Disponible: <https://www.prisma.io/docs/orm/v6/prisma-client/queries/filtering-and-sorting>
- [29] "Prisma findMany where OR — ensure that at least X conditions met?," Stack Overflow. [En línea]. Disponible: <https://stackoverflow.com/questions/77793963/prisma-findmany-where-ensure-that-at-least-x-conditions-met>
- [30] "Introduction," react-i18next Documentation. [En línea]. Disponible: <https://react.i18next.com/>
- [31] "Interpolation," i18next Documentation. [En línea]. Disponible: <https://www.i18next.com/translation-function/interpolation>
- [32] Prisma, "Generating Prisma Client," Prisma Documentation. [En línea]. Disponible: <https://www.prisma.io/docs/orm/prisma-client/setup-and-configuration/generating-prisma-client>
- [33] Prisma, "Prisma schema," Prisma Documentation. [En línea]. Disponible: <https://www.prisma.io/docs/orm/prisma-schema/overview>
- [34] "Data types," Prisma/DB Documentation, v0.4.4. [En línea]. Disponible: <https://prismadb.readthedocs.io/en/0.4.4/data-types/>
- [35] S. Pal, "Implementing JWT authentication and password encryption with bcrypt in a Node.js application," DEV Community. [En línea]. Disponible: <https://dev.to/beincharacter/implementing-jwtauthentication-and-password-encryption-with-bcrypt-in-a-nodejs-application-1572>
- [36] Metana, "Bcrypt and JWT: web app security," Metana Blog. [En línea]. Disponible: <https://metana.io/blog/bcrypt-and-jwt-web-app-security/>
- [37] R. Ambadi, "Setting up auth routes with Express," DEV Community. [En línea]. Disponible: <https://dev.to/aritik/setting-up-auth-routes-with-express-57oi>

ANNEXOS

A1. ESTADO DEL ARTE

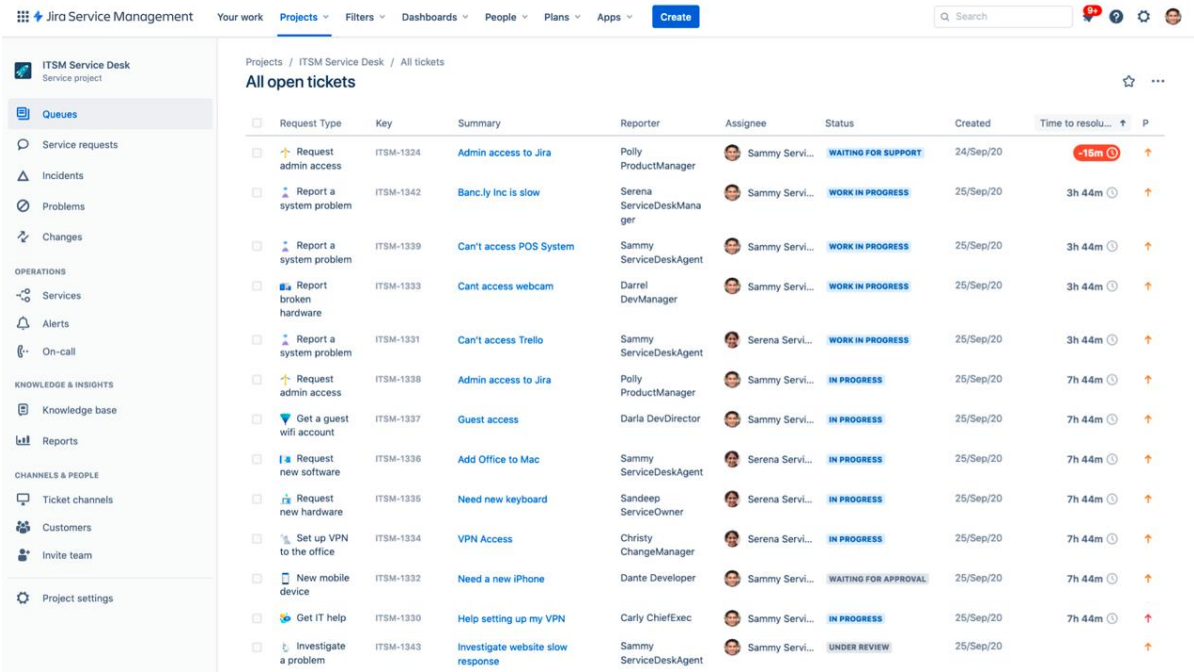


Figura 1. Pantalla principal del Jira Service Management

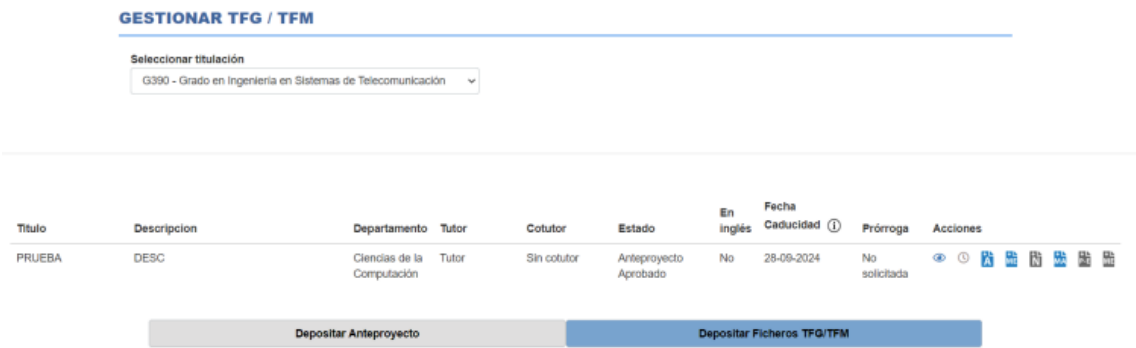


Figura 2. Selección de TFG en la plataforma GestTFx

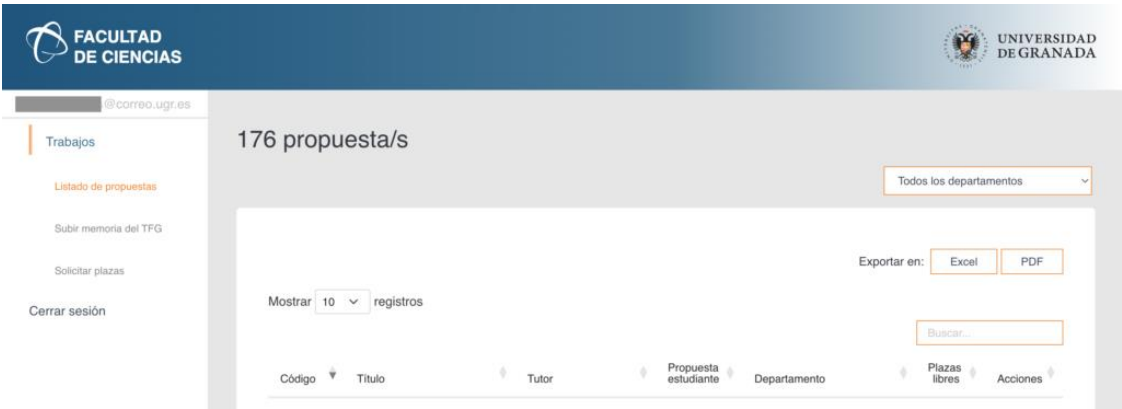


Figura 3. Listado de propuestas de la plataforma de la Universidad de Granada

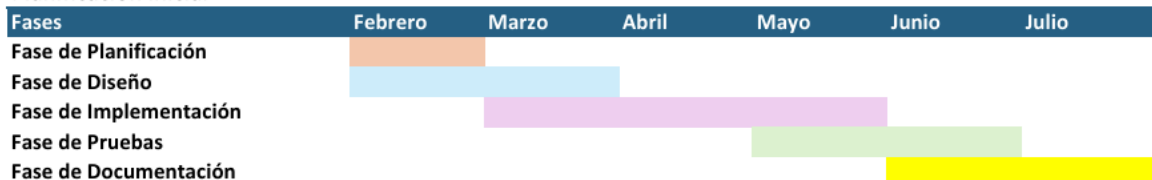
A2. REQUISITOS FUNCIONALES

RF_01	Registrarse en la web	Alta	Completado
RF_02	Iniciar sesión	Alta	Completado
RF_03	Crear propuesta de TFG	Alta	Completado
RF_04	Eliminar propuesta creada	Alta	Completado
RF_05	Ver estado de una propuesta	Alta	Completado
RF_06	Ver detalle de una propuesta	Alta	Completado
RF_07	Consultar propuestas disponibles	Alta	Completado
RF_08	Seleccionar propuestas (alumno)	Alta	Completado
RF_09	Grabar solicitud	Alta	Completado
RF_10	Modificar prioridad de selección	Media	Completado
RF_11	Eliminar una selección	Media	Completado
RF_12	Corregir propuesta devuelta	Alta	Completado
RF_13	Solicitar tutorizar una propuesta	Alta	Completado
RF_14	Ver candidatos y asignar alumno	Alta	Completado
RF_15	Asignar menciones a profesores	Alta	Completado
RF_16	Validar propuesta de alumno	Alta	Completado
RF_17	Ver propuestas pendientes de validación	Alta	Completado
RF_18	Ver listado de alumnos y su estado	Alta	Completado
RF_19	Ver listado de profesores	Alta	Completado
RF_20	Cargar profesores por fichero	Media	Pendiente
RF_21	Ver resumen del sistema (dashboard)	Alta	Completado
RF_22	Exportar datos a fichero Excel	Media	Completado
RF_23	Restablecer contraseña	Media	Pendiente
RF_24	Cierre automático de propuesta	Alta	Completado

Figura 5. Listado de los requisitos funcionales y su estado final.

A3. DIAGRAMA DE GANTT

Planificación inicial



Ejecución final

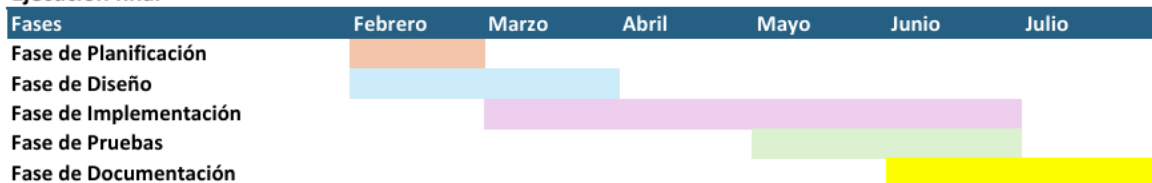


Figura 6. Diagrama de Gantt con la comparativa entre planificación inicial y ejecución final.

A5. MODELO DE DATOS

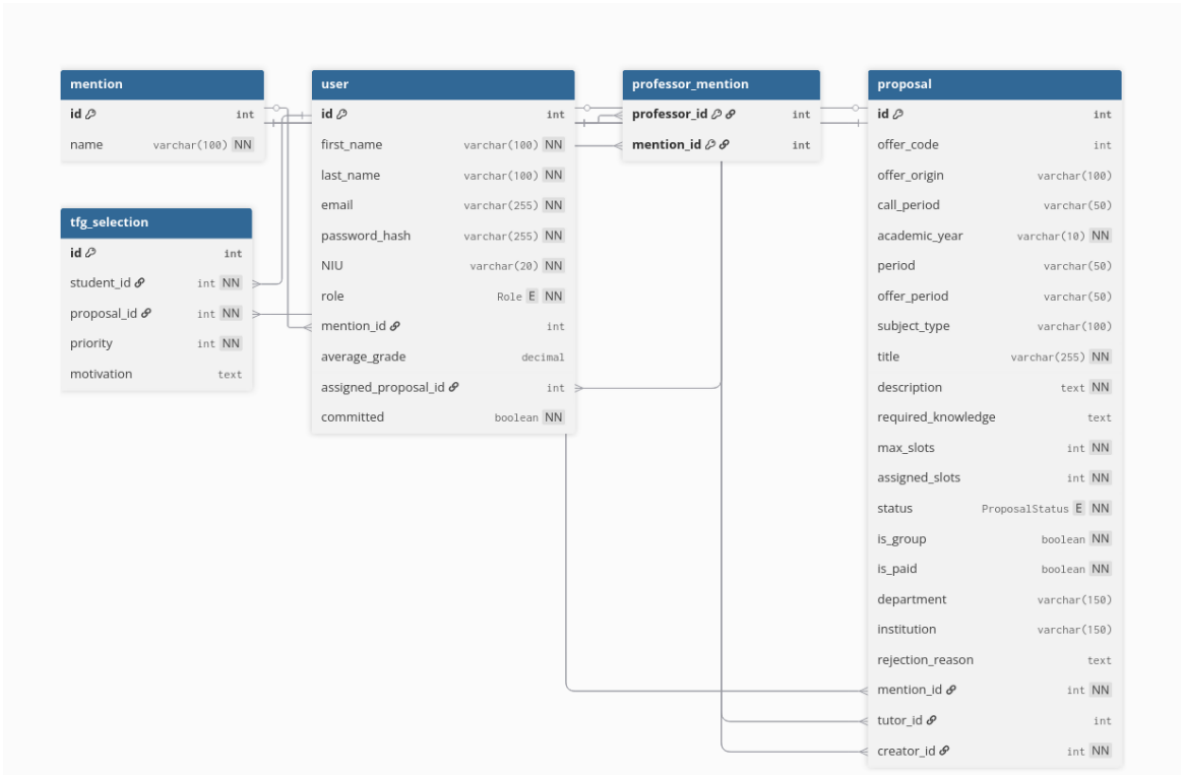


Figura 7. Modelo de datos del sistema.

A3. DIAGRAMAS DE CASOS DE USO

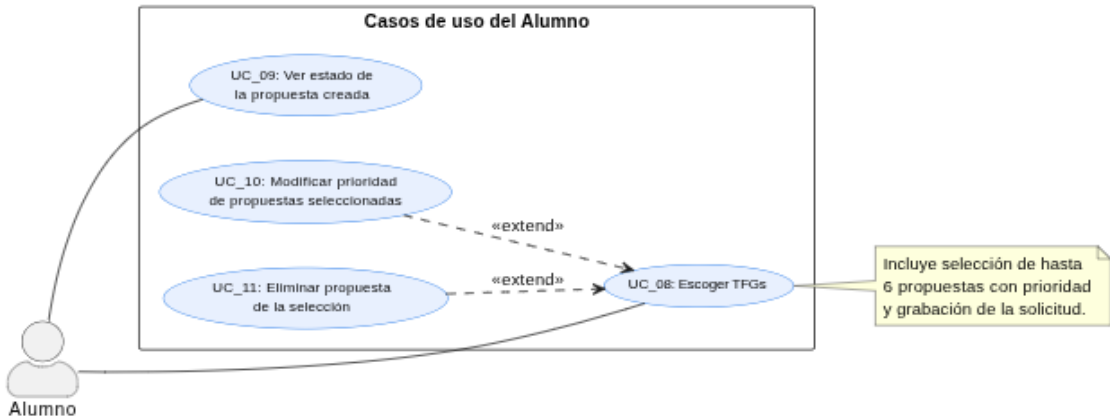


Figura 8. Casos de uso específico a un alumno.

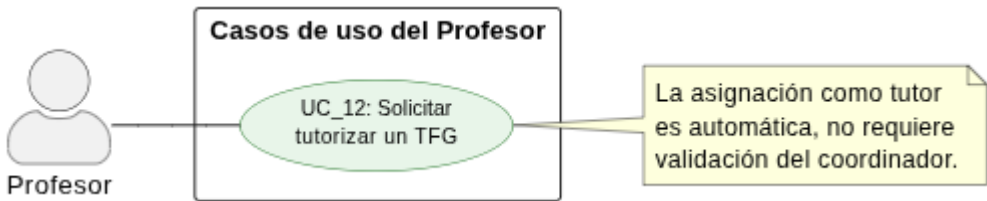


Figura 9. Casos de uso específico a un profesor.

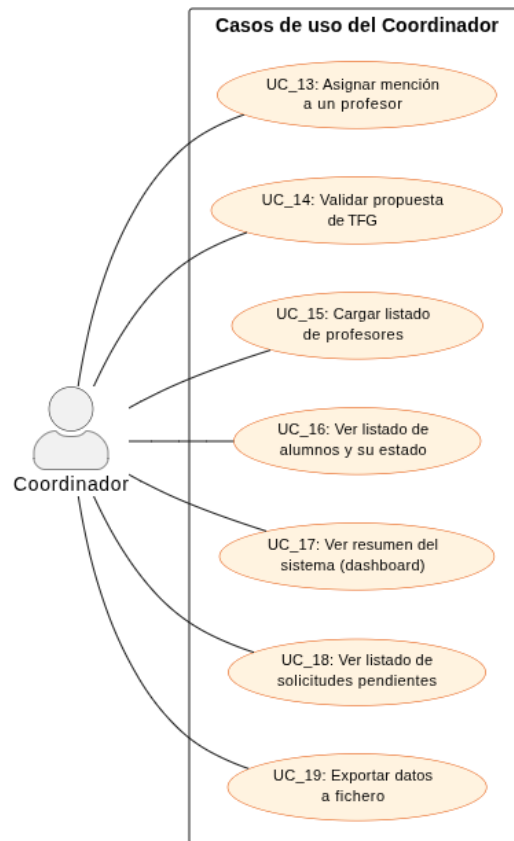


Figura 10. Casos de uso específico a un coordinador

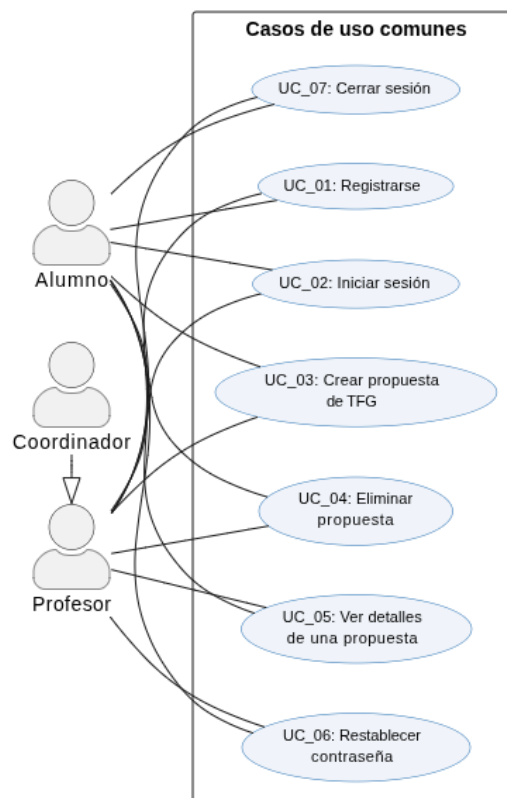


Figura 11. Casos de usos comunes a todos los roles