UNIVERSIDAD DE MÁLAGA

E.T.S. INGENIERÍA
INFORMÁTICA
UNIVERSIDAD DE MÁLAGA

Grado en Ingeniería del Software

Desarrollo de una aplicación web progresiva sobre la gestión de trabajos de fin de grado en universidades

(Development of a progressive web application on the management of end-of-degree projects in universities)

Realizado por
David Bueno Carmona

Tutorizado por
Eduardo Guzmán de los Riscos

Departamento
LENGUAJES Y CIENCIAS DE LA COMPUTACIÓN
UNIVERSIDAD DE MÁLAGA

MÁLAGA, septiembre 2025

UNIVERSIDAD DE MÁLAGA

E.T.S. INGENIERÍA INFORMÁTICA
UNIVERSIDAD DE MÁLAGA

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA

Grado en Ingeniería del Software

Desarrollo de una aplicación web progresiva sobre la gestión de trabajos de fin de grado en universidades

Development of a progressive web application on the management of end-of-degree projects in universities

Realizado por

David Bueno Carmona

Tutorizado por

Eduardo Guzmán de los Riscos

Departamento

LENGUAJES Y CIENCIAS DE LA COMPUTACIÓN

UNIVERSIDAD DE MÁLAGA

MÁLAGA, SEPTIEMBRE DE 2025

Fecha defensa: Septiembre de 2025

Resumen

Este Trabajo de Fin de Grado aborda la problemática de la gestión académica de TFGs, tradicionalmente realizada mediante correos, hojas de cálculo o sistemas cerrados poco adaptables. Estos métodos generan ineficiencias, errores y sobrecarga administrativa.

Como resultado se propone TFGInfo, una aplicación web progresiva, multiplataforma y accesible, diseñada para centralizar la gestión completa de los TFGs: creación, asignación, seguimiento y finalización. El sistema contempla distintos perfiles de usuario (administrador, profesor y estudiante) con roles y permisos diferenciados, lo que garantiza seguridad, integridad de datos y una experiencia adaptada a cada actor.

El desarrollo se ha llevado a cabo siguiendo los principios ágiles, estructurado en sprints, permitiendo iteraciones incrementales y retroalimentación temprana. Se ha utilizado MySQL como base de datos relacional, .NET Core para el backend y Angular con Angular Material para el frontend, priorizando modularidad, escalabilidad y usabilidad. Además, se implementaron pruebas de accesibilidad (WAVE), rendimiento y buenas prácticas (Lighthouse), así como una interfaz responsive y adaptable a distintos dispositivos.

Entre las principales funcionalidades destacan:

- Gestión de usuarios, roles y permisos.

- Administración de líneas de TFG, reservas y solicitudes.
- Creación de grupos de trabajo y canales de comunicación entre alumnos y tutores.
- Notificaciones automáticas vía correo electrónico.
- Diseño orientado a la accesibilidad y la experiencia de usuario.

TFGInfo intenta cubrir un vacío entre las soluciones corporativas rígidas y las herramientas improvisadas, ofreciendo una plataforma ligera, extensible y de fácil adopción.

Palabras clave: MySQL, Angular, .NET Core, Angular Material, TFG

Abstract

This Final Degree Project addresses the challenges of managing academic projects, which are often handled through emails, spreadsheets, or closed institutional systems with limited flexibility. These methods lead to inefficiencies, errors, and administrative overload.

As a result, TFGInfo is presented as a progressive, cross-platform web application that centralizes the full lifecycle of final projects: creation, assignment, monitoring, and completion. The system defines different user roles (Administrator, Professor, Student) with tailored permissions to ensure security, data integrity, and a role-specific user experience.

The project was developed using an agile methodology with sprints, enabling incremental iterations and early feedback. The chosen technologies include MySQL for the relational database, .NET Core for the backend, and Angular with Angular Material for the frontend, focusing on modularity, scalability, and usability. Accessibility and performance were validated using tools such as WAVE and Lighthouse, while the interface was designed to be responsive and user-friendly across devices.

Key features include:

- User, role, and permission management.
- Administration of project lines, reservations, and student requests.

- Creation of workgroups and communication channels between students and supervisors.
- Automatic email notifications.
- Design focused on accessibility and user experience.

The conclusions highlight that TFGInfo bridges the gap between rigid corporate solutions and improvised tools, offering a lightweight, extensible, and user-friendly platform.

Keywords: MySQL, Angular, .NET Core, Angular Material, TFG

Índice

Índice de Ilustraciones

Índice de tablas

1

Introducción

1.1. Motivación

La gestión de los Trabajos de Fin de Grado (TFG) en muchas universidades es un proceso que involucra múltiples actores y tareas administrativas que, cuando no están correctamente coordinadas, pueden derivar en ineficiencias, errores y frustración para estudiantes, profesores y personal de administración.

Tradicionalmente, la asignación de TFGs, la selección de tutores, la formación de grupos y la entrega de documentos se han gestionado a través de correos electrónicos, hojas de cálculo o formularios impresos. Esta falta de integración provoca pérdida de información, duplicidades, y sobrecarga administrativa. Además, los sistemas existentes en algunas universidades suelen ser cerrados, de difícil acceso para los estudiantes, o limitados en cuanto a personalización.

TFGInfo nace como una respuesta a esta problemática, proponiendo una solución web centralizada, accesible desde cualquier dispositivo, y adaptada a las necesidades específicas de los centros educativos. Su desarrollo también representa una oportunidad para aplicar conocimientos adquiridos en el grado en Ingeniería del Software, tales como diseño de bases de datos, desarrollo web full-stack, seguridad, y experiencia de usuario.

Desde una perspectiva académica y profesional, el proyecto busca resolver una necesidad real en el ámbito universitario, al tiempo que constituye una valiosa experiencia de desarrollo de software completo, desde el análisis de requisitos hasta la implementación y pruebas. Además, TFGInfo pone especial énfasis en la modularidad y extensibilidad, de forma que pueda ser evolucionado en el futuro con nuevas funcionalidades, integraciones o adaptaciones a otras instituciones.

1.2. Objetivos

Diseñar e implementar una aplicación web completa para gestionar TFGs

El objetivo principal del proyecto es desarrollar una solución tecnológica integral que abarque tanto el backend como el frontend. La aplicación debe permitir gestionar el ciclo de vida completo de un TFG, incluyendo la creación, asignación, seguimiento y finalización de este. Esta implementación incluye el modelado de la base de datos, el desarrollo de servicios mediante una API REST, y la construcción de una interfaz de usuario moderna, intuitiva y responsive.

Ofrecer una interfaz intuitiva tanto para administradores como para profesores y alumnos

La experiencia de usuario es fundamental para asegurar la adopción de la herramienta por parte de los distintos perfiles que interactúan con el sistema. Por ello, se ha planteado un diseño centrado en el usuario (UX), donde cada tipo de usuario accede únicamente a las funcionalidades relevantes para su rol, minimizando la complejidad y evitando sobrecargas de información. Los alumnos pueden consultar líneas de TFG disponibles, los profesores pueden gestionar sus propuestas y los administradores pueden supervisar y validar todo el proceso.

Integrar roles y permisos diferenciados

El sistema debe contemplar una gestión de usuarios basada en roles que limite el acceso a la información y funcionalidades según el perfil del usuario. Los roles definidos son: Administrador, Profesor y Estudiante. Esta distinción es clave para mantener la seguridad, la integridad de los datos y garantizar que cada usuario interactúe con el sistema de forma adecuada a su función institucional.

Garantizar la integridad de los datos

Uno de los objetivos técnicos esenciales del proyecto es asegurar que la información almacenada sea coherente y no se produzcan inconsistencias en la base de datos. Esto se logra mediante el uso de claves foráneas y validaciones tanto en el backend como en el frontend.

Facilitar el trabajo colaborativo mediante grupos de trabajo

El sistema permite la creación y gestión de grupos de trabajo que pueden estar formados por uno o varios alumnos y profesores. Esta funcionalidad permite reflejar la realidad de muchos TFGs que se desarrollan en entornos colaborativos, interdepartamentales o interinstitucionales. Además, fomenta la transparencia y la comunicación entre todos los implicados en un proyecto de fin de grado.

1.3. Antecedentes

La necesidad de gestionar TFGs de forma organizada y eficiente ha impulsado el desarrollo de diferentes plataformas en el ámbito universitario. A continuación, se analizan algunas de las herramientas más conocidas y utilizadas, así como sus limitaciones en el contexto en el que se enmarca este proyecto.

SIGMA [2]

Es una de las soluciones más extendidas en universidades españolas. Se trata de un sistema de gestión académica muy completo que abarca desde la matriculación hasta la gestión de titulaciones, expedientes y TFGs. Sin embargo, SIGMA presenta ciertas desventajas:

- Su implementación suele requerir acuerdos institucionales complejos y costosos.
- Tiene una curva de aprendizaje pronunciada, especialmente para usuarios no técnicos.
- No es fácilmente adaptable a necesidades específicas de departamentos o facultades concretas.
- Carece de flexibilidad para integrar nuevos flujos de trabajo sin intervención del proveedor.

Universitas XXI [3]

Esta plataforma también es muy potente y utilizada en muchas instituciones de educación superior. Su foco está en la gestión administrativa de las universidades en su conjunto, incluyendo módulos para TFGs. A pesar de sus capacidades:

- Suele estar muy centralizado y controlado por las áreas de administración, lo que limita la autonomía de usuarios como profesores y estudiantes.
- La personalización e incorporación de nuevas funcionalidades no es sencilla.

- En muchos casos, su uso se limita a consultas pasivas, sin interacción fluida entre los distintos actores.

Soluciones caseras: hojas de cálculo, formularios y correo electrónico

En muchos departamentos, especialmente aquellos sin acceso a plataformas institucionales robustas, la gestión de TFGs se realiza mediante hojas de cálculo compartidas (por ejemplo, en Google Sheets), formularios en línea (Google Forms, Microsoft Forms), o incluso por intercambio de correos electrónicos. Esta aproximación es ágil al principio, pero presenta grandes limitaciones:

- No hay control de acceso ni gestión de roles.
- Se generan errores frecuentes por edición simultánea o pérdida de datos.
- Es difícil escalar el sistema cuando aumenta el volumen de TFGs o el número de usuarios.

¿Qué aporta TFGInfo frente a estas alternativas?

TFGInfo nace con el objetivo de situarse en un punto intermedio entre las soluciones corporativas sobredimensionadas y las soluciones improvisadas. Su propuesta de valor incluye:

- **Ligereza y portabilidad**: al ser una aplicación web modular y abierta, puede desplegarse en servidores propios o en la nube sin dependencia de terceros.
- **Acceso multiplataforma**: funciona desde navegadores modernos, sin necesidad de instalaciones adicionales.
- **Gestión por roles**: incorpora una gestión de permisos desde el diseño, asegurando que cada usuario accede solo a las funcionalidades que le corresponden.
- **Personalización**: puede adaptarse fácilmente a las particularidades de cada facultad, titulación o plan de estudios.

En resumen, TFGInfo pretende cubrir el vacío existente entre las soluciones rígidas institucionales y las herramientas improvisadas, proporcionando una plataforma equilibrada, eficaz y accesible para la gestión académica del TFG.

1.4. Metodología

Para realizar este proyecto se ha aplicado una metodología incremental e iterativa basada en los principios ágiles [1]. La finalidad es dividir el trabajo en iteraciones o sprints ya que en el desarrollo de una aplicación existen etapas muy diferenciables. Se ha elegido esta metodología por las ventajas que aporta:

Flexibilidad ante cambios

Durante el desarrollo de un sistema de gestión como TFGInfo, es común que los requisitos evolucionen o se refinen a medida que se avanza. La metodología ágil permite adaptar el rumbo del proyecto sin penalizar el avance previo, facilitando la incorporación de nuevas funcionalidades, cambios en la lógica de negocio o mejoras en la interfaz.

Descomposición en entregas pequeñas

Esta metodología propone dividir el trabajo en unidades más manejables (iteraciones o sprints), lo que permite centrarse en objetivos concretos a corto plazo. Esto hace que el progreso sea visible, medible y fácil de gestionar. En el caso de TFGInfo, esto se tradujo en diseñar la estructura de la aplicación realizando el backend, luego funcionalidades básicas, y más tarde componentes visuales avanzados.

Retroalimentación temprana

Aunque se trata de un proyecto académico, la posibilidad de obtener feedback de profesores, compañeros o incluso usuarios reales (como estudiantes y docentes) durante el desarrollo puede enriquecer el resultado. Con esta metodología, este feedback puede incorporarse entre iteraciones sin necesidad de rehacer todo el trabajo.

Reducción de riesgos

Al ir construyendo y validando el sistema de forma incremental, se identifican errores y cuellos de botella más rápido. Esto reduce el riesgo de que un fallo grave se detecte demasiado tarde. Por ejemplo, si un endpoint presenta problemas de rendimiento o si una vista es confusa, puede corregirse en la siguiente iteración.

Mejor planificación y control del tiempo

En el contexto de un TFG, donde hay una fecha límite clara, una metodología ágil permite planificar de forma realista las tareas según el tiempo disponible. Se priorizan las funcionalidades esenciales primero y se dejan los extras para fases finales, evitando que el proyecto quede incompleto por falta de tiempo.

Entrega de valor desde el principio

Incluso si el proyecto no se completa en su totalidad, al final de cada iteración se tiene una versión funcional del sistema. Esto garantiza que siempre haya una "versión demostrable" del proyecto, algo crucial para presentaciones, tutorías o revisiones intermedias.

Orientación a objetivos claros y medibles

Se favorece la definición de metas concretas por iteración. Esto ayuda a mantener el enfoque y a evitar desviaciones. En el caso de TFGInfo, se traduce en tareas

bien definidas como “crear endpoint de búsqueda de TFGs” o “diseñar formulario de alta de alumno”.

1.4.1. Sprints

El desarrollo del proyecto se organizó siguiendo una aproximación ágil basada en sprints, con el objetivo de dividir el trabajo en iteraciones manejables y asegurar la entrega de valor de forma progresiva. Cada sprint se orientó a alcanzar objetivos concretos y producir entregables verificables, de manera que el sistema pudiera evolucionar de forma controlada y medible.

Cabe destacar que, aunque la filosofía ágil sugiere desarrollar funcionalidades completas en cada iteración (incluyendo tanto backend como frontend), en este proyecto se optó por una adaptación práctica. Se decidió priorizar la implementación del backend en las fases iniciales, con el fin de disponer de una API estable y probada antes de abordar la construcción del frontend. Esta estrategia permitió reducir el riesgo de retrabajos, ya que cualquier modificación en la lógica de negocio o en los modelos de datos habría implicado rehacer componentes visuales si se hubieran desarrollado en paralelo desde el principio.

De esta forma, los primeros sprints se centraron en el diseño y desarrollo de la base de datos y la API REST, mientras que en fases posteriores se implementaron las interfaces de usuario y su integración con los servicios ya consolidados. Este enfoque garantizó un desarrollo más ordenado y eficiente, manteniendo el espíritu iterativo e incremental de la metodología ágil.

1.4.1.1. Sprint 1: Análisis y planificación del sistema

Objetivo:

Definir las bases del proyecto TFGInfo, analizando los requisitos funcionales y no funcionales, los posibles riesgos, y estableciendo una estructura sólida a nivel conceptual y de datos. Este sprint representa el punto de partida del proyecto, marcando la dirección técnica y organizativa del desarrollo.

Tareas:

Identificación de requisitos

En esta fase se recopilaron y documentaron las necesidades que debía cubrir la aplicación. Se realizó un análisis de los diferentes perfiles de usuario (administrador, profesor, alumno), sus interacciones con el sistema, y los procesos clave que debía soportar la plataforma, como la gestión de TFGs, la creación de grupos, la asignación de líneas temáticas y la autenticación basada en roles. Se elaboraron casos de uso, historias de usuario y se identificaron requisitos funcionales (qué debe hacer el sistema) y no funcionales (rendimiento, seguridad, accesibilidad, etc.).

Identificación de riesgos

En esta tarea se evaluaron los factores que podrían comprometer el desarrollo o el éxito del proyecto. Algunos de los riesgos detectados fueron:

- Complejidad de las relaciones entre entidades (TFGs, usuarios, grupos, departamentos).
- Posibles errores al importar datos desde fuentes externas (CSV).
- Dificultades en la sincronización entre frontend y backend.

Diseño y modelado de la aplicación

Se construyó un diagrama de clases que representa las entidades principales del sistema y sus relaciones. Este diseño sirvió como base tanto para la implementación de la base de datos como para la arquitectura del backend. Se optó por un enfoque modular, donde cada componente (usuarios, TFGs, grupos, departamentos) se trató como una unidad autónoma con su propia lógica y controlador. Además, se definieron los flujos de interacción entre componentes y los puntos clave de acceso (API REST).

Diseño del modelo y de la base de datos

Basándose en el modelado anterior, se diseñó una base de datos relacional utilizando MySQL. Se definieron las tablas necesarias, sus atributos, claves primarias y foráneas, así como las relaciones entre ellas. Especial atención se puso en las relaciones muchos a muchos, por ejemplo, entre profesores y TFGs, o entre alumnos y grupos de trabajo. Se diseñaron tablas intermedias para estas relaciones, y se aplicaron restricciones de integridad referencial para asegurar la coherencia de los datos. También se preparó un conjunto inicial de datos para realizar pruebas, incluyendo scripts para crear la base de datos.

Entregables:

Documento de requisitos y casos de uso

Incluye la descripción de cada funcionalidad del sistema, los actores implicados, los escenarios principales, y los requerimientos técnicos y operativos. Sirve como referencia para todos los sprints posteriores.

Diagrama de clases y esquema de base de datos relacional

El diagrama de clases permite visualizar la estructura del sistema desde un punto de vista lógico, mientras que el esquema relacional detalla cómo se implementan

estas relaciones en la base de datos. Ambos documentos son fundamentales para los desarrolladores.

Lista priorizada de funcionalidades (Product Backlog)

Se definió un backlog inicial con todas las funcionalidades necesarias ordenadas por prioridad y complejidad. Esto permitió estructurar los siguientes sprints de forma eficiente, garantizando que se abordaran primero las funcionalidades más críticas para el funcionamiento básico del sistema.

1.4.1.2. Sprint 2: Preparación de entorno y estructura base

Objetivo:

Establecer la base técnica mínima viable del sistema, configurando tanto el entorno de desarrollo como los primeros componentes funcionales del backend. Este sprint sienta las bases para poder comenzar el desarrollo de funcionalidades reales, garantizando que la infraestructura esté correctamente integrada y probada.

Tareas:

Extracción de los datos e integración en la base de datos.

Esta tarea incluyó:

- Revisión y limpieza de los datos: detección de inconsistencias, caracteres extraños, errores tipográficos y formatos incorrectos.
- Creación de scripts SQL para importar los datos a la base de datos MySQL de forma ordenada.
- Normalización de los datos: separación en tablas relacionadas, asignación de claves foráneas y eliminación de redundancias.
- Verificación de la integridad referencial y relaciones entre entidades (por ejemplo, cada alumno debe pertenecer a una carrera válida).

Creación de un back-end .NET

Se configuró un proyecto **ASP.NET Core** como base del backend del sistema. Esta tarea incluyó:

- Creación del proyecto desde Visual Studio y configuración del entorno de desarrollo.
- Instalación de dependencias necesarias.
- Definición del DbContext y clases de entidades que reflejan el modelo de datos diseñado en el sprint anterior.
- Configuración del archivo appsettings.json con los parámetros de conexión a la base de datos MySQL.
- Estructuración del proyecto en carpetas claras: Models, Controllers, Services, DTOs, etc.

Entregables:

Proyecto ASP.NET Core configurado

Se entrega un proyecto backend funcional con la arquitectura base ya creada. Está listo para que se añadan controladores, lógica de negocio y autenticación en los siguientes sprints.

Conexión y prueba de base de datos MySQL

El backend ya está conectado a la base de datos local (o remota) y es capaz de lanzar migraciones automáticas, ejecutar consultas básicas y manipular datos con Entity Framework.

Datos de prueba cargados correctamente desde CSV

Gracias a los scripts desarrollados, la base de datos contiene información realista que permite probar la lógica del sistema sin necesidad de ingresar datos manualmente.

Pruebas básicas de acceso a datos (API mínima)

Se desarrollaron endpoints iniciales de prueba para validar que las operaciones básicas (GET, POST, PUT, DELETE) funcionan correctamente. Estas pruebas demostraron que la API puede leer entidades desde la base de datos y devolverlas en formato JSON, lo cual confirma que la infraestructura está correctamente conectada y operativa.

1.4.1.3. Sprint 3: Desarrollo del backend funcional

Objetivo:

Implementar los controladores principales de la API REST, desarrollando la lógica de negocio asociada a cada uno de los módulos clave del sistema. Este sprint marca el paso de una infraestructura técnica básica a un sistema con funcionalidades específicas para la gestión de TFGs, usuarios y recursos académicos.

Tareas:

Desarrollo de controladores y endpoints

Se implementaron los controladores principales, cada uno encargado de gestionar un conjunto de recursos y operaciones críticas del sistema. Se usó el patrón RESTful, asegurando rutas claras, consistentes y seguras, utilizando métodos HTTP adecuados (GET, POST, PUT, DELETE). Entre ellos se encuentran:

Controlador de Usuarios:

Este controlador es el encargado de gestionar la parte de autenticación en la aplicación. Tiene funcionalidades como, realizar el inicio de sesión, realizar el registro de un usuario nuevo, hacer las comprobaciones sobre el token de autenticación, conceder roles a los usuarios y comprobar que acciones están permitidas a los usuarios según su rol.

Controlador de Reservas:

Este controlador se centra en las acciones relacionadas con las reservas de un TFG, como la solicitud de un TFG por parte del alumno, la aceptación o denegación del TFG por parte del profesor, el envío de correos notificando las actualizaciones en el TFG, el cambio de estado que tiene un proyecto, entre otras funcionalidades.

Controlador de Líneas de TFG:

Gestiona las operaciones relacionadas con los Trabajos de Fin de Grado como podrían ser: La creación y modificación de líneas de TFGs, Asignación de tutores a las líneas, asignación de carreras que pueden optar a hacer el TFG en cuestión, etc.

Otros controladores:

Existe un controlador por cada entidad de la base de datos con la lógica de la que se encarga cada uno. La lógica básica que tienen todos son un CRUD, y las entidades que tienen controladores son: Alumnos, Profesores, Departamentos, Titulaciones, Centros, Líneas de TFG, Instancias de TFG y Canales de trabajo.

Entregables:

API REST funcional con rutas bien definidas

La estructura de rutas sigue convenciones REST claras, como:

- api/usuarios/{id}
- api/reservas/solicitar
- api/departamentos

Esto facilita el uso de la API por parte del frontend y su documentación.

Lógica de negocio modular y testeada

Cada controlador está vinculado a un servicio de negocio independiente (por ejemplo, UserService, TFGService), lo que mejora la organización del código y facilita su testeo.

Validaciones y gestión de errores implementados

Todos los controladores integran validaciones a nivel de entrada (campos requeridos, formatos, relaciones válidas) y gestión de errores con respuestas claras en formato JSON. Se implementaron respuestas estandarizadas (200 OK, 400 Bad Request, 404 Not Found, 500 Internal Server Error) que facilitan la depuración y la experiencia de desarrollo del frontend.

1.4.1.4. Sprint 4: Desarrollo del frontend e integración inicial

Objetivo:

Construir la interfaz de usuario utilizando el framework Angular, creando componentes reutilizables y dinámicos, formularios interactivos y vistas con capacidades de búsqueda. Este sprint también incluyó la primera integración real entre el frontend y la API desarrollada en sprints anteriores.

Tareas:

Creación del front-end en Angular

Se inicializó el proyecto Angular utilizando Angular CLI, definiendo una estructura clara y modular de carpetas:

- components, para los elementos reutilizables (formularios, listas, botones),
- services, para las conexiones HTTP con el backend,
- models, para las interfaces de datos (DTOs),
- pages, para las vistas principales del sistema (dashboard de reservas, gestión de TFGs, usuarios, etc.).

También se configuraron las rutas principales mediante el módulo de enrutamiento de Angular (RouterModule) y se integraron servicios HTTP usando HttpClientModule.

Desarrollo del cliente (formulario y vistas)

Se construyeron formularios dinámicos y validados para que los usuarios puedan crear y editar recursos como TFGs, usuarios, departamentos y reservas.

Características implementadas:

- Validaciones en tiempo real (campos obligatorios, formatos, longitud mínima/máxima).
- Formularios reactivos con FormBuilder.
- Mensajes de error claros y estilizados para mejorar la experiencia del usuario.
- Asociación dinámica de campos dependientes (por ejemplo, seleccionar una carrera actualiza los TFGs disponibles).
- Formularios adaptados a cada rol: el alumno ve formularios distintos a los de un administrador.

Desarrollo de vistas de búsqueda y filtros

Se crearon vistas que permiten listar los elementos principales del sistema (TFGs, usuarios, reservas, departamentos), acompañadas de filtros dinámicos. Funcionalidades incluidas:

- Filtrado por campos como estado, nombre, carrera, departamento.
- Conexión directa con la API mediante servicios (UserService, TfgService, etc.), asegurando que los datos mostrados estén sincronizados con la base de datos en tiempo real.

Entregables:

Componentes funcionales de Angular conectados al backend

Se entregaron los primeros componentes funcionales conectados a los endpoints REST. Estos componentes permiten a los usuarios visualizar y gestionar información almacenada en la base de datos. Además, se implementó el sistema de carga de datos asincrónica (observables con RxJS).

Formularios con validaciones

Todos los formularios críticos fueron validados tanto en frontend como en backend. Esto garantiza que los datos introducidos cumplen con los requisitos del modelo de datos, y mejora la usabilidad al informar al usuario de forma clara sobre cualquier error.

Páginas de búsqueda con filtros activos

Las páginas de listado de TFGs, usuarios o reservas incluyen herramientas completas para buscar, filtrar y ordenar los resultados, lo que facilita una navegación eficiente y rápida por el sistema. Estas vistas están diseñadas para ser reutilizadas y adaptables, según el tipo de entidad que se esté gestionando.

1.4.1.5. Sprint 5: Estilo, adaptación, pruebas y documentación

Objetivo:

Finalizar la capa visual de la aplicación asegurando una experiencia de usuario cuidada, accesible y funcional en distintos dispositivos. Este sprint también se centró en verificar la estabilidad del sistema a través de pruebas exhaustivas y en generar la documentación necesaria para su uso, instalación y entrega final.

Tareas:

Desarrollo del cliente: estilos de la aplicación

En esta tarea se trabajó en la presentación estética y la coherencia visual del sistema.

Acciones realizadas:

- Definición de una paleta de colores corporativa consistente con el entorno académico.
- Aplicación de estilos CSS y SCSS personalizados para botones, formularios, menús y tablas.
- Incorporación de íconos y tipografías que mejoran la usabilidad.
- Mejora de la jerarquía visual de las vistas para facilitar la lectura y la navegación.
- Uso de Angular Material y clases personalizadas para mantener uniformidad.

El objetivo fue lograr una interfaz profesional, clara y visualmente agradable, sin sobrecargar al usuario con elementos innecesarios.

Diseño adaptativo (responsive)

Se adaptó toda la interfaz para que funcione correctamente en distintos dispositivos: ordenadores, tablets y móviles.

Técnicas utilizadas:

- Uso de Flex Layout, Grid y Media Queries para adaptar el contenido a distintos tamaños de pantalla.
- Comprobación del comportamiento de los componentes en resoluciones pequeñas, reorganizando elementos cuando era necesario.
- Optimización de menús, formularios y botones para pantallas táctiles.
- Simplificación de vistas en modo móvil para mejorar el rendimiento y la legibilidad.

Gracias a estas mejoras, TFGInfo puede utilizarse desde casa, en la universidad o desde dispositivos móviles, lo que mejora la accesibilidad para todos los perfiles de usuario.

Redactar la documentación final del proyecto:

Manual de usuario: describe cómo usar la aplicación según el rol (navegación, creación de recursos, gestión de TFGs, etc.).

Manual de instalación: explica paso a paso cómo clonar el repositorio, configurar la base de datos, lanzar el backend y arrancar el frontend localmente. Incluye requisitos técnicos, comandos y posibles errores comunes.

Entregables:

Interfaz visualmente cuidada y responsive

El sistema cuenta con una interfaz moderna, que intenta ser agradable y que se adapta a distintos tamaños de pantalla. Los elementos visuales están organizados y estilizados de forma coherente, cumpliendo estándares de usabilidad.

Sistema probado desde múltiples perfiles de usuario

Todas las funcionalidades han sido validadas desde el punto de vista de alumnos, profesores y administradores. Se confirmó que los permisos están correctamente aplicados y que cada usuario accede solo a las funciones que le corresponden.

Manual de usuario e instalación redactado

Se entregan ambos manuales con capturas, explicaciones paso a paso y lenguaje claro, facilitando tanto el uso del sistema como su despliegue en otros entornos o instituciones.

Proyecto listo para presentación y entrega

Al final del sprint, el proyecto se encuentra en un estado completamente funcional, probado, documentado y visualmente completo. Está preparado para su exposición ante el tribunal académico, así como para ser usado en entornos reales o ampliado en trabajos futuros.

1.5. Organización del documento

El presente documento se estructura en varios apartados que abarcan todo el proceso de desarrollo del proyecto TFGInfo, desde la concepción inicial hasta su implementación y validación.

En primer lugar, en el apartado de Tecnologías y Herramientas se describen las soluciones técnicas utilizadas para el desarrollo, incluyendo la base de datos, el servidor backend, la aplicación frontend, así como las herramientas de control de versiones, diseño y documentación empleadas durante el proyecto.

Posteriormente, en la sección de Especificación y Análisis, se presentan los requisitos funcionales y no funcionales de la aplicación, los casos de uso principales, el modelo de dominio y los diagramas de secuencia, junto con un análisis de riesgos que permitió anticipar y mitigar posibles problemas técnicos, organizativos y legales.

A continuación, en Diseño del Sistema, se aborda la concepción de la interfaz de usuario, los principios de usabilidad y accesibilidad aplicados, así como los bocetos y maquetas que guiaron el desarrollo de la aplicación final.

El apartado de Implementación detalla la estructura del proyecto a nivel técnico, incluyendo la base de datos, el backend y el frontend, además de describir las principales funcionalidades desarrolladas, los servicios auxiliares y el sistema de autenticación y autorización. También se exponen las pruebas realizadas sobre rendimiento, accesibilidad y comportamiento como aplicación web progresiva (PWA).

En la sección de Conclusiones y Trabajos Futuros, se recopilan los resultados alcanzados, destacando las aportaciones más relevantes del proyecto, así como posibles líneas de mejora y extensión de la aplicación para un uso más amplio en entornos universitarios.

Finalmente, los Apéndices incluyen un manual de instalación con los pasos necesarios para desplegar la aplicación en un entorno real y un manual de usuario que explica de manera detallada el uso de la herramienta según cada rol del sistema (administrador, profesor y estudiante).

2

Tecnologías y Herramientas

2.1. Base de Datos

Para el desarrollo de TFGInfo se ha optado por una base de datos relacional, concretamente MySQL, una de las tecnologías SQL más consolidadas, robustas y ampliamente utilizadas en el ámbito académico y profesional.

El sistema gestiona una serie de entidades con relaciones complejas y jerárquicas como estudiantes, profesores, TFGs, departamentos, carreras, universidades y grupos, lo que hace que un modelo relacional estructurado sea especialmente adecuado.

En momentos previos de la elección de base de datos se han estudiado en un primer lugar que tipo de base de datos se iba a utilizar. Sobre este asunto se han tenido en consideración las principales características de los dos tipos de bases de datos principales, SQL y NoSQL [4]. Para una aplicación que se basa mayoritariamente en las relaciones de sus entidades y el control de estas se ha optado por una base de datos SQL para garantizar la integridad de la información y de sus relaciones.

Entre las posibles bases de datos SQL se han contemplado para realizar este proyecto están MySql [5] y PostgreSQL [6]. Se ha optado por MySQL por la facilidad de instalación y la amplia documentación que existe al respecto.

La aplicación que se ha usado para la gestión de la base de datos es **MySQL Workbench**.

2.2. Servidor de Backend

Para el desarrollo de una API para la aplicación TFGinfo, se han contemplado varios lenguajes de programación con sus respectivos frameworks para la creación y gestión de APIs. Entre ellos Spring Framework [7] en Java, FastApi [8] en Python y .NET [9] para C#. Entre estas opciones se ha optado por .NET para C# ya que aporta las siguientes características:

- Una arquitectura robusta y modular que permite gestionar fácilmente funcionalidades como usuarios, TFGs, departamentos, reservas, etc., cada uno con su propia lógica bien encapsulada.

- Alto rendimiento y eficacia, .NET ofrece un rendimiento superior gracias a su motor optimizado, compilación anticipada y bajo consumo de memoria. Esto se traduce en una API ágil, capaz de responder a muchas solicitudes concurrentes sin degradar el rendimiento, algo ideal para una plataforma multiusuario como TFGInfo.

- Seguridad integrada, .NET ofrece una fácil gestión de la autenticación y autorización que resulta muy importante en una aplicación basada en roles (alumno, profesor y administrador)

- Entity Framework [10] es un mapeador objeto-relación que permite trabajar con entidades de forma sencilla segura y rápida. Proporciona facilidades para hacer consultas complejas.

2.2.1. Swagger

Durante el desarrollo de TFGInfo se ha utilizado Swagger como herramienta principal para la documentación y prueba de la API REST. Swagger, integrado a través de Swagger UI en el proyecto backend desarrollado con .NET, permite describir de manera estandarizada todos los endpoints de la aplicación, sus métodos disponibles (GET, POST, PUT, DELETE), los parámetros que requieren y las respuestas esperadas.

El uso de Swagger aporta varias ventajas al proyecto:

Documentación automática y actualizada: cada vez que se modifica o amplía un endpoint en el backend, Swagger genera de forma dinámica la documentación

correspondiente. Esto evita inconsistencias entre la implementación real y los manuales técnicos estáticos.

Interfaz interactiva: es posible realizar peticiones directamente desde la interfaz de Swagger UI, lo que facilita la validación de la API sin necesidad de herramientas externas.

Estandarización mediante OpenAPI: Swagger genera la especificación de la API en formato OpenAPI, lo que asegura que el proyecto cumpla con un estándar ampliamente reconocido, facilitando su integración con otras aplicaciones y sistemas.

En el contexto de TFGInfo, la documentación generada por Swagger describe cada uno de los controladores y sus operaciones: gestión de usuarios, asignación de TFGs, creación de reservas, administración de entidades académicas, entre otros. Cada operación cuenta con ejemplos de solicitud y respuesta en formato JSON.

Gracias a esta integración, el proyecto dispone de una documentación técnica clara, accesible y siempre alineada con la implementación, lo que constituye un soporte fundamental para futuros desarrolladores que deseen mantener, ampliar o integrar la aplicación en entornos reales.

2.3. Aplicación de Frontend

En la actualidad las tecnologías más usadas para el desarrollo de aplicaciones web están Angular [11] y React [12], ambas con los mismos lenguajes de programación: Typescript, para implementar la lógica que va a realizar un usuario de la

aplicación; HTML, para la interfaz que va a ver el usuario en la web; y, por último SCSS, para los estilos que va a tener la interfaz.

Con esos rasgos comunes la principal diferencia es lo que aporta cada framework. Finalmente se ha optado por usar Angular, usando Angular Material por las siguientes razones [13]:

- Se trata de un framework completo (no solo una biblioteca), lo que significa que incluye desde el enrutado hasta la gestión del estado, formularios, internacionalización, herramientas de pruebas y más, todo con una estructura clara y coherente.

- Promueve buenas prácticas desde el inicio: uso de componentes, servicios, inyección de dependencias, tipado con TypeScript, y separación de responsabilidades.

- Se integra perfectamente con APIs mediante HttpClient, ideal para consumir los endpoints expuestos por un backend en .NET.

Y por último las herramientas integradas que facilitan la creación de componentes, módulos, servicios, etc.

2.4. Desarrollo del código

El desarrollo del proyecto TFGInfo ha requerido el uso de herramientas modernas que facilitan la escritura, organización, control y mantenimiento del código fuente. A continuación, se describen las principales:

Visual Studio Code [14]

Visual Studio Code (VS Code) ha sido el entorno de desarrollo integrado (IDE) elegido para el desarrollo del frontend y del backend. Es una herramienta ligera, rápida, multiplataforma y altamente extensible mediante extensiones.

Se ha utilizado para:

- Editar archivos TypeScript, HTML y CSS en el frontend con Angular.
- Editar archivos C# y JSON en la configuración del backend con .NET.
- Gestionar proyectos mediante terminal integrada.
- Depurar código y realizar pruebas de forma eficiente.

Git y GitHub [15] [16]

Para el control de versiones se ha utilizado Git como sistema distribuido, junto con GitHub como plataforma de alojamiento y colaboración.

Git y GitHub han permitido:

- Controlar el historial de cambios del proyecto.
- Trabajar con ramas para implementar funcionalidades de forma ordenada.
- Documentar cada avance mediante mensajes de commit.
- Hacer copias seguras del repositorio y restaurar versiones anteriores en caso necesario.
- Gestionar versiones entregables.

2.5. Creación de diagramas y bocetos

2.5.1. Diagramas

Durante el desarrollo de TFGInfo, se ha utilizado **Mermaid [17]**, una herramienta web basada en texto, para generar de forma rápida y estructurada distintos tipos de diagramas útiles en documentación técnica y de software.

¿Qué es Mermaid?

Mermaid es un lenguaje de marcado ligero que permite crear diagramas a partir de sintaxis textual. Está especialmente orientado a desarrolladores y documentadores técnicos que necesitan incluir diagramas dinámicos y fácilmente editables en sus proyectos.

Ventajas de usar Mermaid

- **Sintaxis simple**: Se escribe con un lenguaje similar a Markdown, sin necesidad de herramientas gráficas complejas.
- **Versatilidad**: Permite generar diagramas de flujo, de clases, de casos de uso, de secuencia, Gantt, entre otros.
- **Visualización inmediata**: Puede visualizarse en navegadores mediante plataformas como mermaid.live o integrarse en editores como Visual Studio Code o herramientas como Notion, Obsidian o GitHub.
- **Mantenimiento fácil**: Al tratarse de código de texto, los diagramas pueden versionarse junto al código fuente y modificarse rápidamente.

En este proyecto se ha usado Mermaid para representar:

- Diagramas de flujo de procesos clave (inicio de sesión, solicitud de TFG, etc.).
- Diagramas de secuencia que muestran la interacción entre usuarios y el sistema.
- Modelos de dominio y relaciones entre entidades.
- Casos de uso y escenarios funcionales.

Gracias a Mermaid, ha sido posible documentar gráficamente el comportamiento y la estructura de la aplicación de forma clara, reproducible y sin depender de herramientas de diseño externas.

2.5.2. Bocetos

Para la creación de bocetos en TFGinfo se ha usado **Figma** [18] que se trata de una herramienta de diseño colaborativo en la nube que se ha convertido en un estándar para la creación de interfaces de usuario, prototipos y bocetos de aplicaciones web y móviles.

A diferencia de programas de diseño tradicionales que requieren instalación y almacenamiento local, Figma funciona directamente en la web, lo que facilita la colaboración en tiempo real sin importar el sistema operativo.

Características principales

- **Diseño de interfaces (UI)**

  Permite crear desde bocetos de baja fidelidad hasta diseños finales de alta resolución, usando un sistema de lienzo infinito, capas, componentes reutilizables y estilos globales.
- **Prototipado interactivo**

  No solo se crean las pantallas, sino que se pueden enlazar con interacciones, transiciones y animaciones para simular el flujo de navegación de una app.
- **Colaboración en tiempo real**

  Varias personas pueden trabajar simultáneamente en el mismo archivo, con un sistema de cursores en vivo similar a Google Docs, facilitando la co-creación y el feedback instantáneo.
- **Sistema de componentes**

Permite crear elementos reutilizables (botones, menús, iconos, etc.) que, al modificarse, actualizan automáticamente todas sus instancias en el proyecto.

- **Control de versiones y comentarios**

  Mantiene un historial de cambios y permite añadir comentarios directamente sobre los elementos del diseño, agilizando las revisiones.

Ventajas para el desarrollo de aplicaciones

- **Alineación entre diseño y desarrollo**: Figma genera especificaciones técnicas, medidas, colores y tipografías directamente para los desarrolladores, reduciendo malentendidos.
- **Accesibilidad multiplataforma**: Funciona en Windows, macOS, Linux (vía web) e incluso en tablets.
- **Integración con otras herramientas**: Compatible con plugins y exportación a formatos como SVG, PNG, PDF, además de integración con sistemas como Jira, Trello o Slack.

3

Especificación y análisis

3.1. Gestión de riesgos

El propósito de un análisis de riesgos es identificar, clasificar y mitigar los riesgos asociados al desarrollo e implementación de una aplicación centrada en la gestión de TFGs. Esta herramienta maneja datos sensibles, opera en un entorno multiusuario y debe integrarse con procesos institucionales, lo que introduce múltiples desafíos.

Riesgos de la seguridad de la información

Uno de los riesgos más importantes en este tipo de sistemas es la seguridad de los datos personales y académicos. La aplicación trata con información sensible, como nombres, correos institucionales, roles académicos y la asignación de Trabajos de Fin de Grado.

Riesgo	**Descripción**	**Impacto**	**Mitigación**
Acceso no autorizado	Usuarios sin permisos acceden a datos sensibles (TFGs, notas, perfiles)	Alto	Implementar autenticación por rol, tokens de acceso y cifrado de credenciales
Inyección SQL / XSS	Uso de formularios para ejecutar código malicioso	Alto	Mediante el uso de un ORM como Entity Framework para sanitizar datos
Pérdida de datos	Caída del servidor o corrupción de la base de datos	Muy alto	Copias de seguridad automáticas y redundancia
Exposición de datos personales	Mal manejo de datos de alumnos y profesores	Alto	Campos mínimos requeridos y acceso restringido

Tabla 1. Riesgos de la seguridad de la información

Riesgos técnicos

A nivel técnico, uno de los principales retos está en la complejidad del modelo de datos. La aplicación debe gestionar relaciones múltiples y cruzadas entre entidades como TFGs, estudiantes, profesores, líneas de investigación y departamentos. Si el modelo no está bien diseñado, pueden surgir problemas de integridad y eficiencia.

Riesgo	Descripción	Impacto	Mitigación
Complejidad de relaciones entre entidades	TFGs, alumnos, departamentos y profesores se relacionan de forma compleja	Medio	Modelado cuidadoso con diagramas ER y uso de claves foráneas
Fallos en la sincronización frontend-backend	Problemas de integración entre Angular y API .NET	Medio	Establecer comunicación usando objetos DTOs para asegurar la integridad
Falta de escalabilidad	Aumento de usuarios podría saturar la app	Medio	Usar buenas prácticas y arquitectura modular

Tabla 2. Riesgos Técnicos

Riesgos de gestión y planificación

Como en cualquier proyecto de desarrollo, existe el riesgo de subestimar los tiempos necesarios para cada tarea. Una mala planificación puede llevar a retrasos importantes. También es posible que, durante el desarrollo, los requisitos del sistema cambien por decisiones del cliente o descubrimientos funcionales.

Riesgo	Descripción	Impacto	Mitigación
Subestimación de tiempos	Tareas de desarrollo más largas de lo previsto	Medio	Planificación ágil (sprints), revisión frecuente de avances
Cambios en los requisitos	Los usuarios finales cambian su opinión sobre cómo debe funcionar la app	Medio	Reuniones regulares de revisión, diseño modular para adaptabilidad
Dependencia de datos de terceros	Problemas al importar datos desde CSV o sistemas externos	Alto	Validación previa de datos, limpieza automatizada de entradas

Tabla 3. Riesgos de gestión y planificación

Riesgos organizativos y de adopción

Aunque el sistema funcione correctamente, puede haber resistencia al cambio por parte del profesorado o personal administrativo. La falta de formación o el rechazo a abandonar métodos tradicionales (como hojas de cálculo o gestión por correo) puede obstaculizar la adopción de la herramienta.

Riesgo	**Descripción**	**Impacto**	**Mitigación**
Resistencia al cambio	Profesores o administrativos rehúsan usar la plataforma	Medio	Formación básica, interfaz amigable y documentación clara
Falta de formación	Los usuarios no saben cómo usar correctamente el sistema	Medio	Incluir un manual de usuario detallado y videotutoriales
Incompatibilidad con sistemas existentes	Falta de integración con bases de datos u otras herramientas institucionales	Alto	Uso de formatos estándar (CSV, JSON), diseño API-first para integraciones futuras

Tabla 4. Riesgos organizativos y de adopción

Riesgos legales

La aplicación debe cumplir con la legislación vigente en protección de datos, especialmente con el Reglamento General de Protección de Datos (GDPR) y la Ley Orgánica de Protección de Datos (LOPDGDD en España). Cualquier fallo en este aspecto, como almacenar datos personales sin consentimiento o no permitir su eliminación, puede tener consecuencias legales.

Riesgo	Descripción	Impacto	Mitigación
Incumplimiento de la LOPD/GDPR [19]	Almacenar datos personales sin base legal	Muy alto	Solicitar consentimiento explícito, limitar datos innecesarios, cifrado en reposo y en tránsito

Tabla 5. Riesgos legales

Desarrollar una aplicación de gestión universitaria implica gestionar riesgos técnicos, organizativos y legales. TFGInfo, como plataforma que maneja información académica y personal, debe diseñarse con una arquitectura segura, escalable y centrada en el usuario. La anticipación y mitigación de riesgos es clave para garantizar su éxito y adopción institucional.

La combinación de buenas prácticas de desarrollo, protección de datos y una estrategia de formación y soporte para los usuarios son los pilares para minimizar estos riesgos.

3.2. Análisis de requisitos

El análisis de requisitos es una etapa clave en el desarrollo de cualquier sistema de software, ya que define qué debe hacer la aplicación y cómo debe comportarse. En el caso de TFGInfo, una aplicación destinada a gestionar TFGs en el entorno universitario es fundamental identificar de forma clara las necesidades de cada tipo de usuario, así como los requisitos técnicos, de seguridad y de usabilidad.

TFGInfo debe permitir la gestión de estudiantes, profesores, departamentos, líneas de TFG, reservas y asignaciones, todo ello en un entorno multiusuario con distintos roles y niveles de acceso.

3.2.1. Historias de usuario - Requisitos funcionales

Los requisitos funcionales describen las funcionalidades específicas que el sistema debe ofrecer. Estos se derivan directamente de las tareas que deben realizar los distintos actores del sistema.

US01 - Gestión de usuarios

El sistema debe permitir registrar, editar y eliminar usuarios (alumnos, profesores).
Como administrador
Quiero poder registrar nuevos usuarios (alumnos, profesores, administradores)
Para proporcionar acceso a la aplicación a quienes lo necesiten

Cada usuario debe autenticarse con credenciales y un rol específico
Como administrador
Quiero poder asignar roles a los usuarios
Para que solo tengan acceso a las funciones correspondientes a su perfil

Los roles determinarán el acceso a las distintas funciones del sistema
Como usuario registrado
Quiero poder iniciar sesión con mis credenciales
Para acceder a mis funcionalidades personalizadas de forma segura

US02 - Gestión de Trabajos de Fin de Grado (TFG)

El sistema debe permitir crear, editar, buscar y eliminar registros de TFG.

Como administrador

Quiero poder crear, editar registros de TFG y asociar profesores y departamentos

Para proponer trabajos a los estudiantes y profesores

La lista de TFGs debe ser accesible para los usuarios, permitiendo consultar la información y solicitar una línea de TFG para su desarrollo.

Como estudiante

Quiero poder ver los TFG disponibles para mi titulación

Para elegir el que más se ajuste a mis intereses

Como profesor

Quiero poder ver los TFG disponibles a mi cargo

Para aceptar o rechazar las solicitudes de los estudiantes

US03 - Gestión de líneas temáticas y reservas

Los estudiantes podrán enviar solicitudes de reserva para una línea de TFG.

Como estudiante

Quiero poder enviar una solicitud de reserva de una línea de TFG

Para expresar mi interés en trabajar con un profesor específico

El sistema debe permitir al tutor aceptar o rechazar una solicitud.

Como profesor

Quiero poder aceptar o rechazar reservas de estudiantes

Para seleccionar a quienes considero más adecuados para el trabajo

El sistema notificará por correo electrónico el resultado de la solicitud.

Como sistema

Quiero enviar una notificación por correo electrónico al estudiante

Para informarle del estado de su solicitud de reserva

US04 - Gestión de departamentos, universidades y titulaciones

La plataforma debe permitir administrar departamentos y carreras, asociando profesores y TFGs a cada uno.

Como administrador

Quiero poder crear y administrar departamentos y titulaciones

Para reflejar la estructura académica real dentro de la aplicación

Debe existir una estructura jerárquica donde las titulaciones pertenezcan a universidades y los profesores a departamentos.

Como profesor

Quiero estar asociado a un departamento

Para que mi perfil refleje correctamente mi afiliación institucional

US05 - Gestión de grupos de trabajo

El sistema permitirá la creación de grupos de alumnos que comparten un mismo TFG.

Como estudiante

Quiero poder formar un grupo de trabajo con otros compañeros

Para colaborar en un mismo TFG

Los grupos podrán tener asignaciones de profesores y TFG asociados.

Como profesor

Quiero poder asignar un grupo de estudiantes a un TFG

Para dirigirlos de forma conjunta en el desarrollo del proyecto

US06 - Comunicación interna básica

El sistema debe enviar correos electrónicos automáticos para notificar acciones como aceptación/rechazo de TFG, confirmaciones de asignación, etc.

Como sistema

Quiero enviar correos automáticos al aceptar o rechazar una solicitud

Para mantener informados a estudiantes y profesores sin intervención manual

3.2.2. Historias de usuario – Requisitos no funcionales

Seguridad

Toda la información debe transmitirse de forma segura (HTTPS).

Como usuario de la aplicación

Quiero que todas las comunicaciones se realicen de forma segura

Para garantizar que mis datos personales no puedan ser interceptados

Los datos personales deben estar protegidos cumpliendo con el RGPD.

Como responsable del sistema

Quiero que los datos personales estén protegidos según el RGPD

Para cumplir con la legislación y proteger la privacidad de los usuarios

El sistema debe implementar autenticación y autorización basada en roles.

Como administrador del sistema

Quiero que se implementen roles y permisos de acceso

Para asegurar que cada usuario solo pueda ver y hacer lo que le corresponde

Rendimiento

Las búsquedas y cargas de datos deben ser rápidas, incluso con grandes volúmenes de información.

Como usuario

Quiero que las búsquedas y cargas de información sean rápidas

Para poder trabajar de forma fluida sin interrupciones

El sistema debe implementar filtros para optimizar el rendimiento.

Como usuario

Quiero que se permita hacer un filtrado en las búsquedas

Para realizar consultas más rápidas y directas

Escalabilidad

La arquitectura debe permitir añadir nuevas funcionalidades sin reestructurar el núcleo.

Como desarrollador del sistema

Quiero que la arquitectura esté diseñada de forma modular

Para poder añadir nuevas funcionalidades sin rehacer el sistema completo

Usabilidad

La interfaz debe ser intuitiva y accesible para usuarios con distintos niveles técnicos.

Como usuario sin conocimientos técnicos

Quiero que la interfaz sea clara e intuitiva

Para poder usar la aplicación sin necesidad de formación previa

Debe adaptarse a dispositivos móviles (diseño responsive).

Como estudiante o profesor

Quiero poder acceder desde mi móvil o tablet

Para gestionar tareas en cualquier momento y lugar

Mantenibilidad

El código debe estar organizado en módulos fácilmente actualizables.

Como desarrollador

Quiero que el código esté organizado en módulos bien definidos

Para poder mantenerlo, depurarlo o ampliarlo fácilmente

Se debe mantener una documentación técnica y funcional clara.

Como equipo técnico

Quiero contar con documentación técnica y funcional

Para asegurar la continuidad del proyecto en futuras versiones

Compatibilidad

El sistema debe funcionar en navegadores modernos (Chrome, Firefox, Edge).

Como usuario

Quiero que la aplicación funcione correctamente en los navegadores más comunes

Para poder acceder desde cualquier entorno sin problemas

3.3. Casos de uso

Los casos de uso describen de forma estructurada cómo interactúan los distintos actores del sistema TFGInfo con las funcionalidades principales de la aplicación. Esta sección tiene como objetivo detallar los flujos de comportamiento esperados, mostrando cómo profesores, estudiantes y administradores utilizan el sistema para realizar tareas clave relacionadas con la gestión de Trabajos de Fin de Grado.

Cada caso de uso parte de una acción concreta iniciada por un actor (como "iniciar sesión", "enviar solicitud", "aceptar TFG" o "crear una entidad"), e incluye los pasos principales que realiza el sistema, así como las decisiones y resultados posibles.

Los casos de uso abordados incluyen acciones fundamentales como:

- **Inicio de sesión y autenticación de usuarios.**
- **Consulta, filtrado y solicitud de TFGs por parte del estudiante.**
- **Aceptación o rechazo de solicitudes de TFG por parte del profesor.**
- **Envío de mensajes por parte del profesor a grupos de trabajo.**
- **Creación de entidades institucionales (universidades, departamentos, carreras)** por parte del administrador.
- **Acceso a canales de trabajo y comunicación grupal entre profesores y alumnos.**

Estos diagramas de flujo ayudan a visualizar el comportamiento del sistema y sirven como guía para el desarrollo, las pruebas y la validación del software. Además, permiten identificar posibles mejoras o extensiones futuras de la plataforma.

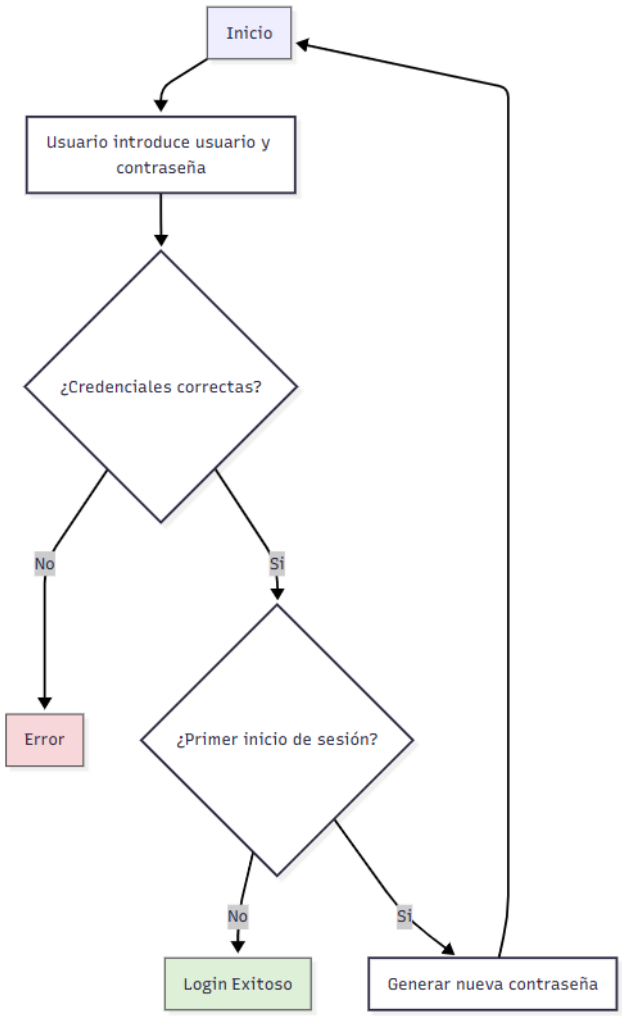


Ilustración 1. Diagrama de casos de uso - Inicio de sesión

Caso de Uso: Inicio de sesión (Historia de usuario US01)

Actor principal: Usuario (estudiante, profesor o administrador)

Objetivo: Acceder al sistema de forma segura mediante credenciales personales.

Resumen:

Este caso de uso describe el proceso mediante el cual un usuario registrado accede a la plataforma TFGInfo introduciendo su correo electrónico y contraseña. El sistema valida las credenciales contra la base de datos. Si son correctas, el usuario es redirigido a su panel correspondiente según su rol. En caso de error, el sistema muestra un mensaje de autenticación fallida y permite reintentar el acceso.

Precondiciones:

- El usuario debe estar registrado previamente en el sistema.
- Debe disponer de un correo electrónico y una contraseña válidos.

Flujo principal:

1. El usuario accede a la página de inicio de sesión.
2. Introduce su correo y contraseña.
3. El sistema envía los datos al backend para validación.
4. Si las credenciales son válidas, se genera un token de acceso y se redirige al panel correspondiente.
5. Si no son válidas, se muestra un mensaje de error.

Postcondición:

El usuario queda autenticado en el sistema y puede acceder a las funcionalidades correspondientes a su perfil.

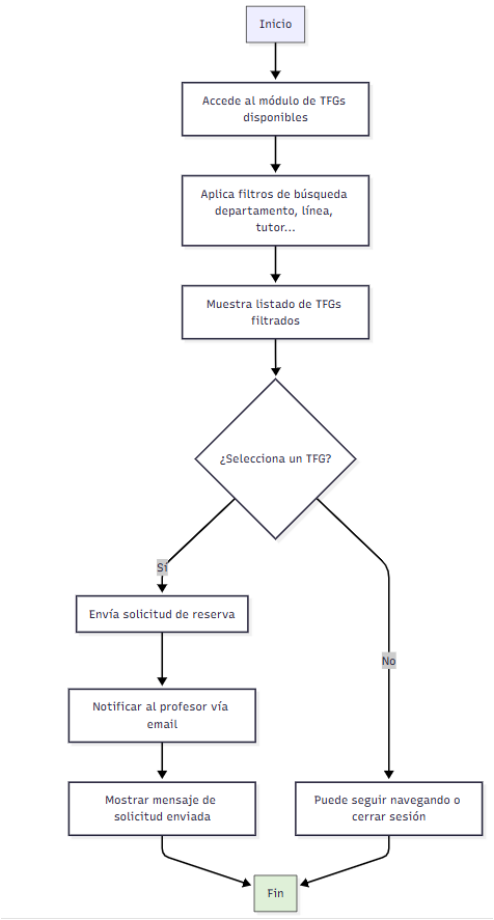


Ilustración 2. Diagrama de casos de uso - Solicitud de TFG

Caso de Uso: Solicitud de TFG (Historia de usuario US03)

Actor principal: Estudiante

Objetivo: Enviar una solicitud formal para reservar un Trabajo de Fin de Grado (TFG) de su interés

Resumen:

Este caso de uso describe el proceso mediante el cual un estudiante, tras iniciar sesión en la plataforma TFGInfo, accede al listado de TFGs disponibles, aplica filtros de búsqueda, consulta los detalles de un TFG y realiza una solicitud para reservarlo. El sistema registra la solicitud y notifica automáticamente al profesor responsable. La solicitud queda pendiente de aprobación.

Precondiciones:

- El estudiante debe estar registrado y autenticado en el sistema.
- Debe existir al menos un TFG disponible asociado a su titulación.

Flujo principal:

1. El estudiante inicia sesión y accede al módulo de TFGs.
2. Filtra y consulta las líneas de TFG disponibles.
3. Selecciona un TFG que le interese.
4. Envía una solicitud de reserva desde el sistema.
5. El sistema guarda la solicitud y notifica al profesor responsable por correo electrónico.
6. Se muestra un mensaje de confirmación al estudiante.

Postcondición:

La solicitud queda registrada como pendiente y el profesor podrá aceptarla o rechazarla posteriormente.

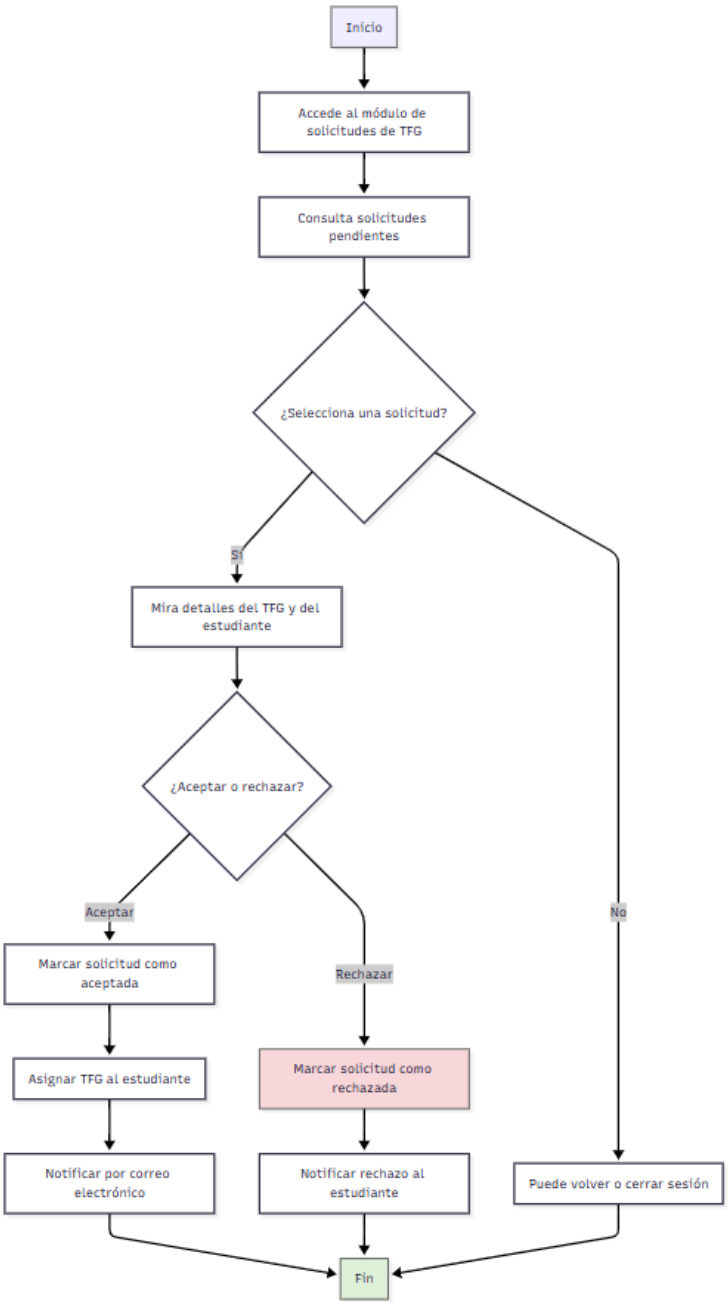


Ilustración 3. Diagrama de casos de uso - Gestión de solicitudes de TFG

Caso de Uso: Gestión de solicitudes de TFG (Historia de usuario US03)

Actor principal: Profesor

Objetivo: Gestionar las solicitudes recibidas por parte de estudiantes para reservar un Trabajo de Fin de Grado (TFG).

Resumen:

Este caso de uso describe cómo un profesor, una vez autenticado en la plataforma, accede al módulo de gestión de solicitudes, revisa las peticiones realizadas por

estudiantes para optar a uno de sus TFGs, y decide aceptarlas o rechazarlas. El sistema actualiza el estado de la solicitud y notifica automáticamente al estudiante por correo electrónico.

Precondiciones:

- El profesor debe estar registrado y autenticado.
- Debe haber al menos una solicitud pendiente de TFG en la que él sea tutor asignado.

Flujo principal:

1. El profesor inicia sesión y accede al módulo de solicitudes.
2. Revisa la lista de solicitudes pendientes asociadas a sus TFGs.
3. Consulta los detalles del estudiante y del TFG solicitado.
4. Decide aceptar o rechazar la solicitud.
5. El sistema actualiza el estado de la solicitud.
6. Se envía una notificación automática al estudiante con la resolución.

Postcondición:

La solicitud queda registrada como aceptada o rechazada, y el estudiante es informado del resultado.

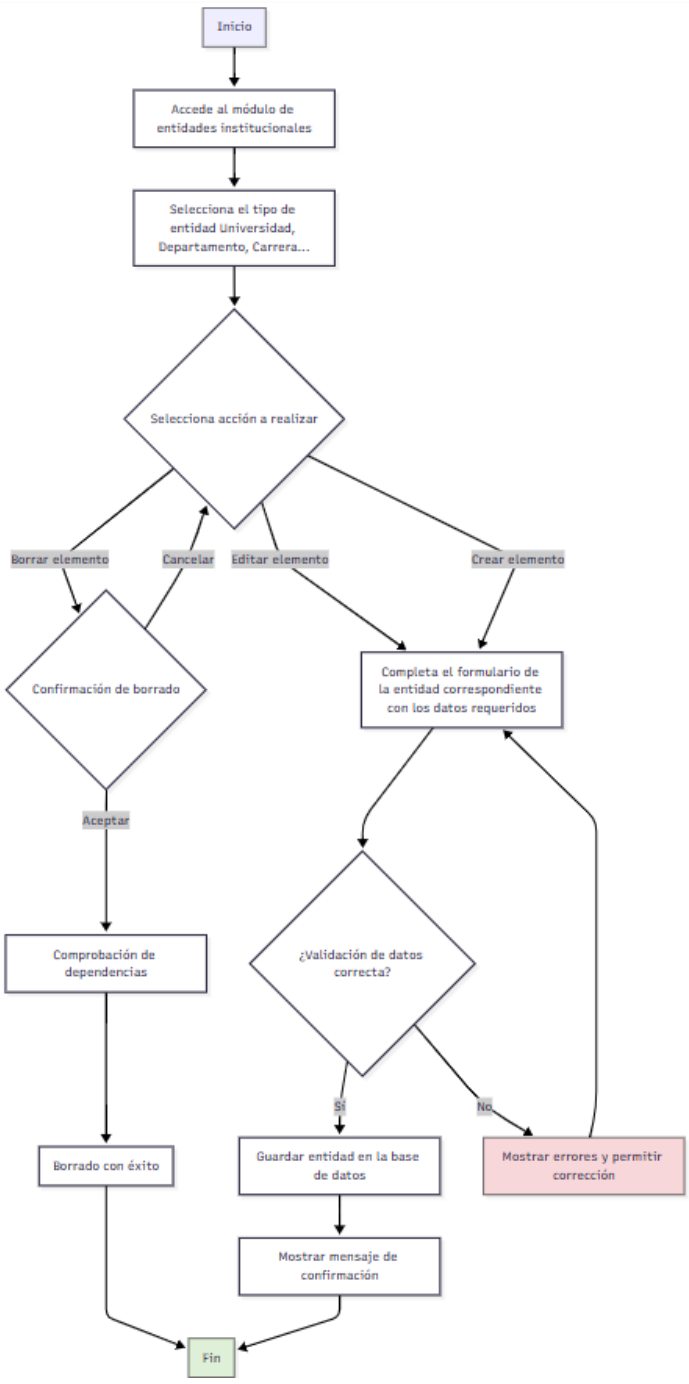


Ilustración 4. Diagrama de casos de uso - Administración de entidades

Caso de Uso: Administración de entidades (Historia de usuario US04)

Actor principal: Administrador

Objetivo: Registrar nuevas entidades académicas dentro del sistema, como universidades, departamentos o titulaciones.

Resumen:

Este caso de uso describe cómo un administrador puede acceder al módulo de gestión institucional y crear nuevas entidades que forman parte de la estructura académica. Estas entidades son necesarias para organizar correctamente los TFGs, los usuarios y sus relaciones jerárquicas. El sistema valida los datos introducidos y guarda la información en la base de datos.

Precondiciones:

- El administrador debe estar autenticado.
- Debe tener permisos de gestión estructural.

Flujo principal:

1. El administrador inicia sesión y accede al módulo de entidades institucionales.
2. Selecciona el tipo de entidad a crear: universidad, departamento o carrera.
3. Rellena el formulario correspondiente con los datos necesarios (nombre, siglas, jerarquía, etc.).
4. El sistema valida los campos introducidos.
5. Si los datos son correctos, se registra la nueva entidad en la base de datos.
6. Se muestra una confirmación del alta exitosa.

Postcondición:

La nueva entidad queda registrada y disponible para su uso en la configuración de usuarios, TFGs y relaciones institucionales.

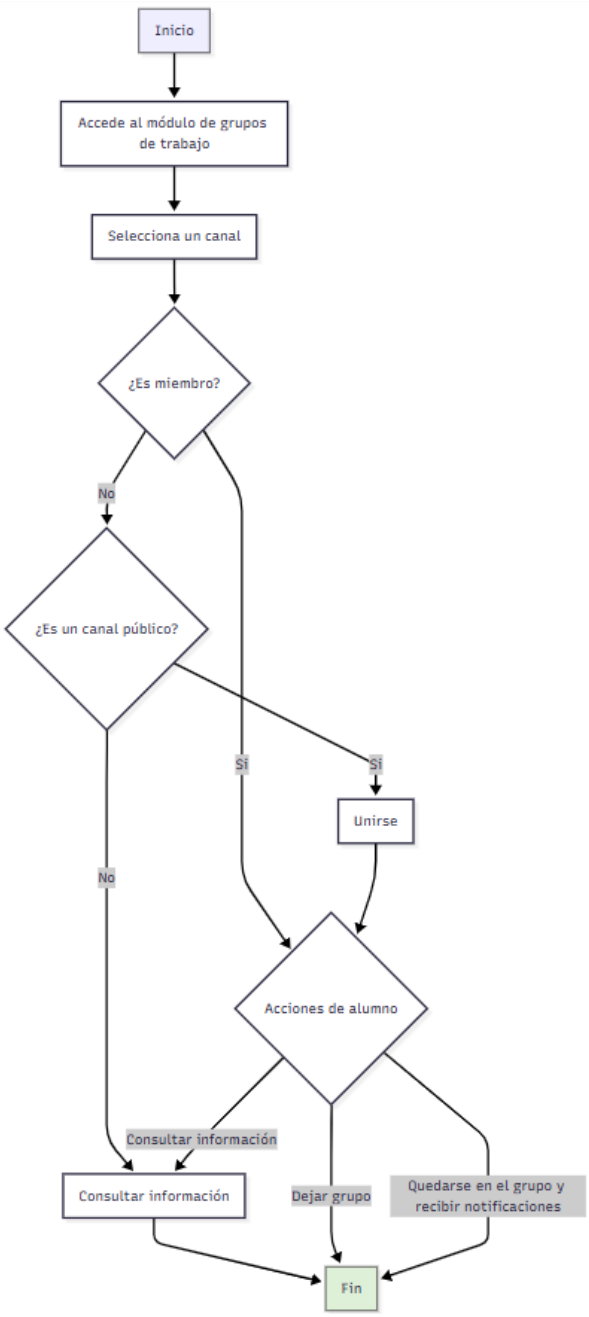


Ilustración 5. Diagrama de casos de uso - Estudiante forma parte de un canal

Caso de Uso: Estudiante forma parte de un canal (Historia de usuario US05)

Actor principal: Estudiante

Objetivo: Formar parte de un canal.

Resumen:

Este caso de uso describe cómo un estudiante puede visualizar, unirse o formar parte de un canal. El canal permite que varios alumnos colaboren en el mismo

proyecto bajo la dirección de uno o varios tutores. La aplicación gestiona las acciones disponibles desde el punto de vista del alumno.

Precondiciones:

- El estudiante debe estar registrado y autenticado en el sistema.
- Debe haber sido invitado o asignado a un grupo, o tener habilitada la opción de unirse a uno.

Flujo principal:

1. El estudiante accede a su panel personal.
2. Entra en la sección de grupos de trabajo.
3. Consulta los grupos disponibles o ya asignados.
4. Puede unirse a un canal (si está público).
5. Una vez dentro del canal, accede a la información compartida: integrantes, TFG asignado, mensajes del profesor, fechas clave, etc.

Postcondición:

El estudiante queda formalmente vinculado a un canal y puede recibir notificaciones de los profesores que se encuentren en él.

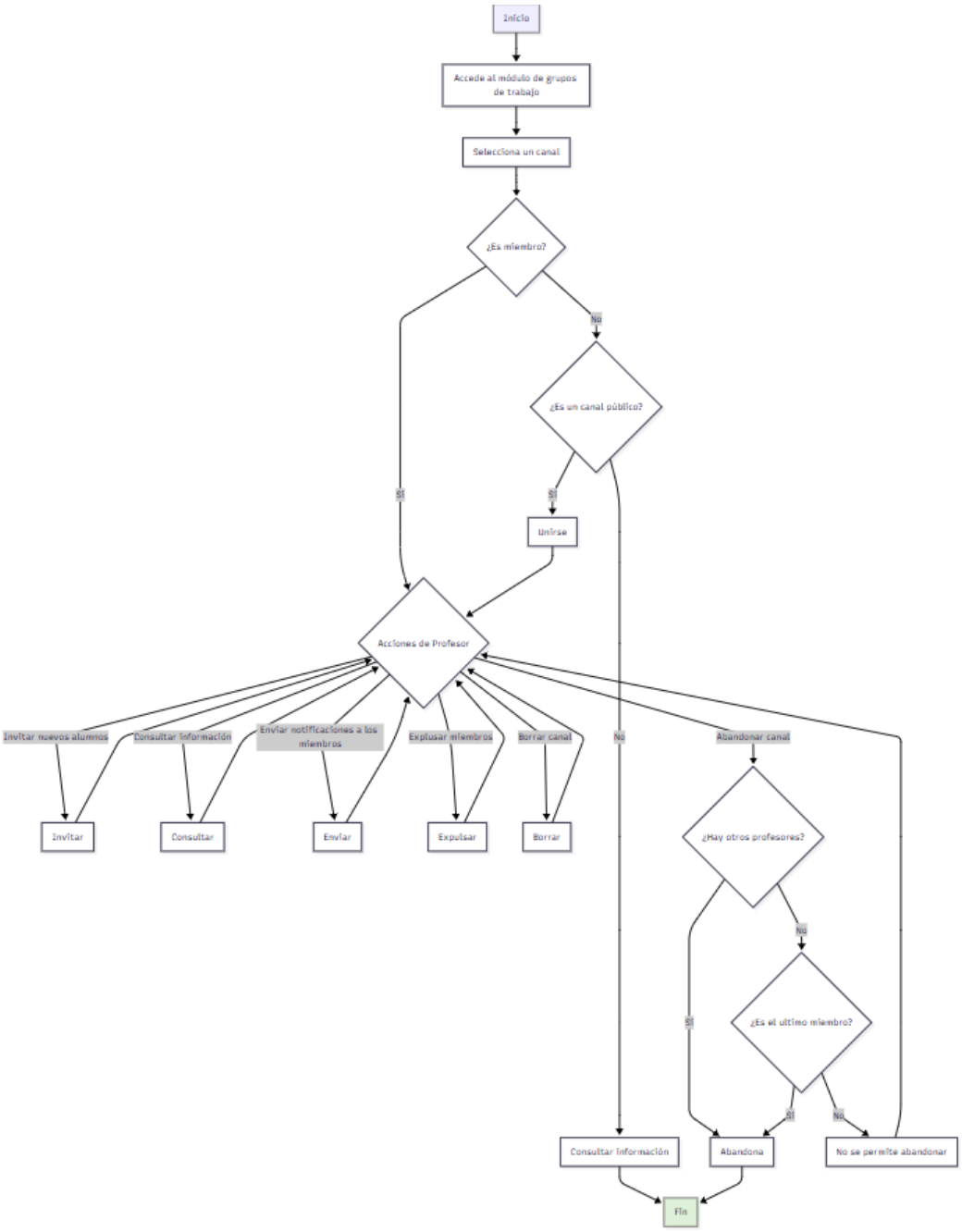


Ilustración 6. Diagrama de casos de uso - Gestión de un canal por parte de un profesor

Caso de Uso: Gestión de un Canal de Trabajo por parte del Profesor (Historia de usuario US05)

Actor principal: Profesor

Objetivo: Administrar un canal de trabajo en el que se agrupan estudiantes asignados a un mismo TFG.

Resumen:

Este caso de uso describe cómo un profesor puede gestionar un canal de trabajo

académico. Si el profesor ya es miembro del canal (o accede a uno público), podrá realizar varias acciones sobre los estudiantes o sobre el canal en sí, incluyendo enviar notificaciones, consultar información, invitar nuevos integrantes, expulsar miembros o incluso eliminar el canal. También puede decidir abandonarlo, si las condiciones lo permiten.

Precondiciones:

- El profesor debe estar autenticado.
- Debe ser miembro del canal, o el canal debe ser público.

Flujo principal:

1. El profesor inicia sesión y accede al módulo de grupos de trabajo.
2. Selecciona un canal de la lista.
3. Si es miembro, o si el canal es público, entra al canal.
4. Una vez dentro, puede realizar las siguientes acciones:
    - **Consultar información del grupo** (integrantes, mensajes, TFG asignado).
    - **Enviar notificaciones** o mensajes al grupo.
    - **Invitar nuevos alumnos** al canal.
    - **Expulsar miembros** del grupo.
    - **Borrar el canal** (si es propietario o único responsable).
    - **Abandonar el canal**, siempre que no sea el único miembro con rol de profesor.
5. Si intenta abandonar el canal y no hay otros profesores o es el último miembro, se bloquea la salida por motivos de integridad.

Postcondición:

El profesor ha gestionado correctamente su canal y sus miembros, manteniendo el flujo de comunicación y supervisión del grupo de trabajo. Si decide abandonarlo, se asegura que no se comprometa la gestión académica del grupo.

3.4. Modelo de dominio

El modelo de dominio constituye la base conceptual de la aplicación TFGInfo, ya que representa las entidades principales del sistema y las relaciones existentes entre ellas. Su diseño permite reflejar de forma estructurada la realidad académica en la que se enmarca la gestión de los TFGs.

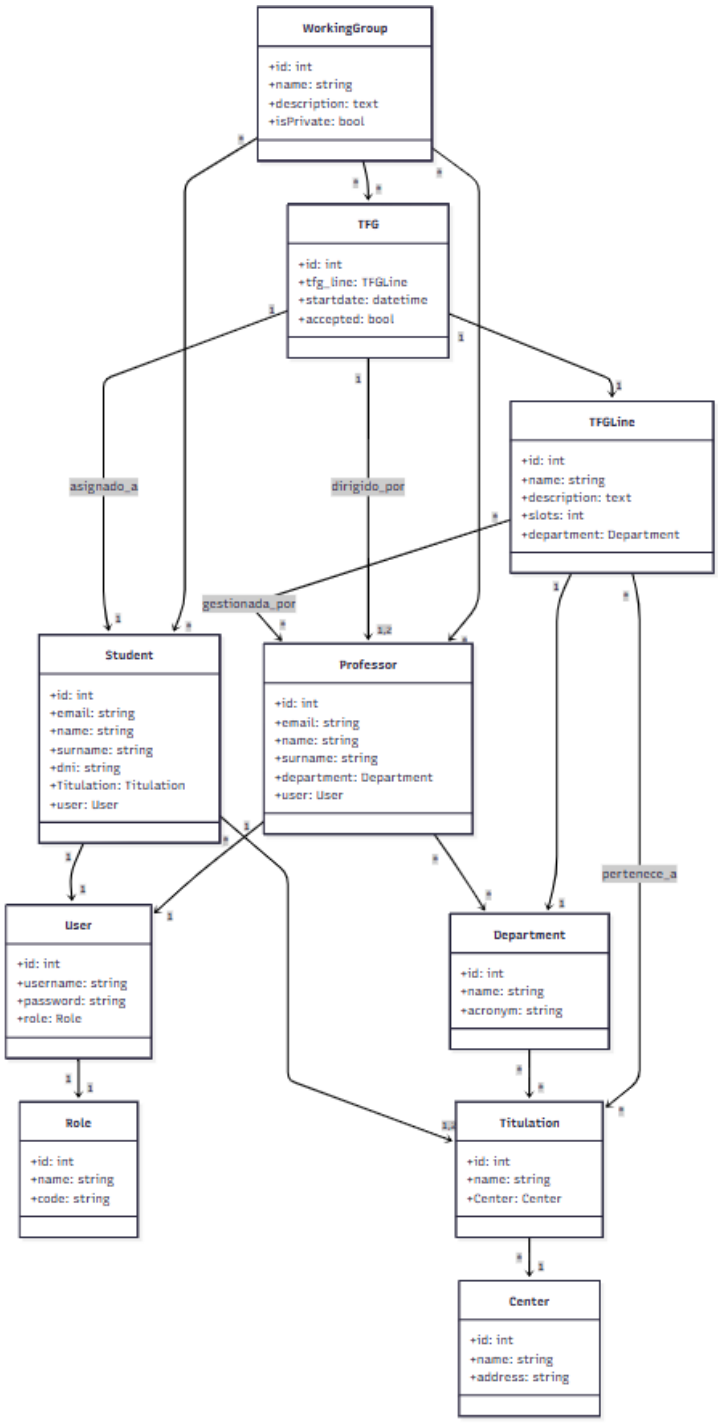


Ilustración 7. Modelo de Dominio

3.5. Diagramas de secuencia

Registro de un usuario

El diagrama de secuencia de la ilustración 8 representa el flujo completo del proceso de registro de una nueva persona en TFGInfo. Este proceso puede ser iniciado por un administrador que crea el usuario manualmente, o por un sistema de preinscripción.

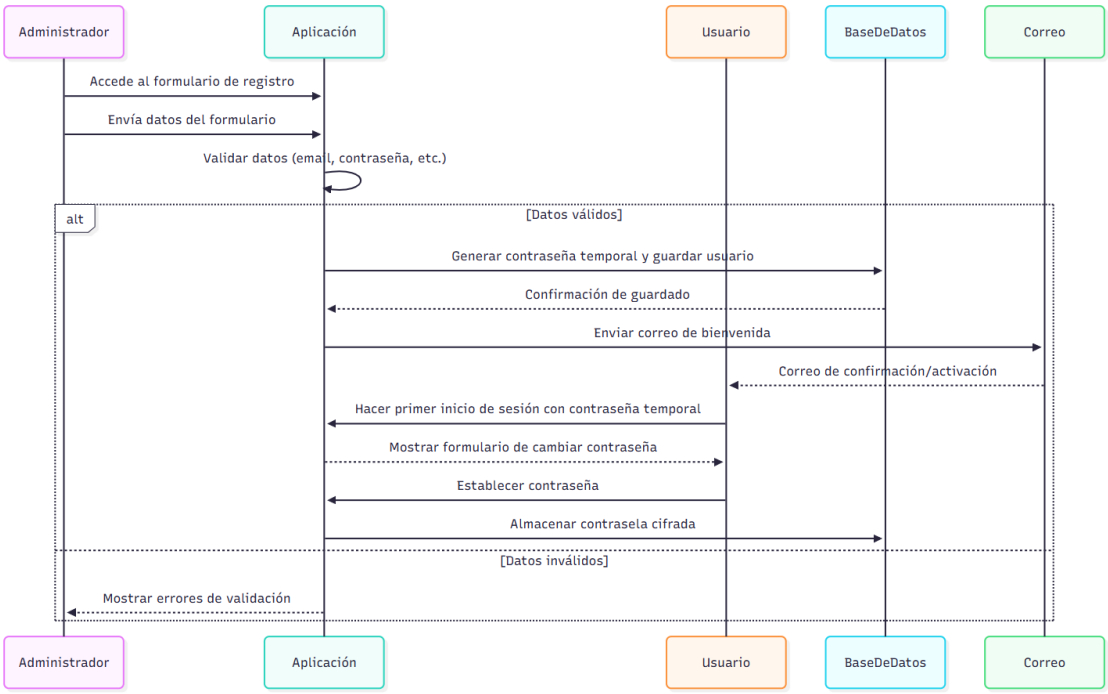


Ilustración 8. Diagrama de secuencia - Registrar un nuevo usuario

Comenzar TFG

El diagrama de secuencia de la ilustración 9 describe el proceso mediante el cual un estudiante solicita comenzar un trabajo de fin de grado y el profesor decide si aceptarlo o rechazarlo.

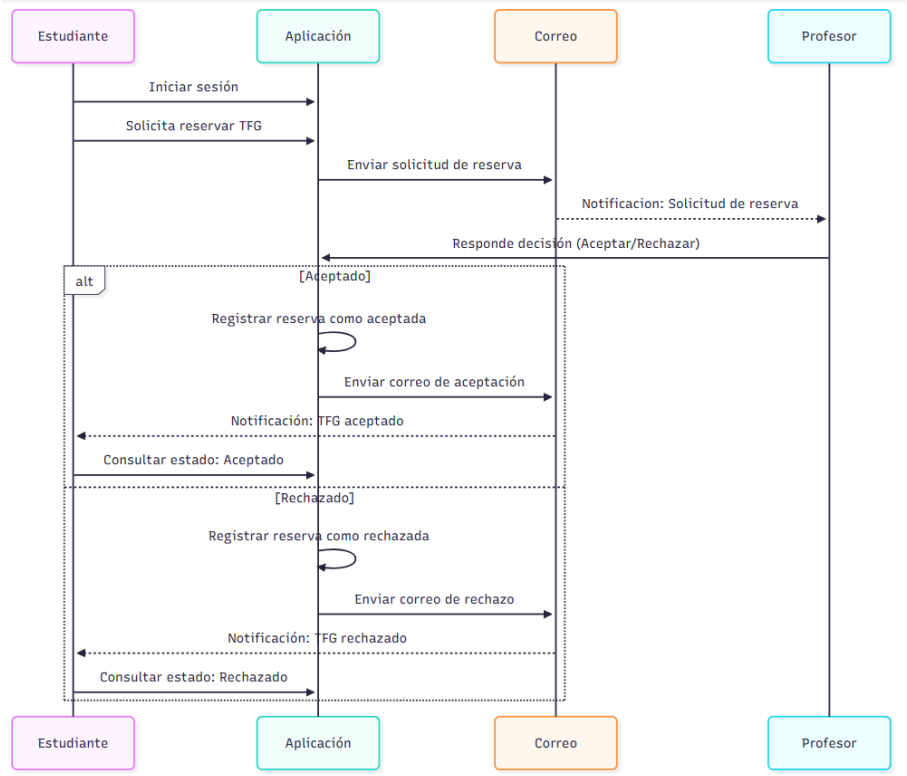


Ilustración 9. Diagrama de secuencia - Comenzar TFG

Envío de correo por canal

El diagrama de la ilustración 10 muestra como un profesor perteneciente a un canal puede notificar a todos los estudiantes que estén en este.

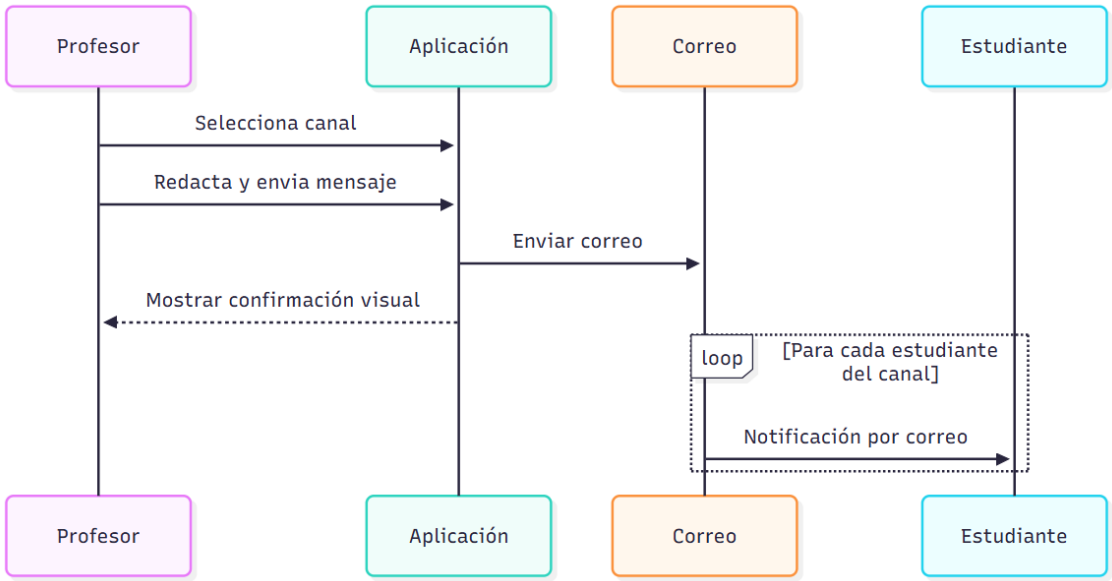


Ilustración 10. Diagrama de secuencia - Envío de mensajes por canal

4

Diseño del sistema

4.1. Base de datos

La base de datos de este proyecto está diseñada con un modelo relacional bien estructurado, pensado para reflejar de forma fiel las entidades del dominio y sus relaciones. Está implementada en MySQL y organizada de manera que mantiene la integridad referencial, evita la redundancia de datos y permite escalabilidad para futuras ampliaciones.

Modelo de datos y organización

El esquema incluye tablas que representan los elementos clave del sistema, como universidades, departamentos, titulaciones, profesores, estudiantes, trabajos de

fin de grado (TFG) y grupos de trabajo. Cada tabla está relacionada con las demás de forma coherente, permitiendo consultas complejas sin perder consistencia.

Algunos ejemplos relevantes:

- **university**: almacena la información de cada universidad, con campos como nombre y dirección.
- **department**: vinculado a la universidad, representa las áreas académicas (informática, robótica, etc.).
- **career**: contiene las titulaciones o programas de estudios, enlazados a una universidad concreta.
- **professor** y **student**: entidades que representan a usuarios académicos, con datos personales, identificación única (DNI) y referencia a su rol en la plataforma.
- **tfg** y **tfg_line**: núcleo del sistema para la gestión de trabajos de fin de grado, permitiendo asignar líneas de investigación, profesores, estudiantes y cupos de plazas.
- **Tablas intermedias** (tfg_line_career, tfg_line_professor, tfg_student, working_group_professor, etc.) que gestionan relaciones muchos a muchos de manera normalizada.

Integridad referencial

Todas las relaciones están reforzadas con claves foráneas (FOREIGN KEY) que garantizan que los datos vinculados existan y estén sincronizados. Por ejemplo:

- Un profesor debe pertenecer a un departamento válido.
- Un estudiante debe estar asociado a una titulación existente.
- Un TFG no puede existir sin una línea de TFG previamente creada.

Esto evita datos huérfanos y asegura que cualquier eliminación o actualización respete las dependencias.

Normalización y eficiencia

El modelo sigue principios de normalización para reducir la duplicación de información:

- Datos como el nombre de una universidad se almacenan **una sola vez** en su tabla y se relacionan con otras entidades por identificadores.
- Las tablas intermedias permiten gestionar relaciones complejas sin repetir información redundante.

Además, se utilizan **índices** (INDEX) en campos de búsqueda frecuente (como acrónimos, códigos, claves externas) para acelerar las consultas y mejorar el rendimiento.

Uso junto a Entity Framework

Aunque la base de datos se define y su contenido se puede rellenar mediante scripts SQL, su interacción principal con la aplicación se hace a través de Entity Framework. Esto significa que:

- la estructura de la base de datos está representada en el código como clases y colecciones (DbSet), lo que facilita trabajar con los datos sin escribir SQL directamente;
- las relaciones entre tablas se reflejan automáticamente como propiedades de navegación, lo que simplifica mucho las operaciones de lectura y escritura;
- cualquier cambio en el modelo de datos puede aplicarse mediante migraciones controladas, manteniendo sincronizados el código y la base de datos.

Ventajas de este diseño

- Consistencia de la información gracias a las restricciones y claves foráneas.
- Flexibilidad para añadir nuevas entidades o relaciones sin romper el modelo existente.
- Rendimiento optimizado con índices y estructura normalizada.
- Escalabilidad para soportar un mayor volumen de datos sin comprometer la integridad.
- Integración fluida con el backend a través de Entity Framework, minimizando errores y simplificando el desarrollo.

En resumen, la base de datos de este proyecto no solo está pensada para almacenar datos, sino también para garantizar calidad, consistencia y rendimiento a largo plazo, funcionando como un pilar estable que soporta tanto la lógica del backend como la experiencia del usuario en el frontend.

4.2. Interfaz de la aplicación

En el desarrollo de cualquier sistema software, la usabilidad y la interfaz de usuario no deben considerarse elementos secundarios o accesorios, sino componentes fundamentales del producto final. De hecho, el éxito o fracaso de una aplicación no suele depender únicamente de lo potente que sea su backend o de lo bien estructurada que esté su base de datos, sino de la experiencia que ofrece a sus usuarios cuando la utilizan. Las características que se han buscado a la hora de diseñar los primeros bocetos de la interfaz han sido los siguientes:

Diseño limpio y minimalista

- La interfaz usa un diseño minimalista, con un fondo blanco y elementos bien espaciados, que favorecen la lectura y reducen la sobrecarga cognitiva.

- Las líneas y bordes suaves transmiten una sensación de modernidad y profesionalismo, evitando distracciones.

Estructura clara y jerarquía visual

- La pantalla está dividida en un menú superior de navegación y un cuerpo principal con la vista actual, logrando una jerarquía muy clara. Manteniendo el menú superior constante durante toda la navegación de la aplicación
- En las vistas de listados cada entidad se presenta en una tarjeta horizontal con campos clave (nombre, departamento, profesores, plazas), lo que permite comparar rápidamente las opciones disponibles.

Funcionalidad intuitiva

- Los listados incluyen un campo de búsqueda bien destacado y un icono de filtros avanzados, que son acciones esperadas por los usuarios para mejorar la experiencia en la exploración de los elementos de la aplicación.
- Los iconos son reconocibles y estándar (persona para el perfil, embudo para filtro, avión de papel para enviar/solicitar), reforzando la comprensión sin necesidad de textos adicionales.

Coherencia y consistencia

- Todos los elementos mantienen un estilo coherente: misma tipografía, colores uniformes y distribución homogénea, lo que genera una experiencia de uso fluida.
- Las columnas están alineadas correctamente, facilitando la lectura horizontal de los datos.

Navegación sencilla

- En la parte superior, la barra de navegación incluye accesos directos a secciones clave como Mis reservas, TFG, Grupos de estudio y Profesores,

así como un icono de usuario para el perfil y otro de menú desplegable de idioma.

- Esto facilita que cualquier usuario pueda desplazarse de manera rápida a otras funcionalidades sin perderse.

Preparada para la adaptabilidad

- Se ha adaptado todas las vistas para cualquier tipo de dispositivo, permitiendo que los usuarios puedan acceder a la aplicación desde cualquier dispositivo y visualizarla con claridad independientemente del tamaño de la pantalla.

4.2.1. Bocetos de la aplicación

Los bocetos de la aplicación han supuesto un elemento clave a la hora del desarrollo de esta ya que permiten partir de una base concreta. Al hacer bocetos sobre la interfaz se han podido detectar posibles errores de usabilidad en momentos previos al desarrollo de la aplicación y, de esta forma, se ha minimizado el tiempo para la corrección de errores. A continuación, se explicarán las diferencias principales entre las interfaces finales y los primeros bocetos.

Resultado Final Inicio de sesión **Boceto Inicio de sesión**

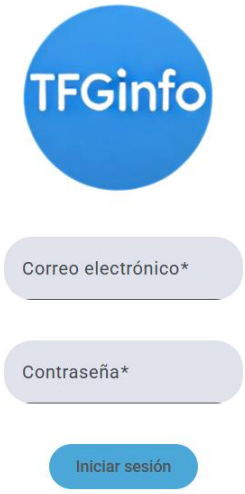


Ilustración 11. Comparativa del inicio de sesión con su boceto

En la ilustración 11 se muestran la interfaz final y el boceto inicial de la ventana de inicio de sesión. En estas vistas la diferencia es muy sutil, ya que tras el desarrollo de la aplicación y el análisis de requisitos se ha eliminado la posibilidad de que un usuario pueda registrarse por sí mismo en la aplicación. Además se ha optado por un estilo más minimalista para el formulario de inicio de sesión.

Boceto de listado de TFGs

Ilustración 12. Boceto del listado de TFGs

Resultado final listado de TFGs

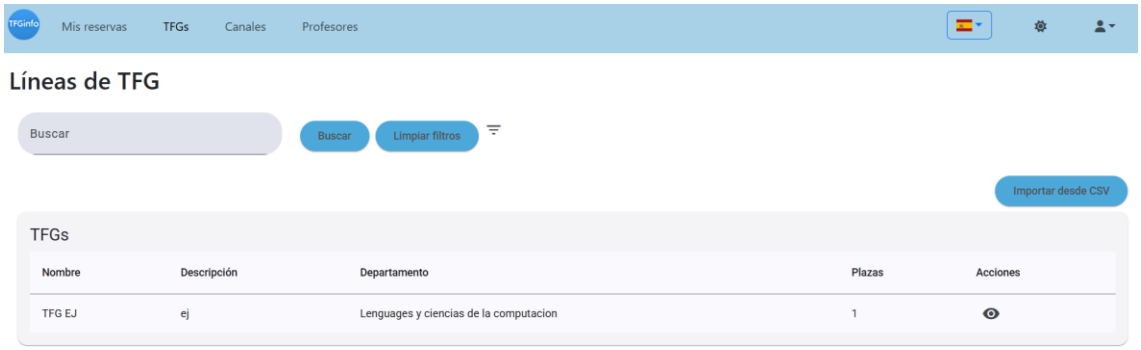


Ilustración 13. Resultado final del listado de TFGs

En las vistas de listado (ilustraciones 12 y 13) se sigue el formato para la búsqueda de entidades, se han agregado el botón de buscar y de limpiar filtros. Sobre el menú superior se ha optado por darle un color diferente al color de fondo habitual en el resto de las vistas, para buscar que destaque por encima ya que se trata de un componente que se va a mostrar constante en la aplicación.

Boceto perfil de un alumno

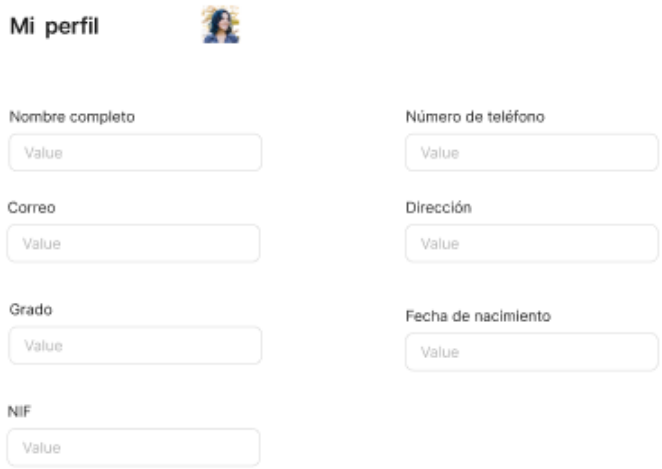


Ilustración 14. Boceto de perfil de un alumno

Resultado final perfil de un alumno

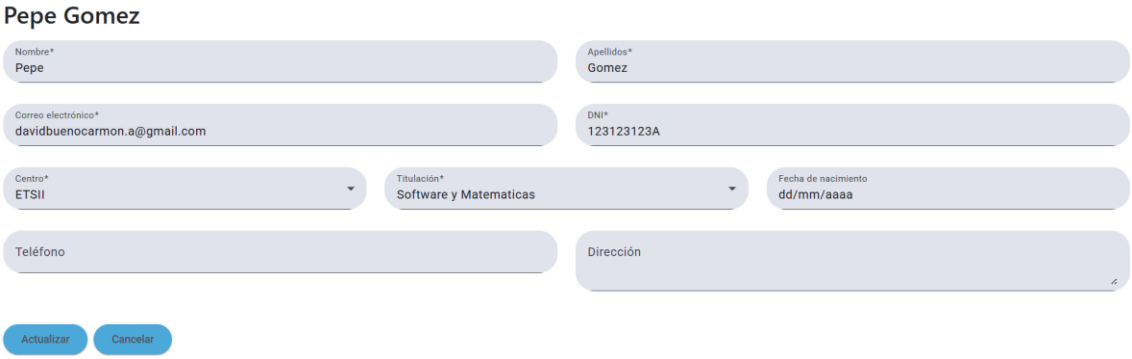


Ilustración 15. Resultado final del perfil de un alumno.

Se han cambiado el formato de los formularios pasando a tener un estilo más minimalista. Como podemos ver en las ilustraciones 14 y 15 el formulario del perfil de un alumno mantiene los campos del boceto a excepción de la imagen de usuario que se ha prescindido en la versión final.

4.2.2. Usabilidad

La usabilidad es fundamental en cualquier producto digital, especialmente en aplicaciones web, porque se trata de qué tan fácil, eficiente y agradable es para los usuarios interactuar con esa aplicación. Algunas ventajas de tener en cuenta la usabilidad en este proyecto son las siguientes:

Navegación consistente y predecible

La interfaz mantiene siempre los mismos elementos de menú y cabecera, lo que reduce la curva de aprendizaje y permite que el usuario sepa siempre dónde está y cómo volver a las secciones principales. Esto disminuye la frustración y optimiza los tiempos de acceso a las funcionalidades.

Acciones claras y confirmadas

Antes de realizar acciones importantes (como eliminar o importar datos) aparecen mensajes de confirmación. Esto genera seguridad y evita errores costosos, aumentando la confianza del usuario en la herramienta.

Búsqueda y filtrado inmediato

La posibilidad de buscar y filtrar directamente sin navegar por varias páginas reduce el tiempo para encontrar información, algo clave en entornos donde se gestiona mucho contenido.

Identidad visual y estructura estable

Elementos como el logotipo y la cabecera permanecen visibles, ayudando a mantener el contexto y reforzando la orientación dentro de la aplicación.

4.2.3. Accesibilidad

La accesibilidad [20] en aplicaciones web es esencial para garantizar que cualquier persona, independientemente de sus capacidades físicas, sensoriales o cognitivas, pueda interactuar con la plataforma sin barreras. Tener en cuenta la accesibilidad en este proyecto aporta ventajas como las siguientes:

Contraste y legibilidad adaptables

La posibilidad de alternar entre un modo claro y uno oscuro permite a los usuarios ajustar el contraste según sus necesidades visuales. Esto es especialmente útil para personas con baja visión, daltonismo o sensibilidad a la luz, facilitando la lectura y reduciendo la fatiga ocular.

Cambio de idioma en tiempo real

La interfaz ofrece la opción de cambiar el idioma sin recargar o interrumpir la navegación. Esto garantiza que personas con distintos niveles de dominio del idioma puedan comprender y utilizar la aplicación con fluidez, mejorando su experiencia global.

Contenido clave siempre accesible

Elementos como la cabecera fija y la navegación persistente evitan que el usuario tenga que desplazarse continuamente para acceder a funciones esenciales. Esto beneficia a personas con movilidad reducida o que usan dispositivos de entrada alternativos, como teclados adaptados.

Diseño adaptable a diferentes dispositivos

La interfaz se reorganiza de manera responsive, garantizando que las personas que usan móviles o tablets, incluyendo aquellas que dependen de un tamaño de letra mayor, puedan acceder a toda la información sin necesidad de hacer zoom o desplazamientos excesivos.

4.2.4. Diseño adaptativo

El diseño adaptativo (*responsive*) [21] es clave para que una aplicación web se ajuste de forma óptima a diferentes dispositivos y tamaños de pantalla, garantizando una experiencia de uso fluida y coherente en ordenadores, tablets y móviles. En este proyecto, su aplicación aporta ventajas como las siguientes:

Adaptación automática del contenido

La interfaz reorganiza y ajusta sus elementos según el ancho y alto de la pantalla, evitando que el usuario tenga que desplazarse lateralmente o hacer zoom para visualizar la información.

Priorización de la información esencial

En pantallas pequeñas, se muestran de forma destacada las funciones y contenidos más relevantes, lo que facilita la interacción y evita la sobrecarga visual.

Interacción cómoda en cualquier dispositivo

Los elementos táctiles, como botones y menús, se dimensionan y separan adecuadamente en dispositivos móviles, reduciendo errores al pulsar y mejorando la experiencia de uso en pantallas táctiles.

Consistencia visual en distintos entornos

El diseño mantiene la misma identidad y estructura general independientemente del dispositivo, lo que permite al usuario reconocer fácilmente la aplicación y usarla con confianza desde cualquier lugar.

5

Implementación

5.1. Estructura del proyecto

5.1.1. Servidor - Backend

El backend de este proyecto está organizado siguiendo un patrón modular y escalable, pensado para separar responsabilidades y facilitar tanto el mantenimiento como la futura ampliación de funcionalidades. La base del backend se encuentra en la carpeta webapi, que centraliza toda la lógica de servidor, la configuración y los controladores que gestionan las distintas áreas de la aplicación.

Organización de carpetas

Dentro de webapi la estructura se divide en varias secciones clave:

- **Common/**

  Esta carpeta contiene utilidades y clases base reutilizables en distintos módulos. Por ejemplo, BaseController.cs y BaseManager.cs proporcionan una estructura común para los controladores y gestores de lógica, asegurando que todos los módulos sigan un mismo patrón. Aquí también se encuentran clases para manejar excepciones específicas (NotFoundException.cs, UnprocesableException.cs), lo que permite capturar y devolver errores de forma controlada y uniforme.

- **Data/**

  Contiene ApplicationDbContext.cs, que es el núcleo del acceso a datos a través de Entity Framework. Aquí se definen los DbSet que representan las tablas y se gestionan las conexiones y el mapeo entre objetos y base de datos.

- **Modules/**

  Esta es la sección más extensa y donde realmente se observa el diseño modular del backend. Cada entidad o área de negocio (por ejemplo, Career, Department, Professor, Student, TFG, etc.) se organiza en su propia carpeta con tres elementos principales:

  - **Controller**: punto de entrada de las peticiones HTTP para esa entidad. Se encarga de recibir las solicitudes, validarlas y delegar la lógica al manager correspondiente.
  - **Manager**: capa intermedia donde reside la lógica de negocio. Aquí se gestionan validaciones más complejas, reglas específicas y procesamiento de la información.
  - **Model y Object**: clases que definen la estructura de datos. El *Model* suele representar la entidad tal y como se gestiona internamente, mientras que el *Object* puede actuar como un DTO

(Data Transfer Object) para controlar qué datos se envían o reciben.

Este esquema de **Controller → Manager → Model** se repite en todos los módulos, consiguiendo que el código sea más limpio, reutilizable y fácil de localizar.

Controladores y Managers

El uso de controladores y managers bien diferenciados es una de las fortalezas de este proyecto.

- Los **controladores** son ligeros y están orientados únicamente a gestionar la comunicación con el cliente. No contienen lógica compleja, lo que evita que se conviertan en piezas monolíticas difíciles de mantener.
- Los **managers** concentran la lógica de negocio. Este diseño sigue el principio de responsabilidad única (SRP), porque cada clase tiene un cometido claro: el controlador "habla" con el exterior y el manager "piensa" cómo resolver la petición.

Esto no solo facilita la lectura del código, sino que también permite que, si en el futuro se quiere cambiar la API o añadir un nuevo punto de entrada, gran parte de la lógica ya esté encapsulada y no sea necesario reescribirla.

Uso de Entity Framework

Entity Framework (EF) actúa como el ORM (Object Relational Mapper) del proyecto. Gracias a él, no es necesario escribir sentencias SQL de forma manual para las operaciones más comunes, ya que EF traduce automáticamente las operaciones con objetos C# en consultas SQL a la base de datos.

En ApplicationDbContext.cs se definen las colecciones DbSet que representan las tablas de la base de datos (Student, Professor, Department, TFG, etc.). Este

contexto también se encarga de aplicar configuraciones, relaciones y restricciones, siguiendo lo definido en los modelos.

El uso de EF aporta varias ventajas:

- **Abstracción del acceso a datos**: se trabaja directamente con objetos y colecciones en lugar de sentencias SQL.
- **Control de migraciones**: permite actualizar el esquema de la base de datos de forma controlada a medida que evolucionan los modelos.
- **Integridad referencial automática**: respeta y gestiona las relaciones definidas entre entidades.

Gracias a esta configuración, el backend consigue ser a la vez **potente** y **mantenible**, con una estructura lógica clara y preparada para crecer. El patrón modular, la separación entre controladores y managers, y el uso de Entity Framework convierten el sistema en una base sólida sobre la que construir nuevas funcionalidades sin comprometer la estabilidad.

5.1.2. Aplicación – Frontend

El frontend está desarrollado en **Angular**, lo que ya de por sí ofrece una arquitectura modular y organizada. La estructura principal se encuentra en la carpeta webapp/src, y está pensada para que cada funcionalidad de la aplicación esté encapsulada en módulos y componentes independientes, facilitando la escalabilidad y la reutilización de código.

Organización de carpetas

Dentro de src la organización sigue un patrón claro:

- **app/**

  Es el núcleo de la aplicación Angular. Contiene:

  - **app.module.ts**: módulo raíz que reúne y declara los demás módulos y componentes.

- **app.component.***: componente principal que sirve como punto de entrada visual, sobre el que se monta el resto de la interfaz.
- **app.routes.ts** y **routes.ts**: definen la navegación interna de la aplicación mediante enrutamiento, permitiendo acceder a las distintas páginas sin recargar la web.

- **core/**

  Esta carpeta concentra elementos comunes y esenciales para toda la aplicación:

  - **Layout**: cabecera, pie, menús, controles de idioma, selector de tema, etc. Mantiene la coherencia visual en todas las páginas.
  - **Services**: servicios generales como autenticación, configuración, interceptores de peticiones y gestión de almacenamiento local.
  - **Pages**: páginas genéricas como la de "no encontrado" (not-found).

- **modules/**

  Aquí está la clave del diseño modular del frontend. Cada área funcional (por ejemplo, admin, groups, tfg, professor, login, profile, etc.) se aloja en su propio módulo, con sus propios componentes, páginas y servicios. Dentro de cada módulo se sigue un patrón uniforme:

  - **components/**: elementos visuales reutilizables en varias páginas del mismo módulo (listas, formularios, cuadros de diálogo, etc.).
  - **pages/**: vistas completas que representan pantallas concretas.
  - **services/**: lógica de acceso a datos y comunicación con el backend para esa parte de la aplicación.

- **assets/**

  Contiene recursos estáticos como imágenes, iconos o ficheros de traducción. En este caso, destaca la carpeta i18n/ con ficheros es.json y en.json para internacionalización.

Modularidad y escalabilidad

La forma en que está dividido el frontend permite que cada módulo sea **autónomo** y se pueda desarrollar o mantener sin interferir en el resto. Esto:

- Permite añadir nuevas funcionalidades con un impacto mínimo en el resto de la aplicación.
- Favorece la reutilización de componentes, por ejemplo, un cuadro de confirmación o un selector de idioma que puede integrarse en múltiples pantallas.

Layouts y consistencia visual

La carpeta core/layout concentra todos los elementos que dan coherencia a la interfaz:

- **Cabecera fija** con acceso a menús, perfil, idioma y tema.
- **Desplegables** para acceder a las distintas secciones.
- **Elementos persistentes** como el logotipo, que refuerza la identidad visual y ayuda a la orientación.

Esto garantiza que el usuario tenga siempre a mano las funciones esenciales sin importar en qué página se encuentre, mejorando la experiencia y reduciendo la necesidad de navegar en exceso.

Servicios y comunicación con el backend

Cada módulo dispone de servicios dedicados para comunicarse con la API, encapsulando las llamadas HTTP y evitando que la lógica de negocio se mezcle con la presentación. Esto significa que:

- Las páginas y componentes solo se ocupan de la parte visual y de interacción con el usuario.

- La comunicación con el backend es consistente y centralizada, lo que facilita cambios futuros (por ejemplo, modificar la URL de la API o la forma en que se manejan los errores).

Internacionalización y accesibilidad

El soporte multilenguaje mediante los ficheros en assets/i18n permite que cualquier texto mostrado en la interfaz se pueda traducir sin modificar el código de los componentes. Esto no solo mejora la accesibilidad para usuarios de diferentes idiomas, sino que también facilita la expansión de la aplicación a otros mercados.

En conjunto, el frontend de este proyecto está diseñado con una arquitectura modular, escalable y mantenible, donde cada módulo es independiente, los layouts aseguran coherencia visual y la separación entre vista y lógica de datos está bien definida. Esto repercute directamente en una experiencia de usuario más fluida, una interfaz coherente y una capacidad de evolución del producto sin comprometer la estabilidad.

5.2. Funcionalidades del proyecto

La funcionalidad central de este sistema gira en torno a la gestión y reserva de TFGs, permitiendo a estudiantes, profesores y administradores interactuar de manera coordinada en todo el ciclo de vida del TFG. El objetivo principal es digitalizar y simplificar un proceso que, tradicionalmente, suele gestionarse de forma manual mediante formularios físicos, correos electrónicos y hojas de cálculo dispersas.

Todas las funcionalidades que se muestran en este apartado son accesibles desde la aplicación. Las vistas y las acciones relacionadas se muestran en el manual de usuario, en el Apéndice 2.

5.2.1. Entidades básicas

Aunque la funcionalidad estrella del proyecto es la gestión y reserva de TFG, la plataforma integra un conjunto de entidades de apoyo que permiten que el proceso sea más controlado, preciso y escalable. Estas entidades no intervienen de forma directa en la solicitud o asignación de un TFG, pero proporcionan la estructura y contexto necesarios para que el sistema sea realmente útil en un entorno universitario.

5.2.1.1. Centros

En este sistema, los Centros representan las escuelas o sedes académicas donde se desarrollan las titulaciones, departamentos y líneas de TFG. Funcionan como el nivel más alto en la jerarquía organizativa de la aplicación, permitiendo segmentar la oferta académica y mantener una estructura ordenada.

Utilidad en la aplicación

- **Organización institucional**

  Evita confusiones entre titulaciones con nombres similares en distintas sedes.
- **Filtrado de información**

  Permite mostrar solo los TFG, titulaciones o departamentos asociados a un centro concreto.
- **Gestión multi-campus**

Soporta múltiples escuelas o centros de una misma universidad dentro del mismo sistema.

Endpoints disponibles

Siguiendo el patrón REST y la estructura de la API, la entidad university suele contar con los siguientes endpoints:

Ilustración 16. Endpoints de Centros

Ejemplo de uso

Si la universidad abre dos nuevas sedes y dispone de un archivo con toda su información:

1. El administrador utiliza POST /university/import para cargarlas de forma masiva.
2. Posteriormente, un usuario que busca una sede concreta puede usar POST /university/search para localizarla por nombre o ubicación.
3. Los departamentos y titulaciones asociados a esas sedes se podrán crear después, asegurando que la estructura jerárquica se mantenga coherente.

5.2.1.2. Titulaciones

La entidad Titulaciones representa las distintas carreras o programas académicos que ofrece una universidad (por ejemplo, Ingeniería Informática, Matemáticas, Robótica, etc.). Su presencia en el sistema es clave para que la asignación de TFG sea coherente y esté alineada con la formación del estudiante.

Cada titulación está vinculada a un centro concreto y puede estar relacionada con varias líneas de TFG. Esto garantiza que un estudiante solo pueda solicitar aquellos TFG que sean relevantes para sus estudios, evitando errores o solicitudes no válidas.

Utilidad en la aplicación

En el flujo general de la plataforma, las titulaciones cumplen varias funciones importantes:

- **Filtrado de TFG por carrera**

  Cuando un estudiante accede al listado de TFG disponibles, la aplicación filtra automáticamente las líneas que son compatibles con su titulación. Esto agiliza la búsqueda y evita opciones no aplicables.

- **Organización académica**

  Permite generar reportes y estadísticas por titulación, ayudando a los departamentos y universiades a entender la distribución de TFG y la demanda en cada carrera.

Endpoints y operaciones

En la API, la entidad **career** cuenta con una serie de endpoints que permiten su gestión.

Siguiendo el patrón REST utilizado en el backend, las operaciones típicas incluyen:

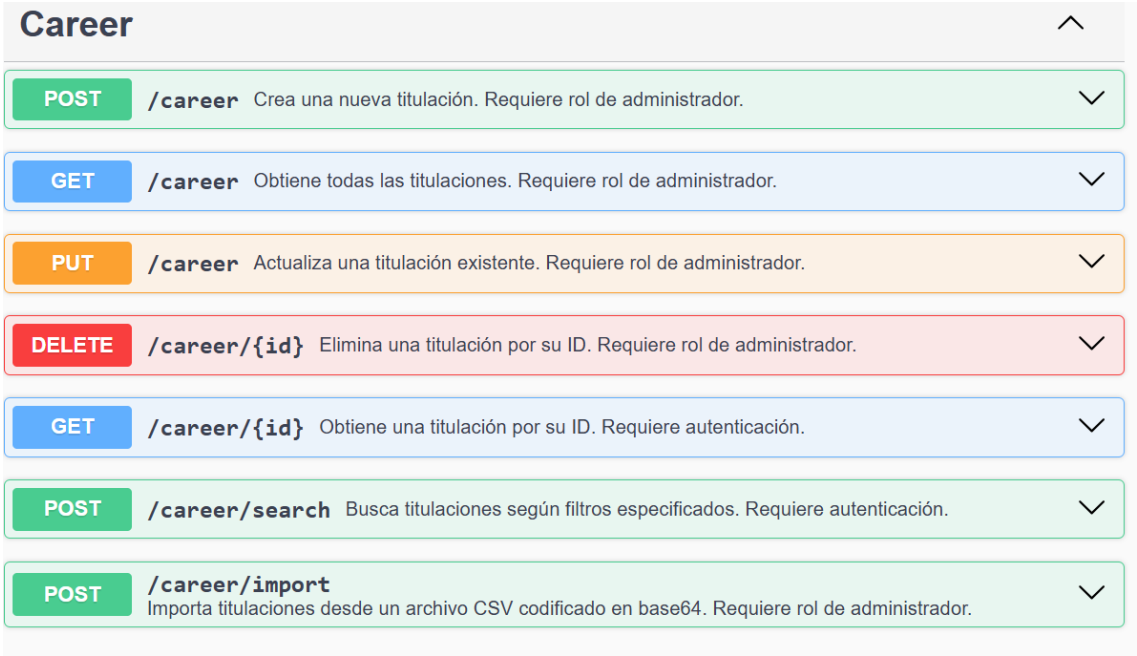


Ilustración 17. Endpoints de Titulaciones

Ejemplo de uso en el sistema

Supongamos que un nuevo programa de estudios, *Ingeniería de Datos*, empieza a impartirse en la universidad.

1. El administrador accede al panel de gestión y crea la titulación mediante POST /career.
2. Posteriormente, un departamento añade nuevas líneas de TFG asociadas a esta titulación.
3. Cuando un estudiante matriculado en *Ingeniería de Datos* inicie sesión, la plataforma le mostrará únicamente TFG compatibles, gracias a la relación directa entre su perfil y la titulación registrada.

5.2.1.3. Departamentos

En la estructura de la aplicación, los Departamentos representan las divisiones académicas dentro de un centro o universidad. Son responsables de ofertar y gestionar las líneas de TFG, así como de coordinar a profesores y titulaciones relacionadas con su área de conocimiento. Un ejemplo sería el *Departamento de Lenguajes y Ciencias de la Computación* o el *Departamento de Matemática Aplicada*.

Utilidad en la aplicación

Los departamentos son un elemento intermedio clave en la jerarquía académica. Gracias a ellos se puede organizar la oferta de TFG por área de conocimiento, asignar responsabilidades a profesores y garantizar que cada línea de investigación esté supervisada por la unidad correspondiente.

Endpoints disponibles

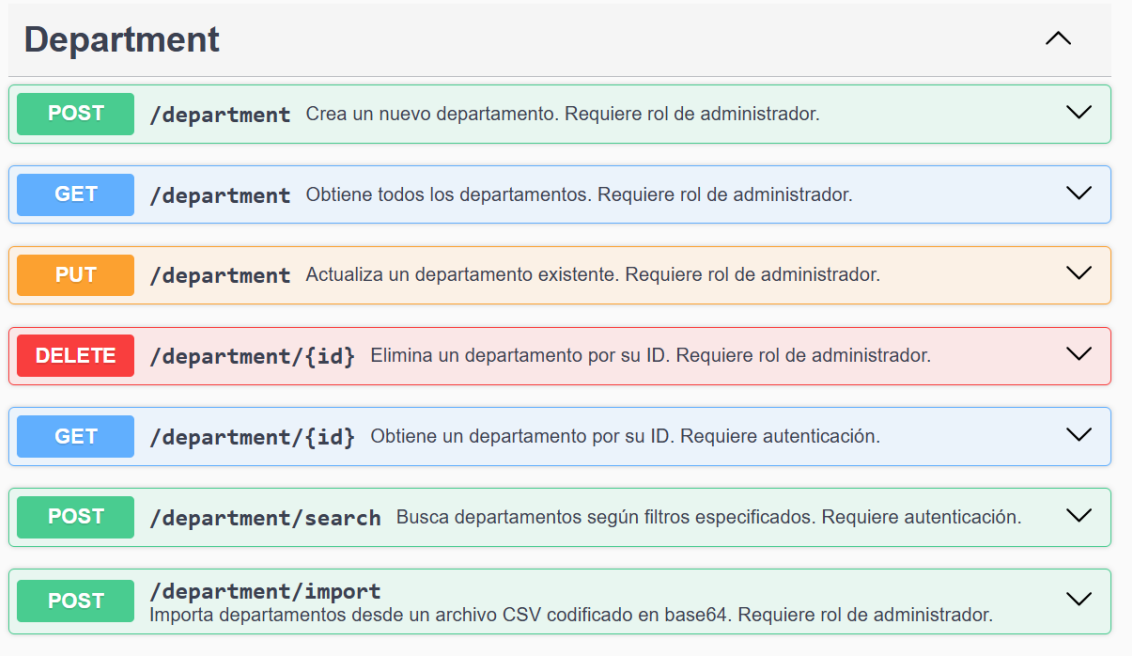


Ilustración 18. Endpoints de Departamentos

Ejemplo de uso

Supongamos que la universidad abre un nuevo centro*: la Escuela de Tecnologías Avanzadas*, con varios departamentos asociados. El proceso sería:

1. El administrador utiliza el endpoint **POST /department/import** para cargar de forma masiva todos los departamentos de ese nuevo centro.
2. Los departamentos recién creados quedan vinculados a su centro correspondiente en la base de datos.
3. Los administradores asocian las líneas de TFG correspondientes a cada departamento.
4. Un profesor, al iniciar sesión, accede a la sección de TFG y visualiza automáticamente todos los proyectos vinculados a su departamento.
5. Un estudiante de una titulación concreta verá únicamente las líneas de TFG compatibles con su plan de estudios, incluso si el departamento ofrece más proyectos para otras titulaciones.

Este flujo asegura que cada usuario tenga acceso únicamente a la información relevante para su perfil, pero al mismo tiempo mantiene centralizada toda la gestión académica en la plataforma.

5.2.2. Estudiantes

En la estructura de la aplicación, los estudiantes pueden participar en la elección, desarrollo y seguimiento de TFG. Cada estudiante está vinculado a una titulación y, a través de ella, a un centro y a las líneas de investigación disponibles. Por ejemplo, un estudiante de Ingeniería Informática o un estudiante de Matemáticas.

Utilidad en la aplicación

Los estudiantes son el eje central de la gestión de TFG, ya que el sistema les permite acceder a la oferta de proyectos, realizar solicitudes, recibir asignaciones

y comunicarse con sus tutores. Además, su información académica es utilizada para filtrar y personalizar la oferta de TFG según su titulación y plan de estudios.

Endpoints disponibles

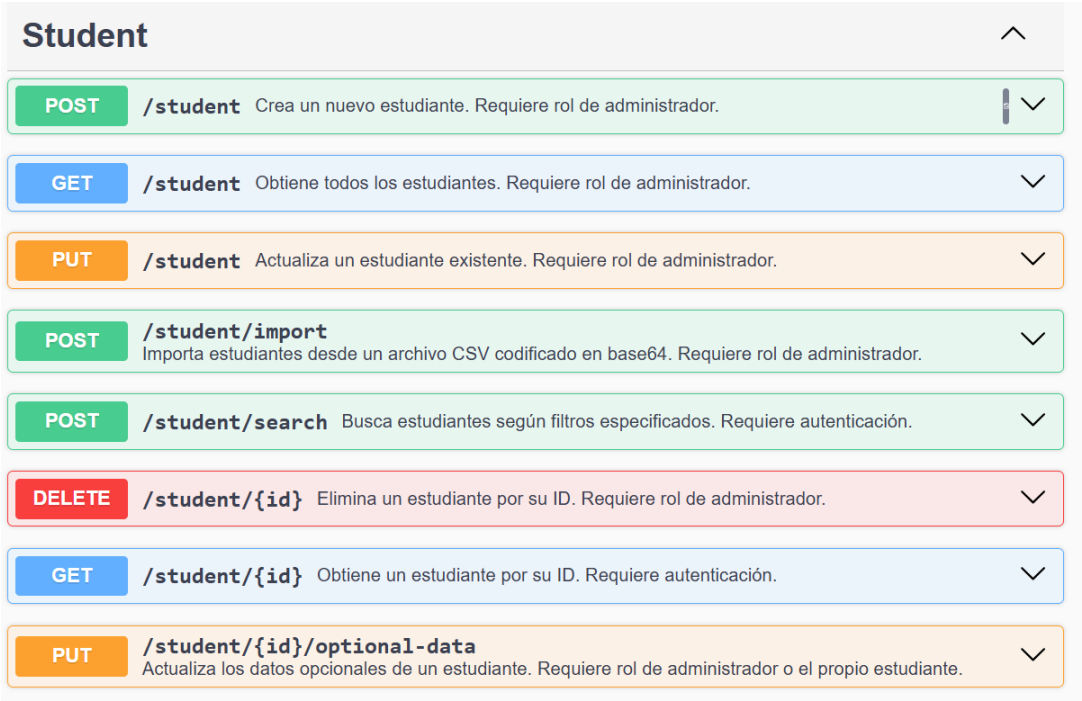


Ilustración 19. Endpoints de Estudiantes

Ejemplo de uso

Supongamos que una universidad matricula a una nueva promoción de estudiantes en varias titulaciones:

1. El administrador utiliza el endpoint **POST /student/import** para cargar de forma masiva todos los nuevos estudiantes.
2. Los registros creados quedan asociados a sus titulaciones y centros correspondientes.
3. Los estudiantes, al iniciar sesión, visualizan automáticamente la oferta de TFG filtrada por su plan de estudios.
4. Un estudiante interesado en un proyecto realiza una solicitud mediante el módulo correspondiente de la plataforma.

5. El tutor asignado puede gestionar y supervisar el progreso del estudiante en su TFG desde el sistema.

Este flujo asegura que la información académica de cada estudiante se gestione de forma centralizada, a la vez que personaliza la experiencia y facilita la coordinación con tutores y administradores.

5.2.3. Profesores

En la estructura de la aplicación, los profesores representan al personal docente universitario encargado de dirigir, supervisar y evaluar los TFG de los estudiantes. Cada profesor está adscrito a uno o varios centros y puede tener una o varias líneas de investigación activas, así como asumir el rol de tutor principal o co-tutor en distintos proyectos.

Utilidad en la aplicación

Los profesores son una pieza clave en la gestión académica de TFG. A través de la plataforma, pueden aceptar solicitudes de estudiantes, coordinar el desarrollo de los proyectos, evaluar entregas y cerrar el proceso académico del TFG. Además, la información de su departamento y su perfil permite a los estudiantes localizar tutores afines a su área de estudio.

Endpoints disponibles

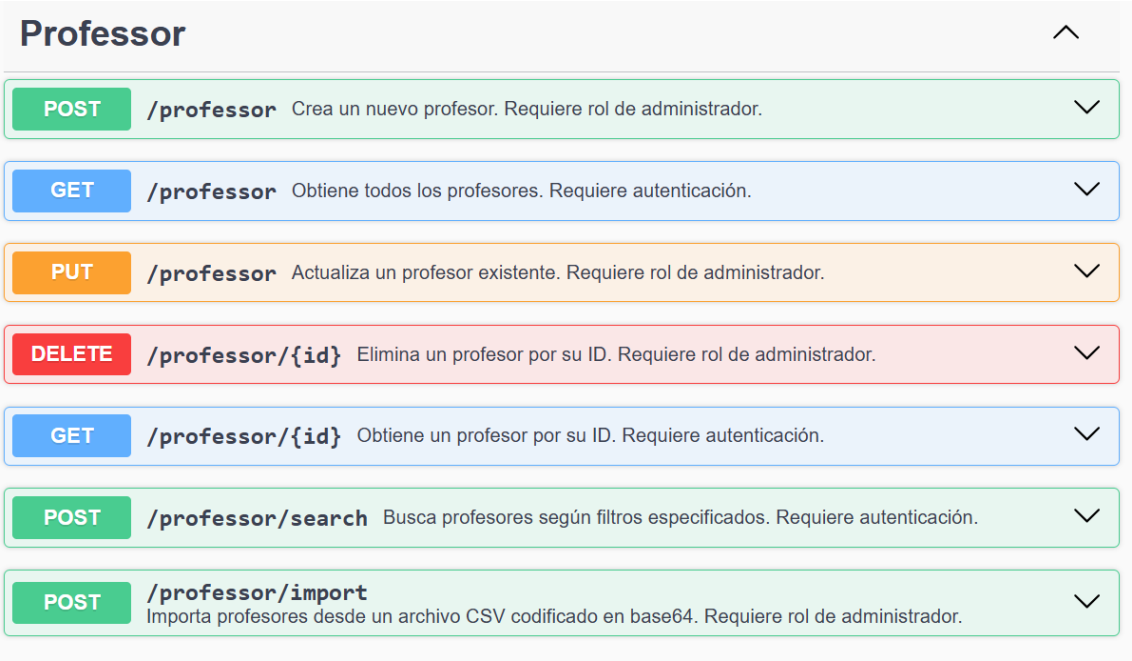


Ilustración 20. Endpoints de Profesores

Ejemplo de uso

Supongamos que una universidad contrata a un nuevo grupo de profesores para reforzar la oferta de TFG en varias áreas:

1. El administrador utiliza el endpoint **POST /professor/import** para dar de alta masivamente a los nuevos docentes.
2. Cada profesor queda vinculado a su departamento en la base de datos.
3. Los profesores acceden a la plataforma y crean sus líneas de TFG.
4. Los estudiantes interesados solicitan TFG a través del sistema, que envía notificaciones a los profesores correspondientes.
5. El profesor acepta la propuesta, inicia el seguimiento del TFG y coordina reuniones con el estudiante.

Este flujo asegura que los profesores tengan control total sobre la gestión de sus TFG y que los estudiantes puedan localizar tutores adecuados a su área de estudio de manera sencilla.

5.2.4. Auth

En la estructura de la aplicación, el módulo Auth es el responsable de la autenticación y validación de credenciales de los usuarios (administradores, profesores y estudiantes), así como de la gestión de contraseñas y la verificación de tokens de sesión.

Utilidad en la aplicación

El controlador de autenticación permite iniciar sesión, cambiar contraseñas, crear cuentas de administrador y validar que un token de acceso es válido. Esto garantiza que solo usuarios autorizados accedan a la plataforma y que su información se mantenga segura.

Endpoints disponibles

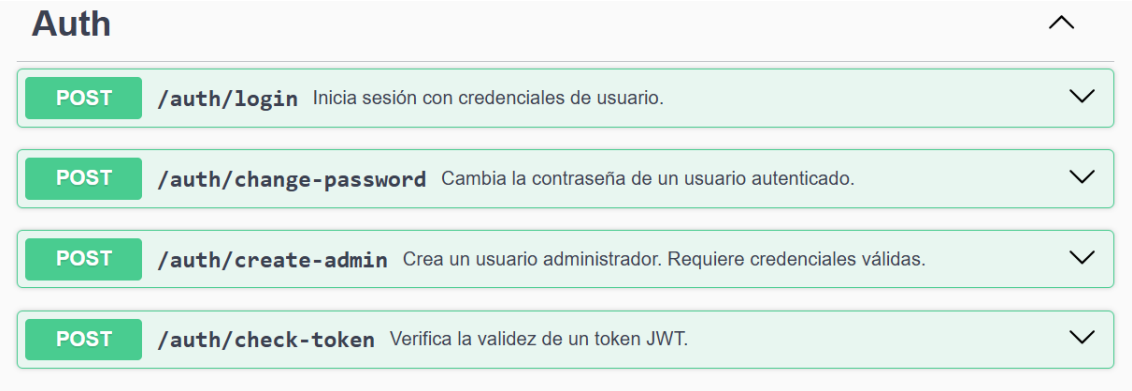


Ilustración 21. Endpoints de Autenticación

Ejemplo de uso

Un administrador necesita dar acceso a la plataforma a un nuevo usuario administrador:

1. El administrador existente envía una petición **POST /auth/create-admin** con las credenciales del nuevo administrador.
2. El nuevo administrador puede iniciar sesión mediante **POST /auth/login**.

3. Si necesita actualizar su clave por seguridad, usa **POST /auth/change-password**.
4. En cada interacción con la API, el cliente puede llamar a **POST /auth/check-token** para confirmar que el token es válido.

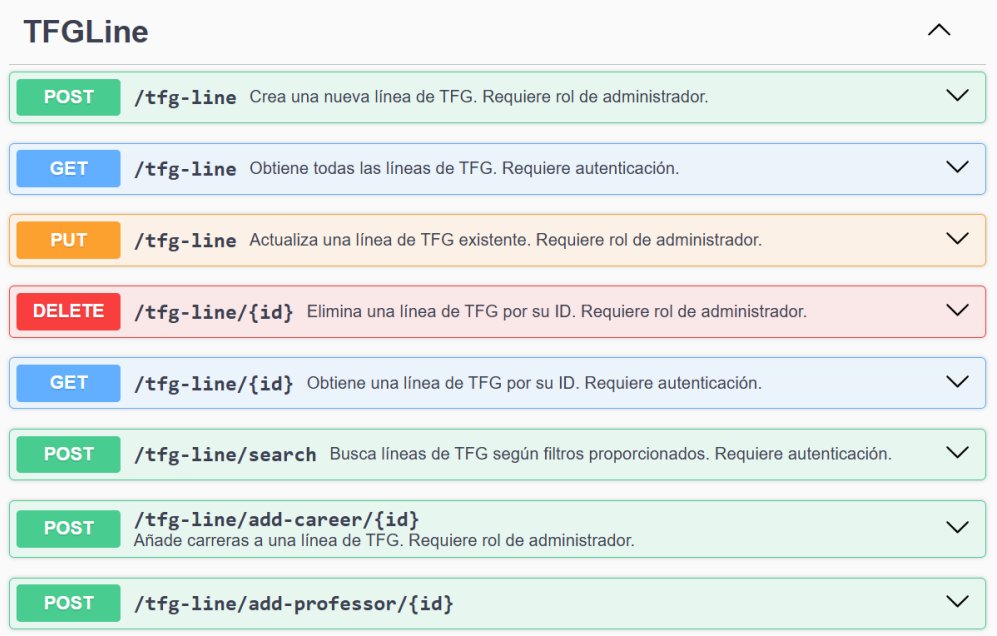
5.2.5. Línea de TFG

En la estructura de la aplicación, las líneas de TFG representan las propuestas temáticas bajo las cuales los estudiantes pueden desarrollar sus TFG. Cada línea está asociada a un departamento, puede tener asignados profesores y titulaciones, y define los cupos disponibles para la asignación de estudiantes.

Utilidad en la aplicación

El controlador TFGLine permite crear, modificar, eliminar, buscar y asignar relaciones a las líneas de TFG. A través de él, los administradores gestionan la oferta de proyectos y definen qué titulaciones y profesores están vinculados a cada línea, facilitando así la organización y asignación de trabajos.

Endpoints disponibles



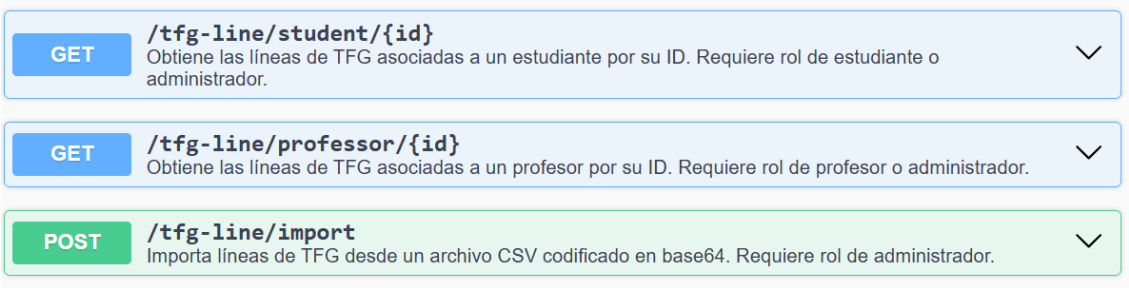


Ilustración 22. Endpoints de Líneas de TFG

Ejemplo de uso

Supongamos que un centro universitario quiere crear nuevas líneas de TFG para el próximo curso:

1. El administrador crea cada línea de TFG mediante **POST /tfg-line**.
2. Posteriormente, asocia titulaciones con **POST /tfg-line/add-career/{id}** y profesores con **POST /tfg-line/add-professor/{id}**.
3. Los estudiantes pueden ver las líneas disponibles mediante **POST /tfg-line/search**.
4. Si se dispone de muchas líneas para cargar, se utiliza **POST /tfg-line/import** para agilizar el proceso.

5.2.6. TFG

En la estructura de la aplicación, TFG representa un Trabajo de Fin de Grado cuando ya existe una solicitud formal de un estudiante para realizarlo. Es decir, se trata de un proyecto ya en proceso de asignación o desarrollo, con uno o más tutores y un estudiante asignado.

La diferencia con líneas de TFG es que en las líneas se gestiona *líneas disponibles* de TFG (temas y proyectos potenciales aún sin asignar), mientras que TFG gestiona *proyectos en curso o en solicitud*, resultado de que un estudiante haya solicitado una línea y se haya iniciado su tramitación.

Utilidad en la aplicación

El controlador TFG permite gestionar el ciclo de vida de los Trabajos de Fin de Grado desde que se realiza una solicitud hasta su aceptación, rechazo o cambio de estado. A través de él, administradores y profesores supervisan y actualizan el estado de los proyectos, mientras que los estudiantes pueden solicitar un TFG disponible.

Endpoints disponibles

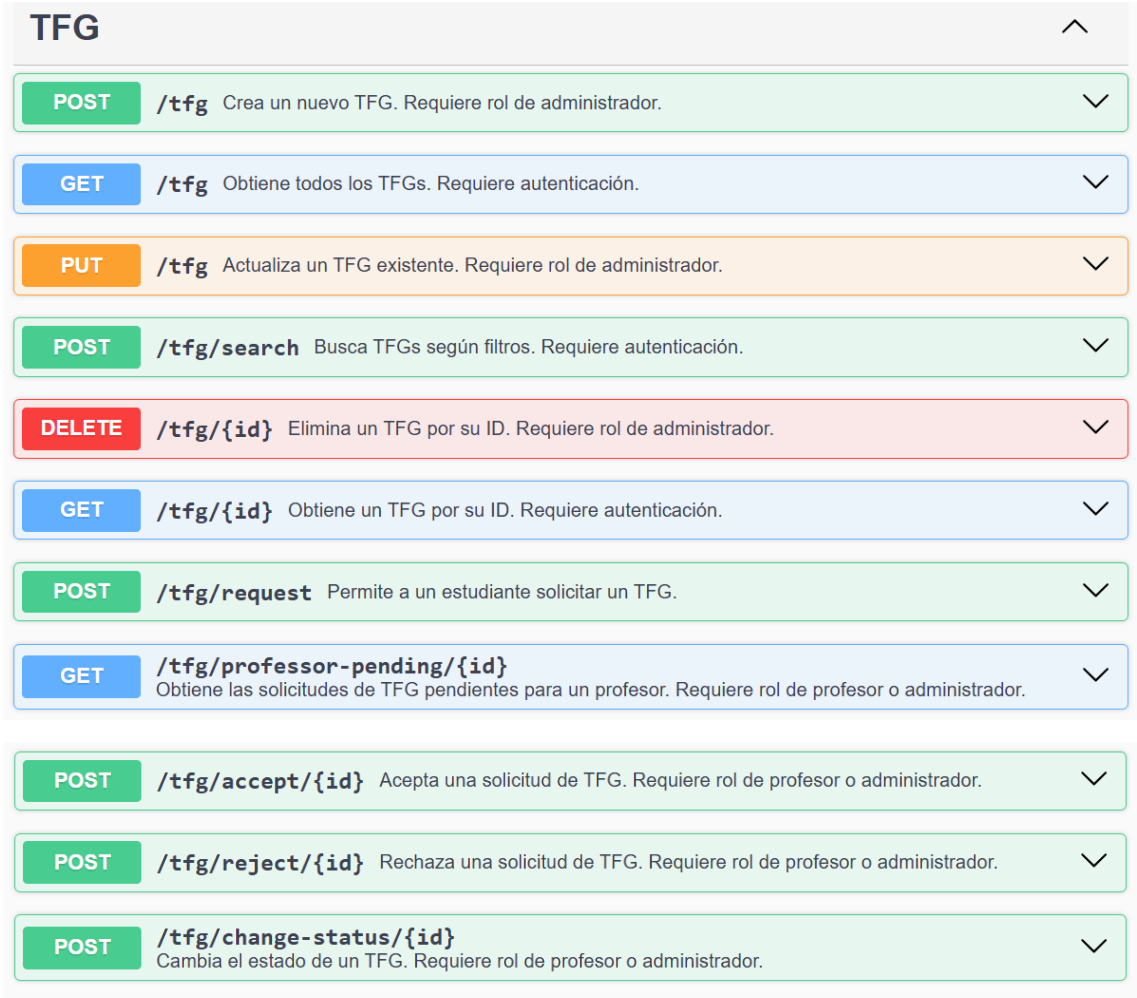


Ilustración 23. Endpoints de TFG

Ejemplo de uso

Un estudiante encuentra en el catálogo una línea de TFG (TFGLine) que le interesa:

1. Solicita ese trabajo mediante **POST /tfg/request**.

2. El profesor responsable revisa la solicitud usando **GET /tfg/professor-pending/{id}**.
3. El profesor acepta la propuesta con **POST /tfg/accept/{id}**.
4. Durante el desarrollo del proyecto, el estado puede actualizarse mediante **POST /tfg/change-status/{id}**.
5. Una vez completado, el TFG queda registrado como finalizado en la plataforma.

Este flujo refleja cómo el gestor de líneas actúa como catálogo de oportunidades, mientras que el gestor de TFG gestiona los proyectos reales con estudiantes asignados.

5.2.7. Canales

En la aplicación, los **Working Groups** (o *canales de trabajo*) son espacios colaborativos donde profesores y estudiantes pueden interactuar en torno a un TFG o un tema concreto.

Endpoints disponibles

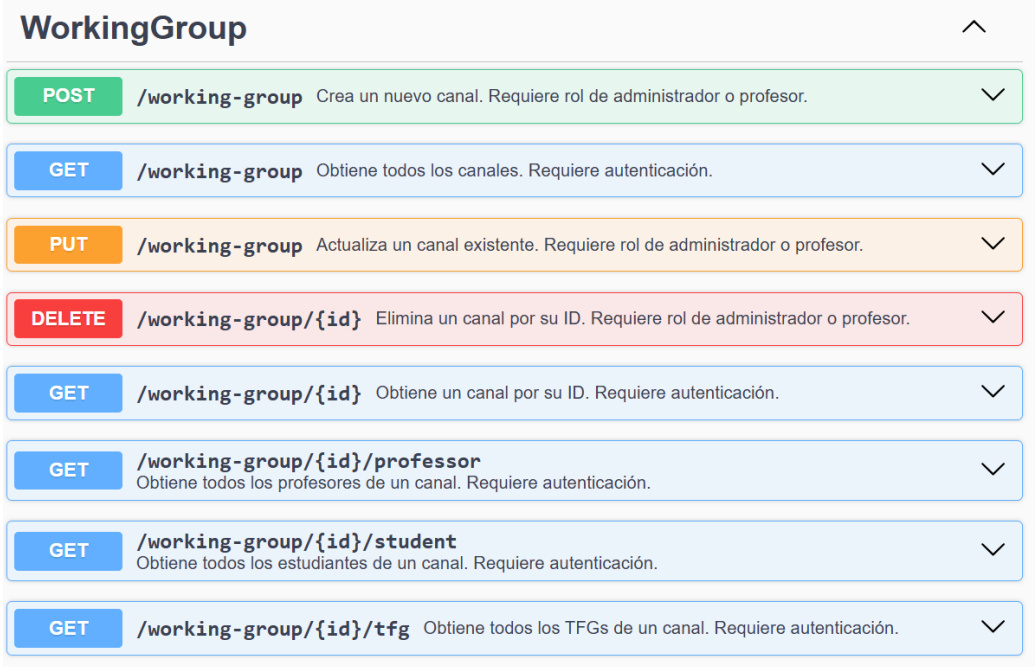


Ilustración 24. Endpoints de Canales

Ejemplo de uso en flujo de trabajo

1. Un profesor al aceptar una solicitud de TFG automáticamente se crea un canal asociado a ese TFG con el tutor y, en caso de que tenga, cotutor y el estudiante que lo solicitó mediante el endpoint **POST /working-group**.
2. En caso de que algún otro profesor vaya a formar parte del proyecto se puede agregar con el endpoint **POST /working-group/add-professor.**
3. Cualquier profesor de los que pertenezcan al grupo podrán enviar mensajes vía correo electrónico para los alumnos mediante el endpoint **POST /working-group/send-message.**

5.3. Servicio de Correo

El servicio de correo es el componente encargado de enviar notificaciones por correo electrónico a los usuarios de la plataforma. Su objetivo es mantener

informados a estudiantes, profesores y administradores sobre eventos clave en el ciclo de vida de un TFG y en la colaboración dentro de grupos de trabajo.

Para realizar este servicio se ha creado una cuenta de Gmail tfginfo.notify@gmail.com.
Con el nombre de TFGinfo. Para establecer la conexión con Gmail, desde un servidor de .NET ha sido necesario registrar la aplicación como una aplicación personal en Google e introducir una clave para habilitar el envío de correo desde la API.

Funcionamiento técnico

- Se apoya en MailKit [22] para la conexión SMTP y el envío de correos.
- Usa las configuraciones definidas en este servicio (servidor SMTP, puerto, credenciales, remitente, etc.).
- Admite envío de correos en formato **HTML** o **texto plano**.
- Cada correo se compone de:
    - **Remitente** (nombre y correo del sistema).
    - **Destinatario** (usuario final).
    - **Asunto** (según el tipo de notificación).
    - **Cuerpo del mensaje** (personalizado para el evento).

Posibles notificaciones enviadas

- **Solicitud de un TFG**
    - Cuando un estudiante solicita un TFG disponible, se notifica al profesor seleccionado como tutor o cotutor con los detalles de la solicitud.
    - Ejemplo de asunto: Nueva solicitud de TFG.
- **Aceptación o rechazo de un TFG**

    - Si el profesor acepta/rechaza la solicitud, el estudiante recibe un correo indicando el resultado y, si aplica, los siguientes pasos.
    - Ejemplo de asunto: Tu solicitud de TFG ha sido aceptada o Tu solicitud de TFG ha sido rechazada.
- **Creación de usuario**
    - Al crear un nuevo usuario por un administrador, se le envía un correo con sus credenciales y un enlace para iniciar sesión.
    - Ejemplo de asunto: Nuevo usuario TFGinfo.
- **Cambio de estado de un TFG**
    - Cuando un TFG cambia de estado (por ejemplo, "En curso" → "Finalizado"), los involucrados reciben una notificación.
- **Nuevo mensaje en un grupo de trabajo**
    - Cuando un profesor envía un mensaje en un canal de trabajo, los miembros del grupo reciben un correo con el contenido del mensaje.
    - Ejemplo de asunto: Nuevo mensaje en tu grupo de trabajo.

Beneficio para la plataforma

Este servicio asegura que los usuarios estén informados sin necesidad de entrar constantemente a la aplicación. Además, centraliza la lógica de notificaciones, lo que facilita modificar el formato o el proveedor de correo sin afectar el resto del sistema.

5.4. Autenticación y Autorización

Flujo de Creación y Autenticación de Usuarios

En este apartado se describe el proceso completo desde la creación de un usuario hasta la autenticación y autorización en la aplicación, utilizando como ejemplo el caso de un estudiante. El flujo es idéntico para el usuario con rol de profesor.

Creación del Usuario

1. **Formulario de creación:**

   Un usuario con rol administrador accede a la página destinada a la creación de estudiantes. En esta página se presenta un formulario que debe ser rellenado con los datos personales del alumno, entre ellos su correo electrónico, que será utilizado como identificador único para el acceso al sistema.

2. **Envío y procesamiento de datos:**

   Al completar el formulario y enviar la información, se genera una petición al servidor que realiza las siguientes acciones:

   - Crea una nueva instancia de estudiante en la base de datos con los datos proporcionados.
   - Crea una instancia asociada de usuario, vinculada al estudiante creado, asignándole el rol de estudiante.
   - Genera una contraseña temporal, la cual se almacena en la base de datos vinculada a dicha cuenta.

3. **Notificación al usuario:**

   Finalizado el proceso de creación, el sistema envía automáticamente un correo electrónico al estudiante, utilizando la dirección proporcionada por el administrador. Este correo incluye:

   - Notificación de la creación de la cuenta en la plataforma TFGinfo.
   - La contraseña temporal asignada para el primer acceso.

Además, el administrador tiene la posibilidad de visualizar la contraseña temporal directamente desde la aplicación, permitiendo así otras formas alternativas de comunicación al usuario.

Primer Acceso y Cambio de Contraseña

1. **Login inicial:**

   El estudiante accede a la pantalla de inicio de sesión e introduce su correo electrónico y la contraseña temporal recibida.

2. **Identificación de primer acceso:**

   El sistema detecta que es el primer inicio de sesión del usuario y, en consecuencia, despliega un modal solicitando el cambio obligatorio de la contraseña temporal. Este formulario requiere:

   - La contraseña antigua (temporal).
   - La nueva contraseña, que debe cumplir con requisitos mínimos de longitud y seguridad.
   - Confirmación de la nueva contraseña.

3. **Actualización segura de contraseña:**

   Una vez confirmado el cambio, la nueva contraseña es almacenada en la base de datos aplicando un algoritmo de hash SHA256, garantizando la seguridad y confidencialidad de las credenciales.

4. **Acceso posterior:**

   Tras el cambio exitoso, el usuario podrá iniciar sesión con su nueva contraseña. En siguientes intentos, el sistema reconocerá que no es la primera vez y permitirá el acceso normal a la plataforma si las credenciales son correctas.

Gestión de Sesiones mediante Tokens JWT [23]

1. **Generación del token:**

   Al iniciar sesión, el servidor genera un token JWT con una validez de una hora. Este token contiene información esencial del usuario, como su identidad y rol asignado.

2. **Almacenamiento y seguridad:**
    - El token JWT es cifrado usando la biblioteca Jose-JWT [24] de .NET para añadir una capa adicional de seguridad que impide la manipulación del token.
    - El token cifrado se almacena en el localStorage del navegador.
3. **Uso del token en la aplicación:**
    - Todas las peticiones HTTP realizadas por el usuario incluyen automáticamente este token para autenticar y autorizar las acciones.
    - La aplicación utiliza la información contenida en el token (como el rol) para habilitar o restringir funcionalidades específicas (por ejemplo, solo los administradores pueden editar líneas de TFG).
4. **Manejo del estado de sesión:**
    - Al refrescar o cerrar y abrir la página, la aplicación lee el token almacenado para restaurar el estado del usuario.
    - Si no existe token o este está caducado, el usuario es redirigido automáticamente a la pantalla de login.

Validación y Autorización en el Servidor

- Cada llamada a la API incluye el token cifrado.
- El servidor descifra el token y valida:
    - Que el token no esté caducado.
    - Que el usuario tenga permisos suficientes para realizar la acción solicitada.
- En caso de que alguna de estas condiciones no se cumpla, la llamada es rechazada devolviendo un error que impide el acceso a los datos o funcionalidades restringidas.

En conclusión, este sistema proporciona una solución robusta para la autenticación y autorización de usuarios mediante el uso de tokens JWT cifrados, gestionando de forma segura la creación de usuarios, el cambio obligatorio de contraseñas temporales y la validación continua de permisos tanto en cliente como en servidor.

5.5. Servicios auxiliares

El sistema cuenta con servicios auxiliares en el lado del cliente (frontend) y en parte del backend, que se encargan de mejorar la seguridad, la experiencia de usuario y la comunicación con la API.

Interceptors (Interceptores de Peticiones HTTP) [25]

Un interceptor es una capa que se ejecuta antes de enviar cada solicitud HTTP al servidor y después de recibir la respuesta.

En esta plataforma, su función principal es:

- Adjuntar automáticamente el token de autenticación en el encabezado Autorización de todas las peticiones protegidas.
- Detectar respuestas no autorizadas (401) y redirigir al usuario a la página de inicio de sesión.
- Centralizar el manejo de **errores** para mostrar mensajes coherentes cuando algo falla.

El uso de esta herramienta evita que el desarrollador tenga que añadir manualmente el token y la gestión de errores en cada llamada a la API.

SnackBar (Notificaciones en pantalla)

El SnackBar es un componente visual que muestra mensajes temporales en la interfaz, normalmente en la parte inferior de la pantalla.

En esta aplicación se utiliza para:

- Mostrar confirmaciones rápidas (por ejemplo, “TFG aceptado con éxito”).
- Avisar de errores (por ejemplo, “No tienes permisos para realizar esta acción”).
- Informar sobre procesos completados (como la importación de líneas de TFG).

El snackbar sirve para ofrecer feedback inmediato al usuario sin interrumpir su flujo de trabajo.

Guards (Guardas de Autorización) [26]

Los guards son filtros que determinan si un usuario puede acceder a una ruta o componente en la aplicación.

En la plataforma, las guardas comprueban:

- Si el usuario está autenticado.
- Si su rol (Admin, Profesor, Estudiante) le permite acceder a la sección solicitada.

Ejemplos:

- Un **AdminGuard** permite el acceso solo a administradores.
- Un **ProfessorGuard** solo a profesores, etc.

Las guardas refuerzan la seguridad evitando que un usuario sin permisos acceda a pantallas o acciones no autorizadas, incluso si manipula la URL.

5.6. Variables de entorno y configuración

En esta aplicación, la configuración no se codifica directamente en el código fuente, sino que se gestiona mediante ficheros de configuración y variables de entorno. Esto permite cambiar parámetros sin recompilar, proteger datos sensibles y tener configuraciones diferentes para desarrollo, pruebas y producción.

Backend – appsettings.json y variables de entorno

En ASP.NET Core, el fichero appsettings.json (y sus variantes para cada entorno) define parámetros clave como:

- Configuración de la base de datos.
- Parámetros del servicio de correo usados por el sistema de notificaciones.
- Configuración para la autenticación mediante JWT.
- Rutas y URLs base de servicios externos.

Generalmente este documento se encuentra en el fichero de .gitignore para evitar que información sensible sea expuesta en un repositorio público.

Frontend – AppConfig y variables de entorno

En el lado del cliente, el fichero AppConfig define parámetros visibles para el navegador, como:

- URL base de la API.
- Dirección de el logo que se usará en la aplicación

En este archivo no se incluyen credenciales sensibles, ya que el código del frontend es accesible por cualquier usuario.

Diferencias clave entre Backend y Frontend

- El backend contiene información interna y segura, mientras que el frontend solo incluye datos no críticos.

- El backend no es visible para el usuario final, el frontend sí.
- La configuración del backend puede modificarse en tiempo de ejecución (reiniciando el servicio), mientras que en el frontend suele requerir recompilar.
- Ambos entornos permiten sobrescribir valores mediante configuraciones específicas para desarrollo, pruebas o producción.

5.7. Pruebas

5.7.1. Usabilidad y Accesibilidad (WAVE)

Para evaluar la usabilidad y accesibilidad de una aplicación web, existen herramientas que permiten detectar de forma automática problemas relacionados con la experiencia de usuario. Una de las más utilizadas es **WAVE**, la cual examina las vistas en busca de errores como falta de contraste en los colores, botones o enlaces sin texto alternativo, o deficiencias en la estructura semántica de la interfaz.

En el caso de TFGInfo, esta herramienta ha sido de gran utilidad, ya que ha permitido identificar aspectos que durante el desarrollo no siempre resultan evidentes. La aplicación sigue un diseño homogéneo, con una paleta de colores consistente y un patrón común en la distribución de botones, formularios, listas y enlaces, lo cual facilita la navegación y reduce la curva de aprendizaje de los usuarios.

Asimismo, TFGInfo incorpora dos modos de visualización: claro y oscuro. En ambos casos la distribución de elementos es idéntica, aunque varían los estilos y contrastes. Esto obligó a realizar pruebas en ambos modos, asegurando que la experiencia fuese coherente y accesible en cualquier contexto. Además, la aplicación está diseñada de forma responsiva, ofreciendo visualización tanto en

escritorio como en dispositivos móviles. En este caso, lo que varía es la distribución de los elementos en pantalla, lo que obligó a comprobar la usabilidad en todas las combinaciones posibles: **Claro-escritorio, Claro-móvil, Oscuro-escritorio y Oscuro-móvil**.

Durante estas pruebas se detectaron y corrigieron diversos problemas, entre los que destacan:

- El contraste insuficiente de los enlaces de la cabecera en el modo oscuro.
- El uso de iconos de perfil e idioma como botones sin texto, que fueron reemplazados por elementos de icono accesibles.
- La falta de contraste en elementos secundarios de las tablas, como el correo electrónico en la vista móvil, tanto en modo claro como en modo oscuro.

Gracias a estas pruebas y a las correcciones implementadas, la aplicación ofrece una experiencia de usuario más consistente, accesible y adaptada a diferentes contextos de uso, reforzando así la calidad final del sistema.

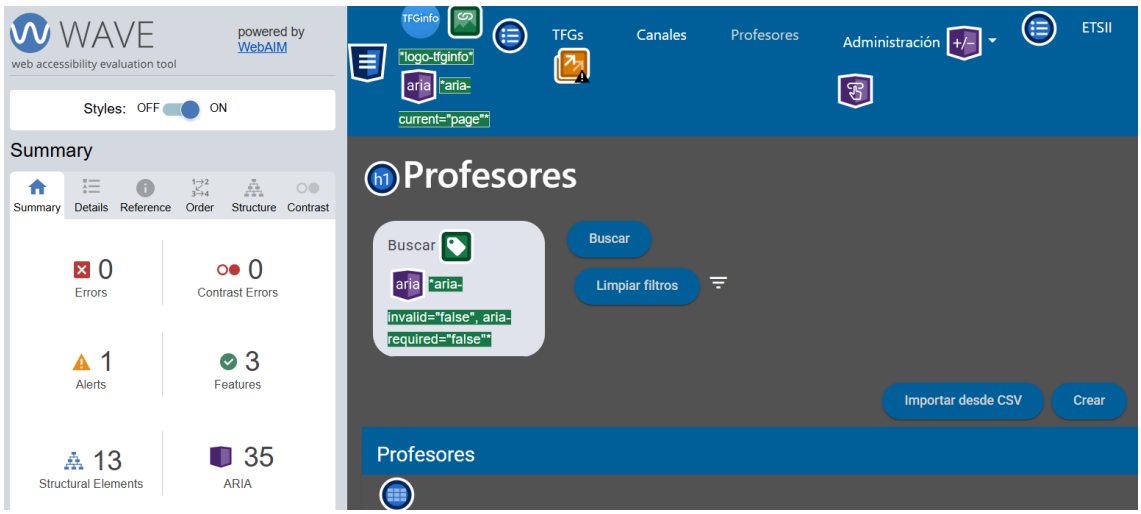


Ilustración 25. Listado de profesores: Oscuro-Escritorio

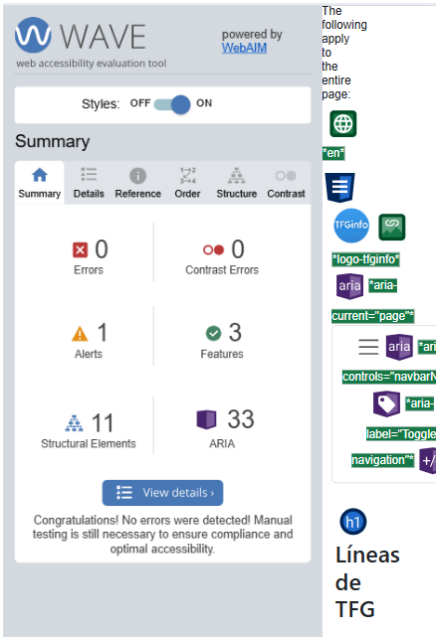


Ilustración 26. Listado de TFGs: Claro-Móvil

Ilustración 27. Detalle de un canal: Claro-Escritorio

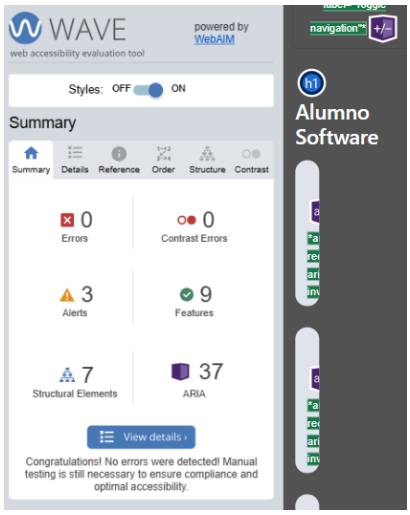


Ilustración 28. Edición de alumno: Oscuro-Móvil

Estos son algunos ejemplos (Ilustraciones 25, 26, 27, 28) de vistas analizadas por WAVE, en modo claro/oscuro y en modo móvil/escritorio. Como se puede apreciar en las imágenes se han evitado los errores y los errores de contraste ya que muestran una puntuación de 0 en los análisis.

Por otra parte, podemos ver que aparecen algunos Alerts. En esta aplicación se ha querido hacer que al pulsar sobre el Logo se redirija al usuario a su vista principal (varía según el rol). Además, se puede ir a esa vista mediante su apartado correspondiente en la cabecera, al poder navegar de varias formas a la misma vista. WAVE lo interpreta como un Alert, ya que no afecta negativamente a la aplicación, pero es algo a tener en cuenta en caso de que se haya hecho por error; en este caso no es así.

Además, en todos los campos de los formularios se usan labels, pero donde hay un selector de Material de Angular aparece un Alert, ya que no reconoce que el label asociado a este, esté siendo correctamente asociado. Por eso, en las vistas de formulario, podemos ver más de un Alert.

5.7.2. Rendimiento, Accesibilidad, Buenas prácticas y SEO (Lighthouse)

Otro aspecto clave para garantizar la calidad de una aplicación web es su rendimiento. Una aplicación rápida y ligera mejora la experiencia del usuario, facilita la navegación y evita abandonos prematuros. Para evaluar este aspecto en **TFGInfo**, se ha utilizado la herramienta FARO (Lighthouse) de Google Chrome, ampliamente reconocida en el ámbito del desarrollo web.

Lighthouse permite auditar aplicaciones web en diferentes categorías, como rendimiento, accesibilidad, buenas prácticas, SEO y compatibilidad con aplicaciones web progresivas (PWA). En lo referente al rendimiento, la herramienta mide métricas relevantes como el tiempo de carga de la primera pintura significativa (*First Contentful Paint*), el tiempo hasta la interactividad (*Time to Interactive*), la velocidad de carga de recursos y la optimización de imágenes, hojas de estilo o scripts.

En las pruebas realizadas sobre TFGInfo, Lighthouse permitió detectar oportunidades de mejora relacionadas con la carga de recursos estáticos y la optimización de imágenes en la interfaz. Asimismo, se comprobó que la aplicación presentaba tiempos de respuesta adecuados en la mayoría de los escenarios, incluso en dispositivos móviles con conexiones simuladas de menor velocidad, lo que asegura que la experiencia de uso no se vea degradada en contextos menos favorables.

Gracias a estas pruebas, se llevaron a cabo ajustes como:

- Compresión y optimización de imágenes para reducir el peso de la interfaz.
- Reducción de ficheros CSS y JavaScript.

Limitaciones en las pruebas de rendimiento en entorno de desarrollo

Durante la evaluación del rendimiento de TFGInfo mediante la herramienta FARO (Lighthouse), se han identificado ciertos aspectos que afectan negativamente a las métricas obtenidas. Sin embargo, es importante destacar que estos resultados no siempre reflejan con exactitud el rendimiento real de la aplicación en un entorno productivo, debido a varias limitaciones propias del entorno de desarrollo y del propio framework utilizado, Angular.

Angular es un framework robusto y modular que facilita el desarrollo de aplicaciones complejas, pero que a su vez introduce una **carga inicial de recursos** (JavaScript, CSS y dependencias internas) que está fuera del control del desarrollador. FARO interpreta esta carga inicial como un retraso en la interactividad de la aplicación, penalizando la puntuación obtenida.

Asimismo, en un entorno de desarrollo:

- no se aplican todas las optimizaciones de compilación que Angular habilita en modo producción, como la minimización, eliminación de código muerto (*tree-shaking*) o el *bundling* eficiente de módulos;
- el servidor local no dispone de configuraciones de caché ni compresión (gzip, brotli), lo que incrementa artificialmente los tiempos de carga medidos;
- las métricas se ven influidas por la herramienta de desarrollo (Angular CLI), que prioriza la recarga rápida de cambios y la depuración por encima del rendimiento.

Por estas razones, el rendimiento observado en desarrollo no debe considerarse definitivo. Una vez desplegada en un servidor de producción con compilación optimizada, caché de recursos estáticos y compresión activada, el comportamiento real de la aplicación es significativamente mejor y más cercano a lo esperado para el usuario final.

En conclusión, aunque FARO pueda señalar advertencias difíciles de resolver directamente desde el código fuente del proyecto, muchas de ellas están vinculadas al funcionamiento interno de Angular y a la naturaleza del entorno de desarrollo, y no representan problemas críticos de la aplicación en producción.

A pesar de ello, se han aplicado los cambios posibles para llevar la puntuación de rendimiento a el valor más alto posible y se ha logrado obtener una puntuación de 100 en las demás categorías.

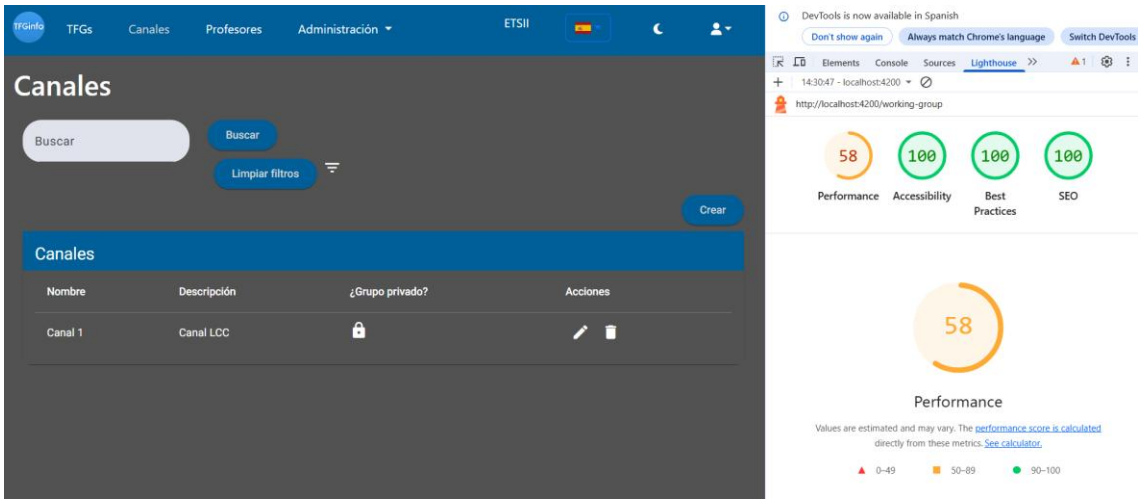


Ilustración 29. Vista de Canales - Evaluación FARO

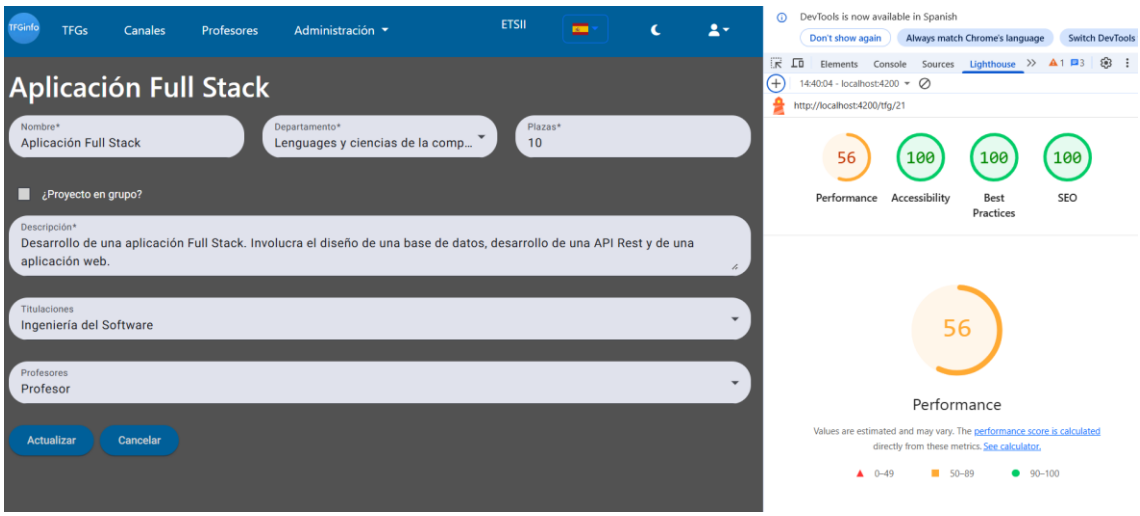


Ilustración 30. Detalle de una línea de TFG - Evaluación FARO

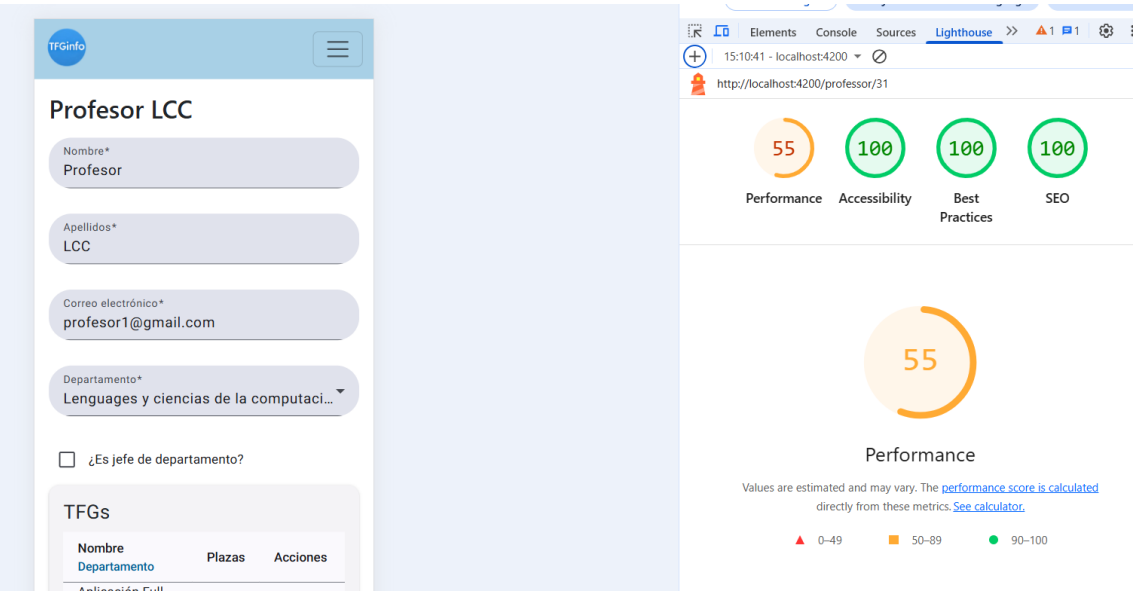


Ilustración 31. Detalle de un profesor - Evaluación FARO

5.7.3. Pruebas con usuarios reales

Con el propósito de validar la usabilidad y el correcto funcionamiento de la aplicación en un entorno próximo al real, se realizaron pruebas con usuarios finales. Debido a las limitaciones de tiempo y recursos disponibles, dichas pruebas se llevaron a cabo con un grupo reducido de participantes (siete usuarios), incluyendo tanto personas con conocimientos previos en gestión académica como usuarios sin experiencia en este ámbito.

Aunque el número de usuarios implicados no fue elevado, los ensayos proporcionaron información valiosa acerca de la experiencia de uso, la claridad de la interfaz y el flujo de navegación. A partir de estas sesiones de prueba se identificaron varias áreas de mejora, que se detallan a continuación:

- **Modelo de departamentos y centros.**

  En una primera versión de la aplicación, se asumió que cada departamento pertenecía a un único centro, premisa que se aplicó también al perfil de los profesores. Tras las pruebas con docentes, se constató que esta premisa no reflejaba adecuadamente la realidad organizativa universitaria. La solución consistió en una refactorización de la lógica y los modelos de datos, permitiendo que un departamento pudiera estar vinculado a varios centros.

- **Estados de los TFGs.**

  Inicialmente, los proyectos únicamente podían estar en tres estados: *pendiente*, *aceptado* o *rechazado*. Los usuarios señalaron la necesidad de contemplar fases intermedias, como el anteproyecto o la presentación final. Se amplió, por tanto, el modelo de estados para reflejar con mayor precisión el ciclo de vida de un TFG.

- **Acceso al detalle de entidades.**

  Algunos usuarios consideraron más intuitivo acceder a la información de entidades (profesores, líneas de TFG, canales, etc.) pulsando directamente sobre el nombre en lugar de usar únicamente los botones "Ver/Editar". Como resultado, se habilitaron ambas opciones de navegación.

- **Importación masiva de datos.**

  En las primeras versiones, la creación de entidades solo era posible mediante formularios individuales. Durante las pruebas, se sugirió la incorporación de una opción para importar registros de forma masiva. Este requerimiento se implementó mediante la carga de ficheros en formato CSV.

- **Envío de credenciales a nuevos usuarios.**

  En la versión inicial, las contraseñas temporales generadas para nuevos usuarios eran proporcionadas únicamente al administrador, quien debía comunicarlas manualmente. Se planteó como mejora la posibilidad de enviar automáticamente dichas credenciales al correo electrónico registrado. Finalmente, se incluyeron ambas modalidades para mayor flexibilidad.

- **Correcciones menores de usabilidad.**

  Durante las pruebas también se identificaron pequeños ajustes que contribuyeron a mejorar la experiencia de usuario:

  - Sustitución del icono de "modo claro" para evitar confusiones con un icono de configuración.
  - Autocompletado automático de la contraseña en el formulario tras su primer cambio.
  - Incorporación de un *tooltip* explicativo al icono de filtros avanzados (embudo), mejorando su comprensión.
  - Implementación de notificaciones internas que informan al usuario tras realizar acciones como reservar un TFG, aceptar/rechazar propuestas o guardar cambios.

5.8. PWA

Una de las grandes ventajas de haber desarrollado TFGInfo con Angular es la posibilidad de transformar la aplicación en una Progressive Web App (PWA), un

modelo híbrido que combina lo mejor de las aplicaciones web y las aplicaciones nativas para escritorio o dispositivos móviles.

¿Qué es una PWA?

Una PWA (Progressive Web App) es una aplicación web que, gracias a ciertas tecnologías, puede comportarse como una aplicación nativa. Se accede inicialmente desde el navegador, pero puede instalarse en el dispositivo del usuario (ya sea ordenador, tablet o smartphone) y ejecutarse como si fuera una app descargada desde una tienda oficial.

Esto significa que TFGInfo puede ofrecer al estudiante, profesor o administrador:

- **Acceso directo desde el escritorio o la pantalla de inicio del móvil** mediante un icono propio.
- **Funcionamiento offline o con conectividad limitada**, gracias al uso de *Service Workers*.
- **Notificaciones push**, útiles para avisar de reservas aceptadas, recordatorios o cambios en los TFGs.
- **Experiencia fluida y similar a una aplicación nativa**, sin necesidad de descargas desde Play Store o App Store.

Angular y su soporte para PWAs

Angular ofrece un módulo oficial llamado @angular/pwa, que simplifica la conversión de una aplicación Angular tradicional en una PWA.

Al añadir este módulo:

- Se genera un Service Worker para gestionar la caché y el acceso offline.
- Se crea un manifest.json, que define el nombre, iconos, colores y comportamiento de la aplicación al instalarse.
- Se habilita la instalación en dispositivos compatibles con solo un clic en el navegador.

Gracias a esta integración, TFGInfo puede ser instalada por los usuarios directamente desde el navegador (Chrome, Edge, Safari, Firefox), lo que reduce las barreras de acceso y elimina la dependencia de tiendas de aplicaciones.

Instalación en dispositivos móviles y escritorio

1. **En móvil (Android o iOS):**
    - El usuario entra en TFGInfo desde el navegador.
    - El navegador detecta que es una PWA y ofrece la opción *"Añadir a la pantalla de inicio"*.
    - A partir de ese momento, TFGInfo se abre como una aplicación independiente, con su icono y sin barra de navegación del navegador.
2. **En escritorio (Windows, macOS, Linux):**
    - Al acceder desde navegadores modernos, aparece la opción *"Instalar aplicación"*.
    - La aplicación se integra en el sistema operativo, con un acceso directo en el menú inicio o en el dock, funcionando como una app nativa.

Beneficios concretos para TFGInfo

- **Acceso instantáneo**: los estudiantes pueden consultar líneas de TFG, reservar plazas o comunicarse en grupos sin necesidad de acceder manualmente al navegador cada vez.
- **Portabilidad total**: funciona en cualquier dispositivo con navegador moderno, sin necesidad de múltiples versiones para iOS, Android o Windows.

6

Conclusiones y trabajos futuros

6.1. Conclusiones

El desarrollo de esta aplicación ha supuesto una experiencia enriquecedora tanto a nivel técnico como organizativo. La creación de un sistema completo, con backend, frontend, autenticación, gestión de datos, notificaciones y comunicación en tiempo real, ha permitido abarcar un ciclo de desarrollo de software muy cercano al que se utiliza en entornos profesionales.

A lo largo del proceso, se han trabajado y aprendido aspectos clave como:

- **Diseño y arquitectura de aplicaciones**

La definición de una estructura clara de controladores, servicios, modelos y componentes ha sido esencial para mantener un código escalable y fácil de mantener.

- **Seguridad y autenticación**

  Implementar roles, permisos y validaciones ha dado una visión práctica de cómo proteger recursos, gestionar sesiones y evitar accesos no autorizados.

- **Comunicación entre backend y frontend**

  El diseño y consumo de API REST ha sido fundamental para conectar las dos partes de la aplicación, garantizando un flujo de datos seguro y consistente.

- **Integración de servicios externos**

  La incorporación de un servicio de correo y notificaciones ha permitido aprender cómo interactuar con sistemas externos y manejar configuraciones sensibles mediante variables de entorno.

- **Gestión de entornos y despliegue**

  Trabajar con configuraciones específicas para desarrollo y producción ha mostrado la importancia de separar entornos y proteger información sensible.

- **Experiencia de usuario y feedback visual**

  Funcionalidades como el snackbar, validaciones en tiempo real y mensajes de error claros han demostrado la importancia de pensar siempre en la experiencia del usuario final.

- **Trabajo iterativo y modular**

  La aplicación se ha desarrollado por módulos independientes pero integrados, lo que ha permitido avanzar de forma progresiva y probar cada parte antes de su despliegue final.

En definitiva, el proyecto no solo ha resultado en una herramienta funcional para la gestión de TFG, líneas de trabajo, grupos y usuarios, sino que también ha sido un ejercicio realista que replica la dinámica de trabajo de un desarrollo profesional: desde el diseño inicial hasta la implementación, pruebas y puesta en producción.

6.2. Trabajos Futuros

Aunque la aplicación desarrollada cumple con los requisitos planteados y ofrece un sistema robusto para la gestión de TFG, líneas, grupos de trabajo y comunicación entre usuarios, existen diversas áreas de mejora y expansión que podrían abordarse en fases posteriores.

Integración con sistemas reales de universidades

Actualmente, la gestión de usuarios se realiza de forma interna en la propia aplicación. Un avance significativo sería la integración con los sistemas de autenticación y gestión de identidades ya existentes en las universidades (por ejemplo, LDAP, Active Directory o Single Sign-On con SAML u OpenID Connect). Esto permitiría:

- que estudiantes y profesores accedieran con sus credenciales institucionales,
- sincronizar automáticamente datos como matrículas, asignaciones de grupos o titulaciones,
- garantizar la coherencia de la información con el sistema académico oficial.

Migración a entornos nativos

Si bien la aplicación es responsive y se adapta a distintos dispositivos, una mejora futura sería adaptar componentes o vistas para Android e iOS, para mostrar una interfaz similar al estándar de cada dispositivo. Esto permitiría:

- Mejor rendimiento y aprovechamiento de funcionalidades específicas de cada sistema operativo.
- Integración con notificaciones push más eficientes.
- Acceso offline y sincronización en segundo plano.
- Experiencia de usuario más fluida gracias a componentes diseñados específicamente para cada plataforma.

Funcionalidades adicionales para grupos de trabajo

Se podrían ampliar las opciones de comunicación en los grupos, incluyendo:

- Videollamadas integradas.
- Compartición de archivos directamente desde la aplicación.
- Historial de mensajes con búsquedas avanzadas.

Módulo de estadísticas y analítica

Incorporar un panel de métricas para administradores y profesores que muestre datos como:

- Número de TFG solicitados, aceptados y rechazados por curso.
- Distribución de líneas de TFG por áreas.
- Evolución del trabajo de los grupos.

Pruebas de carga y optimización del rendimiento

A medida que el sistema se utilice con un mayor número de usuarios concurrentes, sería conveniente realizar pruebas de estrés para optimizar consultas, mejorar la cache de datos y garantizar la escalabilidad del servicio.

Mayor personalización y accesibilidad

Incorporar temas visuales personalizables y mejoras de accesibilidad para cumplir con estándares como WCAG, garantizando que cualquier usuario pueda utilizar la aplicación sin barreras.

Agregar funcionalidades extras de las PWA

Agregar notificaciones push para informar sobre solicitudes aceptadas, fechas límite o cambios importantes, y preparar componentes offline para que los usuarios pueden consultar información previamente cargada (por ejemplo, líneas de TFG disponibles) incluso sin conexión.

Referencias

[1] Persona y Soluciones, "Metodología Agile – Características", Persona y Soluciones, [En línea]. Disponible en: https://personasysoluciones.es/metodologia-agile-caracteristicas . [Accedido: 12-ago-2025].

[2] Sigma AIE, "Sigma Academic", *Sigma AIE*, [En línea]. Disponible en: https://www.sigmaaie.org/es/soluciones/sigma-academic [Accedido: 12-ago-2025].

[3] Universitas XXI, "Universitas XXI", [En línea]. Disponible en: https://universitasxxi.com/co/index.html [Accedido: 12-ago-2025].

[4] Amazon Web Services, "Diferencia entre bases de datos relacionales y no relacionales", *AWS*, [En línea]. Disponible en: https://aws.amazon.com/es/compare/the-difference-between-relational-and-non-relational-databases/ [Accedido: 12-ago-2025].

[5] MySQL, "MySQL", [En línea]. Disponible en: https://www.mysql.com/ [Accedido: 12-ago-2025].

[6] PostgreSQL, "PostgreSQL", [En línea]. Disponible en: https://www.postgresql.org/ [Accedido: 12-ago-2025].

[7] Spring, "Spring Framework", *Spring*, [En línea]. Disponible en: https://spring.io/projects/spring-framework [Accedido: 12-ago-2025].

[8] S. Tiangolo, "FastAPI", [En línea]. Disponible en: https://fastapi.tiangolo.com/ [Accedido: 12-ago-2025].

[9] Microsoft, "Descargar .NET Framework", *Microsoft*, [En línea]. Disponible en: https://dotnet.microsoft.com/es-es/download/dotnet-framework [Accedido: 12-ago-2025].

[10] Microsoft, "Entity Framework Documentation", *Microsoft Learn*, [En línea]. Disponible en: https://learn.microsoft.com/es-es/ef/ [Accedido: 12-ago-2025].

[11] Angular, "Angular", [En línea]. Disponible en: https://angular.dev/ [Accedido: 12-ago-2025].

[12] Meta, “React”, [En línea]. Disponible en: https://es.react.dev/ [Accedido: 12-ago-2025].

[13] Geekflare, “Angular vs React”, *Geekflare*, [En línea]. Disponible en: https://geekflare.com/es/angular-vs-react/ [Accedido: 12-ago-2025].

[14] Microsoft, “Visual Studio Code”, *VS Code*, [En línea]. Disponible en: https://code.visualstudio.com/ [Accedido: 12-ago-2025].

[15] Git, “Documentación de Git”, [En línea]. Disponible en: https://git-scm.com/doc [Accedido: 12-ago-2025].

[16] GitHub, “GitHub”, [En línea]. Disponible en: https://github.com/ [Accedido: 12-ago-2025].

[17] Mermaid, “Mermaid.js”, [En línea]. Disponible en: https://mermaid.js.org/ [Accedido: 12-ago-2025].

[18] Figma, “Figma”, [En línea]. Disponible en: https://www.figma.com/ [Accedido: 12-ago-2025].

[19] Boletín Oficial del Estado, “Ley BOE-A-2018-16673”, *BOE*, [En línea]. Disponible en: https://www.boe.es/buscar/act.php?id=BOE-A-2018-16673 [Accedido: 12-ago-2025].

[20] W3C, “Fundamentos de Accesibilidad Web”, [En línea]. Disponible en: https://www.w3.org/WAI/fundamentals/ [Accedido: 12-ago-2025].

[21] W3C, “Responsive Design” (PDF), [En línea]. Disponible en: https://www.w3.org/2013/Talks/responsive-design.pdf [Accedido: 12-ago-2025].

[22] .NET Foundation, “MailKit – Trabajando con correos”, [.NET Foundation], [En línea]. Disponible en: https://dotnetfoundation.org/news-events/detail/mailkit-working-with-emails [Accedido: 12-ago-2025].

[23] JWT.io, “Introducción a JSON Web Tokens”, [En línea]. Disponible en: https://www.jwt.io/introduction [Accedido: 12-ago-2025].

[24] NuGet, “jose-jwt”, *NuGet*, [En línea]. Disponible en: https://www.nuget.org/packages/jose-jwt/ [Accedido: 12-ago-2025].

[25] Angular, “HttpInterceptor – API Reference”, [En línea]. Disponible en: https://v17.angular.io/api/common/http/HttpInterceptor [Accedido: 12-ago-2025].

[26] Angular, “Route Guards – Guía”, [En línea]. Disponible en: https://angular.dev/guide/routing/route-guards [Accedido: 12-ago-2025].

Apéndice A. Manual de Instalación

Este manual describe el proceso de instalación y despliegue de la aplicación, que se compone de tres elementos principales:

1. **Base de datos (MySQL)**
2. **Servidor Backend (API en .NET)**
3. **Aplicación Frontend (Angular)**

La infraestructura se gestiona de manera conjunta mediante **Docker** y **Docker Compose**, lo que facilita la instalación en diferentes entornos de desarrollo y asegura la portabilidad del sistema.

La aplicación contenida en Docker tiene una información base en la base de datos como profesores, líneas de tfg, centros, etc. Para probar los usuarios se necesitan las siguientes credenciales.

Administrador:

Nombre de usuario: sa@gmail.com

Contraseña: sa

Profesor:

Nombre de usuario: profesor1@gmail.com

Contraseña: profesor1

Alumno:

Nombre de usuario: alumno1@gmail.com

Contraseña: alumno1

Independientemente de estos usuarios, es posible crear nuevos usuarios desde la aplicación usando la cuenta de administración proporcionada.

A.1. Requerimientos

Para realizar la instalación es necesario contar con:

- Docker y Docker Compose instalados en el sistema.
    - Instalar Docker Desktop desde este enlace Docker Desktop: The #1 Containerization Tool for Developers | Docker
    - Una vez instalado es necesario ejecutar Docker Desktop para el uso de contenedores para la aplicación.
- El proyecto completo (El repositorio de github: https://github.com/DavidBuenoCarmona/TFGinfo):
    - Es necesario tener git instalado para clonar el repositorio, es posible descargarlo desde este enlace: Git - Downloads
    - Para clonar el repositorio crearemos una carpeta donde se vaya a guardar y abriremos una terminal del sistema.
    - En la ubicación de esta carpeta ejecutaremos el siguiente comando:

```
// Para clonar el repositorio
$ git clone https://github.com/DavidBuenoCarmona/TFGinfo.git
```

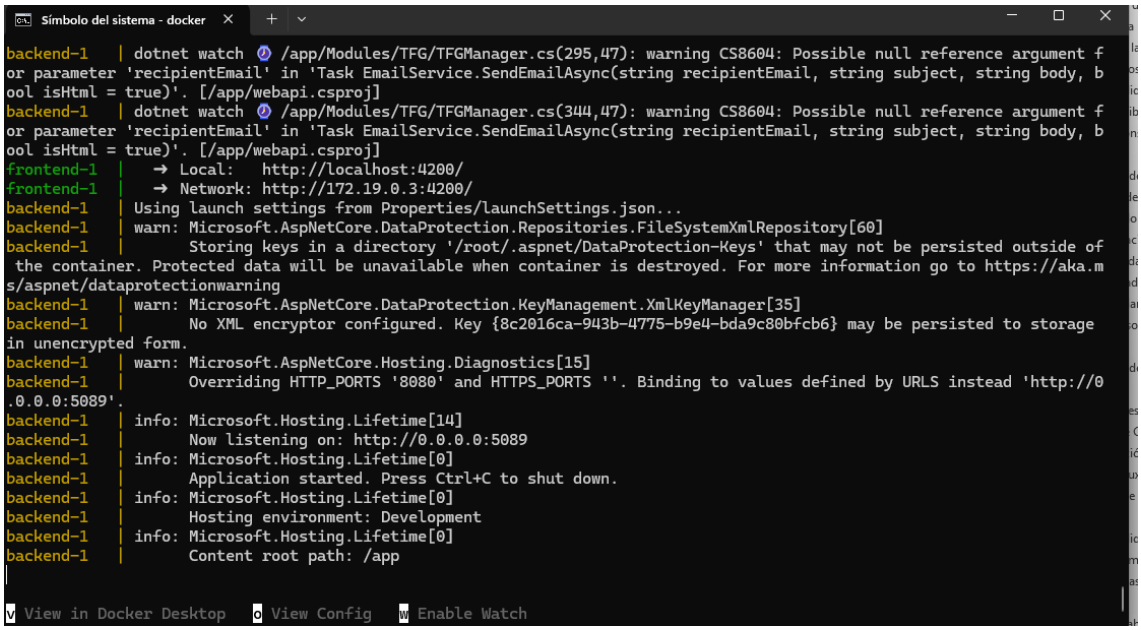
- Ejecutar el archivo de orquestación global
    - Docker Desktop debe estar iniciado para que funcione la llamada a compose up.
    - Previamente se ha clonado el repositorio en una carpeta mediante una terminal, usando esa misma terminal accedemos al proyecto

```
// Accedemos a la carpeta del repositorio
$ cd TFGinfo\tfginfo\src
```

- Una vez en la carpeta del proyecto podemos ejecutar Docker compose up para comenzar con el procesamiento en contenedores

```
$ Docker compose up
```

El resultado debería ser similar a la siguiente pantalla, representado que todo se ha ejecutado correctamente:



Se deberá poder acceder desde un navegador a la dirección http://localhost:4200/ y debería aparecer la pantalla de inicio de sesión, en la cual se podrá acceder con los usuarios y contraseñas que aparecen en la introducción del manual.

A.2. Estructura del proyecto

El proyecto se organiza en tres directorios principales:

- Una carpeta para el **frontend**, que contiene la aplicación Angular junto con su Dockerfile.
- Una carpeta para el **backend**, que incluye la API desarrollada en .NET y su correspondiente Dockerfile.

- Una carpeta destinada a la **base de datos**, donde se ubican los scripts SQL para la creación de la base de datos y carga inicial de datos.

En la raíz del proyecto se encuentra el archivo de **Docker Compose**, que unifica todos los servicios en un mismo entorno.

A.3. Componentes principales

Base de datos (MySQL)

La base de datos se despliega en un contenedor independiente con la imagen oficial de MySQL. Está configurada para:

- usar credenciales definidas mediante variables de entorno,
- exponer un puerto al sistema anfitrión para conexiones externas,
- ejecutar automáticamente los scripts de inicialización que crean las tablas y cargan los datos básicos.

Servidor Backend (API .NET)

El backend se construye a partir de un Dockerfile que:

- utiliza la imagen oficial de .NET SDK para permitir el desarrollo con recarga en caliente,
- restaura dependencias, compila y ejecuta la aplicación en un contenedor,
- expone el puerto donde se publica la API,
- está conectado con la base de datos a través de la cadena de conexión definida en las variables de entorno.

Aplicación Frontend (Angular)

El frontend se construye con un Dockerfile que:

- se basa en la imagen oficial de Node,

- instala las dependencias de Angular y de la aplicación,
- expone el puerto correspondiente al servidor de desarrollo de Angular,
- permite trabajar con recarga en tiempo real para facilitar el desarrollo.

A.4. Orquestación con Docker Compose

El archivo de orquestación global define los tres servicios: **frontend**, **backend** y **base de datos**. Este archivo:

- configura los puertos de exposición hacia el host,
- sincroniza volúmenes para que los cambios en el código del host se reflejen dentro de los contenedores,
- declara dependencias para que los servicios arranquen en el orden correcto (por ejemplo, el backend espera a que la base de datos esté disponible),
- establece las variables de entorno necesarias, como las credenciales de la base de datos y la cadena de conexión del backend.

A.5. Flujo de despliegue

El proceso de despliegue de la aplicación sigue el siguiente flujo:

1. **Inicialización del entorno**:

   Se inicia Docker Compose desde la raíz del proyecto. Esto descarga las imágenes necesarias, construye las imágenes personalizadas para frontend y backend, y levanta los tres contenedores en ejecución.
2. **Inicialización de la base de datos**:

   Al iniciarse por primera vez, el contenedor de MySQL ejecuta automáticamente los scripts SQL de la carpeta correspondiente, creando las tablas y cargando los datos iniciales.
3. **Arranque del backend**:

Una vez la base de datos está en marcha, el backend se conecta a ella utilizando la cadena de conexión definida. La API queda disponible en el puerto configurado.

4. **Arranque del frontend**:

   La aplicación Angular se ejecuta en modo de desarrollo, sirviendo la interfaz web y comunicándose con la API para obtener y enviar datos.

A.6. Acceso a servicios

- **Frontend (Angular)**: disponible en el puerto asignado en el host, accesible desde un navegador web.
- **Backend (API .NET)**: expuesto en el puerto correspondiente, accesible mediante clientes HTTP o directamente desde el frontend.
- **Base de datos (MySQL)**: accesible desde el propio backend y, en caso de ser necesario, también desde clientes externos de gestión de bases de datos mediante el puerto mapeado.

A.7. Mantenimiento y reinicio

- Los servicios pueden detenerse y reiniciarse fácilmente desde Docker Compose.
- Si se requiere reinstalar la base de datos desde cero, basta con eliminar el volumen asociado para que, en el próximo arranque, los scripts SQL vuelvan a ejecutarse y reconstruyan toda la estructura con los datos de prueba.
- Los contenedores de frontend y backend admiten cambios en el código sin necesidad de reconstruir la imagen completa, gracias al uso de volúmenes y recarga en caliente.

Apéndice B. Manual de Usuario

Este manual de usuario muestra el uso de la aplicación, explicando de manera clara y detallada cada uno de sus componentes y funcionalidades. A lo largo de esta sección, se describirán las distintas partes de la interfaz, el propósito de cada módulo y los pasos necesarios para realizar las acciones más habituales. Desde la autenticación y gestión de usuarios, hasta la administración de líneas de TFG, solicitudes, grupos de trabajo y notificaciones, el manual busca ofrecer una referencia completa que permita a cualquier usuario, independientemente de su rol, aprovechar al máximo las capacidades de la aplicación.

B.1. Inicio de sesión

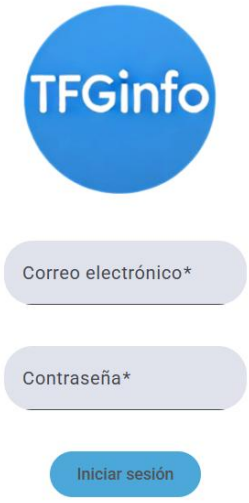


Ilustración 32. Vista de inicio de sesión

La pantalla de inicio de sesión (Illustración 32) es la única vista disponible antes de que un usuario se autentique en el sistema. En esta pantalla se muestra un formulario que solicita los siguientes datos:

- **Correo electrónico** del usuario.
- **Contraseña**.

Una vez ingresados los datos, el sistema procesará la solicitud y podrá producirse una de las siguientes situaciones:

Datos incorrectos

Si el correo electrónico o la contraseña son incorrectos, se mostrará el mensaje de error:

"Usuario o contraseña incorrecto"

Este mensaje está diseñado para no revelar si la cuenta de usuario existe o no, protegiendo así la seguridad del sistema.

Datos correctos – Primer inicio de sesión

Si las credenciales son correctas, pero es la primera vez que el usuario accede a la aplicación, se mostrará un modal solicitando el cambio de contraseña. Esto ocurre porque el usuario está utilizando la contraseña temporal generada por el sistema. Será obligatorio reemplazarla por una contraseña personal antes de continuar.

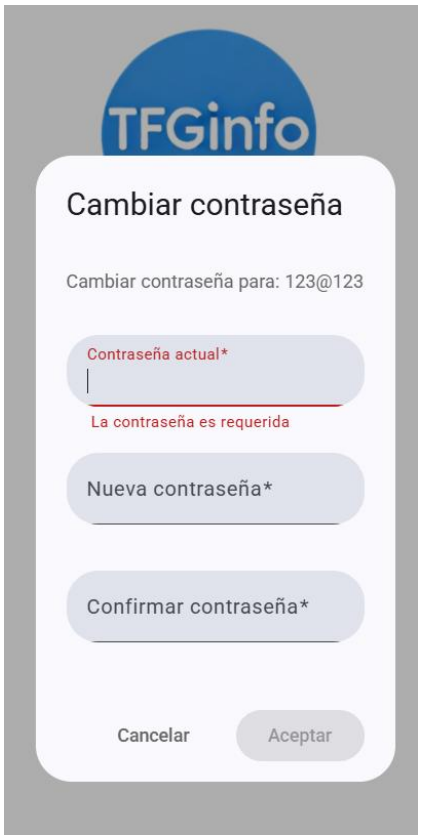


Ilustración 33. Vista de cambio de contraseña

Datos correctos – Inicio de sesión normal

Si las credenciales son correctas y no es el primer inicio de sesión, el sistema permitirá el acceso de forma normal. El usuario será redirigido a la vista principal de la aplicación, sin necesidad de cambiar su contraseña.

B.2. Mis reservas

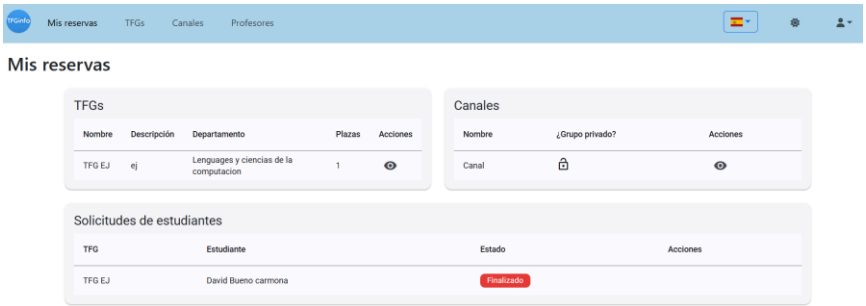


Ilustración 34. Vista de Mis Reservas

La vista de Mis Reservas es la vista principal para los usuarios con rol de Alumno y Profesor. Los usuarios con rol de Administrador no tienen acceso a esta vista, ya que no están relacionados con TFGs ni canales como los otros roles.

Cabecera de la aplicación

Esta cabecera permanece visible en toda la sesión una vez que el usuario ha iniciado sesión. Funciona como el método principal de navegación dentro de la aplicación.

Listado de TFGs

Este listado se muestra tanto a alumnos como a profesores, con diferencias según el rol:

- Alumnos: Visualizan las líneas de TFG que han solicitado.
- Profesores**:** Visualizan las líneas de TFG en las que está asignado como posible tutor o co-tutor.

Ambos roles pueden visualizar la línea de TFG, lo que redirige a la vista de edición correspondiente.

Canales

Muestra los canales a los que pertenece el usuario, independientemente de su rol (alumno o profesor). Al pulsar sobre un canal, el usuario accede a la vista de edición de dicho canal.

Solicitudes de estudiantes (Exclusivo para profesores)

En esta sección, los profesores pueden ver los TFGs activos de los estudiantes en los que él actúa como tutor o co-tutor. Las acciones disponibles dependen del estado del TFG:

- Pendiente de aceptar: El profesor puede aceptar o rechazar la solicitud.

- Aceptado: El profesor puede avanzar el estado del TFG siguiendo el flujo habitual:

Aceptado → Anteproyecto entregado → Anteproyecto aceptado → Presentado → Finalizado

B.3. Cabecera

Ilustración 35. Cabecera de la aplicación

La cabecera de la aplicación es un elemento constante durante toda la navegación, ya que facilita el acceso a las principales secciones y funcionalidades. Esta imagen muestra la vista de un administrador, donde no podemos ver el apartado de Mis Reservas, pero sí el de Administración.

Acciones disponibles en la cabecera

Logo

- Al hacer clic sobre él, el usuario será redirigido a su vista principal correspondiente según su rol.

Mis Reservas

- Disponible solo para usuarios con rol de Profesor o Alumno.
- Al seleccionarlo, carga la vista Mis Reservas.

TFGs

- Accesible para todos los usuarios, sin importar el rol.
- Redirige a la vista del listado de líneas de TFG.

Canales

- Accesible para todos los usuarios, sin importar el rol.
- Muestra el listado de canales.

Profesores

- Accesible para todos los usuarios, sin importar el rol.
- Redirige a la vista del listado de profesores.

Administración

- Exclusivo para usuarios con rol de Administrador.
- Se presenta como un menú desplegable (dropdown) que da acceso a las entidades internas de la universidad:
    - Centros
    - Departamentos
    - Titulaciones
    - Estudiantes
- Cada opción redirige a la vista correspondiente.

Centro Seleccionado

- Elemento informativo exclusivo para administradores.
- Muestra el centro actualmente seleccionado, el cual determina la información visible sobre canales, centros, profesores y TFGs.
- Al hacer clic, redirige a la vista de selección de centro para cambiar esta configuración.

Selector de idioma

- Disponible para todos los usuarios.
- Permite cambiar el idioma de la aplicación entre español e inglés.

Selector de tema

- Permite alternar entre tema claro (representado con un sol) y tema oscuro (representado con una luna).

Dropdown de perfil

- Para usuarios con rol de Profesor o Alumno, permite:
    - Visualizar su perfil.
    - Cerrar sesión.
- Para usuarios con rol de Administrador, permite:

- Cerrar sesión.
- Modificar el centro seleccionado.

B.4. Listado de líneas de TFG

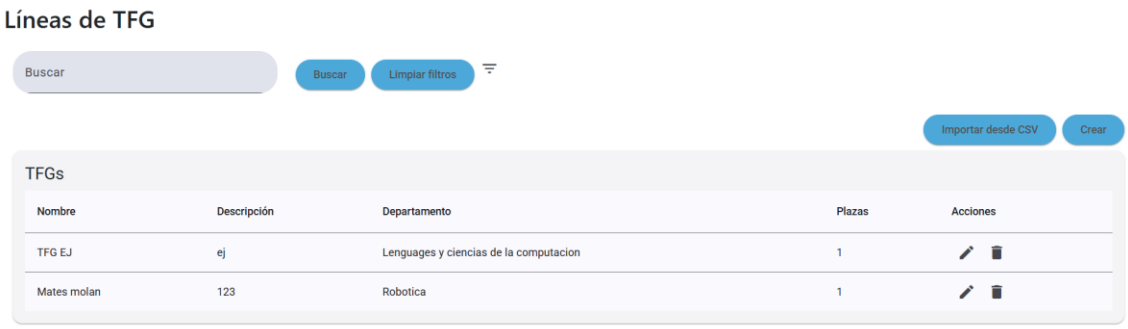


Ilustración 36. Vista del listado de líneas de TFG

Este apartado muestra el listado de líneas de TFG. Aunque la vista es la misma para todos los usuarios, el contenido cambia según el rol y características del usuario.

- **Administradores:**

  Al ingresar por primera vez, no verán ninguna línea. Aparecerá un mensaje en la tabla indicando que deben seleccionar un centro para visualizar los TFGs disponibles en dicho centro. Esto se debe a que los administradores tienen acceso a todos los centros de la universidad y deben elegir uno para cargar el listado correspondiente.

- **Alumnos:**

  Visualizan los TFGs a los que pueden solicitar unirse, filtrados automáticamente según su titulación. Esto significa que la aplicación elimina todas las líneas de TFG cuya titulación no coincide con la del alumno.

- **Profesores:**

  Pueden visualizar todas las líneas de TFG asociadas a su departamento.

De manera automática, se filtran las líneas que no estén relacionadas con el departamento del profesor que está consultando la vista.

Filtros disponibles

Como en todos los listados de la aplicación, aquí se dispone de filtros para facilitar la búsqueda:

- **Filtro general:**

  Permite buscar por cualquier parámetro visible en la tabla, en este caso:

  - Nombre
  - Descripción
  - Departamento

  Al pulsar el botón **Buscar**, se aplican los filtros y se actualiza el listado. El botón **Limpiar filtros** elimina los filtros aplicados.

- **Filtros avanzados:**

  Se acceden pulsando el icono del embudo. Al desplegarlos, aparecen campos para cada parámetro mostrado en la tabla:

  - Nombre
  - Descripción
  - Departamento
  - Plazas

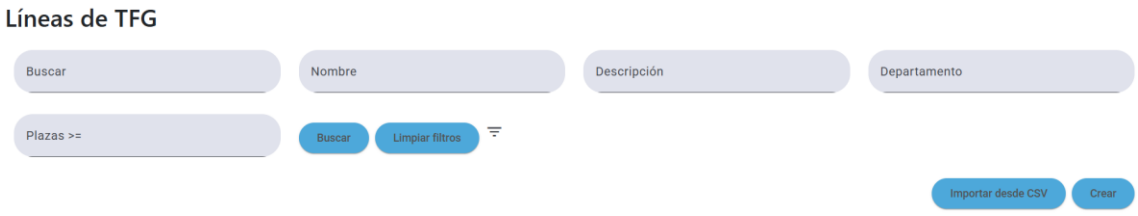


Ilustración 37. Filtros avanzados líneas de TFG

Acciones disponibles según rol

- **Administradores:**
    - Pueden crear nuevas líneas de TFG pulsando el botón **Crear**, que redirige al formulario de creación.
    - Pueden importar líneas en masa desde un archivo CSV, lo cual abre un modal para subir el archivo.
    - Pueden editar o eliminar líneas existentes.
- **Profesores y Alumnos:**
    - Solo pueden visualizar los detalles de cada línea de TFG.

B.5. Canales

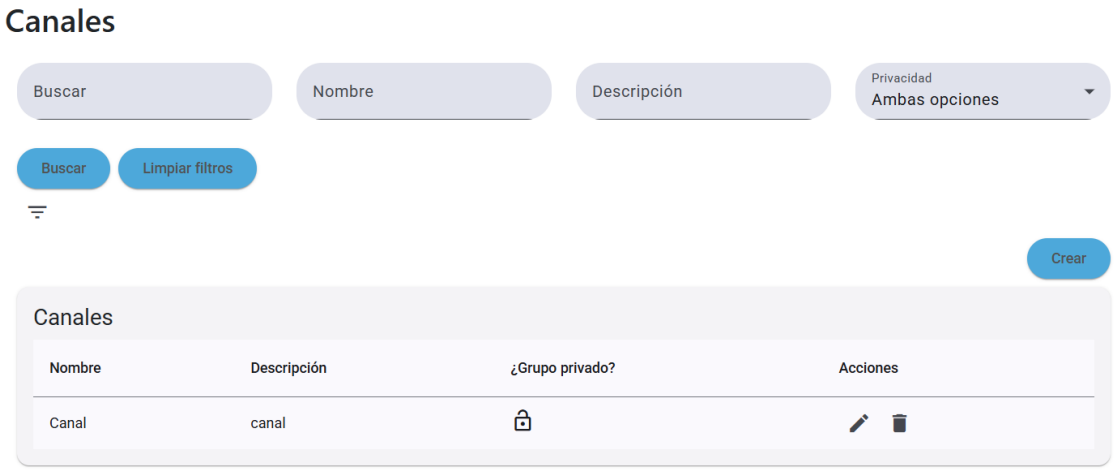


Ilustración 38. Vista de listado de canales

Listado de Canales

Este apartado muestra el listado de canales disponibles para el usuario. Aunque la vista es común para todos, el contenido se ajusta según el rol y permisos del usuario.

- **Administradores:**

  Inicialmente no visualizarán canales hasta seleccionar un centro. Aparecerá

un mensaje en la tabla indicando que deben seleccionar un centro para mostrar los canales asociados a ese centro.

- **Alumnos y Profesores:**

  Visualizan únicamente los canales de su escuela/centro

Información mostrada en la tabla

En el listado se presentan los siguientes datos por cada canal:

- **Nombre**
- **Descripción**
- **Privacidad**
  - Indicada mediante un icono de candado:
    - **Candado cerrado:** Canal privado.
    - **Candado abierto:** Canal público.

Filtros disponibles

Como en todos los listados de la aplicación, se pueden aplicar filtros para facilitar la búsqueda:

- **Filtro general:**

  Permite buscar por cualquiera de los campos visibles en la tabla:
  - Nombre
  - Descripción

  Al pulsar el botón **Buscar**, se aplican los filtros y se actualiza el listado.

  El botón **Limpiar filtros** elimina cualquier filtro activo.
- **Filtros avanzados:**

  Se activan mediante el icono del embudo y muestran campos para filtrar por:

- Nombre
- Descripción
- Privacidad

Acciones disponibles según rol

- **Administradores:**
  - Pueden crear nuevos canales mediante el botón **Crear**, que redirige al formulario de creación.
  - Tienen permisos para editar o eliminar canales existentes.
- **Profesores:**
  - Pueden crear nuevos canales mediante el botón **Crear**, que redirige al formulario de creación.
- **Alumnos:**
  - Solo pueden visualizar los detalles de los canales a los que pertenecen.

B.6. Profesores

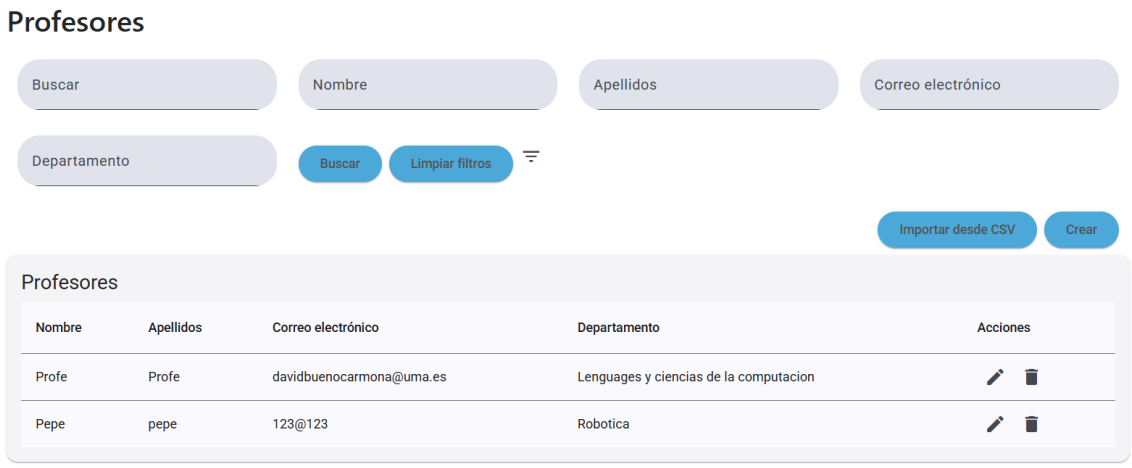


Ilustración 39. Vista del listado de profesores

Este apartado muestra el listado de profesores de la universidad. Aunque la vista es común para todos los usuarios, el contenido y las acciones disponibles dependen del rol.

- **Administradores:**

  Pueden visualizar todos los profesores del centro actualmente seleccionado. Sino se ha seleccionado un centro, no se mostrará ningún registro y aparecerá un mensaje indicando que es necesario seleccionar un centro para cargar el listado.
- **Alumnos y Profesores:**

  Pueden visualizar el listado de profesores correspondientes a su centro.

Información mostrada en la tabla

Cada fila del listado contiene la siguiente información:

- **Nombre**
- **Apellido**
- **Departamento**
- **Correo electrónico**

Filtros disponibles

Como en todos los listados de la aplicación, se cuenta con herramientas de filtrado:

- **Filtro general:**

  Permite buscar por cualquiera de los campos visibles en la tabla:
  - Nombre
  - Apellido
  - Departamento
  - Correo electrónico

    El botón **Buscar** aplica los filtros y actualiza el listado, mientras que **Limpiar filtros** los elimina.

- **Filtros avanzados:**

  Accesibles mediante el icono del embudo, permiten filtrar de forma individual por:

  - Nombre
  - Apellido
  - Departamento
  - Correo electrónico

Acciones disponibles según rol

- **Administradores:**
  - Pueden crear nuevos registros de profesores mediante el botón **Crear**, que redirige al formulario de creación.
  - Pueden importar listados de profesores desde un archivo CSV, abriendo un modal para subir el archivo.
  - Tienen permisos para editar o eliminar registros de profesores.
- **Alumnos y Profesores:**
  - Solo pueden visualizar los datos de los profesores.

B.7. Estudiantes

Ilustración 40. Vista del listado de estudiantes

Este apartado muestra el listado de estudiantes registrados en la universidad, por lo que no es necesario seleccionar un centro para visualizar los alumnos de este. Es exclusivo para usuarios con rol de Administrador, por lo que alumnos y profesores no tienen acceso a esta vista.

Información mostrada en la tabla

Cada registro del listado contiene los siguientes datos:

- **Nombre**
- **Apellido**
- **Titulación (Centro)**
- **Correo electrónico**

Filtros disponibles

Como en todos los listados de la aplicación, se incluyen herramientas de filtrado:

- **Filtro general:**

  Permite buscar por cualquiera de los campos visibles en la tabla:

  - Nombre
  - Apellido
  - Titulación
  - Centro
  - Correo electrónico

  El botón **Buscar** aplica los filtros y actualiza el listado, mientras que **Limpiar filtros** los elimina.

- **Filtros avanzados:**

  Accesibles mediante el icono del embudo, permiten filtrar individualmente por:

  - Nombre
  - Apellido

- Titulación
- Centro
- Correo electrónico

Acciones disponibles

Al ser una vista exclusiva para administradores, se disponen de todas las acciones de gestión:

- **Crear** un nuevo estudiante mediante el botón correspondiente, que redirige al formulario de creación.
- **Importar** estudiantes en masa desde un archivo CSV, abriendo un modal para la carga del archivo.
- **Editar** cualquier registro existente.
- **Eliminar** registros del listado.

B.8. Departamentos

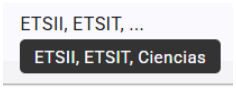
Ilustración 41. Vista del listado de departamentos

Este apartado muestra el listado de departamentos registrados en la universidad. Es exclusivo para usuarios con rol de Administrador, por lo que alumnos y profesores no tienen acceso a esta vista.

Información mostrada en la tabla

Cada registro del listado contiene los siguientes datos:

- **Nombre**
- **Acrónimo**
- **Centros asociados** (lista de centros vinculados al departamento) En caso de haber más de dos centros asociados se verán puntos suspensivos, que al pasar el cursor por encima mostrará todos los centros.



Filtros disponibles

Como en todos los listados de la aplicación, se incluyen herramientas de filtrado:

- **Filtro general:**

  Permite buscar por cualquiera de los campos visibles en la tabla:

  - Nombre
  - Acrónimo
  - Centros asociados

  El botón **Buscar** aplica los filtros y actualiza el listado, mientras que **Limpiar filtros** los elimina.

- **Filtros avanzados:**

  Accesibles mediante el icono del embudo, permiten filtrar individualmente por:

  - Nombre
  - Acrónimo
  - Centros asociados

Acciones disponibles

Al ser una vista exclusiva para administradores, se dispone de todas las acciones de gestión:

- **Crear** un nuevo departamento mediante el botón correspondiente, que redirige al formulario de creación.
- **Importar** departamentos en masa desde un archivo CSV, abriendo un modal para la carga del archivo.
- **Editar** cualquier registro existente.
- **Eliminar** registros del listado.

B.9. Titulaciones

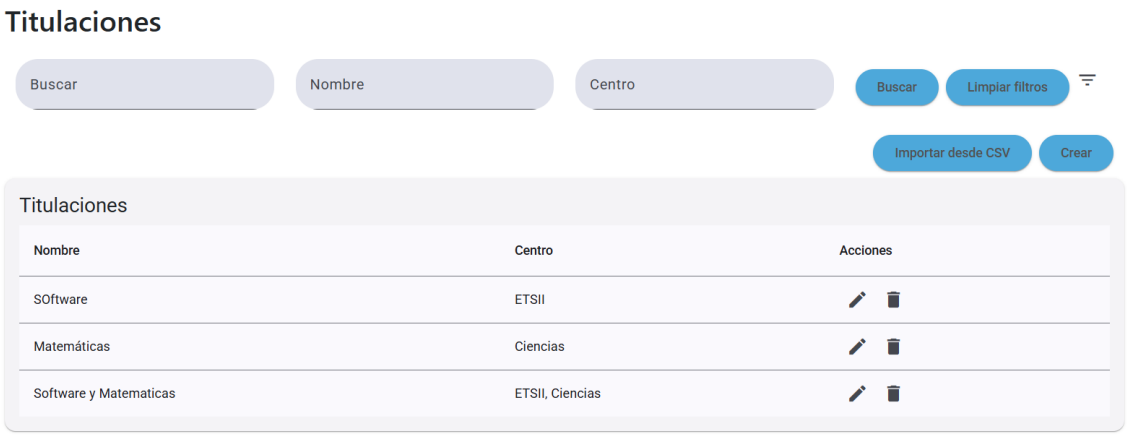


Ilustración 42. Vista del listado de titulaciones

Este apartado muestra el listado de titulaciones registradas en la universidad. Es exclusivo para usuarios con rol de Administrador, por lo que alumnos y profesores no tienen acceso a esta vista. Una titulación se puede relacionar hasta con dos centros, en caso de que sea un doble grado cuyas titulaciones pertenezcan a centros diferentes.

Información mostrada en la tabla

Cada registro del listado contiene los siguientes datos:

- **Nombre**
- **Centro** (al que pertenece la titulación)

Filtros disponibles

Como en todos los listados de la aplicación, se incluyen herramientas de filtrado:

- **Filtro general:**

  Permite buscar por cualquiera de los campos visibles en la tabla:

  - Nombre
  - Centro

    El botón **Buscar** aplica los filtros y actualiza el listado, mientras que **Limpiar filtros** los elimina.
- **Filtros avanzados:**

  Accesibles mediante el icono del embudo, permiten filtrar individualmente por:

  - Nombre
  - Centro

Acciones disponibles

Al ser una vista exclusiva para administradores, se dispone de todas las acciones de gestión:

- **Crear** una nueva titulación mediante el botón correspondiente, que redirige al formulario de creación.
- **Importar** titulaciones en masa desde un archivo CSV, abriendo un modal para la carga del archivo.
- **Editar** cualquier registro existente.
- **Eliminar** registros del listado.

B.10. Centros

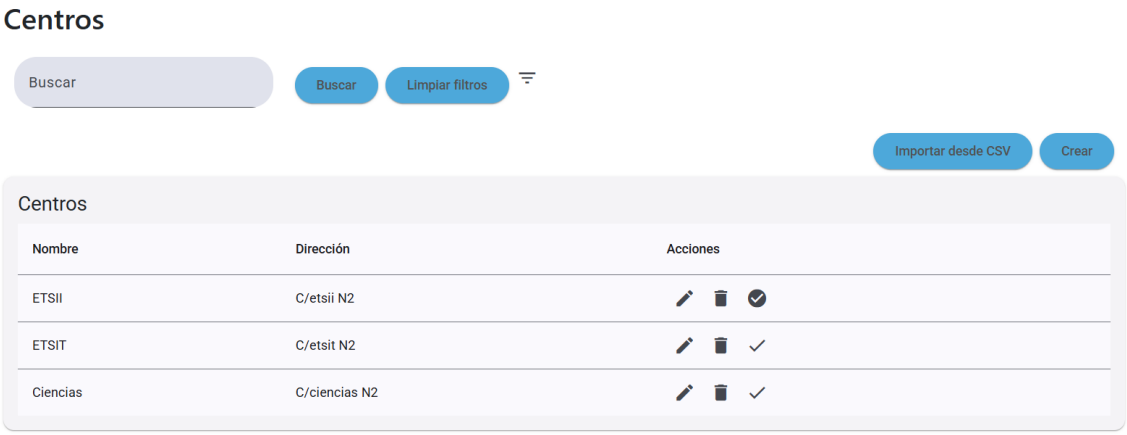


Ilustración 43. Vista del listado de centros

Este apartado muestra el listado de centros registrados en la universidad. Es **exclusivo para usuarios con rol de Administrador**, por lo que alumnos y profesores no tienen acceso a esta vista.

Información mostrada en la tabla

Cada registro del listado contiene los siguientes datos:

- **Nombre**
- **Dirección**

Filtros disponibles

Como en todos los listados de la aplicación, se incluyen herramientas de filtrado:

- **Filtro general:**

  Permite buscar por cualquiera de los campos visibles en la tabla:

  - Nombre
  - Dirección

  El botón **Buscar** aplica los filtros y actualiza el listado, mientras que **Limpiar filtros** los elimina.

- **Filtros avanzados:**

  Accesibles mediante el icono del embudo, permiten filtrar individualmente por:

  - Nombre
  - Dirección

Acciones disponibles

Al ser una vista exclusiva para administradores, se dispone de todas las acciones de gestión:

- **Crear** un nuevo centro mediante el botón correspondiente, que redirige al formulario de creación.
- **Importar** centros en masa desde un archivo CSV, abriendo un modal para la carga del archivo.
- **Editar** cualquier registro existente.
- **Eliminar** registros del listado.
- **Seleccionar centro** para el contexto de la aplicación: esta acción define el centro activo para visualizar y gestionar la información relacionada (canales, departamentos, titulaciones, TFGs, etc.).

B.11. Creación y edición de líneas de TFG

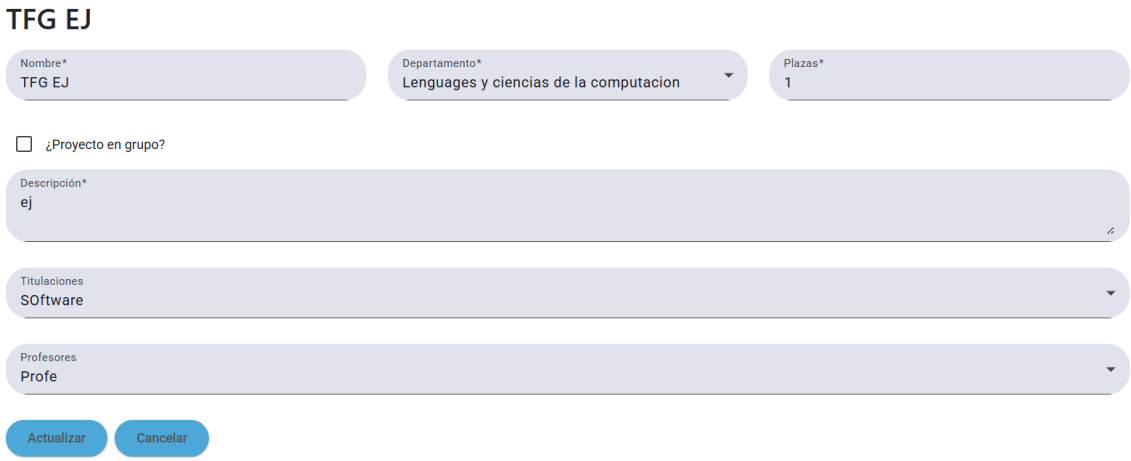


Ilustración 44. Vista del detalle de una línea de TFG

La vista del Formulario de Líneas de TFG permite a los administradores crear o editar una línea de TFG. Dependiendo del contexto, el formulario puede presentarse en modo creación o edición:

- **Modo creación:** El encabezado muestra *"Nuevo TFG"* y el formulario está vacío para introducir nuevos datos.
- **Modo edición:** El encabezado muestra el nombre del TFG y carga los datos existentes para su modificación.

Campos del formulario

1. Nombre

- **Tipo:** Campo de texto.
- **Obligatorio:** Sí.
- **Descripción:** Nombre de la línea de TFG.
- **Validación:** Si se deja vacío, muestra el error **"Nombre requerido"**.

2. Departamento

- **Tipo:** Selector desplegable.
- **Fuente de datos:** Lista de departamentos disponibles.

- **Obligatorio:** Sí.
- **Validación:** Si no se selecciona, muestra el error **"Departamento requerido"**.

3. Plazas

- **Tipo:** Campo numérico.
- **Obligatorio:** Sí.
- **Descripción:** Número de plazas disponibles para la línea de TFG.
- **Validación:** Si se deja vacío o no es un número válido, muestra el error **"Plazas requeridas"**.

4. Es en grupo

- **Tipo:** Casilla de verificación (*checkbox*).
- **Descripción:** Indica si el TFG puede ser desarrollado en grupo.

5. Descripción

- **Tipo:** Área de texto (*textarea*).
- **Obligatorio:** Sí.
- **Descripción:** Resumen o detalle del TFG.
- **Validación:** Si se deja vacío, muestra el error **"Descripción requerida"**.

6. Titulaciones

- **Tipo:** Selector múltiple.
- **Visible:** Solo en **modo edición**.
- **Fuente de datos:** Lista de titulaciones disponibles.
- **Descripción:** Permite asociar la línea de TFG a una o varias titulaciones.

7. Profesores

- **Tipo:** Selector múltiple (*mat-select*).
- **Visible:** Solo en **modo edición**.

- **Fuente de datos:** Lista de profesores disponibles.
- **Descripción:** Permite asignar uno o varios tutores a la línea de TFG.

Acciones disponibles

Las acciones varían según el rol del usuario que accede a esta vista:

- **Administradores:**
    - **Crear / Actualizar:**
        - El botón se habilita únicamente si el formulario es válido.
        - En modo creación, el botón muestra **"Crear"**.
        - En modo edición, el botón muestra **"Actualizar"**.
- **Estudiantes:**
    - **Solicitar TFG:** Disponible solo para estudiantes y en modo edición. Permite enviar una solicitud para unirse a la línea de TFG.
- **Todos los usuarios:**
    - **Cancelar:** Cierra el formulario y regresa a la vista anterior.

Validaciones y comportamiento

- El formulario es **reactivo** (*Reactive Form*), lo que significa que la validación y estado se gestionan de forma automática en tiempo real.
- Los mensajes de error se muestran debajo de cada campo obligatorio si no se cumple la validación correspondiente.
- El botón de creación/actualización se deshabilita mientras el formulario sea inválido.

B.12. Formulario de solicitud de TFG

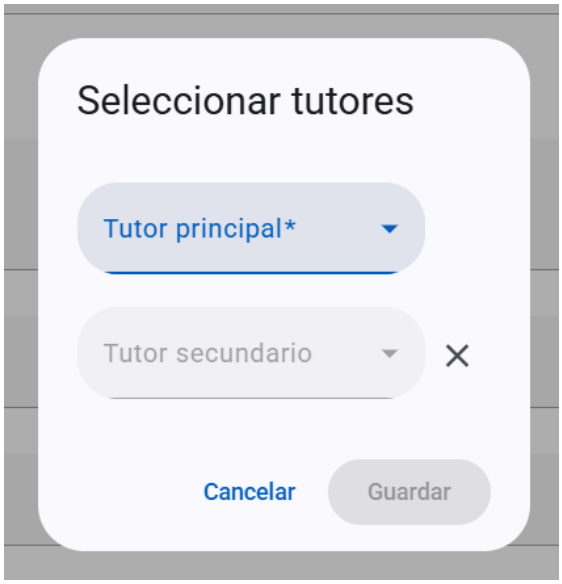


Ilustración 45. Modal de una solicitud de un TFG

La vista Formulario de Selección de Tutores de TFG permite asignar tutores principales y secundarios a una solicitud de TFG. Este formulario se muestra en un modal (diálogo) y está diseñado para que el usuario (alumno) pueda definir la tutoría de su TFG, incluyendo la opción de tutor externo.

Campos del formulario

1. Tutor Principal

- **Tipo:** Selector desplegable
- **Fuente de datos:** Lista de profesores disponibles en la línea de TFG.
- **Obligatorio:** Sí.
- **Validación:** Si no se selecciona, muestra el error **"Tutor principal requerido"**.

2. Tutor Secundario (opcional)

- **Tipo:** Selector desplegable (*mat-select*).

- **Fuente de datos:** Lista filtrada de profesores, sin el profesor seleccionado como tutor principal.
- **Opcional:** Sí.
- **Opciones adicionales:**
    - Seleccionar un tutor interno (profesor listado).
    - Seleccionar un tutor externo (opción resaltada con texto en negrita).
- **Acción adicional:** Botón con icono de "cerrar" para eliminar el tutor secundario seleccionado.

3. Tutor Externo – Nombre *(condicional)*

- **Visible:** Solo si se elige la opción de **tutor externo** en el campo *Tutor Secundario*.
- **Tipo:** Campo de texto.
- **Obligatorio:** Sí, si está visible.
- **Validación:** Si se deja vacío, muestra el error **"Nombre requerido"**.

4. Tutor Externo – Correo electrónico *(condicional)*

- **Visible:** Solo si se elige la opción de **tutor externo** en el campo *Tutor Secundario*.
- **Tipo:** Campo de texto.
- **Obligatorio:** Sí, si está visible.
- **Validación:** Si se deja vacío, muestra el error **"Correo electrónico requerido"**.

Acciones disponibles

- **Cancelar:**

  Botón que cierra el formulario sin guardar cambios.
- **Guardar:**

- Disponible solo cuando el formulario es válido.
- Al pulsarlo, se envía el formulario y se actualizan los tutores del TFG.

Validaciones y comportamiento

- El formulario es **reactivo** (*Reactive Form*).
- Los campos obligatorios muestran mensajes de error si no cumplen la validación.
- La aparición de los campos de **tutor externo** depende de la selección en *Tutor Secundario*.
- El botón de **Guardar** permanece deshabilitado hasta que todos los campos requeridos estén correctos.

B.13. Creación y edición de canales

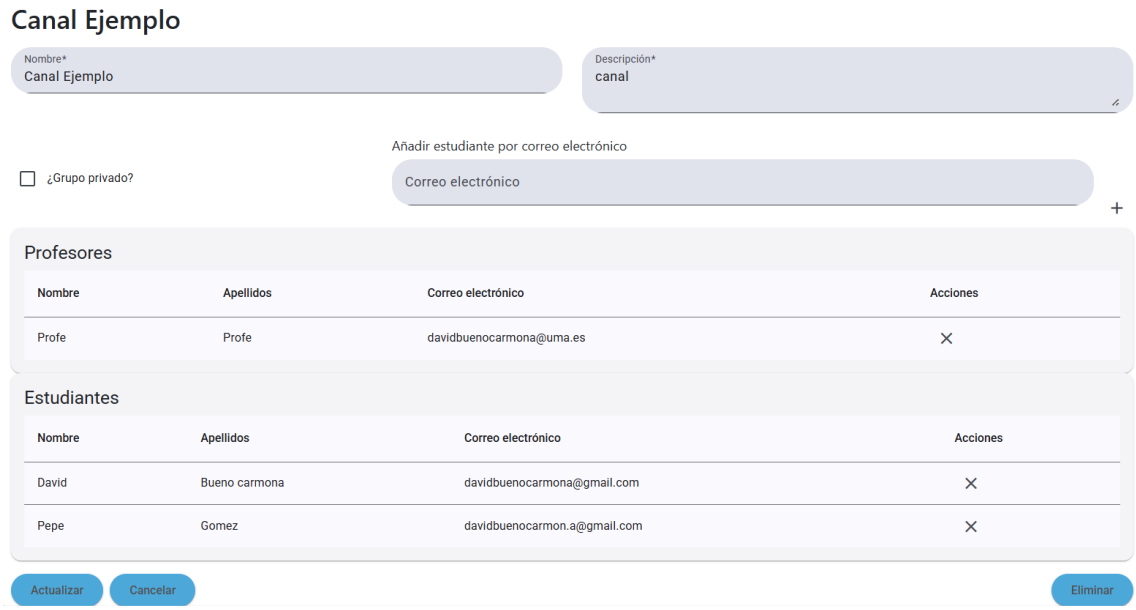


Ilustración 46. Vista de detalle de un canal

La vista Formulario de Creación y Edición de canales permite a los usuarios con los permisos correspondientes crear nuevos grupos o editar la información de grupos existentes. El formulario se muestra tanto en modo creación como en modo

edición, adaptando sus campos y opciones disponibles según el contexto y el rol del usuario.

Campos del formulario

1. Nombre

- **Tipo:** Campo de texto (*input*).
- **Obligatorio:** Sí.
- **Validación:** Si está vacío, muestra el error **"Nombre requerido"**.

2. Descripción

- **Tipo:** Área de texto (*textarea*).
- **Obligatorio:** Sí.
- **Validación:** Si está vacía, muestra el error **"Descripción requerida"**.

3. Privado

- **Tipo:** Casilla de verificación (*checkbox*).
- **Función:** Determina si el grupo es privado.
    - **Canal privado:** Representado por un candado cerrado. Solo se puede entrar por invitación.
    - **Canal público:** Representado por un candado abierto. Cualquier usuario puede unirse.

4. Añadir estudiante por correo electrónico *(solo en edición y si el usuario puede editar)*

- **Tipo:** Campo de texto (*input*) + botón con icono **“+”**.
- **Función:** Permite añadir un estudiante al canal introduciendo su correo electrónico.
- **Validación:** El campo debe contener un correo electrónico válido para habilitar el botón.

5. Enviar mensaje solo para miembros *(solo en edición, solo profesores)*

- **Tipo:** Área de texto (*textarea*) + botón con icono **"enviar"**.
- **Función:** Permite enviar un mensaje a los miembros del grupo.

6. Selección de profesor *(solo para administradores en modo creación)*

- **Tipo:** Selector desplegable.
- **Función:** Asigna un profesor al grupo en el momento de su creación.
- **Fuente de datos:** Lista de profesores disponible en el centro donde se esté creando el canal.

Listados adicionales

Ambos listados son visibles para todos los usuarios, pero las acciones están reservadas para usuarios con permisos de edición, es decir profesores y administradores.

- **Lista de Profesores:**

  Muestra los profesores asignados al grupo con la opción de eliminarlos.
- **Lista de Estudiantes:**

  Muestra los estudiantes miembros del grupo con la opción de eliminarlos.

Acciones disponibles

- **Crear / Actualizar:**
  - **Visible:** Si el usuario puede editar.
  - **Función:** Guarda los cambios o crea el grupo.
  - **Restricción:** Botón deshabilitado si el formulario no es válido.
- **Unirse al grupo:**

    - **Visible:** Si el usuario no es miembro, no es administrador, no está en modo creación y el grupo no es privado.
- **Abandonar el grupo:**
    - **Visible:** Si el usuario es miembro, no es administrador y no está en modo creación.
- **Cancelar:**
    - Cierra el formulario sin guardar cambios.
- **Eliminar grupo:**
    - **Visible:** Si el usuario puede editar y no está en modo creación.

Validaciones y comportamiento

- El formulario es **reactivo** (*Reactive Form*).
- Los campos obligatorios muestran mensajes de error si no se cumplen las condiciones.
- El contenido y las acciones visibles cambian dinámicamente según:
    - Si el formulario está en **creación** o **edición**.

B.14. Creación y edición de profesores

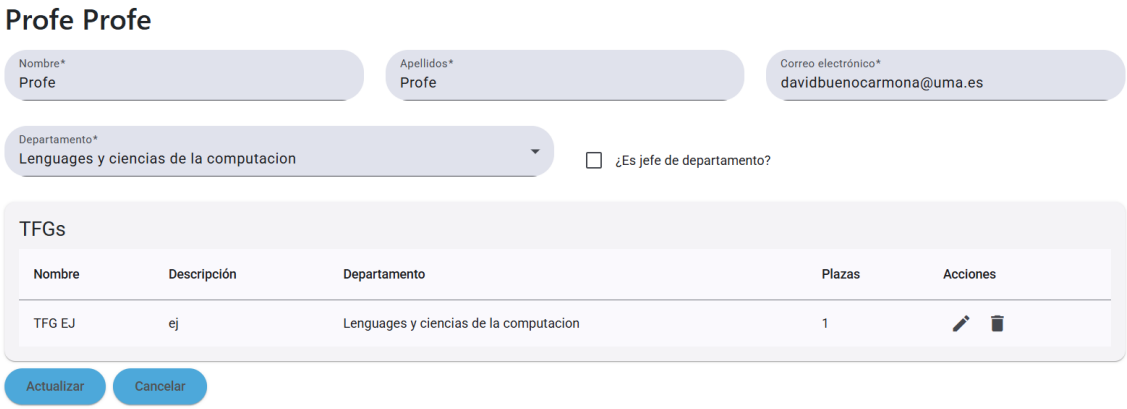


Ilustración 47. Vista del detalle de un profesor

Esta vista permite a los administradores crear nuevos registros de profesores o modificar los datos de profesores existentes. Los usuarios no administradores

pueden ver el formulario sin la posibilidad de aplicar ningún cambio solo por consultar la información.

Cuando un profesor accede a la sección de la cabecera de Mi Perfil, es llevado a esta vista.

El contenido del formulario y las acciones disponibles cambian según si se está en **modo creación** o **modo edición**.

Campos del formulario

1. Nombre

- **Tipo:** Campo de texto (*input*).
- **Obligatorio:** Sí.
- **Validación:** Si está vacío, muestra el mensaje **"Nombre requerido"**.

2. Apellido

- **Tipo:** Campo de texto (*input*).
- **Obligatorio:** Sí.
- **Validación:** Si está vacío, muestra el mensaje **"Apellido requerido"**.

3. Correo electrónico

- **Tipo:** Campo de texto (*input*) con validación de formato de email.
- **Obligatorio:** Sí.
- **Validación:** Si el campo está vacío o no tiene un formato válido, muestra **"Email requerido"**.

4. Departamento

- **Tipo:** Selector desplegable.
- **Obligatorio:** Sí.
- **Contenido:** Lista de departamentos disponibles.
- **Validación:** Si no se selecciona un departamento, muestra **"Departamento requerido"**.

5. Jefe de departamento

- **Tipo:** Casilla de verificación (*checkbox*).
- **Función:** Indica si el profesor es el jefe del departamento seleccionado.

Información adicional en modo edición

- **Listado de TFGs:**
  - Se muestra solo cuando se edita un profesor existente.
  - Contiene los TFGs en los que participa el profesor.
  - Presentado a través del componente de listado de líneas de TFG.

Acciones disponibles

- **Crear / Actualizar:**
  - Visible solo para administradores.
  - Guarda los datos introducidos en el formulario.
  - Deshabilitado si el formulario no es válido.
- **Cancelar:**
  - Cierra el formulario y vuelve a la vista anterior sin guardar cambios.

Validaciones y comportamiento

- El formulario es **reactivo** (*Reactive Form*).
- Los campos obligatorios muestran mensajes de error si no se cumplen las condiciones.
- En **modo creación**, no se muestra la lista de TFGs.
- En **modo edición**, el encabezado de la vista muestra el nombre y apellido del profesor.

B.15. Creación y edición de estudiantes

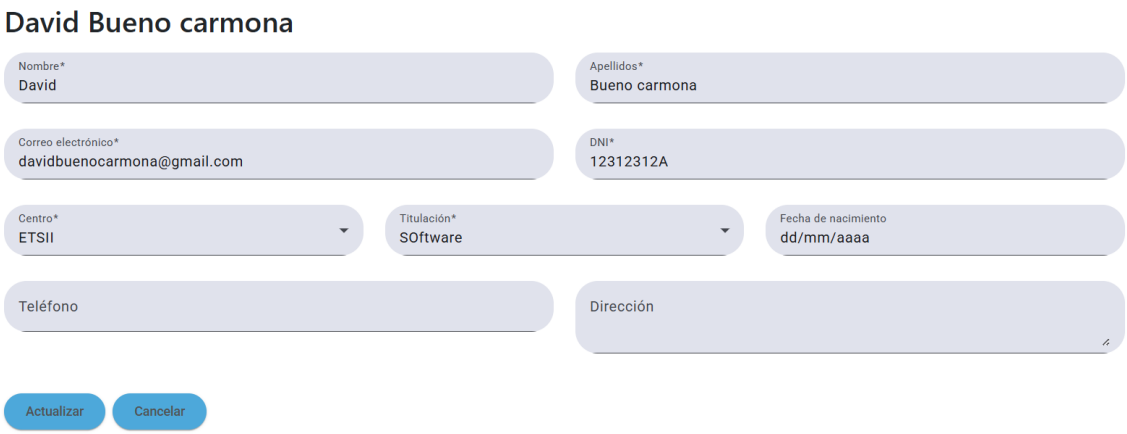


Ilustración 48. Vista del detalle de un alumno

Esta vista permite crear un nuevo alumno o editar el perfil de un alumno existente. El encabezado y los datos mostrados se adaptan dependiendo de si se está en **modo creación** o **modo edición**.

Cuando un alumno selecciona la sección de la cabecera de Mi Perfil, es redirigido a esta vista, con su información. No puede modificar información sensible como su DNI o correo electrónico. Puede modificar datos opcionales como su fecha de nacimiento, dirección o número de teléfono.

Campos del formulario

1. **Nombre:**
   - **Tipo:** Campo de texto (*input*).
   - **Obligatorio:** Sí.
   - **Validación:** Si está vacío, muestra **"Nombre requerido"**.
2. **Apellido:**
   - **Tipo:** Campo de texto (*input*).
   - **Obligatorio:** Sí.
   - **Validación:** Si está vacío, muestra **"Apellido requerido"**.

3. **Correo electrónico:**
    - **Tipo:** Campo de texto (*input*) con validación de formato de email.
    - **Obligatorio:** Sí.
    - **Validación:** Si está vacío o no tiene un formato válido, muestra **"Email requerido"**.
4. **DNI:**
    - **Tipo:** Campo de texto (*input*).
    - **Obligatorio:** Sí.
    - **Validación:** Si está vacío, muestra **"DNI requerido"**.
5. **Centro:**
    - **Tipo:** Selector desplegable.
    - **Obligatorio:** Sí.
    - **Contenido:** Lista de centos disponibles.
    - **Validación:** Si no se selecciona, muestra **"Centro requerida"**.
6. **Titulación (Carrera):**
    - **Tipo:** Selector desplegable.
    - **Obligatorio:** Sí.
    - **Contenido:** Lista de titulaciones disponibles en el centro seleccionado.
    - **Validación:** Si no se selecciona, muestra **"Carrera requerida"**.
7. **Fecha de nacimiento:**
    - **Tipo:** Campo de fecha.
    - **Obligatorio:** No.
8. **Teléfono:**
    - **Tipo:** Campo de texto (*input*).
    - **Obligatorio:** No.
9. **Dirección:**
    - **Tipo:** Área de texto (*textarea*).

- **Obligatorio:** No.

Acciones disponibles

- **Crear / Actualizar:**
  - Guarda los datos introducidos en el formulario.
  - Deshabilitado si el formulario no es válido.
- **Cancelar:**
  - Cierra el formulario y vuelve a la vista anterior sin guardar cambios.

Validaciones y comportamiento

- El formulario es **reactivo** (*Reactive Form*).
- Los campos obligatorios muestran mensajes de error cuando no cumplen las condiciones mínimas.
- En **modo creación**, el encabezado muestra *"Nuevo alumno"*.
- En **modo edición**, el encabezado muestra el nombre y apellidos del alumno.

B.16. Creación y edición de departamentos

Ilustración 49. Vista del detalle de un departamento

Esta vista permite **crear un nuevo departamento** o **editar uno existente**. El contenido se adapta según el modo en que se abra el formulario:

- **Modo creación:** Encabezado *"Nuevo departamento"*.
- **Modo edición:** Encabezado con el nombre del departamento.

Campos del formulario

1. Nombre

- **Tipo:** Campo de texto (*input*).
- **Obligatorio:** Sí.
- **Validación:** Muestra **"Nombre requerido"** si está vacío.

2. Acrónimo

- **Tipo:** Campo de texto (*input*).
- **Obligatorio:** No.
- **Uso:** Representa una abreviatura oficial o interna del departamento.

3. Centros asociadas

- **Tipo:** Selector múltiple.
- **Obligatorio:** No.
- **Contenido:** Lista de universidades disponibles para vincular el departamento.
- **Uso:** Permite asociar el departamento a uno o varios centros universitarios.

Acciones disponibles

- **Crear / Actualizar:**
  - Guarda la información introducida.
  - Está deshabilitado mientras el formulario no sea válido.
- **Cancelar:**
  - Cierra el formulario sin guardar cambios y regresa a la pantalla anterior.

Validaciones y comportamiento

- Formulario **reactivo** (*Reactive Form*), con control automático de errores y estado de campos.
- Solo los administradores tienen acceso a esta vista.
- El campo "Nombre" es el único obligatorio; los demás son opcionales.
- En **modo creación**, todos los campos aparecen vacíos.
- En **modo edición**, se muestran los datos actuales del departamento para ser modificados.

B.17. Creación y edición de titulaciones

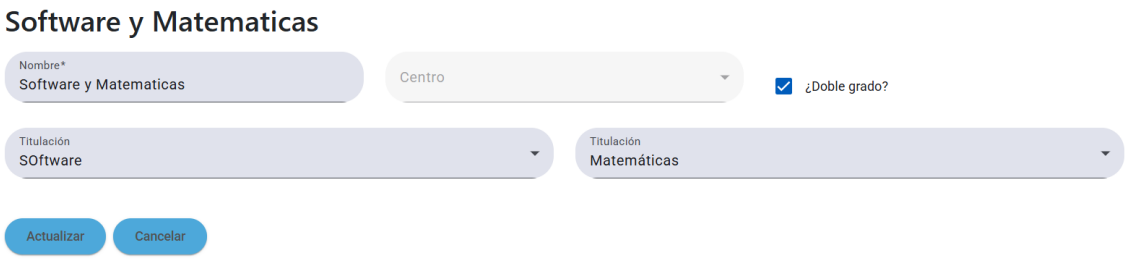


Ilustración 50. Vista del detalle de una titulación

Esta vista permite **crear una nueva titulación** o **editar una existente**.

El encabezado se adapta según el modo:

- **Modo creación:** Muestra *"Nueva titulación"*.
- **Modo edición:** Muestra el nombre de la titulación actual.

Campos del formulario

1. Nombre de la titulación

- **Tipo:** Campo de texto (*input*).
- **Obligatorio:** Sí.
- **Validación:** Error si el campo está vacío.

2. Universidad

- **Tipo:** Selector único.
- **Obligatorio:** Sí.
- **Contenido:** Lista de universidades disponibles.
- **Uso:** Define a qué universidad pertenece la titulación.

3. Doble titulación

- **Tipo:** Casilla de verificación.
- **Obligatorio:** No.
- **Comportamiento:**
    - Si se activa, se muestran dos selectores adicionales: *titulación principal* y *titulación secundaria*.
    - Permite configurar titulaciones que se cursan de manera conjunta.

4. Titulación principal y secundaria *(solo si doble titulación activada)*

- **Tipo:** Selector único para cada una.
- **Obligatorio:** Sí, si está habilitado el modo de doble titulación.
- **Contenido:** Lista de todas las carreras disponibles.
- **Uso:** Define qué titulaciones forman la doble titulación.

Acciones disponibles

- **Crear / Actualizar:**
    - Guarda los cambios realizados en la titulación.
    - Se deshabilita mientras el formulario no sea válido.
- **Cancelar:**
    - Cierra el formulario sin guardar cambios.

Validaciones y comportamiento

- Formulario **reactivo**, con control de errores en tiempo real.

- El campo nombre siempre es obligatorio.
- El campo centro, relaciona la titulación con un centro. En caso de que la titulación sea un doble grado, se bloquea el campo de centro ya que, al seleccionar otras dos titulaciones existentes, esta nueva se relacionará con ambos centros a los que pertenezcan las otras titulaciones.
- La sección de doble titulación solo aparece si el usuario marca la casilla correspondiente.
- Permite tanto creación como edición de carreras existentes.

B.18. Creación y edición de centros

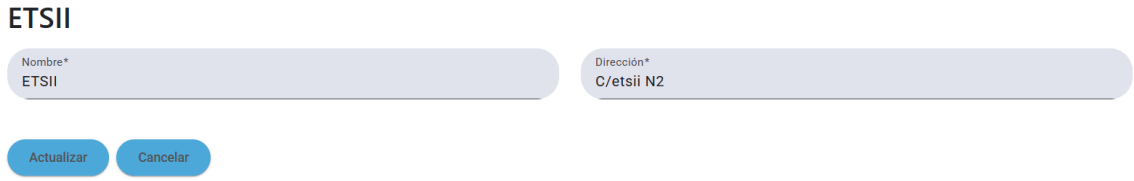


Ilustración 51. Vista del detalle de un centro

Esta vista permite crear un nuevo centro o editar uno existente. El encabezado se adapta según el modo:

- **Modo creación:** Muestra "Nuevo centro".
- **Modo edición:** Muestra el nombre del centro actual.

Campos del formulario

1. Nombre del centro:

- **Tipo:** Campo de texto (input).
- **Obligatorio:** Sí.
- **Validación:** Error si el campo está vacío.

2. Dirección:

- **Tipo:** Campo de texto (input).

- **Obligatorio:** Sí.
- **Uso:** Dirección donde se ubica el centro.
- **Validación:** Error si el campo está vacío.

Acciones disponibles

- **Crear / Actualizar:**

  Guarda los cambios realizados en el centro. Se deshabilita mientras el formulario no sea válido.
- **Cancelar:**

  Cierra el formulario sin guardar cambios.

Validaciones y comportamiento

- Formulario reactivo con control de errores en tiempo real.
- Permite tanto creación como edición de centros existentes.

UNIVERSIDAD DE MÁLAGA | uma.es

E.T.S. DE INGENIERÍA INFORMÁTICA

E.T.S de Ingeniería Informática

Bulevar Louis Pasteur, 35

Campus de Teatinos