



UNIVERSIDAD DE GRANADA

TRABAJO FIN DE GRADO
INGENIERÍA INFÓRMATICA

ErasmusUGR

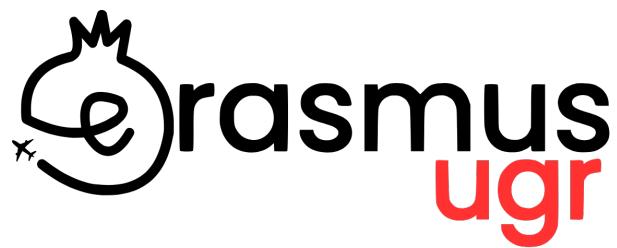
Plataforma Web para la Gestión de Movilidades Erasmus y
Equivalencias en la UGR
Autor
Blanca Girón Ricoy

Directores
Javier Medina Quero
Nuria López Ruiz



ESCUELA TÉCNICA SUPERIOR DE INGENIERÍAS INFORMÁTICA Y DE
TELECOMUNICACIÓN

—
Granada, mes de 2025



ErasmusUGR

Plataforma Web para la Gestión de Movilidades Erasmus y
Equivalencias en la UGR.

Autor

Blanca Girón Ricoy

Directores

Javier Medina Quero
Nuria López Ruiz

ErasmusUGR: Plataforma Web para la Gestión de Movilidades Erasmus y Equivalencias en la UGR

Blanca Girón Ricoy

Palabras clave: Erasmus+, movilidad, acuerdos de estudios, convalidaciones, Universidad de Granada, plataforma web, dashboard de estudiantes, dashboard de tutores, Flask, MongoDB, React, Tailwind CSS

Resumen

Este TFG desarrolla una plataforma web para facilitar el proceso de movilidad internacional a estudiantes de la Universidad de Granada que participan en programas Erasmus+. La herramienta centraliza información dispersa sobre destinos y equivalencias académicas, simplificando la elaboración del Acuerdo de Estudios, documento clave para garantizar la validez académica del intercambio. La plataforma permite a los estudiantes explorar universidades de destino mediante diversos filtros, consultar experiencias previas, visualizar asignaturas ya convalidadas y crear su acuerdo académico con asistencia guiada. Para los tutores académicos, ofrece un sistema eficiente de revisión y aprobación de solicitudes. El proyecto surge de la observación directa de las dificultades que enfrentan los estudiantes durante la planificación de su movilidad, donde la falta de información clara y accesible genera incertidumbre y, en ocasiones, lleva a desistir de estas oportunidades. La plataforma busca eliminar estos obstáculos burocráticos, optimizando el proceso administrativo y mejorando la experiencia global de movilidad para la comunidad universitaria de Granada.

ErasmusUGR: Online Platform for Erasmus Mobility and Validation Management at the University of Granada

Blanca Girón Ricoy

Keywords: Erasmus+, mobility, study agreements, equivalences, University of Granada, web platform, student dashboard, tutor dashboard, Flask, MongoDB, React, Tailwind CSS

Abstract

This Final Degree Project develops a web platform to facilitate the international mobility process for University of Granada students participating in Erasmus+ programs. The tool centralizes scattered information about destinations and academic credit transfers, simplifying the creation of the Learning Agreement, a key document to ensure the academic validity of the exchange. The platform allows students to explore destination universities through various filters, consult previous experiences, view courses that have already been validated, and create their academic agreement with guided assistance. For academic tutors, it offers an efficient system for reviewing and approving applications. The project emerges from direct observation of the difficulties students face when planning their mobility, where the lack of clear and accessible information creates uncertainty and sometimes leads to abandoning these opportunities. The platform seeks to eliminate these bureaucratic obstacles, optimizing the administrative process and improving the overall mobility experience for the University of Granada community.

Yo, **Blanca Girón Ricoy**, alumno de la titulación Ingeniería Informática de la **Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación de la Universidad de Granada**, con DNI 75575344N, autorizo la ubicación de la siguiente copia de mi Trabajo Fin de Grado en la biblioteca del centro para que pueda ser consultada por las personas que lo deseen.

Fdo: Blanca Girón Ricoy

Granada a 4 de septiembre de 2025 .

D. **Javier Medina Quero**, Profesor del Departamento de Ingeniería de Computadores, Automática y Robótica de la Universidad de Granada.

D. **Nuria López Ruiz**, Profesora del Departamento de Electrónica y Tecnología de Computadores de la Universidad de Granada.

Informan:

Que el presente trabajo, titulado ***ErasmusUGR, Plataforma Web para la Gestión de Movilidades Erasmus y Equivalencias en la UGR*** ha sido realizado bajo su supervisión por **Blanca Girón Ricoy**, y autorizamos la defensa de dicho trabajo ante el tribunal que corresponda.

Y para que conste, expiden y firman el presente informe en Granada a 5 de mes de 2025 .

Los directores:

Javier Medina Quero Nuria López Ruiz

Agradecimientos

A mis tutores, Nuria y Javier, por su confianza en esta idea y por sus ánimos, que hacen que guarde un recuerdo especial de los profesores que conforman esta escuela.

A mis padres, que me quieren como soy y, con cariño, me muestran cómo puedo ser mejor. Me enseñan cada día a esforzarme y a trabajar con honestidad, y están conmigo cuando dudo, recordándome que puedo más de lo que creo. Soy fruto de su amor, de su constancia y de los sacrificios que han hecho por mí. Este logro también es suyo, y espero que al leer estas líneas se sientan tan orgullosos de mí como yo lo estoy de ellos.

A mi hermana, que siempre está cerca y me cuida de forma incondicional. Aunque seamos distintas —yo más de ciudad y ella capaz de imaginarse viviendo en nuestro pequeño pueblo— sé que seguiremos igual de unidas. Porque, al final, esté donde esté, mi hogar siempre será con ella.

A mi tía Tere, por su fuerza y su valor, por cuidarnos tantas veces como una madre más y por enseñarme con su ejemplo lo que significa sostener a los demás incluso cuando no es fácil. Le agradezco su entrega y el lugar tan especial que ocupa en mi vida. Ojalá algún día tenga yo la fortaleza y el coraje que siempre le he visto.

A mis amigos. A los de siempre y a los que han llegado después, quienes me han acompañado en todas mis versiones y dado seguridad cuando dudaba y cariño en cada paso. Por vosotros sé que lo importante no es la meta, sino lo que ocurre mientras caminamos juntos: cada risa, cada momento y cada gesto. Me habéis enseñado a querer y a dejarme querer. Y creo de verdad que, si todo el mundo pudiera contar con unos amigos como los míos, el mundo sería un lugar mucho más amable.

Al final, la persona que hoy ha conseguido esta carrera, y de la que creo que me puedo sentir orgullosa, no está hecha solo de mi propio esfuerzo. Soy también todo lo que me han dado quienes han formado parte de mi vida, sobre todo quienes me han querido: su cariño, su fuerza y la parte de sí mismos que dejaron en mí.

A todos vosotros, gracias.

Índice general

1. Introducción	1
1.1. Contexto	1
1.2. Motivación	2
1.3. Objetivos	3
1.4. Estructura del documento	4
2. Descripción del problema	7
2.1. Contexto	7
2.1.1. Contexto y marco normativo	8
2.1.2. Procedimiento actual para la gestión de acuerdos de estudios . . .	11
2.1.3. Análisis crítico y principales limitaciones. Problema a resolver . . .	13
3. Estado del arte	15
3.1. Introducción	15
3.1.1. Contexto, problema y objetivos	15
3.2. Análisis de soluciones existentes	16
3.2.1. Plataformas institucionales europeas. European Student Card Initiative	17
3.2.2. Iniciativas comunitarias y no institucionales	28
3.3. Análisis comparativo. Conclusiones	33
4. Planificación	35
4.1. Metodología	35
4.2. Análisis de costes	38
4.2.1. Recursos humanos	38
4.2.2. Licencias Software y Herramientas	39
4.2.3. Equipamiento informático	39
4.2.4. Infraestructura	40
4.2.5. Equipamiento Informático	40
4.2.6. Resumen de costes	40

5. Análisis del problema	43
5.1. Análisis de requisitos	43
5.1.1. Requisitos funcionales	44
5.1.2. Requisitos no funcionales	46
5.2. ¿Qué actores participan y qué hacen?	47
5.3. Casos de uso	49
5.3.1. Descripción de los casos de uso	50
5.3.2. Diagramas de casos de uso	54
5.4. Diagramas de secuencia	57
6. Diseño de la plataforma	65
6.1. Diseño de la arquitectura	65
6.1.1. Microservicios vs. Sistemas monolíticos: principales diferencias	66
6.1.2. Modelo cliente-servidor	69
6.2. Diseño de la base de datos	72
6.2.1. Modelado de la información	73
6.3. Diseño de la interfaz de usuario	78
6.3.1. Introducción. Intencionalidad del diseño	78
6.3.2. Identidad visual y personalización	78
6.3.3. Prototipado y herramientas	79
6.3.4. Tecnologías empleadas	80
6.3.5. Organización general, experiencia de usuario y diagramas de flujo .	81
7. Implementación	85
7.1. Entorno y herramientas de desarrollo	85
7.1.1. Lenguajes de programación	85
7.1.2. Frameworks. Bibliotecas utilizadas.	86
7.2. Desarrollo	88
7.2.1. Desarrollo del Backend	88
7.2.2. Desarrollo del Frontend	93
8. Verificación y pruebas funcionales	97
8.1. ¿Qué es Postman?	97
8.2. Diseño de la colección de pruebas	98
8.2.1. Entorno de pruebas	99
8.2.2. Ejecución de las pruebas	99
8.3. Pruebas de Navegación y Flujo del Usuario	99
8.3.1. Validación de formularios y acciones básicas	100
8.3.2. Comprobaciones adicionales	100
8.3.3. Pruebas de visualización en distintos dispositivos	100
9. Conclusiones. Trabajo futuro	103
9.1. Consecución de objetivos. Resultados	103
9.2. Trabajo futuro	105

Índice de figuras

2.1. Proceso de aprobación del Acuerdo de Estudios. Fuente: European Commission [20]	11
3.1. Diseño del Carné Europeo de Estudiante [29]	17
3.2. Logo de Erasmus+ App [3]	18
3.3. Vista de la aplicación de <i>Erasmus+ App</i> . Imagen sacada de Play Store [30].	19
3.4. Logo de <i>Erasmus Without Paper</i> [9].	20
3.5. Proceso digital de nominaciones digitales de estudiantes mediante <i>Erasmus Without Paper</i> [41].	20
3.6. Proceso digital de firmado de acuerdos de estudios mediante <i>Erasmus Without Paper</i> [41].	21
3.7. Proceso digital de envío de registros académicos mediante <i>Erasmus Without Paper</i> [41].	21
3.8. Proceso digital de firmado de acuerdos interinstitucionales mediante <i>Erasmus Without Paper</i> [41].	22
3.9. Logo de <i>OLA. Online Learning Agreement</i>	22
3.10. Selección de tipos de movilidad en la plataforma <i>OLA. Online Learning Agreement</i> . Elaboración propia	23
3.11. Inserción de la información de la institución de origen en la plataforma <i>OLA. Online Learning Agreement</i> . Elaboración propia	24
3.12. Acción de firmar un nuevo Acuerdo de Estudios usando <i>OLA. Online Learning Agreement</i> . Elaboración propia	25
3.13. Diferentes logos de <i>EWP Dashboard</i> . [28]	26
3.14. Vista de secciones del Dashboard	27
3.15. Denegación de un Acuerdo de Estudios de un estudiante	27
3.16. Aplicación a una movilidad	28
3.17. Logo de <i>Erasmus Student Network</i> [8].	29
3.18. <i>ESN Card</i> [10]	30
3.19. Logo de <i>Erasmusu</i> [12].	31
3.20. Diferentes vistas de la plataforma <i>Erasmusu.com</i> . Elaboración propia.	32
4.1. Diagrama de Gantt. Cronología de los sprints.	37

5.1.	Diagrama de caso de uso de gestión de usuarios: registro, inicio/cierre de sesión y administración del perfil.	55
5.2.	Diagrama de caso de uso de exploración y búsqueda de destinos.	56
5.3.	Diagrama de caso de uso de gestión del acuerdo de estudios.	56
5.4.	Diagrama de caso de uso de administración del sistema: gestión de colecciones (usuarios, grados, centros, asignaturas, destinos) y validación de equivalencias.	57
5.5.	Diagrama de secuencia. Inicio de sesión y exploración de destinos.	58
5.6.	Diagrama de secuencia. Asignación de destinos a estudiante.	59
5.7.	Diagrama de secuencia. Asignación de destinos a tutor.	59
5.8.	Diagramas de secuencia para la gestión de grados y centros en la plataforma.	60
5.9.	Diagramas de secuencia. Gestión de destinos	61
5.10.	Diagramas de secuencia. Gestión de asignaturas.	62
5.11.	Diagrama de secuencia. Gestión del Acuerdo de Estudios.	63
6.1.	Diseño de la arquitectura en 3 niveles	66
6.2.	Diagrama de las arquitecturas. Fuente: https://aws.amazon.com/es/compare/the-difference-between-monolithic-and-microservices-architecture/	67
6.3.	Esquema general de la arquitectura cliente-servidor de la plataforma Erasmus. Se muestra la interacción entre las distintas capas: presentación (SPA React + Vite), lógica de negocio (API REST con Flask y middleware CORS) y persistencia (base de datos MongoDB).	71
6.4.	Entidades de la plataforma y sus relaciones	72
6.5.	Logo de <i>MongoDB</i>	73
6.6.	Gráficos SVG creados. Tipografía usada y paleta de colores de la plataforma	79
6.7.	Logo de <i>ErasmusUGR</i>	79
6.8.	Comparación entre el prototipado visual realizado en Canva y la versión final desarrollada de la sección de perfil.	80
6.9.	Diagrama de flujo de los distintos roles	83
7.1.	Logo de <i>Cloudinary</i>	93
8.1.	Logo de <i>Postman</i>	98

Índice de cuadros

3.1. Comparativa de funcionalidades entre herramientas del ecosistema Erasmus+	34
4.1. Coste mensual estimado con tarifas de mercado.	38
4.2. Coste de licencias de software y herramientas utilizadas	39
4.3. Coste de equipamiento informático (gasto único)	39
4.4. Costes mensuales de infraestructura (supuesto de despliegue)	40
4.5. Coste de equipamiento informático	40
4.6. Resumen de costes del primer mes (tarifas de mercado).	40
4.7. Resumen de meses 2, 3 y 4 (tarifas de mercado).	40
4.8. Coste total estimado del proyecto (tarifas de mercado).	41
9.1. Resumen de la consecución de los objetivos específicos	104

Capítulo 1

Introducción

La movilidad internacional es una de las oportunidades más valiosas y enriquecedoras que puede experimentar un estudiante durante su etapa universitaria. A través de programas como Erasmus+, miles de alumnos amplían cada año sus horizontes académicos y personales, desarrollando competencias clave en un entorno multicultural. Según la Comisión Europea [7], más de 12 millones de personas han participado en el programa Erasmus+ desde su creación hace unos 38 años, lo que demuestra su impacto positivo en la formación de los jóvenes europeos y su papel en la construcción de un espacio educativo más integrado y diverso. Esta experiencia supera el ámbito puramente académico, fomentando la adquisición de una perspectiva internacional que resulta fundamental en el actual contexto globalizado.

1.1. Contexto

Cuando un estudiante decide participar en un programa de movilidad, debe atravesar un complejo proceso administrativo, en el que uno de los elementos clave es la correcta planificación académica. La elección del destino no solo depende de las preferencias personales o del prestigio de la universidad de destino, sino también de la posibilidad de reconocer las asignaturas cursadas en el extranjero. Para ello, es necesario completar un documento denominado *Acuerdo de Estudios (Learning Agreement)*, en el que se establece qué materias se estudiarán en la universidad de destino y su equivalencia en la universidad de origen. La correcta elaboración de este acuerdo es fundamental para garantizar la validez académica del intercambio.

A pesar de la importancia de este documento, muchos estudiantes enfrentan dificultades para elaborarlo. La información sobre asignaturas previamente reconocidas es dispersa, poco accesible y, en muchas ocasiones, desactualizada. Para tomar una decisión informada, los alumnos deben recurrir a múltiples fuentes, como páginas web oficiales, bases de datos de su facultad, foros de antiguos Erasmus o incluso consultas directas a compañeros que ya han realizado una movilidad. Este proceso, además de consumir una

gran cantidad de tiempo, genera incertidumbre y estrés, lo que en algunos casos desmotiva a los estudiantes o incluso les impide completar a tiempo la solicitud de movilidad.

Desde el punto de vista institucional, los tutores académicos encargados de gestionar estos acuerdos de estudios también se ven afectados por la falta de un sistema centralizado. La revisión de las solicitudes es un proceso laborioso que requiere verificar la compatibilidad entre los programas de estudio de diferentes universidades, lo que en muchas ocasiones se traduce en retrasos administrativos y desmotivación del tutor y del estudiante.

Ante este escenario, surge la necesidad de una solución que simplifique el proceso de selección del destino y la planificación académica de los estudiantes Erasmus. Este Trabajo de Fin de Grado propone el desarrollo de una **plataforma web** diseñada específicamente para facilitar la gestión de la movilidad en la Universidad de Granada. A través de esta herramienta, los alumnos podrán explorar los destinos disponibles, consultar información detallada sobre cada universidad, acceder a experiencias y valoraciones de otros estudiantes y, lo más importante, conocer qué asignaturas han sido previamente reconocidas y cuáles requieren una nueva solicitud de homologación.

Esta plataforma no solo servirá como un repositorio de información, sino que también ofrecerá funcionalidades avanzadas, como la posibilidad de filtrar universidades en función de las asignaturas que el estudiante desea reconocer, ayudando así a reducir la carga burocrática del proceso. Además, permitirá una mejor comunicación con los tutores académicos, agilizando la validación de acuerdos de estudios y minimizando errores en el reconocimiento posterior.

1.2. Motivación

La idea de desarrollar esta plataforma nace de una combinación de factores personales y académicos. A lo largo de estos cuatro años de carrera, he sido testigo de cómo compañeros de clase han tenido dificultades durante el proceso de movilidad, especialmente en la gestión del acuerdo de estudios. La falta de información clara y accesible sobre asignaturas reconocidas ha generado estrés y, en algunos casos, ha llevado a estudiantes a desistir de su intención de realizar un intercambio académico. En otros casos, los alumnos han descubierto problemas con el reconocimiento de sus asignaturas una vez que ya estaban en la universidad de destino, lo que ha afectado negativamente su experiencia académica y personal.

Desde el inicio de mis estudios en ingeniería, he aprendido a identificar problemas y a diseñar soluciones eficientes. Este proyecto representa una oportunidad ideal para aplicar esos conocimientos en un entorno real, creando una herramienta que no solo resuelva un problema concreto, sino que también tenga un impacto positivo en la comunidad universitaria. La posibilidad de reducir la carga burocrática y de proporcionar un acceso más directo a la información sobre la movilidad académica motiva el desarrollo de esta plataforma.

Además, siempre he sentido interés por el diseño de interfaces y la creación de experiencias digitales atractivas. Desarrollar una plataforma web con una interfaz intuitiva y accesible es un reto apasionante que combina aspectos técnicos con consideraciones de usabilidad y diseño. La idea de que esta herramienta pueda ayudar a cientos de estudiantes en el futuro, facilitando su acceso a la movilidad internacional, refuerza aún más mi motivación para llevar a cabo este proyecto.

Este Trabajo de Fin de Grado no solo me permite aplicar los conocimientos adquiridos durante la carrera en un problema real, sino que también tiene el potencial de convertirse en una herramienta útil y práctica para la comunidad universitaria. Con esta plataforma, se espera mejorar la experiencia de movilidad de los estudiantes de la Universidad de Granada, facilitando el acceso a información clave, optimizando el proceso de reconocimiento y reduciendo los obstáculos administrativos que actualmente dificultan la planificación de una estancia Erasmus.

El trabajo se encuentra en el siguiente repositorio: <https://github.com/blancagiron/erasmus-ugr>

1.3. Objetivos

Teniendo en cuenta lo mencionado anteriormente, los objetivos que nos planteamos son:

1. **OE1:** Comprender los procesos administrativos de acuerdos de estudio y equivalencias en la UGR.
 - a) **OE1.1:** Estudiar cómo se gestionan estos procesos actualmente.
 - b) **OE1.2:** Detectar dificultades y posibles mejoras.
2. **OE2:** Investigar soluciones similares en otras universidades.
 - a) **OE2.1:** Analizar plataformas Erasmus de otras instituciones.
 - b) **OE2.2:** Identificar funcionalidades útiles y aplicables a la UGR.
 - c) **OE2.3:** Determinar carencias y limitaciones de las soluciones actuales desde la perspectiva del estudiante.
 - d) **OE2.4:** Estudiar tecnologías usadas en estas plataformas.
3. **OE3:** Diseñar e implementar una plataforma web usable y accesible.
 - a) **OE3.1:** Definir arquitectura y estructura de la aplicación.
 - b) **OE3.2:** Crear prototipos centrados en la experiencia del usuario.
 - c) **OE3.3:** Desarrollar la interfaz con tecnologías modernas (React).
4. **OE4:** Crear una base de datos para almacenar información de destinos y equivalencias.

- a) **OE4.1:** Diseñar el modelo de datos (universidades, asignaturas, etc.).
 - b) **OE4.2:** Implementar la base de datos (MongoDB).
5. **OE5:** Establecer comunicación eficiente entre frontend y backend.
 - a) **OE5.1:** Desarrollar una API RESTful (Flask).
 - b) **OE5.2:** Gestionar datos desde la API.
 - c) **OE5.3:** Asegurar una comunicación segura usando JSON.
 6. **OE6:** Implementar la gestión de solicitudes de convalidación.
 - a) **OE6.1:** Diseñar el flujo de trabajo de las solicitudes.
 - b) **OE6.2:** Permitir a los estudiantes enviar solicitudes.
 - c) **OE6.3:** Crear un panel para que los coordinadores las gestionen.
 - d) **OE6.4:** Incluir notificaciones sobre el estado de las solicitudes.
 7. **OE7:** Integrar mecanismos para fomentar la interacción entre estudiantes Erasmus.
 8. **OE8:** Aplicar buenas prácticas de seguridad web.
 - a) **OE8.1:** Investigar medidas básicas de seguridad.
 - b) **OE8.2:** Implementar autenticación y autorización de usuarios.

1.4. Estructura del documento

Capítulo 1: Introducción Capítulo actual en el que se describe el problema que se trata de resolver, el contexto y los objetivos marcados.

Capítulo 2: Descripción del problema

Capítulo 3: Estado del Arte Se realiza un análisis del estado del arte, explorando soluciones previas y herramientas existentes relacionadas con la gestión de la movilidad académica.

Capítulo 4: Planificación En esta sección se recoge una estimación del tiempo necesario para realizar el proyecto y las tareas que se deben cumplir. Además, se explica la metodología seguida.

Capítulo 5: Análisis del problema Se recogen los detalles que definen y limitan el funcionamiento y uso de la plataforma web.

Capítulo 6: Diseño de la plataforma En este capítulo se detallan las decisiones de diseño de la arquitectura de la plataforma, quedando descritos los componentes del sistema, la estructura de la base de datos y el diseño de la interfaz de usuario.

Capítulo 7: Implementación Este capítulo recoge el proceso de desarrollo de la plataforma. Se describe la recopilación de información y cómo se han llevado a la práctica las funcionalidades del sistema.

Capítulo 8: Verificación y pruebas funcionales Se evalúa el impacto de la plataforma a través de pruebas y validaciones, destacando los resultados obtenidos.

Capítulo 9: Conclusiones. Trabajo futuro Se presentan las conclusiones y se proponen posibles mejoras y futuras líneas de desarrollo.

Este proyecto es software libre, y está liberado con la licencia [36].

Capítulo 2

Descripción del problema

2.1. Contexto

La participación en el programa **Erasmus+** supone para los estudiantes de la Universidad de Granada una experiencia de gran valor, ya que no solo amplía su formación académica, sino que también les permite crecer a nivel personal, conocer nuevas culturas y desarrollar competencias muy útiles para su futuro. Sin embargo, la gestión de este proceso todavía presenta ciertas limitaciones e ineficiencias que, en muchos casos, dificultan que la experiencia sea tan sencilla y accesible como cabría esperar. Aunque la normativa y los manuales institucionales describen los pasos a seguir [49, 15] y las guías Erasmus+ establecen buenas prácticas para las convocatorias KA131 y KA171 [18, 19], no existe un único punto de acceso que reúna de forma actualizada los planes de estudio, las equivalencias previas y los requisitos idiomáticos de cada destino. Esta dispersión obliga a realizar búsquedas manuales en distintos portales, intercambiar numerosos correos y completar formularios PDF, lo cual causa retrasos, errores en la planificación y una excesiva carga administrativa tanto para el estudiantado como para los coordinadores académicos.

En las secciones siguientes:

- Definiremos los conceptos y normas que rigen el proceso.
- Describiremos el procedimiento vigente de elaboración y validación del Acuerdo de Estudios.
- Analizaremos críticamente sus principales limitaciones.

2.1.1. Contexto y marco normativo

Definiciones clave

A continuación, se presentan algunas definiciones fundamentales extraídas de la normativa vigente de la Universidad de Granada (UGR), concretamente del Reglamento de Movilidad Internacional de Estudiantes [49] y del Reglamento sobre Reconocimiento de Créditos de la ETSIIT [14]. Estas definiciones son fundamentales para analizar con rigor las carencias del proceso:

- **Estudiantes salientes / estudiantes enviados:** Estudiantes de la Universidad de Granada que realizan una estancia académica temporal en una universidad o entidad de derecho público o privado de otro país
- **Reconocimiento de créditos:** La aceptación por parte de la Universidad de Granada de los créditos que, habiendo sido obtenidos en enseñanzas universitarias oficiales o en enseñanzas universitarias no oficiales, en la misma u otra universidad, son computados en otras enseñanzas distintas cursadas en la Universidad de Granada a efectos de la obtención de un título oficial.
- **Acuerdo de Estudios (Learning Agreement):** Documento en el que quedarán reflejadas, con carácter vinculante, las actividades académicas que se desarrollarán en la universidad de acogida, así como las que serán reconocidas en la universidad de origen y la valoración, en su caso, en créditos ECTS.
- **Tutor/a docente:** miembro del Personal Docente e Investigador a tiempo completo, asignado por el Centro o por la Escuela Internacional de Posgrado al estudiantado acogido o enviado, con las funciones de asesoramiento y tutoría.
- **Responsable académico de la movilidad en el Centro:** Miembro del Personal Docente e Investigador a tiempo completo, encargado de coordinar la movilidad saliente y entrante en el Centro. En el caso de la ETS de Ingenierías Informática y de Telecomunicación, en la actualidad esta responsabilidad está asignada al Subdirector o Subdirectora de Relaciones Externas e Internacionalización.
- **Transcript of Records:** Certificado oficial de la universidad de destino donde se haga constar las asignaturas cursadas por el estudiante, la calificación obtenida y el número de créditos involucrados.

Normativa aplicable

El proceso de movilidad internacional de estudiantes en el ámbito universitario, y en particular la elaboración del **Acuerdo de Estudios** está regulado por un conjunto de normativas y documentos oficiales que garantizan la calidad académica y la protección de los derechos del estudiantado participante. El desarrollo de cualquier solución que

pretenda facilitar esta movilidad debe ajustarse a estas directrices y que, de esta forma, quede garantizada su legalidad y operabilidad.

Las principales normativas que resultan de interés para este proyecto son las siguientes:

Erasmus+ KA171: Movilidad entre países del programa y terceros países no asociados La acción clave **KA171** del programa Erasmus+ regula la movilidad de alumnos entre países del programa y terceros no asociados. Las directrices establecidas por la Comisión Europea [19] indican que el objetivo del Acuerdo de Estudios es garantizar una planificación transparente del período de estudios en el extranjero y asegurar el reconocimiento académico de las asignaturas superadas.

El Acuerdo de Estudios debe:

- Ser **digital** o usar una plantilla no digital provisional si no se dispone de plataforma electrónica.
- Incluir, antes del inicio de la movilidad, el cumplimentado de las tablas A, B y C:
 - **Tabla A:** Asignaturas a cursar en la universidad de destino.
 - **Tabla B:** Componentes virtuales, si procede. Es decir, actividades académicas que el estudiante puede cursar de manera online, como pueden ser cursos, talleres o seminarios online.
 - **Tabla C:** Asignaturas del plan de estudio que serán reconocidas.
- Favorecer el reconocimiento automático de créditos, reduciendo al mínimo los trámites o requisitos adicionales de validación.
- Contemplar la gestión formal de cambios excepcionales en el programa de estudios mediante las tablas A2, B2 y C2 (tablas de modificaciones).
- Estar firmado por las tres partes implicadas (estudiante, institución de origen e institución de destino) antes del inicio de la movilidad, admitiéndose tanto firmas escaneadas como electrónicas.

Erasmus+ KA131: Movilidad entre países del programa La acción clave **KA131** regula la movilidad de estudiantes entre instituciones de los Estados miembros de la Unión Europea y los países asociados al programa Erasmus+.

Entre sus requisitos fundamentales [18], se destacan los siguientes:

- La obligatoriedad de emplear **Acuerdos de Estudio en formato digital**, aspecto que se desarrolla con mayor detalle en la sección *Estado del Arte* [3].

- La cumplimentación de las tablas A (estudios en destino) y B (reconocimiento en la institución de origen), con la posibilidad de añadir una tabla C en caso de componentes virtuales.
- El reconocimiento automático de los créditos obtenidos, evitando requisitos adicionales de validación.
- La existencia de un procedimiento formalizado para la gestión de cambios excepcionales en el plan de estudios durante la movilidad.
- El registro de los resultados académicos en el *Transcript of Records* (2.1.1).

Normativa institucional: Universidad de Granada A nivel institucional, el proceso de movilidad internacional en la **Universidad de Granada** se rige por:

- El **Reglamento de Movilidad Internacional de Estudiantes de la Universidad de Granada** [49], que establece las condiciones generales de participación, reconocimiento académico y obligaciones de los estudiantes de movilidad saliente y entrante.
- El **Reglamento sobre Reconocimiento de Créditos en Programas de Movilidad Internacional de la ETSIIT** [14], que regula específicamente los procedimientos de reconocimiento de créditos para los estudiantes de esta Escuela Técnica Superior.

Estos reglamentos recogen las distintas modalidades de participación, los derechos y obligaciones de los estudiantes, así como los requisitos y criterios generales aplicables. Además, regulan aspectos específicos como la ampliación de la estancia, las posibles renuncias o la gestión de situaciones extraordinarias.

En el contexto de la **Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación**, además de la normativa oficial, se pone a disposición de los estudiantes un **Manual para la Elaboración del Acuerdo de Estudios** [15], que ofrece orientaciones prácticas sobre la selección de asignaturas, el procedimiento de aprobación del acuerdo y las herramientas disponibles para su gestión.

Este manual supone una fuente complementaria muy útil para entender las necesidades actuales y las posibles áreas de mejora que pretende abordar la plataforma propuesta en este trabajo.

Además, se utilizará para explicar y comprender el proceso de acuerdos de estudios, ya que este se encuentra regulado bajo el **Reglamento de la Universidad de Granada sobre Movilidad Internacional de Estudiantes** [49]. En este proceso, la **Sede Electrónica de la UGR** adquiere un papel clave como plataforma oficial para la formalización del acuerdo. A través de la sede, el estudiante puede cumplimentar, firmar y registrar de forma electrónica su solicitud, así como añadir la propuesta acordada con su tutor docente.

2.1.2. Procedimiento actual para la gestión de acuerdos de estudios

Cómo ya se ha definido en el apartado anterior de conceptos clave [2.1.1], el Acuerdo de Estudios es un documento esencial que garantiza el reconocimiento de las asignaturas cursadas durante la estancia en la universidad de destino. Debe entenderse como un contrato, y como tal, obliga a las partes firmantes a cumplir y respetar una serie de cláusulas [15].

Para que el reconocimiento de créditos sea válido, el Acuerdo de Estudios debe:

- Establecer la equivalencia entre las asignaturas cursadas en destino y asignaturas de la UGR.
- Ajustarse a un máximo del 50 % de los créditos totales de la titulación cursada en la UGR.
- Ser firmado por las tres partes (estudiante, UGR y universidad de destino).

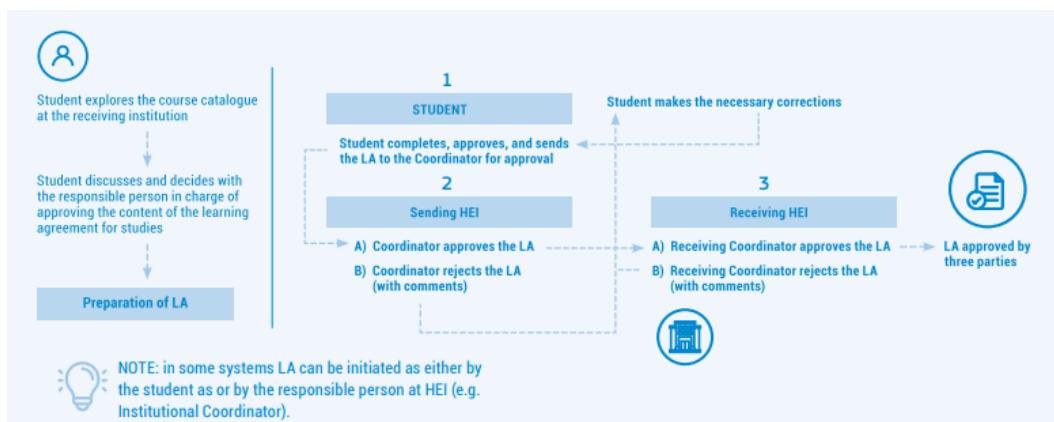


Figura 2.1: Proceso de aprobación del Acuerdo de Estudios. Fuente: European Commission [20]

En el apartado de *Motivación 1.2* ya se detalla que la elaboración de este documento y la búsqueda de información son uno de los aspectos más críticos del programa de movilidad. Conocer a fondo el procedimiento actual es esencial para entender las necesidades que motivan la plataforma propuesta en este Trabajo de Fin de Grado, cuyo objetivo principal es facilitar, sistematizar y agilizar dicho proceso.

Elaboración de la propuesta

La base fundamental que debe guiar la elaboración del Acuerdo de Estudios es asegurar la formación del estudiante durante su estancia en el extranjero.

1. **Búsqueda de información.** El estudiante debe recopilar los programas oficiales de las asignaturas de destino (contenidos, competencias, carga ECTS), preferentemente desde las webs oficiales o contactando con responsables académicos.
2. **Elaboración del dossier.** El estudiante debe realizar un dossier en formato electrónico (.doc, .docx, .odt o .pdf) que incluya:
 - Programas oficiales de las asignaturas de destino.
 - Carga lectiva de cada asignatura.
 - Fuente oficial de los datos (preferiblemente URLs).
3. **Redacción de la propuesta de Acuerdo de Estudios.** Usando el formulario normalizado proporcionado por la UGR (en formato .doc), el estudiante elaborará una propuesta que relacione las asignaturas de destino con las asignaturas de la UGR, conforme a los principios de equivalencia académica (contenidos, competencias y cargas docentes similares).
4. **Negociación y validación con el tutor docente.** El estudiante se pondrá en contacto con el tutor docente del intercambio para que valore y evalúe su propuesta inicial. El tutor podrá solicitar modificaciones, y tras las correcciones oportunas, aprobará el envío al Subdirector/a de Internacionalización de su facultad para su revisión definitiva.

Una vez se ha validado la propuesta, la **Sede Electrónica de la Universidad de Granada** adquiere un papel clave en la formalización de forma telemática del Acuerdo de Estudios. A través de esta plataforma el estudiante debe llenar el formulario correspondiente, reflejando la propuesta acordada, adjuntar el documento en formato PDF y firmar electrónicamente la solicitud. El sistema genera de manera automática el acuerdo que será firmado posteriormente por los responsables académicos pertinentes y enviado a la universidad de destino. La Sede Electrónica actúa como canal oficial de notificaciones mediante el sistema Hermes, y permite al estudiante consultar el estado del proceso y descargar el acuerdo una vez formalizado [51].

El reconocimiento de las asignaturas puede realizarse a través de varios mecanismos:

- **Reconocimiento individual.** Cuando una asignatura de destino presenta similitud en contenidos, competencias y carga lectiva con una asignatura de la UGR, se realiza un reconocimiento directo.
- **Reconocimiento de asignaturas individuales más créditos de optatividad.** Si la carga cursada en una asignatura de destino supera notablemente la carga de su equivalente en la UGR, el exceso puede reconocerse como créditos de optatividad.

- **Reconocimiento únicamente como optatividad.** Si no existe una asignatura en la UGR que guarde suficiente relación académica con la asignatura de destino, la materia podrá reconocerse como créditos de optatividad, siempre que el contenido tenga valor formativo.
- **Reconocimiento de bloques de asignaturas.** Puede darse el caso de que dos o más asignaturas de destino se correspondan con una o más asignaturas de la UGR. En este caso, es obligatorio aprobar todas las asignaturas del bloque para que se pueda reconocer en la UGR.
- **Reconocimiento de cursos de idiomas.** Los cursos de idiomas realizados en la universidad de destino podrán reconocerse hasta un máximo de 6 créditos ECTS como créditos de optatividad, siempre que estén reflejados en el certificado oficial de calificaciones (Transcript of Records) y se indique su carga lectiva.
- **Reconocimiento del Trabajo Fin de Grado (TFG).** Se puede reconocer si se realiza una memoria específica y se presentan contenidos equivalentes a los exigidos en la UGR.

2.1.3. Análisis crítico y principales limitaciones. Problema a resolver

Tras describir el procedimiento actual para la elaboración del Acuerdo de Estudios en el contexto de la movilidad internacional, es posible señalar diversas limitaciones que afectan a su eficiencia, claridad y equidad.

- **Falta de centralización de la información:** Los datos necesarios para la elaboración del acuerdo (planes de estudio, reconocimientos previos, idiomas requeridos, etc.) se encuentran dispersos en múltiples fuentes no estandarizadas, como páginas web de facultades, correos personales, documentos no oficiales o testimonios de estudiantes anteriores.
- **Dificultad para consultar equivalencias anteriores:** No existe una base de datos o una plataforma accesible para conocer qué asignaturas han sido previamente reconocidas en cada destino. Esta información se transmite de manera informal.
- **Proceso manual y repetitivo:** El estudiante debe elaborar documentos Word o PDF, consultar manualmente las asignaturas ofertadas en cada destino y comunicarse con tutores por correo electrónico. No hay una herramienta unificada para automatizar o facilitar estos pasos.
- **Falta de herramientas de búsqueda avanzada:** No es posible realizar una búsqueda inversa del tipo “¿en qué universidades puedo reconocer esta asignatura?”, lo que afecta a la capacidad de decisión del estudiante.

- **Carga administrativa para el profesorado:** El tutor docente y el responsable académico deben revisar manualmente cada propuesta, sin apoyo digital para verificar equivalencias ya aceptadas o realizar sugerencias más rápidamente.
- **Limitada trazabilidad y transparencia:** No hay forma estructurada de hacer seguimiento del estado de la solicitud, las correcciones solicitadas o los criterios aplicados en años anteriores.

Estas limitaciones ponen de manifiesto la necesidad de una herramienta tecnológica que integre la información dispersa, facilite el proceso de planificación y solicitud, y permita una toma de decisiones más eficiente, transparente y equitativa. Esta necesidad es la que motiva el desarrollo de la plataforma propuesta en el presente proyecto.

Con esta plataforma, se reducirá la incertidumbre, la carga administrativa y se mejorará la equidad y eficiencia del proceso Erasmus+ en la UGR.

Capítulo 3

Estado del arte

3.1. Introducción

3.1.1. Contexto, problema y objetivos

Tal y como se señaló en la sección *Motivación* [1.2], los programas de movilidad conllevan importantes desafíos de carácter administrativo. A partir de esta problemática, previamente descrita en el capítulo anterior [2], esta revisión del estado del arte examina las soluciones tecnológicas existentes orientadas a optimizar la experiencia del estudiantado en programas como Erasmus+.

La transformación digital en la administración educativa, acelerada tanto por las demandas actuales como por situaciones excepcionales como la pandemia de la COVID-19, ha dado lugar a numerosas iniciativas tecnológicas pensadas para simplificar trámites que antes resultaban pesados y complejos. Aun así, estas soluciones siguen estando muy dispersas y poco conectadas entre sí, lo que hace necesario revisar de forma crítica el estado actual de las plataformas de gestión Erasmus.

Esta revisión del estado del arte se alinea con el objetivo específico **OE2** establecido en el Capítulo *Introducción* [1.3], así como con cada uno de sus subobjetivos derivados (OE2.1-OE2.4). Su propósito es analizar qué herramientas, prácticas y enfoques tecnológicos se han utilizado hasta ahora en la gestión de la movilidad Erasmus, con especial atención a la accesibilidad de información importante para el alumnado y el proceso de elaboración del Acuerdo de Estudios. Además, se busca detectar carencias o vacíos que justifiquen el desarrollo de una nueva solución adaptada a las necesidades reales del estudiantado, en especial, en el contexto de la Universidad de Granada.

Esta revisión ha sido realizada siguiendo un enfoque sistemático, basado en la pregunta: *¿Qué plataformas o herramientas web existen para la gestión de la movilidad Erasmus y cómo integran acuerdos de estudios, experiencias y equivalencias?*

Así mismo, en lo relativo a la estrategia de búsqueda seguida, se han empleado:

- **Palabras clave :** “student platform”, “Erasmus platform”, “student exchange platform”, “Erasmus Without Paper”, “digital learning agreement”, “student mobility platform”, entre otras.
- **Fuentes consultadas:** Buscadores web convencionales para identificar soluciones no académicas disponibles públicamente en la web, incluyendo plataformas no oficiales de la Comisión Europea[23, 32, 26], documentaciones técnicas [34], redes y comunidades de estudiantes [11], y herramientas de gestión de movilidad [43, 31].
- **Criterios de inclusión:**
 - Plataformas o sistemas digitales dirigidos específicamente a la gestión de movilidad estudiantil internacional.
 - Soluciones que abordan acuerdos de estudios o exploración de equivalencias ya existentes.
 - Documentación académica o técnica sobre digitalización de procesos Erasmus.
 - Disponibilidad actual y accesibilidad para análisis.

- **Criterios de exclusión:**

- Plataformas generales de gestión académica sin funcionalidades específicas para movilidad internacional.
- Documentación que no incluya análisis de funcionalidades o implementación.
- Prototipos no implementados o sin evidencia de uso real.

3.2. Análisis de soluciones existentes

Para cada recurso identificado se ha seguido una metodología de búsqueda y clasificación de proyectos inspirada en la propuesta por Alonso de Castro y García-Peñalvo [1], originalmente diseñada para el análisis de iniciativas recogidas en la plataforma oficial de Erasmus+ [24]. Dado que en dicho repositorio no se han encontrado proyectos que aborden de forma directa la problemática que nuestra plataforma pretende resolver, se ha aplicado esta misma metodología al estudio y comparación de otras herramientas y plataformas web consideradas relevantes.

1. Análisis de sus funcionalidades principales.
2. Evaluación de su enfoque hacia el usuario (institucional vs estudiante).
3. Determinación de su alcance en los procesos de movilidad.
4. Identificación de fortalezas y debilidades.

3.2.1. Plataformas institucionales europeas. European Student Card Initiative

La Iniciativa del Carné Europeo de Estudiante (*European Student Card Initiative, ESCI*) consiste en una estrategia de la Unión Europea para promover la participación del alumnado en actividades educativas y culturales y la creación de un Espacio Europeo de Educación para 2025 [27]. Está estructurada en tres componentes fundamentales:

- **Carné Europeo de Estudiante:** *European Student Card*. Sistema que simplifica la movilidad de los estudiantes en Europa facilitando la verificación del estatus académico a nivel europeo. Permite que los estudiantes en movilidad accedan a servicios universitarios y beneficios externos (transporte público, descuentos) en su institución de acogida sin necesidad de gestionar documentación adicional. Busca eliminar barreras administrativas.



Figura 3.1: Diseño del Carné Europeo de Estudiante [29]

- **Erasmus+ App:** Aplicación oficial para gestionar la movilidad Erasmus+, con búsqueda de oportunidades, trámites, descuentos y guías paso a paso [22].
- **Erasmus Without Paper:** Infraestructura digital que conecta los sistemas informáticos de las instituciones adscritas para gestionar las movilidades Erasmus+ completamente en línea [25].

En esta revisión del estado del arte, analizaremos con mayor profundidad los componentes **Erasmus+ App** y **Erasmus Without Paper**, en los apartados [3.2.1] y [3.2.1] respectivamente. Sin embargo, si bien el Carné Europeo de Estudiante constituye un componente muy importante de la estrategia de digitalización del Espacio Europeo de Educación, no se va a profundizar en su análisis al no tratarse de una herramienta orientada directamente a los procesos de gestión o experiencia del estudiante en movilidad. Aun así, es importante mencionarlo por su relevancia en el marco de la *European*

Student Card Initiative ya que facilita la identificación del estudiante en servicios a nivel europeo.

Erasmus+ App

Las aplicaciones móviles han cambiado por completo la forma en que usamos nuestros dispositivos, haciéndolos más prácticos y accesibles en el día a día. Siguiendo esta tendencia, la Comisión Europea, viendo la necesidad de hacer más sencilla y rápida la gestión de la movilidad estudiantil, puso en marcha iniciativas para digitalizar y mejorar estos procesos. En 2017 impulsó el desarrollo de la aplicación gratuita **Erasmus+ App**. El proyecto fue llevado a cabo por la Eötvös Loránd University, la European University Foundation y la Erasmus Student Network AISBL, en colaboración con la Aristotle University of Thessaloniki y EWORX, como parte de la conmemoración del trigésimo aniversario del programa Erasmus [4]. Esta aplicación constituyó el primer intento institucional de ofrecer una herramienta digital específicamente orientada al estudiantado en movilidad, poniendo en el centro de su diseño la experiencia del usuario.



Figura 3.2: Logo de Erasmus+ App [3]

A grandes rasgos, sus funcionalidades son:

- **Acceso a Información y Servicios.** Los estudiantes pueden acceder a la información y servicios necesarios a lo largo de su movilidad.
- **Búsqueda de Eventos y Ofertas.** Los estudiantes pueden buscar eventos verificados y ofertas especiales para estudiantes en toda la UE.
- **Compartir Consejos y Experiencias.** Los estudiantes pueden compartir experiencias, consejos y trucos sobre cómo aprovechar su experiencia. Esto se encuentra en las secciones “Top Tips” y “Stories”.
- **Checklist de pasos administrativos.** Los estudiantes pueden seguir su proceso mediante listas de tareas.
- **Soporte lingüístico online.** Los estudiantes pueden acceder a recursos online para aprender nuevos idiomas y practicar con conversaciones en vivo con un mentor.

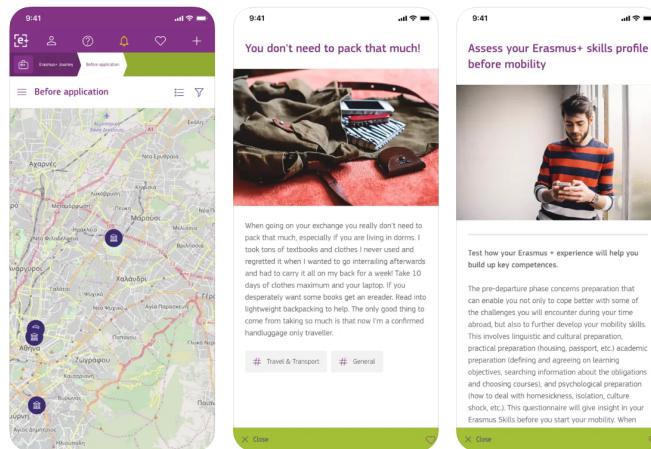


Figura 3.3: Vista de la aplicación de *Erasmus+ App*. Imagen sacada de Play Store [30].

Las principales limitaciones de esta aplicación tienen que ver con su alcance real. Aunque resulta útil como punto de información, no permite centralizar todos los trámites. Por ejemplo, la elaboración del Acuerdo de Estudios sigue siendo manual: la app solo ofrece la posibilidad de consultarla una vez completado y redirige al usuario a la plataforma externa *OLA. Learning Agreement*. Del mismo modo, el reconocimiento de créditos no se gestiona directamente desde la aplicación.

A esto se añaden ciertas barreras técnicas, ya que la herramienta depende de sistemas externos, como portales nacionales, que en ocasiones presentan problemas de usabilidad. Además, no todas sus funcionalidades están abiertas a quienes todavía no forman parte de Erasmus+, lo que limita su utilidad para estudiantes que se encuentran en una fase de exploración de destinos o programas.

Por último, la aplicación carece de características clave que serían de gran ayuda, como un buscador de asignaturas reconocidas o una herramienta que permita explorar de forma sencilla los destinos disponibles.

Erasmus Without Paper (EWP)

Erasmus Without Paper es la columna vertebral de la estrategia de digitalización del programa Erasmus+ de la Comisión Europea. Como se ha mencionado con anterioridad, es uno de los componentes principales de *European Student Card Initiative* y surge para digitalizar los procesos administrativos entre instituciones de educación superior (IES). Es una plataforma digital de carácter institucional (no una aplicación móvil) cuyo propósito principal, según su documentación oficial [34], es sustituir los tradicionales flujos de trabajo en papel por intercambios digitales seguros entre universidades. Al mismo tiempo, busca reducir la carga administrativa a través de APIs estandarizadas y permitir que cada institución mantenga sus propios sistemas de gestión conectándolos,

a la vez, a una red común e interoperable.



Figura 3.4: Logo de *Erasmus Without Paper* [9].

Funciona a través de dos módulos complementarios [34]: el **EWP Network**, que conecta los sistemas de información de las universidades mediante APIs, y el **EWP Dashboard**, una solución web que se analizará con mayor detalle en el apartado [3.2.1].

Las funcionalidades principales de ***Erasmus Without Paper*** se centran en digitalizar los trámites esenciales de la movilidad. Entre ellas se encuentra la posibilidad de realizar nominaciones electrónicas de estudiantes (3.5), que permiten enviar candidaturas de manera online y facilitan la comunicación directa entre los sistemas de las universidades.

Otra de las grandes ventajas es la gestión electrónica de los Acuerdos de Estudios (3.6). Con este sistema, el documento puede crearse, modificarse y validarse de forma tripartita (estudiante, universidad de origen y universidad de destino), además de admitir modificaciones posteriores al acuerdo inicial.

También se está probando el intercambio automatizado de registros académicos o *Transcript of Records* (3.7), que busca un formato común para todas las instituciones y pretende evitar el envío manual de expedientes. Por último, destaca la firma digital de convenios interinstitucionales (3.8), que sustituye el papeleo físico y garantiza la validez oficial de estos acuerdos.

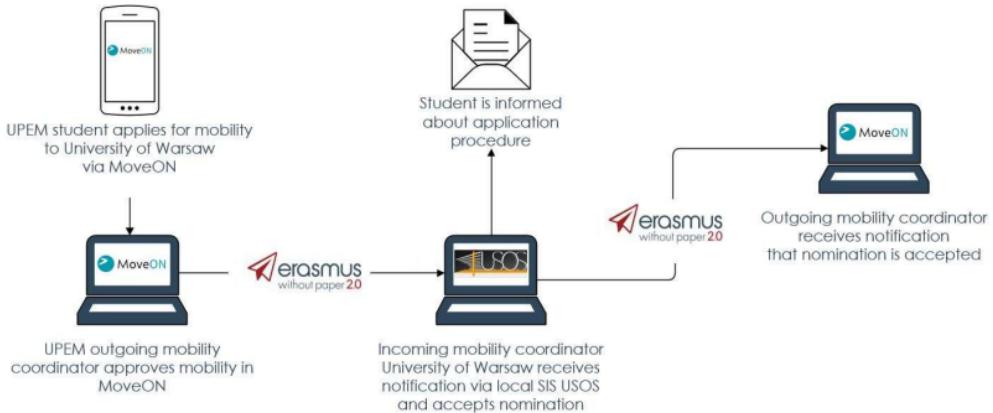


Figura 3.5: Proceso digital de nominaciones digitales de estudiantes mediante *Erasmus Without Paper*[41].

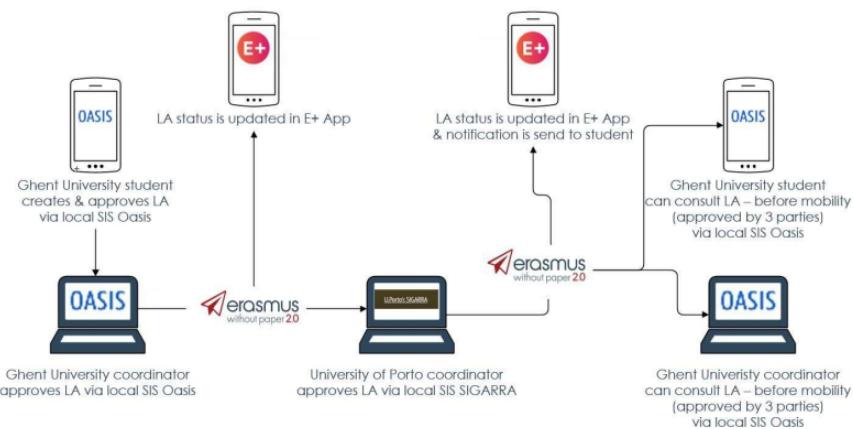


Figura 3.6: Proceso digital de firmado de acuerdos de estudios mediante *Erasmus Without Paper*[41].



Figura 3.7: Proceso digital de envío de registros académicos mediante *Erasmus Without Paper*[41].

Ahora bien, pese a estos avances, la herramienta presenta limitaciones importantes. Desde la perspectiva del estudiantado, la primera barrera es que no interactúan directamente con EWP: dependen siempre de que su universidad habilite los procesos. Esto reduce su autonomía y no resuelve otras necesidades, como la búsqueda de información sobre equivalencias, la selección de destinos o la recepción de notificaciones sobre el estado de sus trámites, lo que genera cierta incertidumbre.

En el plano institucional, la implementación es desigual. Muchas universidades europeas han tenido dificultades técnicas para integrar sus sistemas con EWP y, en el caso de la Universidad de Granada, en 2025 solo se utilizaba para los acuerdos interinstitucionales, mientras que otros procesos, como el Acuerdo de Estudios o el *Transcript of Records*, continuaban gestionándose manualmente desde la Sede Virtual (véase capítulo 2).

A ello se suma que algunos procesos clave, como la confirmación de llegada o salida de

estudiantes, o la validación completa del *Transcript of Records*, siguen en fase piloto. Y, finalmente, persiste cierta incertidumbre sobre el futuro de la plataforma: el contrato del consorcio EWP+ finaliza en 2025/26 y aún no está claro si el proyecto seguirá operativo con las mismas funcionalidades [33].

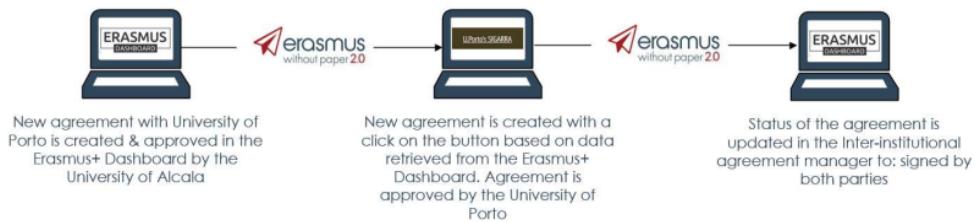


Figura 3.8: Proceso digital de firmado de acuerdos interinstitucionales mediante *Erasmus Without Paper*[41].

Online Learning Agreement (OLA)

La plataforma online **Online Learning Agreement** ha sido desarrollada por varias universidades y redes de estudiantes Erasmus. Se trata de un proyecto cofinanciado por la Comisión Europea, responsable de la información oficial del programa Erasmus+ [35]. Su objetivo es digitalizar el proceso de firma y gestión de los acuerdos de estudios en movilidades Erasmus+. Representa el estándar de referencia para este trámite clave en el programa Erasmus+, cuya importancia ya ha sido explicada. Dada la naturaleza de la herramienta que propone este Trabajo de Fin de Grado, el OLA representa una referencia valiosa.



Figura 3.9: Logo de *OLA*. *Online Learning Agreement*

Como parte del ecosistema de *Erasmus Without Paper (EWP)*, la plataforma **OLA (Online Learning Agreement)** ofrece un conjunto de funcionalidades diseñadas para digitalizar y simplificar la gestión de los acuerdos de estudios.

En primer lugar, permite la **creación digital del acuerdo** a través de una interfaz web intuitiva y fácil de usar. El estudiante puede seleccionar las asignaturas de movilidad

con sus créditos ECTS correspondientes, mientras el sistema valida automáticamente que se cumplen los requisitos académicos mínimos. Este proceso no solo reduce tiempos de gestión, sino que también disminuye errores derivados del manejo manual de documentos.

El OLA incorpora además un **flujo de aprobación tripartito**, en el que intervienen de manera secuencial el propio estudiante, la universidad de origen y la universidad de destino. Cada firma queda registrada digitalmente y el sistema genera un historial de versiones con sello temporal que garantiza la trazabilidad.

Otra de sus ventajas es la **gestión dinámica de modificaciones**. Si durante la estancia el plan de estudios necesita cambios, estos pueden introducirse directamente en la plataforma, siguiendo el mismo proceso de validación que el acuerdo inicial. La herramienta incluso permite comparar versiones para visualizar qué se ha modificado en cada actualización.

Finalmente, el OLA se integra con otros sistemas institucionales, como el *EWP Dashboard*, y puede consultarse también desde la *Erasmus+ App*, lo que asegura coherencia y conectividad entre las distintas plataformas del programa Erasmus+.

El acceso a la plataforma se realiza mediante las credenciales académicas de cada estudiante. En mi caso, he probado las distintas funcionalidades utilizando mis propias credenciales, lo que me ha permitido experimentar de primera mano con el proceso de creación y firma de un acuerdo de estudios, así como con la introducción de datos personales e institucionales. Para poder acceder a la plataforma, hay que identificarse con las credenciales académicas. Para probar las distintas funcionalidades y la interfaz, he accedido con mis propias credenciales:

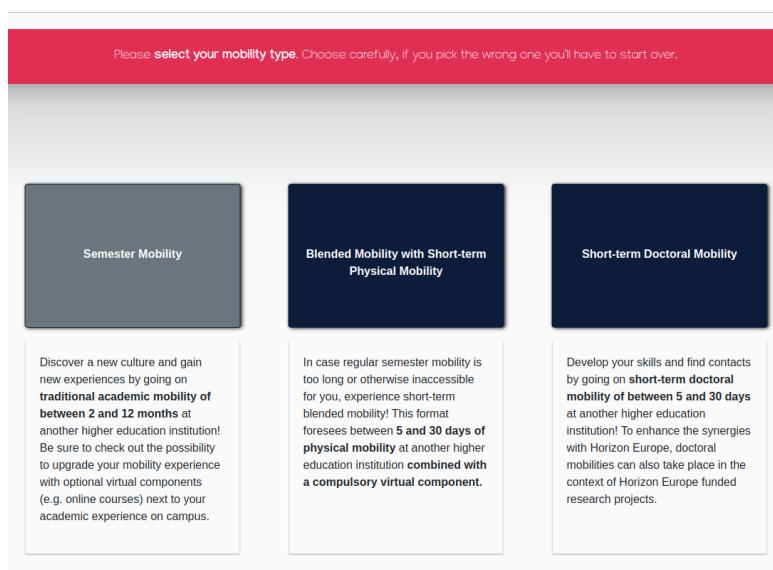


Figura 3.10: Selección de tipos de movilidad en la plataforma OLA. *Online Learning Agreement*. Elaboración propia

3.2. Análisis de soluciones existentes

The screenshot shows the 'OLA Online Learning Agreement' interface. At the top, there's a navigation bar with links for 'ABOUT', 'FAQ', 'ELDER OLA', 'FOR TRAINEES', 'MY LEARNING AGREEMENTS', 'MY ACCOUNT', and 'LOG OUT'. Below the navigation, a red banner displays the instruction: 'Select your home institution from the list and indicate the contact and responsible person. It can be the same person but it is the responsible who will receive the invitation to review and sign the agreement.' A horizontal progress bar at the top indicates the user is on step 2 of 6, titled 'Sending Institution Information'. The main form area is divided into sections for 'Sending Institution' and 'Sending Responsible Person'. The 'Sending Institution' section requires fields for Country (Spain), Name (UNIVERSIDAD DE GRANADA), Faculty/Department (ETS. INFORMATICA Y TELECOMUNI.), Address (Granada), and Erasmus Code (E_GRANADA01). The 'Sending Responsible Person' section requires fields for First name(s) (Pepito), Last name(s) (Perez), Position (Coordinador), Email (pepito@ugr.es), and Phone number (+). A note below this section specifies that the responsible person must be filled in unless it differs from the contact person. The 'Sending Administrative Contact Person' section is partially visible on the right, showing fields for First name(s), Last name(s), Position, Email, and Phone number, along with a note about administrative contact persons. At the bottom, there are 'Previous' and 'Next' buttons, and a footer containing logos for it.auth, ESN (Erasmus Student Network), EUF (European University Foundation), and the Erasmus+ Programme, along with links for 'Privacy Policy and Terms and Conditions', 'RELEASE NOTES', 'PRIVACY POLICY', and 'TERMS AND CONDITIONS'.

Figura 3.11: Inserción de la información de la institución de origen en la plataforma *OLA. Online Learning Agreement*. Elaboración propia

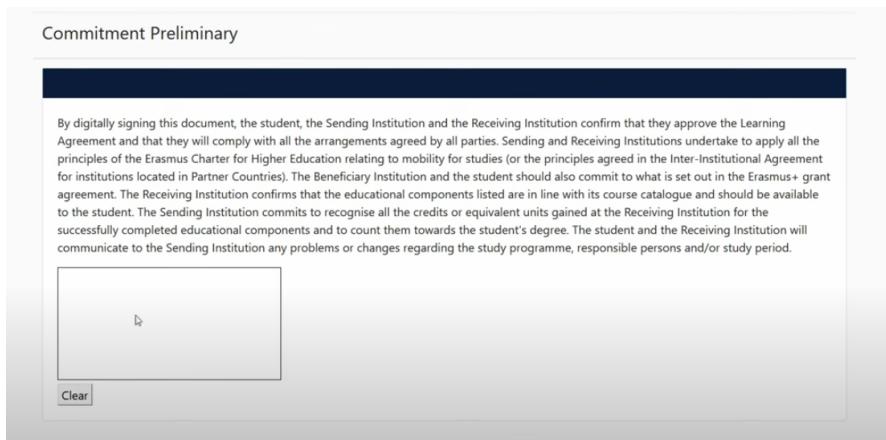


Figura 3.12: Acción de firmar un nuevo Acuerdo de Estudios usando *OLA*. *Online Learning Agreement*. Elaboración propia

Las limitaciones más destacadas de esta plataforma [35] tienen que ver, sobre todo, con el alcance real de sus funcionalidades. El OLA está **centrado únicamente en una fase del proceso** —la elaboración del Acuerdo de Estudios—, pero no va más allá. Esto significa que no ofrece herramientas complementarias que podrían resultar muy útiles, como la posibilidad de consultar experiencias de otros estudiantes, explorar destinos o buscar asignaturas compatibles. Además, aunque lo utilizan estudiantes y responsables académicos como subdirectores o coordinadores, no contempla la participación de figuras intermedias, por ejemplo, los tutores de destino.

Otro aspecto importante es la fuerte **dependencia institucional**: para que el sistema funcione, es necesario que tanto la universidad de origen como la de destino estén plenamente integradas en la plataforma. A ello se suma que el acceso está condicionado, ya que los estudiantes solo pueden entrar con sus credenciales académicas oficiales, lo que limita su uso en fases tempranas de exploración o planificación de la movilidad.

EWP Dashboard

Se trata de una plataforma administrativa orientada a gestores universitarios, que ofrece una interfaz web para la gestión de acuerdos interinstitucionales, nominaciones y *Online Learning Agreements* (OLA). Constituye uno de los componentes clave de ***Erasmus Without Paper*** y está dirigida exclusivamente a las oficinas de relaciones internacionales de las universidades, quedando fuera del alcance de estudiantes o tutores académicos [21]. Sus principales objetivos son sustituir los procesos en papel mediante intercambios digitales estandarizados y servir como solución para aquellas instituciones que no disponen de sistemas propios compatibles con ***Erasmus Without Paper***. De esta forma, facilita la gestión digital de acuerdos, nominaciones y OLAs. Además, la plataforma se integra con la *Erasmus+ App*, lo que permite a las instituciones interactuar

directamente con estudiantes entrantes y salientes a través de la aplicación.



Figura 3.13: Diferentes logos de *EWP Dashboard*. [28]

Esta herramienta incorpora varias funcionalidades que resultan especialmente útiles para avanzar en la digitalización de la movilidad Erasmus+. Por un lado, **facilita la gestión** de acuerdos y procesos administrativos al centralizar trámites institucionales, permitir la creación, negociación y firma digital de convenios entre universidades, y ofrecer la exportación de documentos en formatos estandarizados como PDF o JSON.

También **simplifica la administración de nominaciones**, ya que las candidaturas de estudiantes pueden enviarse de manera electrónica a las universidades de destino, con la ventaja añadida de integrarse con la *Erasmus+ App*, lo que mejora la comunicación entre instituciones y alumnado.

En cuanto al seguimiento del Acuerdo de Estudios, el sistema **habilita la firma digital trilateral** (estudiante, universidad de origen y universidad de destino), reduce la carga burocrática de los coordinadores y permite introducir modificaciones después de la llegada del estudiante, manteniendo siempre un historial transparente de cambios.

A pesar de su papel en la digitalización de los procesos Erasmus+, el *EWP Dashboard* presenta varias limitaciones que dificultan su eficacia y adopción. Una de las más relevantes son las **barreras de adopción**. Para que el sistema funcione, necesita integrarse con la infraestructura tecnológica ya existente en cada universidad, lo que puede resultar complejo cuando se trata de plataformas antiguas o poco compatibles. Además, no todas las instituciones cuentan con los recursos técnicos o el presupuesto necesario para realizar estas integraciones. A ello se suma que el acceso está restringido exclusivamente al personal administrativo, dejando fuera a perfiles como estudiantes o tutores, lo que reduce su alcance real.

También se han señalado **problemas técnicos y de usabilidad**. La interfaz está pensada para personal especializado y resulta poco intuitiva, lo que dificulta su manejo por usuarios menos familiarizados. Incluso procesos clave, como las nominaciones de estudiantes, en ocasiones siguen requiriendo intervención manual, lo que limita la prometida automatización.

Por último, existe un **futuro incierto** respecto a la continuidad de la herramienta.

El contrato del consorcio EWP+ finaliza en 2025/26 y no está claro si el *EWP Dashboard* mantendrá sus funcionalidades actuales o si sufrirá cambios significativos en su alcance [33].

A continuación, se presentan capturas de pantalla de *EWP Dashboard* que sirven para ilustrar su interfaz y funcionalidades principales.

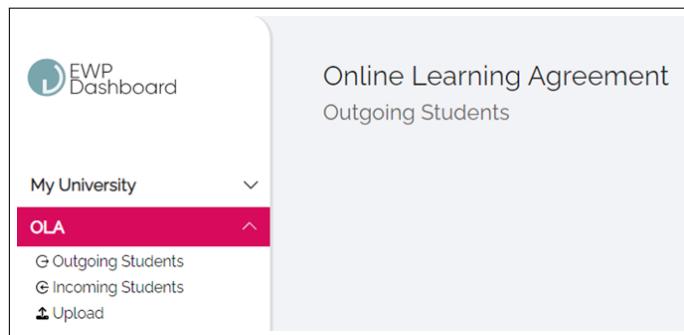


Figura 3.14: Vista de secciones del Dashboard

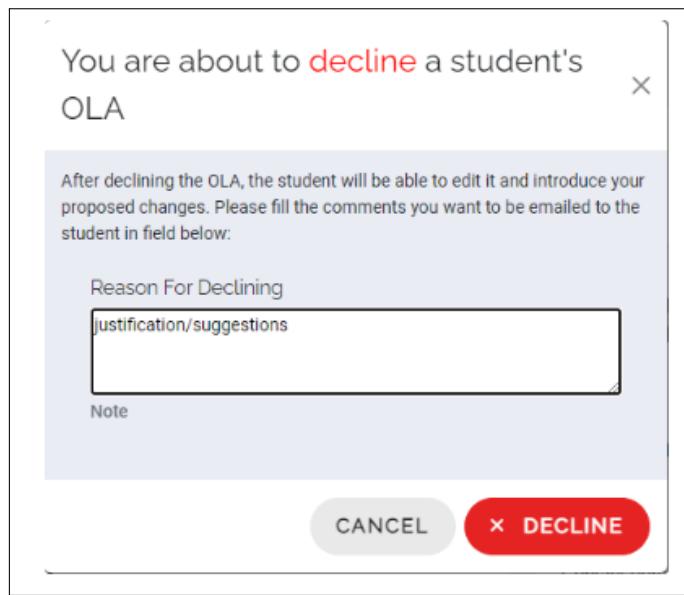


Figura 3.15: Denegación de un Acuerdo de Estudios de un estudiante

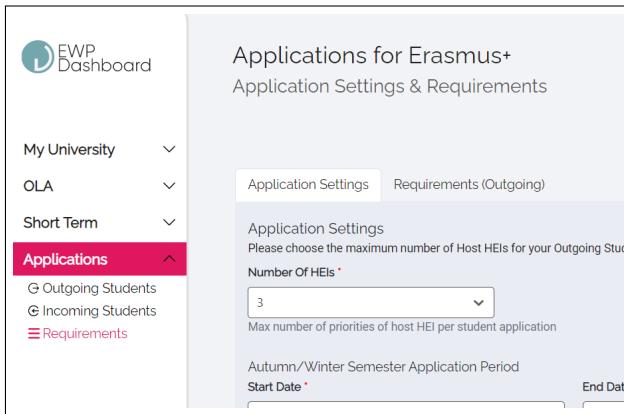


Figura 3.16: Aplicación a una movilidad

Tras analizar las herramientas oficiales que constituyen la *European Student Card Initiative*, queda claro que funcionan como un ecosistema interconectado pero fragmentado. La aplicación *Erasmus+* sirve como punto de entrada, mientras que la plataforma EWP (*Erasmus Without Paper*) y su panel de control (*EWP Dashboard*) permiten a las instituciones gestionar, revisar, nominar y confirmar las solicitudes de manera centralizada y digitalizada.

No obstante, más allá del ámbito institucional, también han surgido diversas iniciativas impulsadas por comunidades estudiantiles y organizaciones independientes. A continuación, se analizan algunas de estas herramientas no institucionales que aportan valor al proceso de movilidad, ya que también son referentes a tener en cuenta para el desarrollo de la solución que se propone en este Trabajo de Fin de Grado.

3.2.2. Iniciativas comunitarias y no institucionales

Junto a las soluciones institucionales ofrecidas por la Comisión Europea, han surgido varias plataformas desarrolladas por organizaciones independientes o asociaciones estudiantiles. Estas iniciativas suelen estar impulsadas por la propia comunidad Erasmus y nacen con el objetivo de responder a necesidades que no se cubren con las herramientas oficiales. A menudo, ofrecen servicios complementarios relacionados con la integración social, información de experiencias de otros estudiantes o preparación del viaje.

En esta sección se analizan algunas de las más relevantes: *Erasmus Student Network (ESN)*, *ErasmusPlay* y *Erasmusu.com*.

Erasmus Student Network (ESN)

Erasmus Student Network (ESN) es la mayor asociación estudiantil europea sin ánimo de lucro, enfocada en apoyar la movilidad internacional. Fue fundada en 1989

y actualmente está presente en más de 520 instituciones de 42 países. Esta asociación opera a 3 niveles: local, nacional e internacional [17].

ESN trabaja bajo el principio “estudiantes ayudando a estudiantes” y para ello trabaja por los siguientes objetivos:

- Facilitar la integración social de los estudiantes internacionales.
- Proporcionar información relevante sobre programas de movilidad académica y contribuir a la mejora de estos programas.
- Representar las necesidades y defender los intereses de los estudiantes.

Si bien no constituye una plataforma única, ESN ha desarrollado numerosas herramientas digitales orientadas a la comunidad Erasmus, haciendo énfasis en aspectos sociales y de integración, siendo su página web oficial (<https://esn.org/>) el punto central de acceso a toda la información y servicios ofrecidos por la organización.



Figura 3.17: Logo de *Erasmus Student Network* [8].

Esta plataforma ofrece diversas funcionalidades clave:

- **Cobertura internacional con adaptación local**
 - Existen distintas plataformas locales y comunidades virtuales por destinos.
- **Portal informativo**
 - Es un lugar de acceso a información detallada sobre los programas de movilidad.
 - Los estudiantes pueden consultar guías por países, calendarios de eventos y testimonios de otros estudiantes.
 - Permite acceder también a asesoramiento académico, con información sobre reconocimiento de créditos o distintos procedimientos de este ámbito.
 - Su enfoque está especialmente orientado hacia las necesidades reales de los estudiantes.

■ ESNcard

- Es un sistema de registro que proporciona a los estudiantes descuentos exclusivos en transportes, alojamiento y establecimientos locales.
- También facilita el acceso preferente a eventos organizados por la ESN.

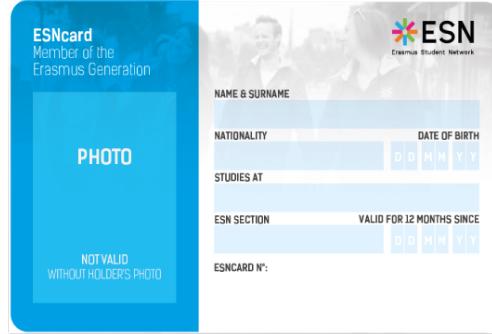


Figura 3.18: *ESN Card*[10]

■ Buddy System

- Conexión con estudiantes locales que actúan como mentores, facilitando su integración social.
- Es la primera plataforma de networking de tutoría entre estudiantes [16].
- Proporciona apoyo, acelera el aprendizaje del idioma local y la participación en actividades extracurriculares.

■ Acceso a recursos

- Estudiantes pueden acceder a materiales para su preparación lingüística y cultural, antes y durante su estancia en el extranjero.

Aunque la plataforma ofrece varias fortalezas, también presenta limitaciones que reducen su impacto real. Una de las más importantes es su **escasa integración con los procesos administrativos formales de las universidades**. Al no estar conectada con los sistemas institucionales, los estudiantes deben duplicar gestiones, y los documentos generados en la plataforma carecen de validez oficial en muchos trámites académicos.

Otro aspecto a tener en cuenta es su **enfoque secundario** en lo académico. La herramienta pone mayor énfasis en los componentes socioculturales de la movilidad, dejando en un segundo plano los contenidos académicos que son fundamentales para el reconocimiento de estudios.

Por último, la **fragmentación de sus secciones** puede afectar a la experiencia de uso. Al funcionar de forma descentralizada, cada área dispone de sus propios servicios, lo que dificulta que el usuario perciba la plataforma como un entorno único y cohesionado.

Erasmusu.com

Erasmusu.com es una plataforma digital que, desde su creación, se ha consolidado como una comunidad en línea. Ofrece una variedad de servicios para facilitar la experiencia de estudio en el extranjero. Fue desarrollada por exparticipantes del programa Erasmus+ para compartir experiencias y consejos sobre movilidad, así como para encontrar alojamiento y conocer gente.



Figura 3.19: Logo de *Erasmusu* [12].

Esta plataforma constituye una de las iniciativas más longevas y completas de apoyo a la comunidad Erasmus, y se diferencia de otras herramientas en que gran parte de su contenido proviene directamente de los propios usuarios [13].

Entre sus funcionalidades más destacadas se encuentra la **búsqueda de alojamiento**, que pone a disposición de los estudiantes un buscador especializado para encontrar opciones de vivienda. En él se pueden consultar habitaciones, pisos y residencias, con información detallada que facilita la comparación de precios y características.

La herramienta también integra **foros de discusión**, donde los estudiantes intercambian información, resuelven dudas y comparten experiencias relacionadas con sus destinos, universidades o asignaturas. Este espacio de interacción directa enriquece el conocimiento colectivo y ofrece un apoyo práctico para quienes están en proceso de movilidad.

Además, ofrece una amplia variedad de **puntos de información**. Los usuarios pueden crear sus propios blogs o consultar los de otros estudiantes, acceder a guías detalladas sobre ciudades y universidades, informarse acerca de los trámites más comunes, o incluso consultar rankings de destinos Erasmus. La plataforma incluye también secciones dedicadas a oportunidades de prácticas y empleo, lo que amplía su utilidad más allá de la experiencia académica y social.

3.2. Análisis de soluciones existentes

Ránking de universidades Erasmus

Posición	Universidad	Puntuación	Gente
1	Università degli Studi di Siena	*****	52
2	University of Manchester	*****	32
3	Uniwersytet Ekonomiczny w Krakowie	*****	27
4	Vilnius Gediminas Technological University	*****	26
5	Vrije Universiteit Amsterdam	*****	26

(Ranking de universidades)

Experiencias Erasmus

Añadir experiencia

- CURSO COMPLETO EN BRNO**
Me estoy adaptando bien, segunda ciudad más grande de la República checa, en la Universidad Mendel. Antes de nada comentar que los profesores en general tardan mucho en responder a los emails o mensajes, hasta varias semanas en contestar a una sola consulta. Me...
- Un erasmus diferente**
¿Cómo es vivir en Francia? Yo recomendaría como es la ciudad: es una ciudad muy pequeña con pocas cosas que hacer pero tiene mucha cultura. Yo estoy lleno de Erasmus y acabo cogiéndole mucho cariño. ¿Cómo es el ambiente estudiantil? Yo no lo sé...
- Medicina en Varsavia. La mejor experiencia de vida**
Hola :) Me llamo Iria. Hace dos años, estuve de erasmus en Varsavia estudiando medicina. Así me decidí por no hacer esta experiencia por los posts que lei aquí, pero estos posts no pueden estar más alejados de mi realidad y de lo de todos mis compañeros de...

(Foro de experiencias)

Expertos en alquiler para estudiantes

Alquila 100% online tu habitación ideal de forma fácil, rápida y segura

Encuentra tu nuevo piso en alquiler 100% online

Q A dónde irás? Buscar

EN TENDENCIA: Barcelona, Dublin, Madrid, Lisboa, + Ciudades +

+100.000 pisos para estudiantes disponibles en todo el mundo

+80.000 propiedades verificadas por nuestros homecheckers

+1 MILLÓN estudiantes internacionales compartiendo experiencias

(Buscador de alojamiento)

Figura 3.20: Diferentes vistas de la plataforma *Erasmusu.com*. Elaboración propia.

Si bien esta herramienta está orientada a cubrir las necesidades integrales del estudiante, también presenta varias limitaciones importantes. La primera es la **falta de integración institucional**, ya que no está conectada con los sistemas oficiales de las universidades ni con plataformas como OLA o EWP. Esto hace que su uso quede restringido en algunos aspectos clave del proceso de movilidad.

Otra limitación es la **ausencia de garantías sobre la veracidad de la información**. Al no contar con un sistema de verificación, los contenidos dependen totalmente de lo que aportan los usuarios. Esto provoca que la calidad y el nivel de detalle varíen considerablemente entre ciudades, generando una experiencia desigual.

Por último, se aprecia una **escasa atención al proceso académico**. La plataforma centra sus esfuerzos en la parte social de la movilidad, así como en la vivienda, la vida diaria o el ocio, mientras que apenas aborda cuestiones cruciales como la compatibilidad de asignaturas, el reconocimiento de créditos o la tramitación de los acuerdos de estudios.

3.3. Análisis comparativo. Conclusiones

Tras analizar individualmente las principales soluciones digitales existentes, tanto institucionales como comunitarias, se puede observar una fragmentación en cuanto al enfoque y los usos. Las plataformas oficiales se centran en aspectos administrativos concretos, mientras que herramientas como *Erasmusu* o *Erasmus Student Network* abordan experiencias estudiantiles o información sobre destinos, pero sin conexión directa con trámites académicos. Esta separación obliga al estudiante a consultar múltiples fuentes y a realizar tareas redundantes o manuales, afectando a la eficiencia y claridad del proceso.

Para facilitar una visión global de las soluciones analizadas, se presenta a continuación una tabla comparativa que sintetiza sus características más relevantes en relación con las necesidades identificadas para la gestión de la movilidad Erasmus+. Este análisis permitirá identificar con claridad los vacíos existentes y justificar cómo la plataforma web propuesta en este Trabajo de Fin de Grado puede aportar valor: unificando la consulta de experiencias, la selección de destinos, la gestión del Acuerdo de Estudios y la comunicación entre estudiantes y tutor.

Cuadro 3.1: Comparativa de funcionalidades entre herramientas del ecosistema Erasmus+

Funcionalidad	Erasmus App	EWP	OLA	EWP Dashboard	ESN	Erasmus+	Nuestra Plataforma
Gestión de acuerdos	X*	✓	✓	✓	X	X	✓
Búsqueda de destinos	X	X	X	X	✓	✓	✓
Exploración de asignaturas	X	X	X	X	X	X	✓
Información sobre equivalencias	X	X	X	X	X	X	✓
Buscador de alojamiento	X	X	X	X	X	✓	X
Soporte lingüístico	✓	X	X	X	X	X	X
Compartir experiencias	✓	X	X	X	✓	✓	X
Interfaz para estudiantes	✓	X	✓	X	✓	✓	✓
Interfaz para tutores	X	✓	X	✓	X	X	✓

* La Erasmus+ App permite consultar el Acuerdo de Estudios creado en OLA, pero no crearlo directamente.

Del análisis de las soluciones digitales actuales para la movilidad Erasmus+ se concluye que, aunque existen herramientas útiles, estas siguen estando fragmentadas y con un alcance limitado. Esta falta de integración obliga al estudiantado a utilizar diferentes recursos para completar el proceso: una plataforma para la gestión del acuerdo y las equivalencias, otra para la búsqueda de destinos, otra para consultar experiencias previas, y además el correo electrónico para comunicarse con su tutor y validar las propuestas.

Además, muchas de estas herramientas están diseñadas exclusivamente para los estudiantes, y otras únicamente para las universidades. El papel del tutor docente, quien acompaña y revisa el proceso de convalidación académica, apenas se contempla.

Frente a este panorama, existe una clara oportunidad para desarrollar una solución web que centralice en un único entorno las funcionalidades más relevantes para la gestión académica de la movilidad Erasmus. Su objetivo es resolver la fragmentación y proporcionar una experiencia unificada, centrada en las necesidades reales del estudiante y del tutor, descartando funciones más orientadas a la vida personal del estudiante, como la búsqueda de alojamiento, poniendo el foco en lo importante: ayudar al estudiante a tomar decisiones informadas sobre su plan de estudios, consultar equivalencias de asignaturas y equivalencias, ahorrar tiempo y papeleo, y facilitar al tutor su tarea de revisión.

Capítulo 4

Planificación

4.1. Metodología

Para organizar el desarrollo del proyecto se optó por una metodología ágil basada en **Scrum**, adaptada a las particularidades de un Trabajo Fin de Grado desarrollado de manera individual.

En su forma tradicional, Scrum distingue tres roles principales: el *Product Owner*, responsable de definir los requisitos y establecer prioridades; el *Scrum Master*, encargado de velar por la correcta aplicación de la metodología; y el equipo de desarrollo. En este proyecto se realizó una adaptación práctica de dichos roles: los tutores asumieron la función de *Product Owner*, orientando las prioridades y validando los avances al final de cada iteración, mientras que el alumno combinó los papeles de *Scrum Master* y desarrollador, ocupándose tanto de planificar los sprints¹ como de gestionar el proceso y ejecutar la implementación técnica.

El trabajo se dividió en **sprints mensuales**, cada uno con un conjunto de objetivos concretos y revisiones periódicas con los tutores. Estas revisiones funcionaron como instancias de validación, permitiendo comprobar el grado de cumplimiento, detectar problemas y reordenar las prioridades si era necesario.

En la práctica, se aplicó uno de los principios más importantes de Scrum: la **priorización de tareas**. Primero se desarrollaron aquellas que garantizaban un producto funcional lo antes posible —como la definición del modelo de datos, el backend básico y la primera interfaz en React—, y más adelante se fueron incorporando características adicionales como las equivalencias complejas, los dashboards de tutor y administrador o la mejora del diseño responsivo. Finalmente, se dejaron para los últimos sprints las tareas de menor prioridad, como refinar la comunicación entre estudiante y tutor más

¹Un *sprint* es un periodo de tiempo breve y definido (habitualmente de dos a cuatro semanas) en el que se establecen objetivos concretos de desarrollo. Al finalizar cada sprint se revisan los progresos y se ajusta la planificación para el siguiente ciclo.

allá de los comentarios en los acuerdos.

Aunque la metodología aplicada fue Scrum, se tuvieron que hacer ciertas **adaptaciones**. Al tratarse de un único desarrollador, el reparto de roles no podía seguirse de forma estricta, y en lugar de trabajar con un backlog muy dinámico, el proceso fue relativamente lineal: partiendo de lo más básico hasta llegar a las funcionalidades más específicas. Esta linealidad no fue un problema, ya que ayudó a mantener un orden natural en el desarrollo y permitió comprobar en todo momento que la plataforma era funcional antes de seguir ampliándola.

Durante el primer cuatrimestre (octubre 2024 – enero 2025) no se comenzó todavía con el desarrollo técnico, sino que se dedicó el esfuerzo a comprender los procesos administrativos de la movilidad Erasmus en la UGR y a recopilar información sobre destinos y equivalencias. Esta fase inicial sirvió para sentar las bases y evitar empezar a programar sin un marco claro.

A partir de febrero de 2025 se inició el desarrollo en **iteraciones sucesivas**, cada una con un alcance específico y una revisión final con los tutores. Este esquema permitió avanzar de manera constante, añadir funcionalidades de forma progresiva y cerrar el proyecto con un producto completo y validado.

La planificación se concretó en los siguientes sprints:

- **Sprint 0 (Octubre 2024 – Enero 2025): Análisis inicial.** Reunión inicial el 8 de octubre para cerrar el tema. Durante este sprint no se desarrolló código, sino que se estudió el procedimiento Erasmus, se analizaron plataformas similares (OLA, EWP, Erasmus+ App), se seleccionó el stack tecnológico y se recopiló información para las colecciones.
- **Sprint 1 (Febrero 2025): Inicio del desarrollo.** Tras la reunión del 25 de febrero, se definieron los casos de uso principales, la arquitectura cliente–servidor y el modelo de base de datos en MongoDB. Se sentaron así las bases del backend y del frontend.
- **Sprint 2 (Marzo 2025): Backend inicial.** Reunión de seguimiento el 18 de marzo. Se resolvieron dudas sobre la gestión de equivalencias y se validaron los primeros endpoints en Flask (usuarios, centros y grados) con autenticación básica.
- **Sprint 3 (Abril 2025): Primer prototipo de frontend.** Revisión intermedia el 15 de abril. Se diseñaron los primeros prototipos en Canva de la interfaz de usuario y las distintas vistas. Se construyó la estructura inicial en React con navegación lateral y un primer dashboard de estudiante. Se validó la conexión básica con el backend.
- **Sprint 4 (Mayo 2025): Decisión estratégica.** El 20 de mayo se decidió posponer la entrega a septiembre debido a las prácticas externas. En este sprint se consolidaron los requisitos y se planificó el desarrollo intensivo de verano.

- **Sprint 5 (Junio 2025): Funcionalidades clave.** Durante junio se avanzó en la gestión de destinos, equivalencias simples y el editor de acuerdos de estudios. Se probó la generación de PDFs oficiales y la integración con Cloudinary para imágenes. Avances enviados periódicamente al tutor.
- **Sprint 6 (Julio 2025): Roles avanzados.** Implementación de los dashboards de tutor y administrador, gestión de usuarios desde el rol admin y refinamiento de equivalencias complejas (bloques y optatividad). También se mejoró el diseño responsivo de la interfaz.
- **Sprint 7 (Agosto 2025): Cierre y validación final.** Se mostraron dos demos clave al tutor: la primera el 5 de agosto, con la plataforma ya funcional para estudiantes y tutores, y la segunda el 11 de agosto, con las funcionalidades finales (panel admin, notificaciones, equivalencias avanzadas). En la segunda mitad de agosto se realizaron ajustes menores, se cerró el desarrollo y se preparó la memoria y defensa.

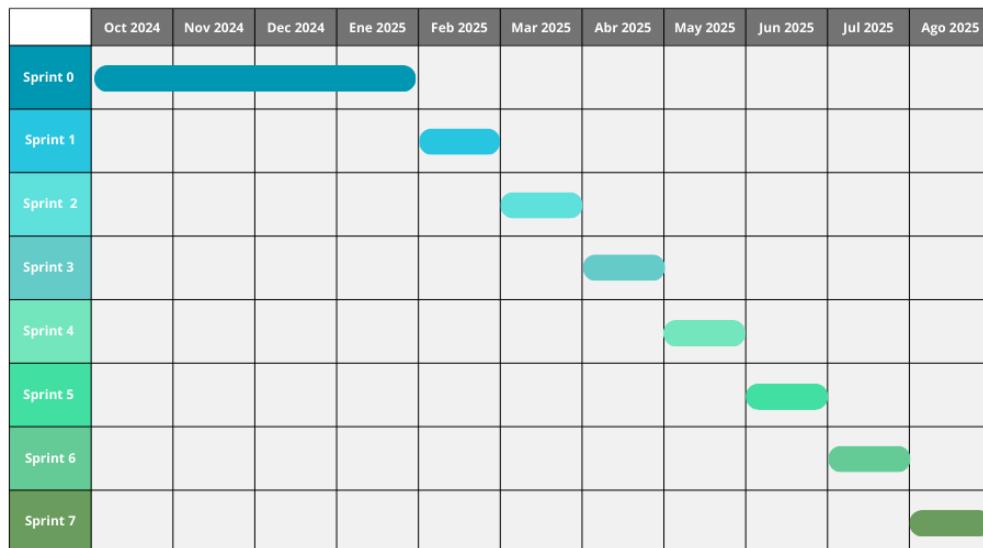


Figura 4.1: Diagrama de Gantt. Cronología de los sprints.

En conjunto, esta planificación permitió avanzar de forma progresiva y controlada, adaptándose a las circunstancias (como la necesidad de compaginar con las prácticas) sin perder de vista los objetivos esenciales del TFG.

4.2. Análisis de costes

Aunque este proyecto se desarrolla con fines académicos, elaborar un presupuesto aporta una visión realista del trabajo realizado. Este análisis da contexto técnico y económico al proyecto, acercándolo al mundo real, además de proporcionar una estimación útil para futuras implementaciones reales o propuestas a entidades como la Universidad de Granada. También ayuda a justificar decisiones técnicas en función de su viabilidad económica.

Para calcularlo, vamos a abarcar desde los recursos humanos hasta el equipamiento técnico y las herramientas software que se han utilizado. El análisis se basa en las tarifas medias del sector en España para perfiles de programación, así como en los recursos técnicos necesarios. La información sobre los salarios se ha obtenido de la plataforma **Glassdoor** <https://www.glassdoor.es/Sueldos/index.html>. Para los cálculos, se estima una duración total del proyecto de 4 meses.

A continuación, se desglosan las partidas principales.

4.2.1. Recursos humanos

Para estimar el coste del desarrollo de la plataforma en un contexto real (fuera del ámbito académico), se contempla un equipo formado por dos desarrolladores: un perfil **senior**, encargado de la arquitectura, planificación y supervisión, y un perfil **junior**, que colaboraría principalmente en el desarrollo del frontend, backend y pruebas.

A diferencia del cálculo estrictamente basado en salario bruto, en el sector se emplean **tarifas de facturación por hora** que incluyen costes indirectos (seguros sociales, vacaciones, licencias de software, equipamiento, gastos generales de oficina, etc.). Considerando valores habituales en el mercado español:

- **Programador Junior:** 25 €/hora.
- **Programador Senior:** 45 €/hora (rango 40–50 €/h, se adopta un valor intermedio).

Se mantiene la hipótesis de una jornada completa de 8 horas al día, 21 días laborables al mes, lo que equivale a **168 horas/mes por persona**.

Rol	Precio/hora (€)	Horas/mes	Coste mensual (€)
Programador Senior	45,00	168	7.560,00
Programador Junior	25,00	168	4.200,00
Total			11.760,00

Cuadro 4.1: Coste mensual estimado con tarifas de mercado.

El desarrollo se estima en **4 meses**, por lo que el coste total en recursos humanos sería:

$$11.760,00 \text{ €} \times 4 = \mathbf{47.040,00 \text{ €}}$$

4.2.2. Licencias Software y Herramientas

Para el desarrollo del proyecto se utilizarán diversas herramientas y tecnologías. A continuación, se presenta una tabla con el desglose de costes asociados a las licencias y herramientas necesarias:

Software/Herramienta	Licencia/Servicio	Coste	Coste (€)
React	MIT	Gratis	0,00
Visual Studio Code	MIT	Gratis	0,00
Flask	BSD	Gratis	0,00
MongoDB	SSPL	Gratis (Community Edition)	0,00
MongoDB Atlas	Servicio	0,00 €/mes (Free Tier)*	0,00
Tailwind CSS	MIT	Gratis	0,00
GitHub	Servicio	0,00 €/mes (Free Tier)	0,00
Postman	Freeware/Servicio	0,00 €/mes (versión gratuita)	0,00
Cloudinary	Servicio	0,00 €/mes (Free Tier)	0,00
Total			0,00

Cuadro 4.2: Coste de licencias de software y herramientas utilizadas

Cabe destacar que la mayoría de las herramientas utilizadas son de código abierto o disponen de versiones gratuitas adecuadas para el desarrollo del proyecto.

4.2.3. Equipamiento informático

Se estiman los siguientes recursos por persona:

- Ordenador portátil o de sobremesa: 1.000 €
- Monitor externo Full HD: 100 €
- Periféricos (teclado, ratón, webcam): 50 €

Concepto	Coste (€)
Equipamiento por persona	1.150
Total para 2 personas	2.300

Cuadro 4.3: Coste de equipamiento informático (gasto único)

4.2.4. Infraestructura

En un escenario futuro en el que se decidiese desplegar la plataforma, se ha planteado el uso de herramientas económicas o gratuitas que faciliten la puesta en producción, como es el caso de *Heroku*. Para realizar la estimación de costes se ha consultado el sitio web oficial de Heroku y, considerando el alcance de la aplicación desarrollada, se ha supuesto la elección del plan *Standard* [37].

Concepto	Coste mensual (€)
Heroku (backend)	23,00
Dominio web	1,00
Total	24,00

Cuadro 4.4: Costes mensuales de infraestructura (supuesto de despliegue)

$$24 \text{ €} \times 4 = \mathbf{96 \text{ €}}$$

4.2.5. Equipamiento Informático

Concepto	Coste (€)
Equipamiento informático	2.300,00
Total	2.300,00

Cuadro 4.5: Coste de equipamiento informático

4.2.6. Resumen de costes

Concepto	Coste (€)
Mano de obra	11.760,00
Equipamiento informático	2.300,00
Infraestructura	24,00
Total primer mes	14.084,00

Cuadro 4.6: Resumen de costes del primer mes (tarifas de mercado).

Concepto	Coste (3 meses) (€)
Mano de obra	35.280,00
Infraestructura	72,00
Total	35.352,00

Cuadro 4.7: Resumen de meses 2, 3 y 4 (tarifas de mercado).

Concepto	Coste total (€)
Mano de obra (4 meses)	47.040,00
Equipamiento informático	2.300,00
Infraestructura (4 meses)	96,00
Total estimado	49.436,00

Cuadro 4.8: Coste total estimado del proyecto (tarifas de mercado).

Capítulo 5

Análisis del problema

Una vez se ha introducido el problema en el capítulo [2] y revisado el estado del arte en el capítulo [3], este capítulo se centra en analizar el problema que aborda la plataforma web desarrollada para estudiantes de la Universidad de Granada. Se expone cómo se han identificado las necesidades reales de los usuarios, se definen los distintos perfiles que interactúan con el sistema y se especifican los requisitos que guían su diseño. Además, se modelan los principales casos de uso con el objetivo de representar el comportamiento que se espera del sistema y las interacciones entre los usuarios y el software.

5.1. Análisis de requisitos

Los requisitos de un sistema son las descripciones de lo que dicho sistema debe hacer: los servicios que proporciona y las restricciones bajo las cuales opera. Estos requisitos reflejan las necesidades de los usuarios y clientes para un sistema que cumpla un propósito concreto, como puede ser controlar un dispositivo, gestionar pedidos o recuperar información. El proceso de descubrir, analizar, documentar y validar estos servicios y restricciones se conoce como *ingeniería de requisitos* (*requirements engineering*),

Según Sommerville, los requisitos se clasifican usualmente como *requisitos funcionales* o *requisitos no funcionales*. Los *requisitos funcionales* describen los servicios que el sistema debe proporcionar, cómo debe reaccionar ante determinadas entradas y cómo debe comportarse en situaciones específicas. Por otro lado, los *requisitos no funcionales* imponen restricciones sobre los servicios o funciones ofrecidas por el sistema. Estas restricciones pueden estar relacionadas con el rendimiento, la seguridad, la usabilidad o con normas y estándares específicos que debe cumplir el sistema.

En la práctica, la distinción entre ambos no siempre se nota fácilmente. Por ejemplo, un requisito de seguridad como limitar el acceso a usuarios autorizados puede considerarse no funcional, pero puede derivar en requisitos funcionales concretos, como la necesidad de implementar autenticación de usuarios. Esto demuestra que los requisitos

no son independientes entre sí: uno puede generar o restringir a otros. Por tanto, los requisitos del sistema no solo especifican qué debe hacer el sistema, sino también la funcionalidad necesaria para asegurar que esas características se entreguen correctamente [48].

Para la recopilación de requisitos, se han seguido una serie de pasos basados en técnicas de ingeniería de requisitos. Estos pasos han permitido identificar las necesidades reales de los distintos usuarios implicados y traducirlas en especificaciones que guiarán al diseño y desarrollo del sistema.

- **Identificación de partes interesadas:** Se determinaron los perfiles clave involucrados en el proceso de movilidad Erasmus+, incluyendo a estudiantes y tutores docentes.
- **Entrevistas:** Se llevaron a cabo conversaciones con los perfiles clave para conocer de primera mano con qué dificultades se encontraban y las necesidades no cubiertas por las herramientas actuales.
- **Análisis de documentación:** Se revisaron documentos institucionales y normativas oficiales, como el Manuel de Acuerdo de Estudios de la ETSIIT o el Reglamento de Movilidad Internacional de la Universidad de Granada. También se consultaron herramientas actuales para estudiar sus funcionalidades y limitaciones.

Este enfoque centrado en el usuario permite asegurar que la plataforma propuesta responde a necesidades reales, evitando desarrollar funcionalidades innecesarias y priorizando aquellas que aportan más valor a los usuarios.

5.1.1. Requisitos funcionales

Los requisitos funcionales de un sistema describen lo que el sistema debería hacer. En principio, la especificación de estos requisitos debería ser completa y consistente. Por completa, entendemos que todos los servicios requeridos por el usuario deberían estar definidos, mientras que por consistente, entendemos que los requisitos no deberían tener definiciones contradictorias [48].

Después de recolectar y redactarlos, los requisitos funcionales se han clasificado de la siguiente forma:

- Autenticación y gestión de usuarios.
- Exploración de destinos.
- Gestión del Acuerdo de Estudios.
- Gestión de equivalencias.

- Interfaz y Dashboard.
- Funcionalidades Avanzadas y Escalabilidad

RF-1: Autenticación y gestión de usuarios

- RF-1.1: El sistema permitirá el registro e inicio de sesión de estudiantes, tutores y administradores.
- RF-1.2: El inicio de sesión podrá hacerse con credenciales propias.
- RF-1.3: El sistema gestionará perfiles personalizados para cada tipo de usuario (estudiante, tutor, administrador).
- RF-1.4: Los usuarios podrán editar la información de su perfil académico/personal.
- RF-1.5: Los administradores podrán gestionar (crear, editar, eliminar) usuarios.

RF-2: Exploración de destinos

- RF-2.1: El sistema mostrará un catálogo de universidades en vista de lista.
- RF-2.2: Los estudiantes podrán aplicar filtros (idioma, plazas, país, curso académico).
- RF-2.3: Se permitirá búsqueda avanzada por asignaturas ofertadas en el destino.
- RF-2.4: Cada destino tendrá una ficha de detalle con descripción, plazas, requisitos e información académica.

RF-3: Gestión del Acuerdo de Estudios

- RF-3.1: El estudiante podrá crear, guardar y editar su acuerdo de estudios.
- RF-3.2: El acuerdo se llenará con datos personales y de movilidad desde la base de datos.
- RF-3.3: Se permitirá seleccionar asignaturas para convalidación (1↔1 o combinadas 1↔2).
- RF-3.4: El acuerdo podrá enviarse al tutor para revisión.
- RF-3.5: El sistema gestionará el flujo de estados: borrador → enviado → revisado → aprobado/rechazado.
- RF-3.6: El acuerdo podrá exportarse en formato PDF.

RF-4: Gestión de equivalencias

- RF-4.1: El sistema permitirá almacenar relaciones de convalidación entre asignaturas UGR y de destino.
- RF-4.2: Se soportará la creación de bloques flexibles (ej. (6 ECTS UGR↔ 5+2 ECTS en destino)).

- RF-4.3: Los tutores podrán consultar, modificar y validar equivalencias propuestas por estudiantes.

RF-5: Interfaz y Dashboard

- RF-5.1: Cada tipo de usuario dispondrá de un dashboard con widgets adaptados a su rol.
 - Estudiante: estado del proceso, notificaciones, accesos rápidos.
 - Tutor: acuerdos pendientes, lista de estudiantes y destinos gestionados.
 - Administrador: accesos a gestión de colecciones (usuarios, destinos, grados, asignaturas).
- RF-5.2: Se mostrará el estado actual del proceso Erasmus de forma clara en el dashboard del estudiante.
- RF-5.3: El sistema notificará a los usuarios sobre cambios relevantes (acuerdo revisado, asignación de destino, etc.).

RF-6: Funcionalidades Avanzadas y Escalabilidad

- RF-6.1: El sistema será escalable para incorporar nuevos centros académicos y titulaciones.
- RF-6.2: Se permitirá la gestión de valoraciones y experiencias de estudiantes sobre los destinos (opcional).

5.1.2. Requisitos no funcionales

Los requisitos no funcionales, como su nombre indica, son los requisitos que no están directamente relacionados con los servicios específicos que el sistema proporciona a sus usuarios. Estos requisitos se centran en las propiedades del sistema, pueden especificar restricciones de implementación y su incumplimiento puede hacer que el sistema sea inutilizable.

Su implementación es compleja ya que pueden afectar a la arquitectura general del sistema, su implementación puede estar dispersa en varios componentes y pueden generar requisitos funcionales adicionales [48].

Para su clasificación, se va a seguir el modelo propuesto por [48] en el libro “Software Engineering, 9th Edition”, donde se categorizan los requisitos no funcionales en:

- Requisitos de Producto. Especifican o restringen el comportamiento del software.
- Requisitos Organizacionales. Derivados de políticas y procedimientos de la organización.
- Requisitos Externos. Derivados de factores externos al sistema.

Requisitos de Producto: Rendimiento, Usabilidad, Seguridad, Escalabilidad, Fiabilidad

Requisitos de Producto

- *Usabilidad*: RNF-1: La interfaz deberá ser intuitiva y clara para estudiantes, tutores y administradores, priorizando una curva de aprendizaje baja.
- *Rendimiento*: RNF-2: El sistema deberá responder en menos de 3 segundos en operaciones clave (búsqueda de destinos, carga de dashboard, visualización de acuerdos).
- *Seguridad*: RNF-3: Las credenciales de usuario se almacenarán cifradas y las sesiones estarán protegidas contra accesos no autorizados.
- *Mantenibilidad*: RNF-4: El código deberá seguir estándares de estilo (PEP8 en Python, ESLint/Prettier en React) y estructurarse modularmente para facilitar futuras mejoras.

Requisitos Organizacionales

- *Tecnología*:
 - RNF-5: El sistema usará tecnologías abiertas y mantenibles (MongoDB, Flask, React) para garantizar continuidad y escalabilidad.
 - RNF-6: Las imágenes de usuario deberán almacenarse en un servicio externo de almacenamiento de archivos en la nube (p. ej. Cloudinary), garantizando disponibilidad y escalabilidad.
- *Documentación*: RNF-7: Se proporcionará documentación técnica y manuales de usuario para facilitar el despliegue y uso de la plataforma.

Requisitos Externos

- *Legales*: RNF-8: El sistema cumplirá con el Reglamento General de Protección de Datos (RGPD) en el tratamiento de la información de los usuarios.
- *Compatibilidad*: RNF-9: La plataforma deberá ser accesible desde los principales navegadores web y dispositivos (ordenadores, tablets y móviles).

5.2. ¿Qué actores participan y qué hacen?

Para comprender quiénes interactúan con la plataforma y qué funciones desempeñan, se ha realizado un análisis detallado de actores, siguiendo las enseñanzas de la asignatura *Fundamentos de Ingeniería del Software*. Se han aplicado los principios y definiciones recogidos en el documento [47], donde se define a un **actor** como cualquier entidad externa que se comunica (interactúa) con el sistema para alcanzar un objetivo concreto.

Es importante señalar que, aunque en la mayoría de los casos un actor corresponde a un usuario humano —por ejemplo, un estudiante o un tutor—, también pueden considerarse actores a otros sistemas o plataformas que intercambian información con la aplicación, como servicios de autenticación o bases de datos externas.

En este contexto, un **actor** es, por tanto, cualquier persona o sistema que utiliza, accede o recibe información de la plataforma, contribuyendo de alguna manera al flujo de trabajo y al cumplimiento de los objetivos del sistema. El análisis de actores es fundamental porque permite identificar los distintos perfiles implicados, sus expectativas y las funcionalidades específicas que requieren para desempeñar sus tareas de forma eficaz.

Con el objetivo de obtener una visión precisa de estos actores y de los posibles usos de la plataforma, se han mantenido conversaciones con figuras clave, como la subdirectora de Relaciones Internacionales de la ETSIIT, y se ha consultado a compañeros que han participado en el programa Erasmus en años anteriores. Gracias a estas interacciones, ha sido posible identificar necesidades reales, problemas habituales y oportunidades de mejora en los trámites existentes, asegurando que el diseño de la plataforma se ajuste a la realidad y aporte verdadero valor a sus usuarios.

A partir de este análisis, se han identificado los siguientes actores principales, cada uno con sus correspondientes atributos y funciones específicas dentro del sistema:

Actores

Actor Estudiante	
Identificador	ACT-1
Descripción	Alumno de la UGR que solicita participar en un programa Erasmus. Interactúa con la plataforma para gestionar su perfil, explorar destinos y crear acuerdos de estudios.
Características	<ul style="list-style-type: none"> ■ Puede registrarse e iniciar sesión. ■ Personaliza su perfil académico y lingüístico. ■ Busca y consulta destinos disponibles. ■ Crea, edita y envía acuerdos de estudios. ■ Consulta el estado de sus solicitudes y notificaciones.
Relaciones	Tutor, Administrador del sistema
Referencias	Casos de uso CU-1, CU-2, CU-3, CU-4, CU-5, CU-6, CU-7, CU-10
Versión	1.0

Actor Tutor	
Identificador	ACT-2
Descripción	Profesor de la UGR encargado de revisar y validar los acuerdos de estudios de los estudiantes.
Características	<ul style="list-style-type: none"> ▪ Inicia sesión con perfil de tutor. ▪ Revisa acuerdos enviados por estudiantes. ▪ Aprueba, rechaza o solicita modificaciones. ▪ Se comunica con los estudiantes mediante comentarios en acuerdos.
Relaciones	
Referencias	
Versión	1.0

Actor Administrador del sistema	
Identificador	ACT-3
Descripción	Persona responsable del correcto funcionamiento de la plataforma. Tiene acceso completo para gestionar usuarios, universidades, grados, asignaturas y configuraciones. En el contexto del programa Erasmus, este rol correspondería normalmente al Coordinador Erasmus .
Características	<ul style="list-style-type: none"> ▪ Da de alta, modifica o elimina usuarios. ▪ Administra destinos, grados, titulaciones y asignaturas. ▪ Asigna destinos a estudiantes y tutores. ▪ Supervisa el sistema y corrige inconsistencias.
Relaciones	
Referencias	
Versión	1.0

5.3. Casos de uso

Los casos de uso son una herramienta esencial en el desarrollo de software, ya que permiten describir, de manera estructurada y comprensible, cómo interactúan los diferentes usuarios o sistemas externos con la aplicación. Cada caso de uso representa una unidad funcional coherente: es decir, una secuencia de acciones o procesos que el sistema lleva a cabo para satisfacer una necesidad específica de uno o varios actores[47].

El modelo de casos de uso no solo ayuda a visualizar las funcionalidades principales, sino que también sirve como base para el diseño, la validación y la posterior implementación del sistema. Además de los propios casos de uso, este modelo está formado por los siguientes elementos clave:

- **Actores:** Representan los distintos roles que interactúan con el sistema, ya sean usuarios humanos (como estudiantes, tutores o administradores) o sistemas externos que intercambian información con la plataforma. Cada actor tiene un objetivo o necesidad concreta que justifica su interacción. En este proyecto, los actores principales se han identificado y descrito previamente en la subsección 5.2.
- **Relaciones:** Las relaciones vinculan a los actores con los distintos casos de uso mediante diagramas específicos. Estos diagramas ayudan a visualizar de un vistazo quién puede realizar cada acción dentro del sistema y cómo se conectan las diferentes funcionalidades.

5.3.1. Descripción de los casos de uso

Con el fin de comprender mejor el funcionamiento, se han definido y documentado los principales casos de uso del sistema. Para ello, se ha utilizado una plantilla que incluye los elementos clave para su análisis, como los actores implicados, de qué tipo es el caso de uso, los requisitos funcionales incluidos, el propósito, las condiciones previas - condiciones sobre el estado del sistema que se deben cumplir para que se pueda llevar a cabo - y poscondiciones - resultados inmediatos tras la ejecución - relativas al caso, además de un resumen que describe de forma general la secuencia de acciones.

CU-1 Registrar usuario	
Actores	Estudiante, Tutor, Administrador
Tipo	Primario, básico y esencial
Referencias	RF1.1, RF1.3
Precondición	El usuario no está registrado en el sistema.
Poscondición	Usuario registrado con credenciales válidas y rol asignado.
Propósito	Permitir crear cuentas para acceder a la plataforma según rol.
Resumen	El sistema registra nuevos usuarios (estudiante/tutor/admin) almacenando datos mínimos y asignando rol.

CU-2 Iniciar sesión	
Actores	Estudiante, Tutor, Administrador
Tipo	Primario, esencial
Referencias	RF1.1, RF1.2
Precondición	Usuario registrado y credenciales disponibles.
Poscondición	Sesión iniciada, usuario redirigido a su <i>dashboard</i> .
Propósito	Autenticar de forma segura y determinar el rol.
Resumen	El usuario introduce credenciales; el sistema valida y abre sesión mostrando el panel según rol.

CU-3 Gestionar perfil	
Actores	Estudiante, Tutor, Administrador
Tipo	Primario, esencial
Referencias	RF1.4, RF1.5, RF5.1
Precondición	Sesión iniciada; permisos según rol.
Poscondición	Datos de perfil actualizados en BD conforme a permisos; cambios validados.
Propósito	Mantener la información de perfil con control de permisos por rol.
Resumen	El usuario accede a su perfil y edita datos conforme a su rol; el sistema valida y persiste los cambios.

CU-4 Explorar destinos (búsqueda y filtrado)	
Actores	Estudiante
Tipo	Primario, esencial
Referencias	RF2.1, RF2.2, RF2.3
Precondición	Sesión iniciada; catálogo de destinos disponible.
Poscondición	Lista de destinos de interés identificada por el estudiante.
Propósito	Facilitar la localización de destinos adecuados.
Resumen	El estudiante aplica filtros (idioma, plazas, país, curso) y consulta resultados en mapa/lista.

CU-5 Ver detalle de destino y experiencias	
Actores	Estudiante
Tipo	Primario, básico
Referencias	RF2.4, RF5.1
Precondición	CU-4 realizado; existe ficha del destino.
Poscondición	Información del destino analizada para la toma de decisión.
Propósito	Conocer requisitos, plazas, asignaturas, guías docentes y experiencias.
Resumen	El estudiante abre la ficha y revisa descripción, requisitos, asignaturas/equivalencias y valoraciones.

CU-6 Crear/editar acuerdo de estudios	
Actores	Estudiante
Tipo	Primario, esencial
Referencias	RF3.1, RF3.2, RF3.3, RF5.1
Precondición	Sesión iniciada; datos personales y de movilidad disponibles.
Poscondición	Acuerdo guardado en estado <i>borrador</i> .
Propósito	Construir el acuerdo con bloques de convalidación (1↔1 / 1↔2).
Resumen	El estudiante añade bloques, ajusta ECTS y guarda el acuerdo para posteriores envíos/revisiones.

CU-7 Enviar acuerdo a tutor	
Actores	Estudiante
Tipo	Primario, esencial
Referencias	RF3.4, RF3.5, RF5.3
Precondición	CU-6 con datos mínimos completos y sin inconsistencias bloqueantes.
Poscondición	Acuerdo en estado <i>enviado</i> ; tutor notificado.
Propósito	Someter el acuerdo a revisión formal por el tutor.
Resumen	El estudiante confirma el envío; el sistema cambia estado y genera notificación al tutor.

CU-8 Revisar/comentar/aprobar acuerdo	
Actores	Tutor
Tipo	Primario, esencial
Referencias	RF3.5, RF5.1, RF5.3
Precondición	CU-7 realizado; tutor asignado al estudiante/destino.
Poscondición	Acuerdo marcado como <i>revisado/aprobado/rechazado</i> ; estudiante notificado.
Propósito	Validar bloques y créditos, y emitir resolución/comentarios.
Resumen	El tutor analiza ECTS y equivalencias, añade comentarios y decide: pedir cambios, aprobar o rechazar.

CU-9 Gestionar equivalencias	
Actores	Tutor
Tipo	Secundario, de apoyo
Referencias	RF4.1, RF4.2, RF4.3
Precondición	Permisos de tutor; catálogo de asignaturas disponible.
Poscondición	Equivalencias creadas/actualizadas disponibles para consulta y acuerdos.
Propósito	Mantener relaciones UGR ↔ destino (incluye 1↔2 y complementos).
Resumen	El tutor revisa equivalencias nuevas propuestas por el estudiante; el sistema valida coherencia de ECTS y publica para uso.

CU-10 Exportar acuerdo a PDF	
Actores	Estudiante
Tipo	Secundario, útil
Referencias	RF3.6
Precondición	Acuerdo en estado <i>revisado o aprobado</i> .
Poscondición	Documento PDF generado y disponible para descarga/archivo.
Propósito	Obtener el documento oficial para trámites.
Resumen	El usuario solicita exportación; el sistema compone el PDF con bloques, datos personales y de movilidad.

CU-11 Asignar destinos a usuarios	
Actores	Administrador
Tipo	Primario, esencial
Referencias	RF5.1 (admin), RF3.2 (datos de movilidad)
Precondición	Sesión admin; catálogo y usuarios disponibles.
Poscondición	Destinos asignados a estudiantes y tutores; notificaciones emitidas.
Propósito	Gestionar la asignación de destinos según procedimientos internos.
Resumen	El admin selecciona destino y usuario; el sistema registra la asignación y notifica a los implicados.

CU-12 Gestionar colecciones (CRUD)	
Actores	Administrador
Tipo	Secundario, de soporte
Referencias	RF1.5, RF5.1 (admin)
Precondición	Sesión admin; acceso a panel de administración.
Poscondición	Colecciones de usuarios, destinos, grados, centros y asignaturas actualizados.
Propósito	Mantener datos maestros consistentes y actualizados.
Resumen	El admin crea/edita/elimina registros de las colecciones principales para garantizar información vigente.

5.3.2. Diagramas de casos de uso

Una vez presentados los casos de uso en tablas, es muy útil plasmar de forma visual cómo interactúan los distintos actores con la plataforma. Por ello, a continuación se muestran cuatro diagramas de casos de uso, cada uno representando el flujo principal de acciones y responsabilidades en un área funcional clave del sistema.

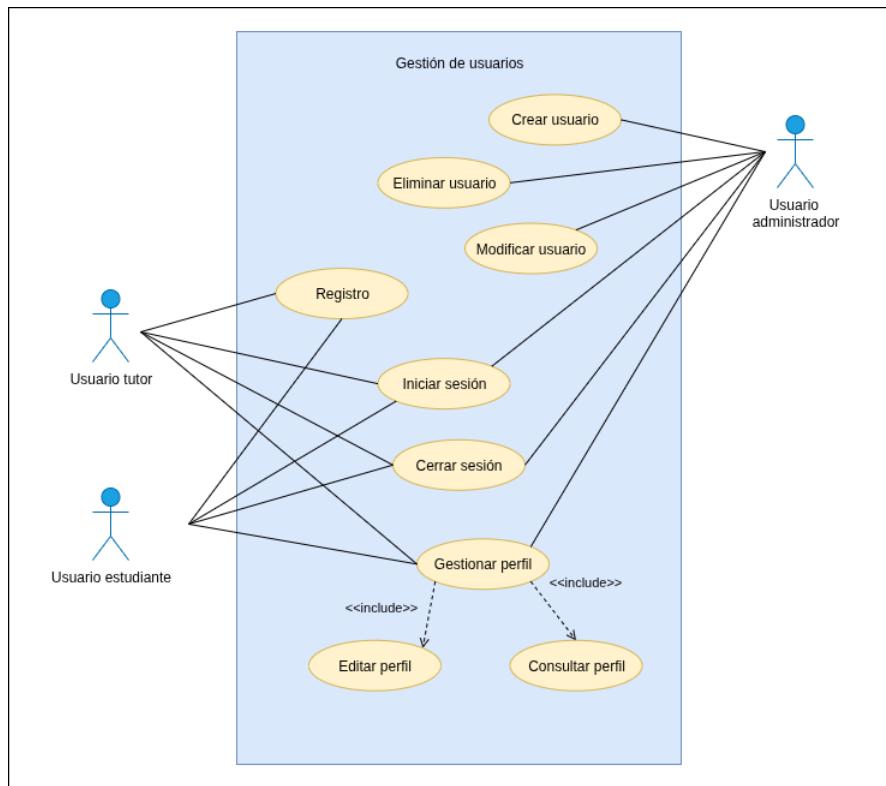


Figura 5.1: Diagrama de caso de uso de gestión de usuarios: registro, inicio/cierre de sesión y administración del perfil.

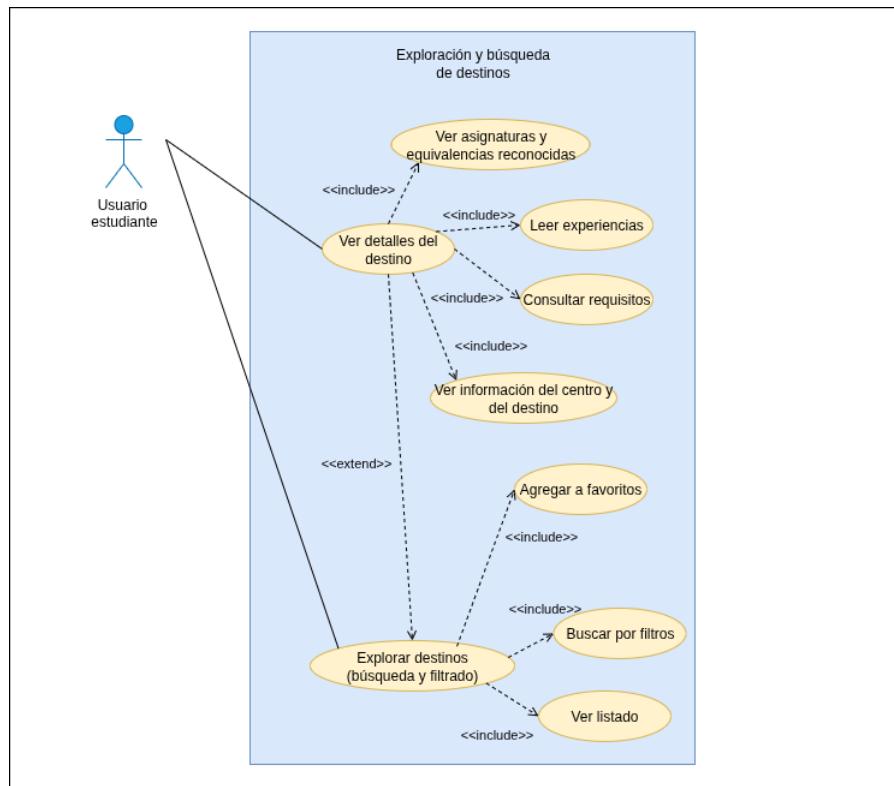


Figura 5.2: Diagrama de caso de uso de exploración y búsqueda de destinos.

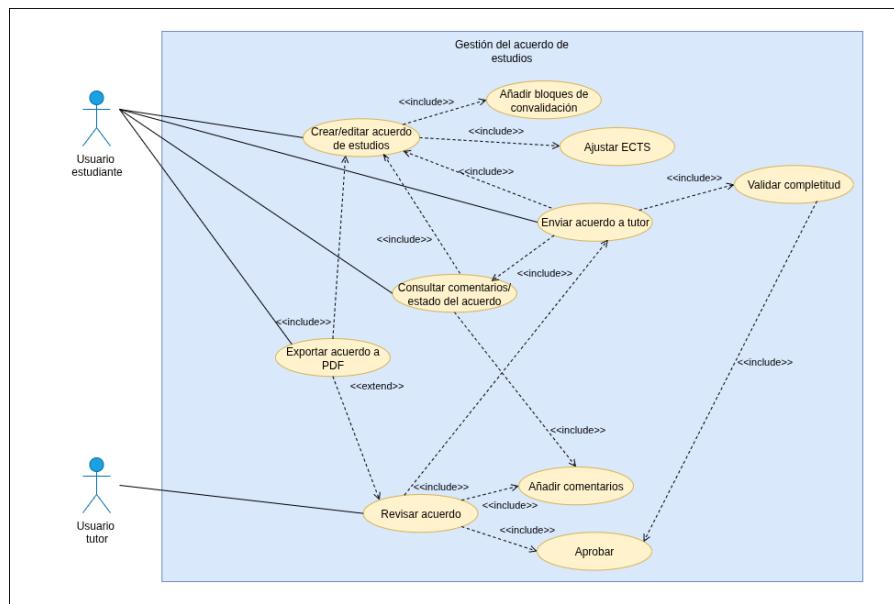


Figura 5.3: Diagrama de caso de uso de gestión del acuerdo de estudios.

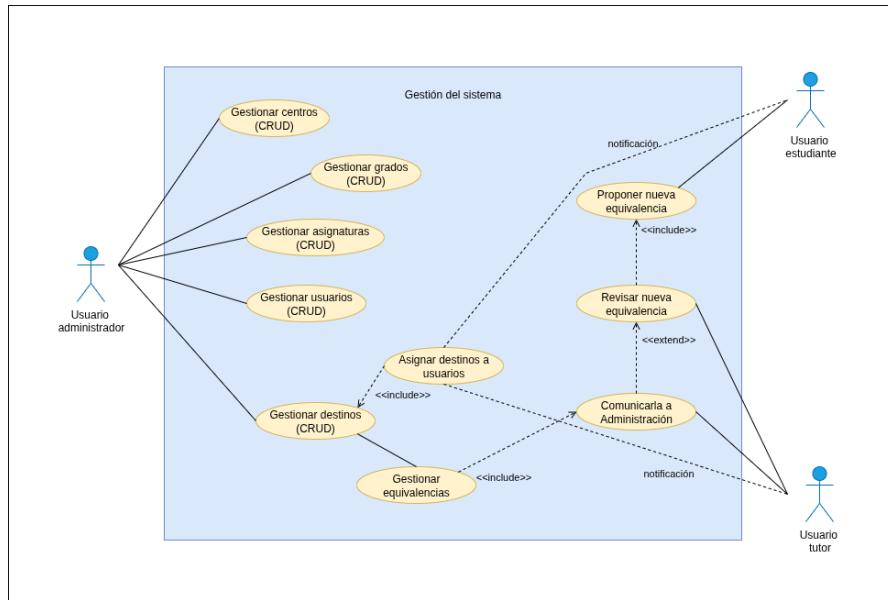


Figura 5.4: Diagrama de caso de uso de administración del sistema: gestión de colecciones (usuarios, grados, centros, asignaturas, destinos) y validación de equivalencias.

5.4. Diagramas de secuencia

Un diagrama de secuencia es un tipo de diagrama de interacción que describe cómo —y en qué orden— un grupo de objetos funciona conjuntamente para llevar a cabo un proceso. Tanto los desarrolladores de software como los profesionales de negocio emplean estos diagramas para comprender los requisitos de un sistema nuevo o para documentar procesos existentes. Por este motivo, a los diagramas de secuencia también se les denomina diagramas de eventos o escenarios de eventos[40].

En el contexto de este proyecto, los diagramas de secuencia resultan especialmente útiles para ilustrar cómo se produce la comunicación entre los distintos componentes de la plataforma y los actores implicados en operaciones concretas. Permiten visualizar, paso a paso y de forma cronológica, el intercambio de mensajes entre el usuario y el sistema, facilitando así la comprensión de procesos clave, la identificación de posibles puntos de fallo y la mejora de la lógica de interacción.

A continuación, se presentan los principales diagramas de secuencia correspondientes a los flujos más relevantes de la plataforma.

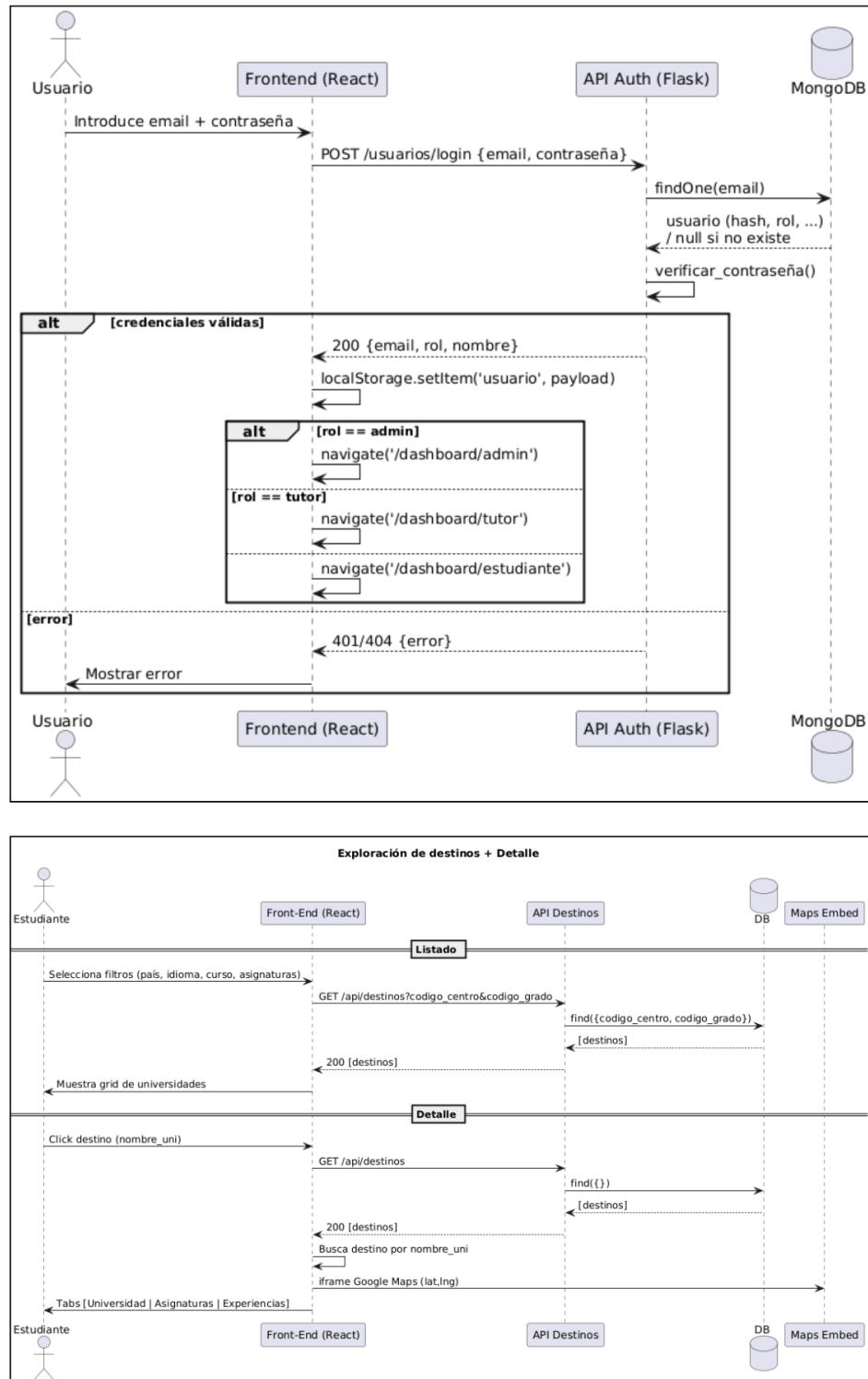


Figura 5.5: Diagrama de secuencia. Inicio de sesión y exploración de destinos.

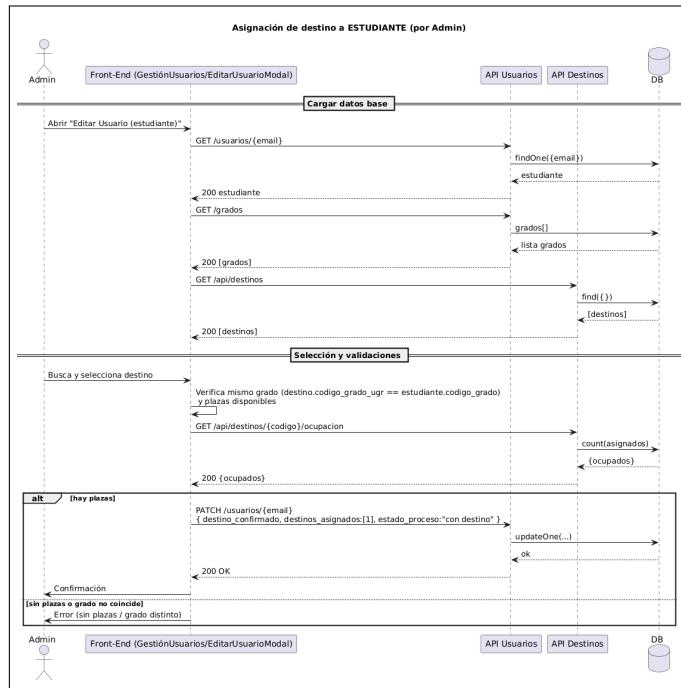


Figura 5.6: Diagrama de secuencia. Asignación de destinos a estudiante.

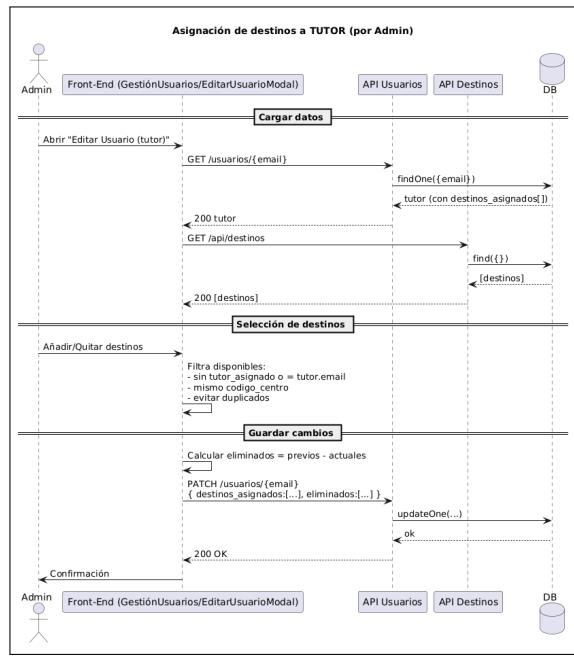


Figura 5.7: Diagrama de secuencia. Asignación de destinos a tutor.

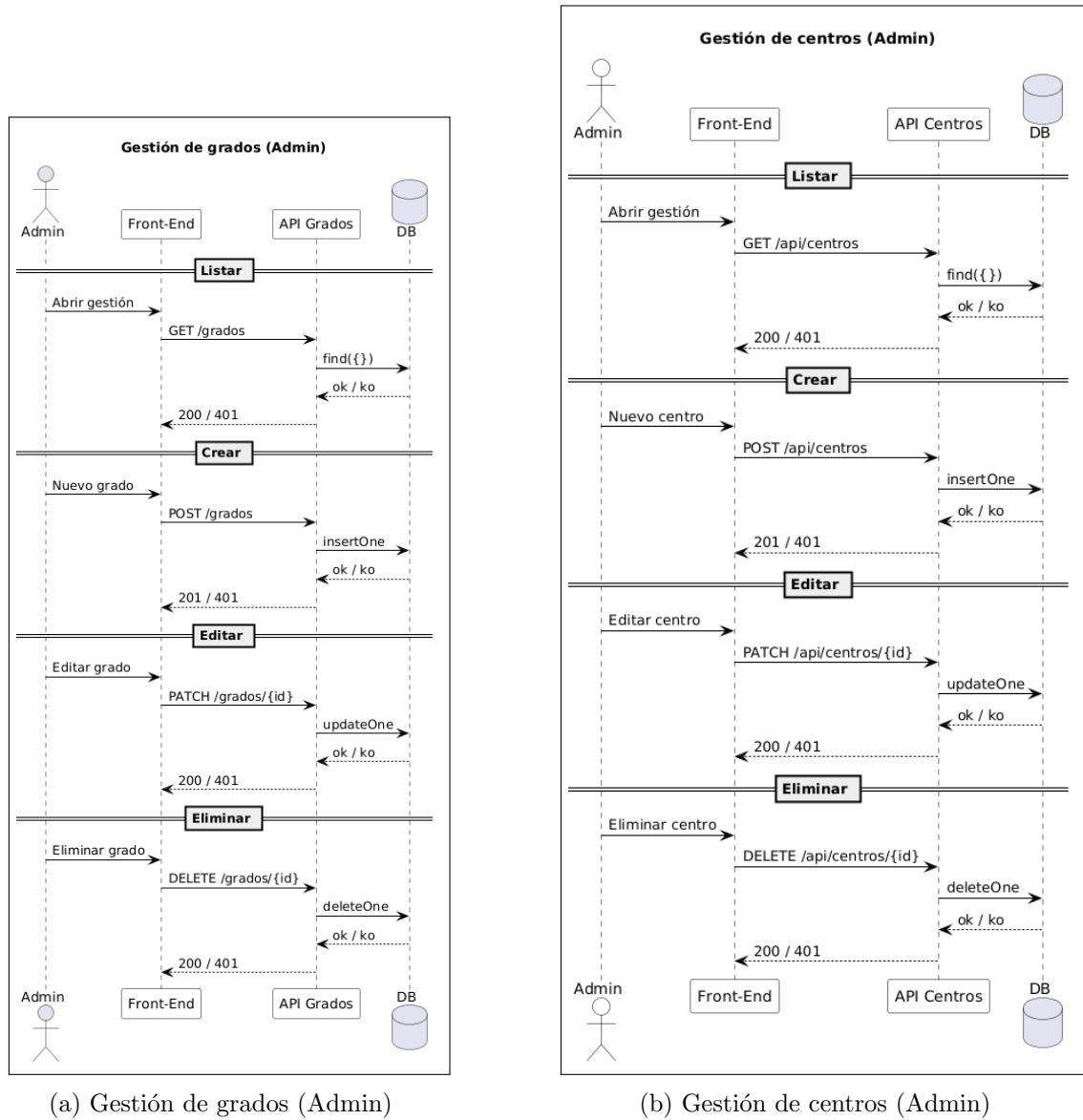


Figura 5.8: Diagramas de secuencia para la gestión de grados y centros en la plataforma.

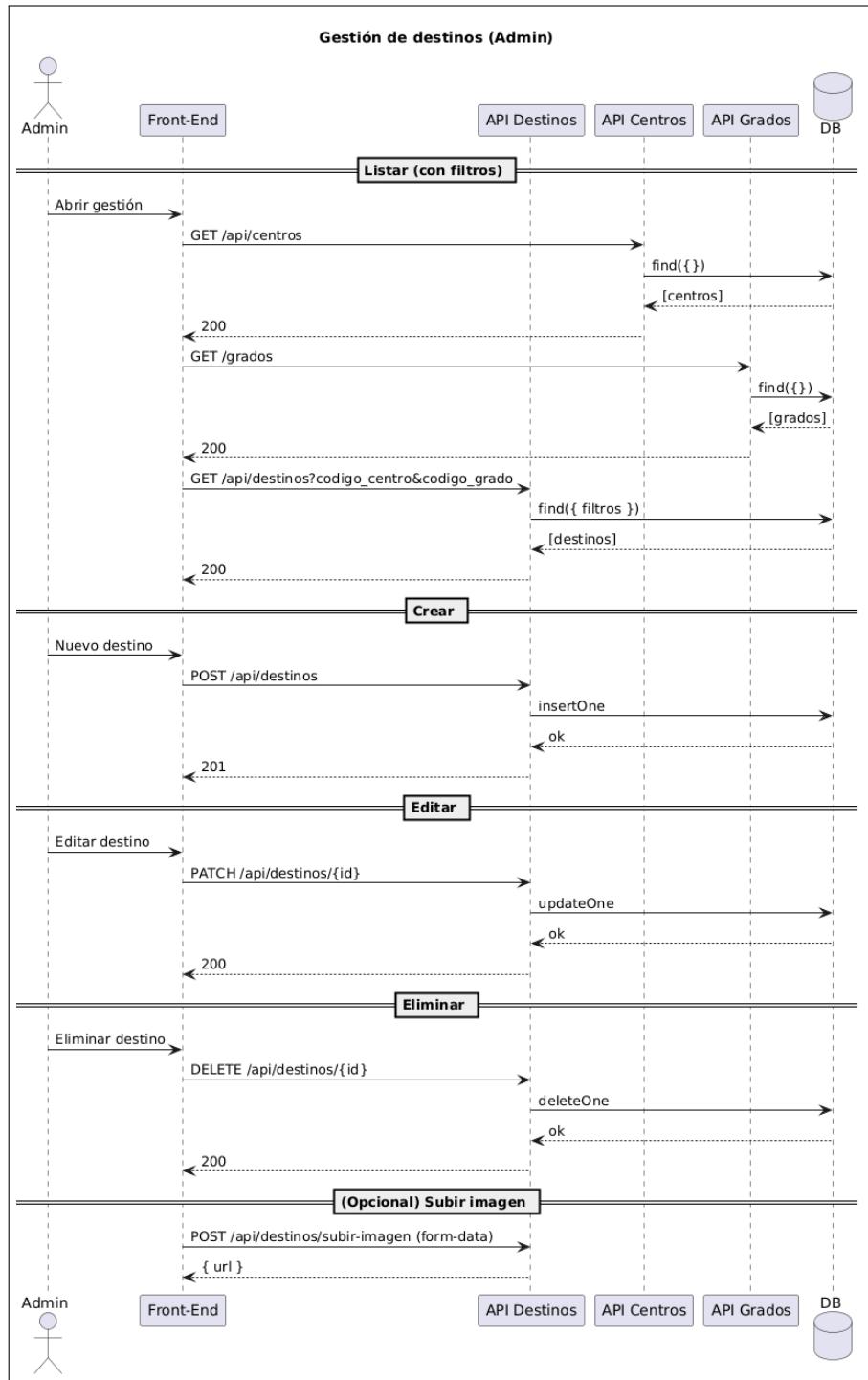


Figura 5.9: Diagramas de secuencia. Gestión de destinos .

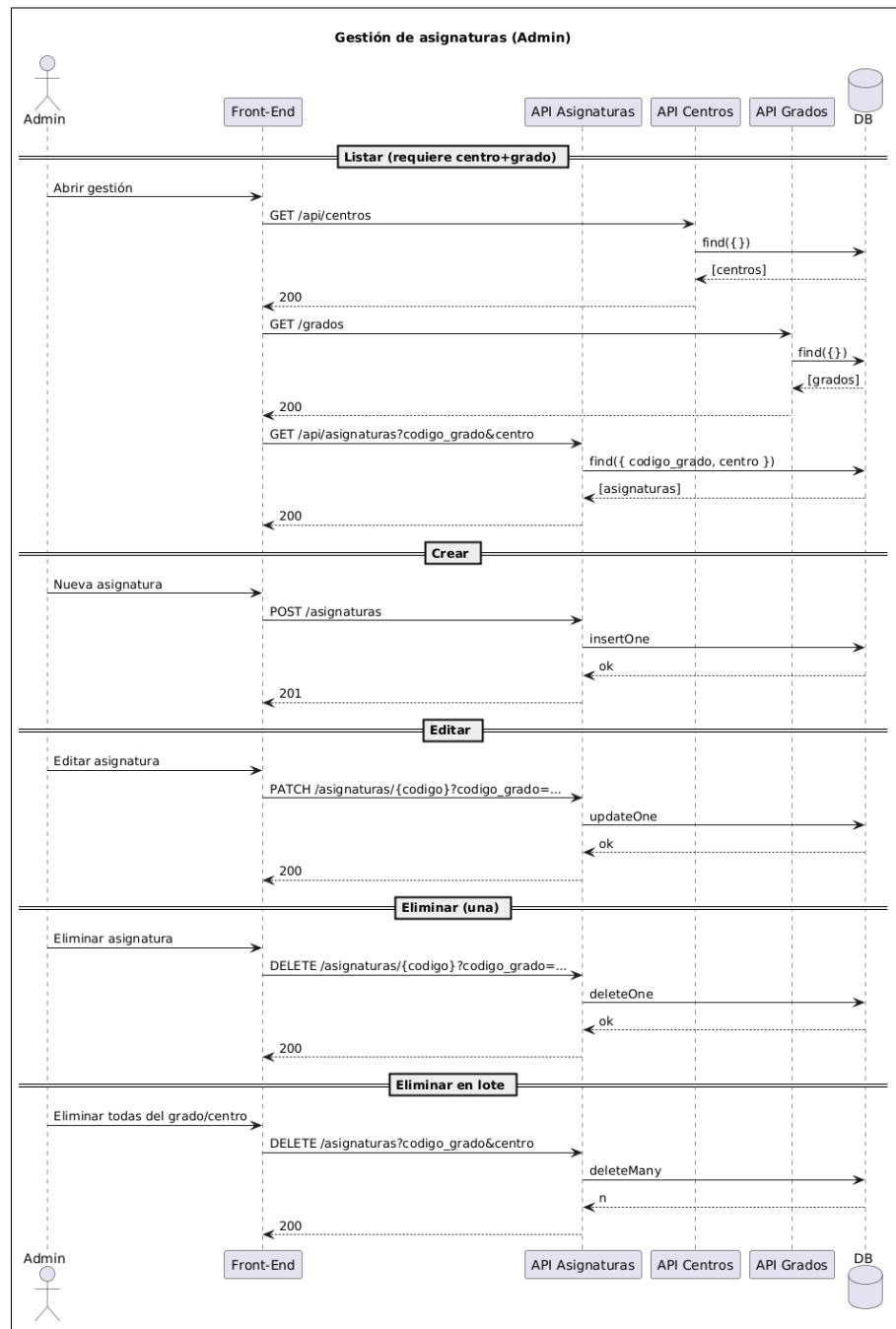


Figura 5.10: Diagramas de secuencia. Gestión de asignaturas.

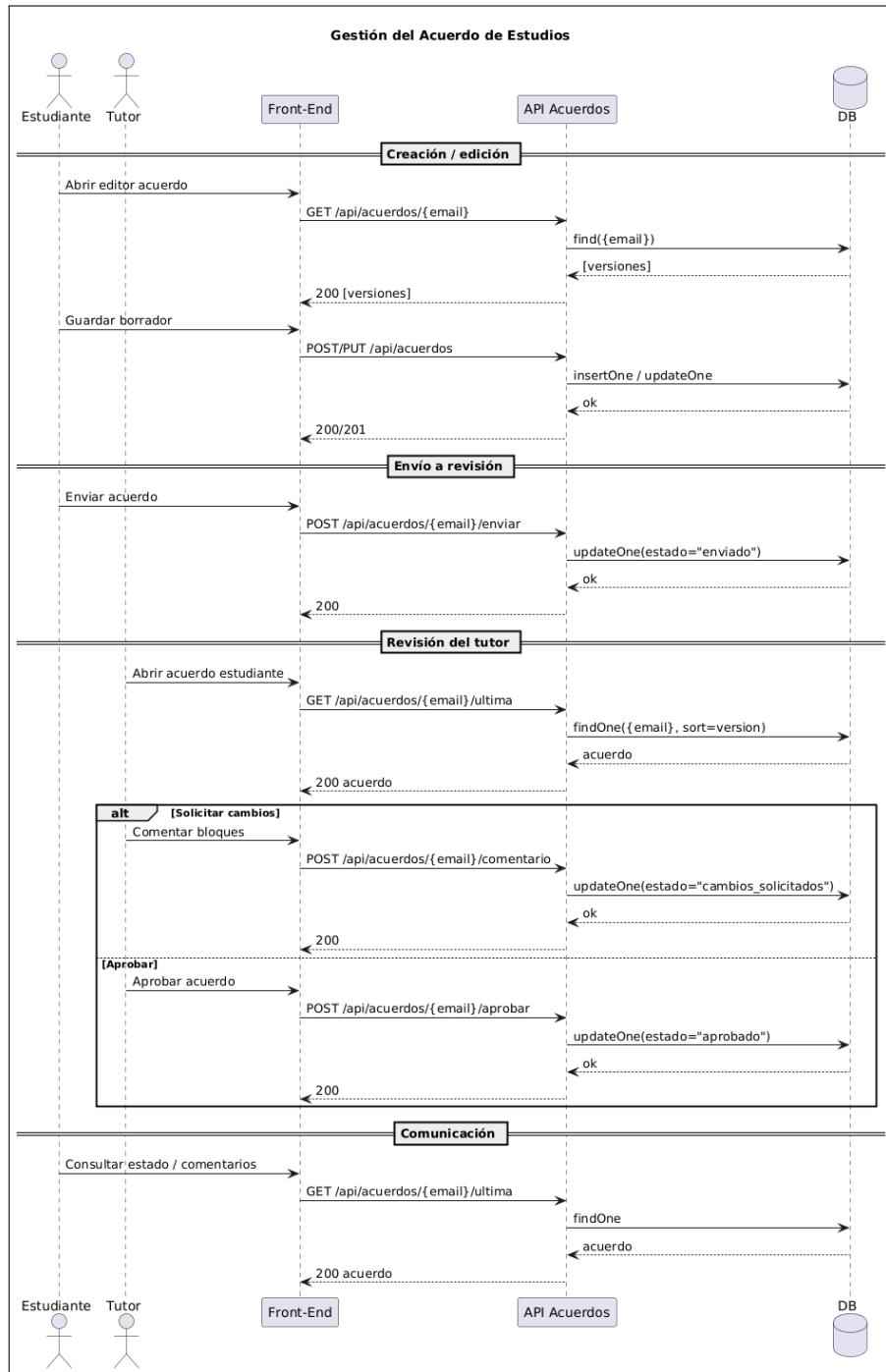


Figura 5.11: Diagrama de secuencia. Gestión del Acuerdo de Estudios.

Capítulo 6

Diseño de la plataforma

En este capítulo se detalla el diseño técnico de la plataforma propuesta. El objetivo principal es ofrecer una solución modular, fácilmente escalable—lo cual resulta esencial para poder añadir nuevos grados y centros en el futuro—y mantenible, capaz de adaptarse a las necesidades de los diferentes perfiles de usuario implicados en el proceso.

Para alcanzar estos objetivos, se ha optado por una arquitectura basada en el modelo cliente-servidor, estructurada en tres capas claramente diferenciadas. Esta organización facilita la separación de responsabilidades entre las distintas partes del sistema, lo que a su vez simplifica tanto el desarrollo inicial como el mantenimiento y la evolución futura de la plataforma. A lo largo de este capítulo se profundizará en cada una de estas capas y en su función dentro del sistema.

A continuación, se describe el modelo de datos y la estructura de la base de datos empleada, diseñada específicamente para ofrecer la flexibilidad que requiere la gestión de los acuerdos de estudios, el reconocimiento de asignaturas y la personalización según el perfil del usuario. La elección de una base de datos documental (MongoDB) responde precisamente a esta necesidad de flexibilidad, ya que permite modelar entidades con relaciones y estructuras anidadas, como las equivalencias entre asignaturas, los idiomas disponibles o el número de plazas por destino.

Por último, se presenta el diseño de la interfaz de usuario, donde se detalla la estructura de las pantallas principales y los criterios seguidos para garantizar una experiencia intuitiva, accesible y adaptada a los distintos roles que utilizan la plataforma. Se pretende así asegurar que cada usuario pueda interactuar con el sistema de forma sencilla y eficaz, independientemente de su perfil o de la tarea que desee realizar.

6.1. Diseño de la arquitectura

La arquitectura de la plataforma sigue el modelo cliente-servidor y está organizada en tres capas principales: presentación (frontend), lógica de negocio (backend) y persistencia

de datos (base de datos). Este enfoque, el cual puede verse en la figura [6.1], facilita la separación de responsabilidades, mejora la escalabilidad y simplifica el mantenimiento del sistema.

Antes de analizar en detalle cada una de estas capas y su interacción, revisaremos las decisiones que han ido guiando el diseño arquitectónico del sistema: desde la elección entre una estructura monolítica o basada en microservicios, hasta el tipo de aplicación web adoptado (SPA frente a aplicaciones tradicionales). Estas decisiones iniciales son clave para entender el porqué de la organización final y cómo cada componente encaja en el conjunto de la plataforma.

A continuación, se presenta un esquema general de la arquitectura y, en los siguientes apartados, se profundiza en cada decisión y en el funcionamiento de las distintas capas.

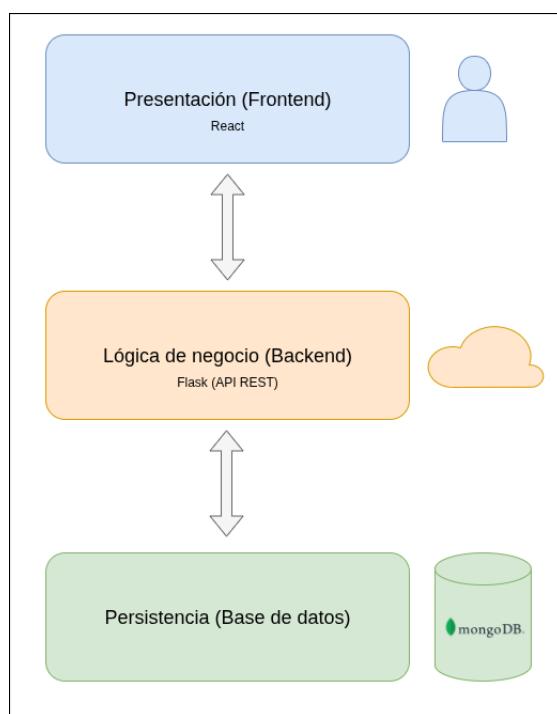


Figura 6.1: Diseño de la arquitectura en 3 niveles

6.1.1. Microservicios vs. Sistemas monolíticos: principales diferencias

En el diseño de la plataforma, una de las primeras decisiones importantes fue elegir entre dos modelos principales: la arquitectura monolítica y la basada en microservicios.

La **arquitectura monolítica** es el modelo más tradicional. Aquí, toda la aplicación—including la interfaz de usuario, la lógica de negocio y el acceso a los datos—se implementa como una unidad única de código que se despliega y ejecuta en un solo servidor o entorno. Esto facilita mucho el desarrollo inicial y la puesta en marcha de la

plataforma, ya que basta con desplegar una única aplicación, y cualquier cambio se realiza sobre el mismo código base. Sin embargo, a medida que el sistema crece, mantener y modificar una aplicación monolítica puede ser más costoso, ya que cualquier modificación puede afectar a múltiples partes de la aplicación, y las actualizaciones requieren desplegar todo el sistema de nuevo.

Por otro lado, la **arquitectura de microservicios** fragmenta la aplicación en pequeños servicios independientes, cada uno responsable de una única característica o función específica. Estos microservicios se comunican con una API y pueden estar escritos en lenguajes diferentes o ejecutarse en servidores distintos. Esto permite desplegar, actualizar y escalar cada componente por separado, lo que es ideal para aplicaciones de gran tamaño, con equipos de desarrollo grandes o que requieran máxima escalabilidad y flexibilidad [2].

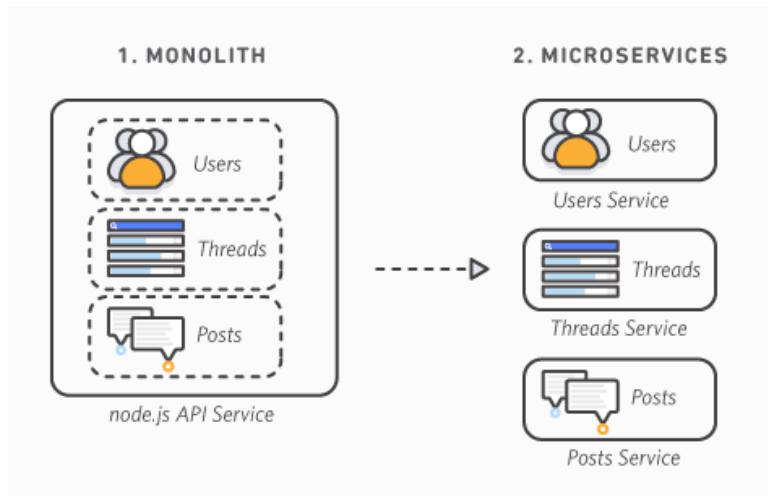


Figura 6.2: Diagrama de las arquitecturas. Fuente: <https://aws.amazon.com/es/compare/the-difference-between-monolithic-and-microservices-architecture/>

Ventajas y desventajas en el contexto de la plataforma ErasmusUGR

- **Monolítico:**

- *Ventajas:* desarrollo y despliegue mucho más sencillo por lo que la complejidad técnica es menor.
- *Desventajas:* escalabilidad más limitada y menor flexibilidad a largo plazo, además de un mayor riesgo de que los cambios afecten a todo el sistema.

- **Microservicios:**

- *Ventajas:* máxima escalabilidad, despliegue y mantenimiento independientes por cada componente, fácil integración con tecnologías modernas y servicios

cloud, el riesgo de fallo total es menor.

- *Desventajas:* requiere más planificación y experiencia en arquitectura distribuida. La complejidad en la implementación y el mantenimiento diario es mayor, además se necesitan una serie de herramientas y costes adicionales para la orquestación e infraestructura.

Arquitectura escogida

En el caso de esta plataforma, tras analizar el alcance y las necesidades presentes y futuras, se ha optado por una arquitectura monolítica modularizada internamente. Esta decisión corresponde principalmente a criterios de simplicidad, cohesión y eficiencia en el desarrollo. En el contexto de un proyecto que arranca con un número limitado de grados y centros, optar por un modelo monolítico ofrece varias ventajas. Por un lado, facilita el desarrollo y despliegue inicial, ya que toda la lógica del backend queda reunida en una sola aplicación, evitando tener que coordinar distintos servicios o configurar múltiples entornos o contenedores. Esto hace más fácil centrarse en construir la funcionalidad y reduce el tiempo y la complejidad técnica. Además, la cohesión funcional entre los módulos del sistema (gestión de usuarios, destinos, centros, etc.) es alta; por ejemplo, la asignación de un destino depende de la existencia de un usuario, que a su vez pertenece a un grado y un centro. En este escenario, separar en microservicios aportaría más complejidad que ventajas, al necesitar mecanismos de comunicación y sincronización adicionales que no son necesarios en este punto del proyecto.

¿Cómo se refleja la arquitectura en la plataforma?

A nivel técnico, la arquitectura monolítica se manifiesta en la plataforma Erasmus UGR de forma muy concreta. Toda la lógica de negocio está unificada en una sola aplicación desarrollada con Flask, que, aunque internamente se organiza de manera modular (utilizando Blueprints¹ para gestionar usuarios, destinos, acuerdos, etc.), se ejecuta siempre en el mismo proceso, servidor y dominio. Todas las colecciones de la base de datos—usuarios, grados, destinos, acuerdos, entre otras—se gestionan desde esta misma aplicación y comparten la conexión con una única instancia de MongoDB. La comunicación interna entre los distintos módulos del backend se realiza a través de funciones y llamadas directas dentro del propio código, sin necesidad de llamadas HTTP entre servicios ni de implementar colas de eventos. Por último, aunque el frontend (implementado en React) está desacoplado del backend, todas las peticiones y operaciones que realiza pasan por un único endpoint REST, lo que facilita la integración y el mantenimiento de la plataforma.

En resumen, el modelo monolítico es la opción más práctica y mantenible para la plataforma en su estado actual, ya que permite mantener el control completo sobre la

¹En Flask, un *Blueprint* es un componente que permite organizar la aplicación en módulos independientes y reutilizables. Cada blueprint puede definir rutas, vistas y lógica específica para una parte de la aplicación (por ejemplo, usuarios, destinos o acuerdos), facilitando así el mantenimiento y la escalabilidad del código.

lógica de negocio, hace más sencillo el despliegue y la comunicación interna, y favorece un desarrollo ágil y enfocado en validar el producto.

Si en el futuro la plataforma crece significativamente (por ejemplo, para dar soporte a muchas universidades o miles de usuarios activos de manera simultánea), se podría evolucionar gradualmente hacia una arquitectura de microservicios, aprovechando la modularidad que ya existe en la organización interna del código.

6.1.2. Modelo cliente-servidor

Una vez elegida la arquitectura interna del backend, es fundamental comprender cómo se comunican entre sí los distintos componentes de la plataforma. Aquí es donde interviene el modelo cliente-servidor, un estándar ampliamente utilizado en aplicaciones web modernas.

En este modelo, existen dos actores principales: el cliente, que en la mayoría de casos es el navegador web donde se ejecuta la aplicación web, y el servidor, que procesa las peticiones, accede a la base de datos y devuelve las respuestas. El ciclo básico es sencillo: el cliente inicia la comunicación enviando una petición (por ejemplo, para obtener información o realizar una acción), el servidor la procesa y devuelve una respuesta. Todo este intercambio sigue el protocolo HTTP y las reglas que establece para el envío y la recepción de datos entre ambas partes. El cliente y el servidor son elementos diferentes, por lo que pueden tener un ciclo de desarrollo diferente, así como usar tecnologías y un equipo diferente entre sí. Generalmente, son construidos como monolíticos, si bien no son autosuficientes ya que existe una dependencia mutua entre los dos.

Este enfoque ofrece una serie de ventajas. Por un lado, como se verá posteriormente, permite utilizar tecnologías diferentes en cada parte (por ejemplo, React para el frontend y Flask para el backend), lo que facilita la especialización y el desarrollo independiente de cada módulo. Además, esta arquitectura mejora la escalabilidad, ya que el servidor puede atender a múltiples clientes simultáneamente, y si es necesario, se pueden desplegar varias instancias del servidor para soportar un mayor volumen de usuarios. Otra ventaja es la modularidad, ya que las interfaces y funcionalidades pueden evolucionar de forma relativamente autónoma: es posible rediseñar la interfaz de usuario sin necesidad de cambiar la lógica de negocio, o viceversa.

Sin embargo, el modelo también presenta algunos desafíos. La comunicación depende de la red y el protocolo HTTP, por lo que pueden haber problemas de latencia o conectividad. Además, como se ha mencionado anteriormente, existe una dependencia mutua: el cliente necesita que el servidor esté disponible y siga respondiendo correctamente, mientras que el servidor requiere que el cliente use adecuadamente la API.

En la plataforma propuesta, el cliente está implementado mediante **React**, una librería declarativa para construir interfaces de usuario modernas. React permite desarrollar una **Single Page Application (SPA)**, en la que toda la navegación y las actualizaciones de la interfaz ocurren en el navegador del usuario, sin necesidad de re-

cargar la página completa. En la práctica, la aplicación se estructura en **componentes** (ficheros `.jsx`), cada uno responsable de una parte concreta de la interfaz—por ejemplo, el formulario de registro, el header de la landing page o la vista de destinos—. La gestión de la navegación interna se realiza con librerías como `react-router-dom`, definiendo en el archivo principal `App.jsx` las distintas rutas de la aplicación y los componentes que deben renderizarse en función de la URL, lo que permite cambiar de página sin tener que recargar toda la aplicación.

Cuando el usuario interactúa con la interfaz—por ejemplo, cuando consulta destinos Erasmus, envía un acuerdo de estudios o actualiza su perfil—, React realiza peticiones HTTP (como GET, POST, PUT o DELETE) al backend. Por su parte, el servidor está desarrollado en **Flask**, un framework de Python, y organizado en diferentes **Blueprints**². Cada Blueprint agrupa rutas y lógica asociada a una funcionalidad específica (usuarios, destinos, asignaturas, acuerdos, etc.), exponiendo los endpoints de la API REST que son consumidos por el frontend para obtener o modificar información.

Cada vez que el backend recibe una petición, la procesa (consultando o actualizando los datos en MongoDB), y responde con los datos necesarios (normalmente en formato JSON). Así, el frontend puede actualizar la interfaz según la respuesta que reciba o mostrar mensajes de error si algo no sale como se esperaba.

Para que esta comunicación sea posible, es necesario habilitar mecanismos como **CORS** (Cross-Origin Resource Sharing), que permiten que el navegador acepte respuestas de dominios diferentes. De esta manera, React puede hacer peticiones al backend Flask, incluso si cada uno está alojado en un servidor o puerto distinto, asegurando así una integración fluida y segura entre ambos [39].

En la figura [6.3] se puede ver la organización en capas y el flujo de comunicación de la plataforma. En este esquema, el cliente (SPA en React) interactúa con el backend (API REST en Flask) a través de peticiones HTTP que pasan por el middleware CORS, mientras que el servidor gestiona el acceso y la actualización de los datos en MongoDB. El diagrama sintetiza cómo se conectan y comunican las distintas partes del sistema.

²Un *Blueprint* en Flask es un módulo reutilizable que permite dividir la aplicación en componentes independientes, agrupando rutas y lógica relacionada bajo una misma estructura. Esto facilita la organización del código y el mantenimiento de aplicaciones grandes.

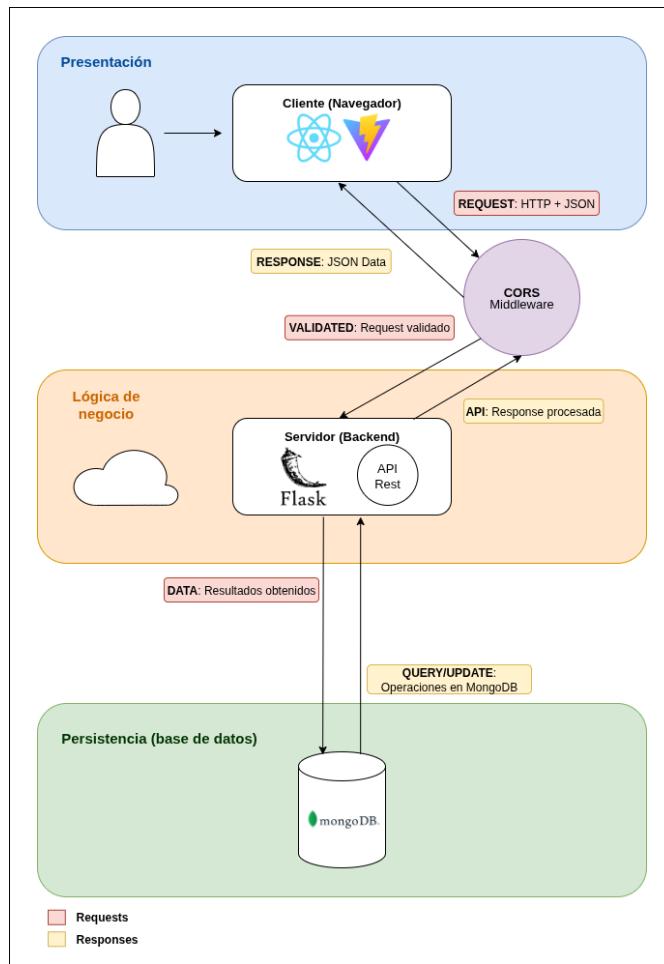


Figura 6.3: Esquema general de la arquitectura cliente-servidor de la plataforma Erasmus. Se muestra la interacción entre las distintas capas: presentación (SPA React + Vite), lógica de negocio (API REST con Flask y middleware CORS) y persistencia (base de datos MongoDB).

6.2. Diseño de la base de datos

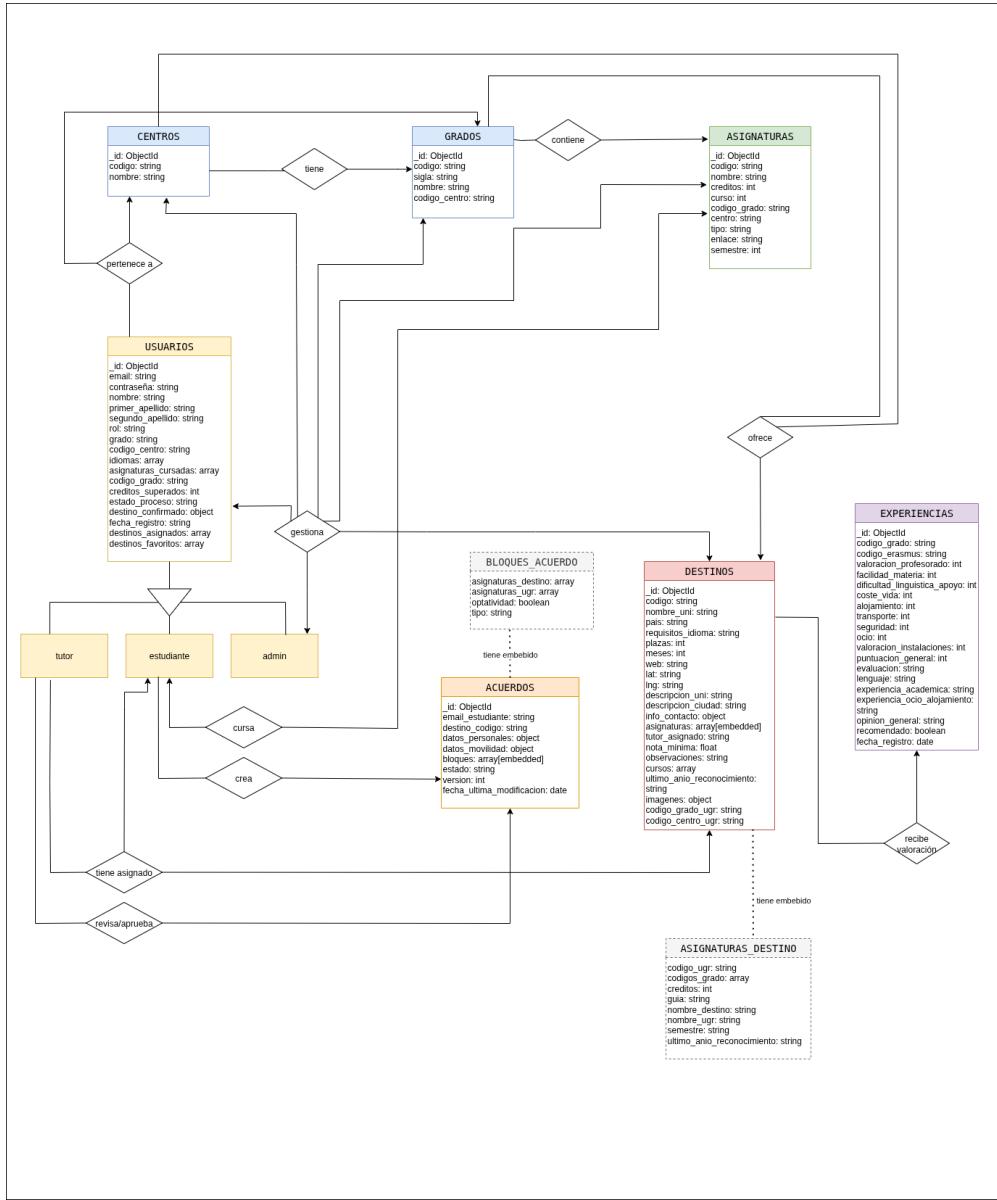


Figura 6.4: Entidades de la plataforma y sus relaciones

En el desarrollo de la plataforma, se ha optado por utilizar **MongoDB** como sistema de gestión de bases de datos. ¿Por qué MongoDB? Pues bien, a la hora de elegir una base de datos para una aplicación como esta, uno de los principales retos es la necesidad de flexibilidad. Los datos que se gestionan (usuarios, destinos, acuerdos, experiencias...) pueden diferir en estructura y cantidad, y es muy probable que la plataforma siga evo-

lucionando y requiera adaptaciones rápidas en el futuro.

MongoDB destaca precisamente por su capacidad de adaptación y su modelo flexible. Diferenciándose de los sistemas tradicionales, en los que el esquema y las relaciones deben quedar definidas con precisión antes de empezar, MongoDB permite trabajar con documentos que pueden crecer, cambiar o incluir nuevos campos según las necesidades de cada momento. Esto hace que desarrollar nuevas funcionalidades o hacer que la plataforma se adapte a los nuevos cambios en los requisitos, sea mucho más sencillo y ágil, sin tener que rediseñar de nuevo toda la estructura de la base de datos cada vez.

Por otro lado, este sistema de gestión de bases de datos, facilita el manejo de datos complejos y anidados. Por ejemplo, en esta plataforma, un acuerdo de estudios puede contener bloques con varias asignaturas de la universidad de origen y de la universidad de destino, reconocimientos, versiones... Todo esto se puede modelar de forma natural usando documentos y colecciones, sin tener que dividir la información en múltiples tablas ni recurrir a operaciones complicadas para reconstruir la información cuando se necesite.

Además, MongoDB está pensado para crecer con el proyecto. Si la plataforma recibe más usuarios o almacena cada vez más datos, puede escalarse horizontalmente sin mayores complicaciones, lo que la convierte en una opción robusta para aplicaciones que pueden evolucionar mucho a medio y largo plazo.

En palabras de IBM,

“Como solución de base de datos NoSQL, MongoDB no requiere un sistema de gestión de bases de datos relacionales (SGBDR), por lo que proporciona un modelo de almacenamiento de datos elástico que permite a los usuarios almacenar y consultar tipos de datos multivariados con facilidad. Esto no solo simplifica la gestión de bases de datos para los desarrolladores, sino que también crea un entorno altamente escalable para aplicaciones y servicios multiplataforma” [38].



Figura 6.5: Logo de *MongoDB*

Por todo ello, MongoDB es una herramienta tecnológica adecuada para la plataforma, ya que permite afrontar su desarrollo con garantías de flexibilidad, agilidad y futuro.

6.2.1. Modelado de la información

Una vez tomada la decisión de qué base de datos utilizar, el siguiente paso es pensar en cómo modelar la información para que la plataforma sea útil, eficiente y fácil de mantener.

Aprovechando las características previamente mencionadas de MongoDB, en vez de ajustarse a un esquema fijo y tradicional como en una base de datos tradicional, el enfoque ha sido utilizar la estructura orientada a documentos que ofrece MongoDB. Esto significa que cada colección representa una entidad principal de la plataforma (usuarios, acuerdos, destinos, grados, centros, experiencias, asignaturas, notificaciones...), y dentro de cada documento se almacena toda la información relevante para esa entidad, incluyendo, cuando es necesario, objetos anidados o listas con datos relacionados.

Esta forma de modelar los datos permite, por ejemplo:

- Que un usuario almacene en su propio documento su destino asignado o sus destinos favoritos, los idiomas que conoce o las asignaturas superadas, sin necesidad de hacer múltiples tablas o relaciones complejas.
- Que un acuerdo de estudios incluya todos los bloques de reconocimiento entre asignaturas UGR y de destino, versiones y estados del acuerdo, en un único documento autocontenido y fácil de consultar.
- Que las experiencias de estudiantes, las asignaturas disponibles en cada destino, o los mensajes del sistema, se gestionan de manera sencilla y natural.

Además, aunque MongoDB no cuenta con mecanismos de integridad referencial como los que existen en los sistemas relacionales, el diseño sí mantiene una lógica clara: las referencias entre documentos (por ejemplo, el código de un destino o el email de un usuario) se utilizan para poder cruzar información y mostrar todos los datos necesarios en la plataforma, pero siempre priorizando la simplicidad y el rendimiento.³

A continuación se describe cómo se han estructurado las principales colecciones y la lógica seguida en el diseño:

- **Usuarios:** La colección de **usuarios** reúne toda la información personal y académica relevante sobre cada persona que utiliza la plataforma, como nombre, apellidos, correo electrónico, centro, grado, idiomas o asignaturas superadas. Un aspecto fundamental es el campo **rol**, que determina el tipo de usuario y, sobre todo, **cómo será su experiencia y qué podrá hacer dentro del sistema**.

En función de si el usuario es **admin**, **tutor** o **estudiante**, la plataforma adapta completamente tanto su interfaz como los permisos y la forma en la que se utilizan los datos almacenados en cada documento. Por ejemplo, algunos campos sólo tienen sentido para ciertos roles: los estudiantes disponen de arrays de destinos asignados y asignaturas superadas, los tutores pueden tener un listado de estudiantes bajo su supervisión y los administradores cuentan con campos relacionados con

³La integridad referencial es un concepto en bases de datos relacionales que asegura que las relaciones entre tablas se mantengan consistentes y válidas. En esencia, garantiza que si una tabla hace referencia a otra (a través de una clave foránea), esa referencia siempre apunte a un registro válido en la tabla referenciada (con la clave primaria).

la gestión del sistema. Así, cada documento almacena únicamente la información relevante para el perfil correspondiente, y los mismos campos pueden tener un uso o significado diferente según el caso.

Esta diferenciación se refleja directamente en el **dashboard** que ve cada usuario al acceder:

- El **administrador** accede a paneles de gestión y herramientas exclusivas para administración, y no visualiza componentes destinados a la experiencia del estudiante o del tutor.
- El **tutor** ve los destinos y estudiantes que gestiona, con widgets y accesos adaptados para la supervisión y validación de acuerdos.
- El **estudiante** puede consultar destinos y experiencias, crear y editar acuerdos y gestionar su propia movilidad, accediendo únicamente a las secciones necesarias para su caso.

De este modo, cada usuario accede sólo a la información y funcionalidades que realmente necesita, haciendo la plataforma más segura, intuitiva y eficiente para todos. Además, la flexibilidad de MongoDB permite que todos los perfiles se almacenen juntos en una única colección y que cada documento⁴ se adapte dinámicamente a los campos requeridos por su rol, evitando esquemas rígidos y simplificando tanto el desarrollo como la evolución futura de la plataforma.+

- **Acuerdos:** La colección de **acuerdos** constituye es el centro del proceso académico de movilidad. Cada documento representa un acuerdo de estudios concreto entre un estudiante y un destino internacional. Para ello, el acuerdo almacena tanto los datos personales del estudiante y los datos relativos a la movilidad, como la información relevante del destino, enlazando ambos mediante los campos `email_estudiante` y `destino_codigo`. El elemento más distintivo es la inclusión de un array de bloques, donde se recogen todas las asignaturas reconocidas y su correspondencia entre la UGR y la universidad de destino. Cada bloque puede contener varias asignaturas de ambos lados, lo que permite modelar casos 1x1, 1xN o incluso propuestas personalizadas. Gracias a esta estructura embebida, es posible consultar, editar o exportar todo un acuerdo de estudios con una sola operación, y a su vez cruzar información con otras colecciones (como estudiantes, destinos o asignaturas) a través de los identificadores comunes.
- **Destinos:** En la colección de **destinos** se guarda toda la información de las ciudades y centros internacionales con los que existen acuerdos Erasmus. Cada destino incluye detalles como el nombre, país, requisitos de idioma o de nota mínima, grado y centro al que están asociados, número de plazas y, especialmente, un array de equivalencias reconocidas. Estas asignaturas no sólo llevan sus propios datos

⁴En MongoDB, un documento es la unidad básica de almacenamiento. Se trata de un objeto en formato BSON (similar a JSON), que agrupa los datos relacionados de manera estructurada y flexible, permitiendo distintos campos según las necesidades.

(nombre, créditos, guía docente...), sino que también referencian mediante códigos a los grados de la UGR para los que resultan equivalentes. Así, es posible filtrar fácilmente desde la plataforma qué asignaturas de un destino puede reconocer un estudiante en función de su grado. Además, la estructura de la colección permite relacionar rápidamente cada destino con los acuerdos de estudiantes y con las experiencias previas, lo que facilita la navegación y la consulta cruzada.

- **Asignaturas:** La colección de **asignaturas** recoge las materias que se ofrecen en los grados y centros de la UGR, en nuestro caso, las asignaturas de los grados de la E.T.S. de Ingenierías Informática y de Telecomunicación. Cada documento incluye información fundamental sobre la asignatura (nombre, créditos, curso, tipo, enlace a la guía docente, etc.), además de los códigos de grado y centro que permiten ubicarla de forma correcta en el contexto académico.

Esta colección cumple un papel esencial como base de referencia tanto para los acuerdos de estudios como para el expediente de cada estudiante. Por un lado, al crear un acuerdo de estudios o gestionar una convalidación, se recurre a esta colección para seleccionar y validar las asignaturas de la UGR que pueden ser reconocidas en cada movilidad. Por otro, las asignaturas superadas por cada estudiante se obtienen a partir de aquí, permitiendo mostrar su progreso académico, validar requisitos y automatizar comprobaciones.

Gracias a la relación directa de las asignaturas con grados y centros, la plataforma puede personalizar la información mostrada a cada usuario, asegurando que sólo vea las materias relevantes para su titulación y contexto, y facilitando tanto la gestión administrativa como la experiencia diaria del estudiante.

- **Grados y centros:** Las colecciones de **grados** y **centros** cumplen una función fundamental como tablas de referencia dentro de la plataforma. En ellas se almacena la información esencial sobre los distintos grados de la UGR y los centros participantes, como el nombre, la sigla y el código identificador. Estos códigos se utilizan a lo largo de toda la base de datos como “pegamento” para enlazar usuarios, acuerdos y destinos, permitiendo mantener la coherencia y la organización de la información sin duplicar datos innecesariamente.

Gracias a esta estructura, cada usuario puede acceder directamente a los recursos que le corresponden: por ejemplo, un estudiante de informática ve automáticamente los destinos y asignaturas asociados a su grado y centro, sin necesidad de filtrar entre opciones que no sean relevantes. Del mismo modo, la plataforma puede crear y mostrar únicamente las asignaturas reconocidas y los acuerdos válidos para cada titulación y centro, lo que mejora notablemente la experiencia de usuario, agiliza la búsqueda y reduce los posibles errores de selección.

En resumen, estas colecciones no sólo garantizan la integridad y el orden en el modelo de datos, sino que permiten personalizar la plataforma y hacer que sea útil y eficiente para cada perfil académico.

- **Experiencias:** La colección de **experiencias** recoge las valoraciones y opiniones de estudiantes que han participado en movilidades internacionales. Cada experiencia queda vinculada a un destino concreto (mediante `codigo_erasmus`) y a un grado (`codigo_grado`), de modo que al consultar un destino o un grado es posible mostrar las valoraciones medias y los testimonios reales asociados a cada combinación. Los campos numéricos permiten calcular automáticamente medias y tendencias, mientras que los textos aportan una visión cualitativa que resulta muy útil para estudiantes que estén interesados en escoger un destino u otro.

- **Notificaciones:** Finalmente, la colección de **notificaciones** permite registrar y mostrar eventos importantes del sistema (avisos de nuevos destinos, cambios de estado, mensajes administrativos, etc.), vinculando cada mensaje al usuario correspondiente a través de su email. De esta forma, cada participante recibe únicamente la información relevante para su perfil, mejorando la comunicación y la experiencia de uso dentro de la plataforma.

A pesar de que MongoDB no es una base de datos relacional en sentido clásico, el diseño de la plataforma asegura una coherencia lógica entre todas las colecciones gracias al uso de campos de referencia—como el email, el código de destino o el código de grado. Así, aunque no existan claves foráneas como tal, es muy sencillo cruzar información: por ejemplo, para mostrar todos los acuerdos de un estudiante, basta con buscar los documentos cuyo campo `email_estudiante` coincide con su correo; o para ver las asignaturas reconocidas en un destino concreto para un grado específico, simplemente se filtra por los correspondientes códigos.

El hecho de organizar la información en colecciones bien estructuradas y aprovechar la anidación de documentos y el uso de arrays facilita mucho la gestión y la consulta de los datos: en la mayoría de los casos, toda la información relevante se puede obtener de forma muy eficiente, en una sola operación. Esto permite que la plataforma sea ágil y responda rápidamente incluso cuando crece el volumen de datos.

Otra gran ventaja de este modelo es su escalabilidad. Si en el futuro se incorporan más centros, basta con añadirlos a la colección correspondiente; a partir de ahí, se pueden crear nuevos grados asociados a esos centros, y a su vez ir añadiendo las asignaturas propias de cada grado o los destinos internacionales que correspondan. De este modo, la estructura del sistema se puede ir ampliando de forma natural y ordenada, sin grandes complicaciones técnicas ni necesidad de rediseñar la base de datos, lo que asegura que la plataforma pueda evolucionar y adaptarse con facilidad a nuevas necesidades.

El diseño de la base de datos y las relaciones entre las entidades se pueden ver en la figura [6.4]

6.3. Diseño de la interfaz de usuario

6.3.1. Introducción. Intencionalidad del diseño

El diseño de la interfaz de la plataforma se ha ideado con el objetivo de proporcionar una experiencia clara, intuitiva y moderna a todos los perfiles implicados en el proceso Erasmus. Desde el principio, la prioridad ha sido crear un entorno accesible y sencillo de usar, que ayude a los estudiantes y tutores a navegar por las diferentes funcionalidades de forma sencilla y eficaz. En esta sección se abordará el proceso de diseño, desde la elección de estilos hasta la creación de prototipos y el flujo de los usuarios.

6.3.2. Identidad visual y personalización

El primer paso antes de hacer el prototipo, fue decidir los estilos que se usarían en la plataforma. Uno de los objetivos ha sido transmitir de forma clara la identidad de la Universidad de Granada (UGR) en la plataforma. Para conseguirlo, se ha utilizado una paleta de colores que evoca los tonos institucionales de la UGR, siguiendo el manual de identidad visual que proporciona la propia universidad [50]. Esta elección de colores, junto a otras tonalidades más llamativas y personalizadas, facilita la identificación y el atractivo visual.

Al mismo tiempo, se ha buscado dotar a la plataforma de una personalidad propia, de modo que se pueda diferenciar de otros portales académicos tradicionales. Siguiendo esta línea, la página de inicio (landing) se diseñó para ser más llamativa, integrando imágenes, composiciones y elementos decorativos, mientras que el resto de la plataforma mantiene un estilo más minimalista, intuitivo y funcional, priorizando la facilidad de uso y la claridad en la presentación de la información.

Para reforzar la asociación con la UGR y la ciudad de Granada, se han creado gráficos SVG personalizados, inspirados en motivos de azulejos granadinos. Estos recursos se han integrado en *banners*, fondos y distintos componentes, enriqueciendo visualmente la plataforma sin recargarla. Todas estas decisiones de diseño están pensadas para facilitar la identificación y la familiaridad de los usuarios, permitiendo que se sientan cómodos y ubicados en un entorno propio de su universidad, pero con un aire moderno y adaptado a sus necesidades.

Tipografía y estilos de los textos

Con el objetivo de lograr una interfaz legible y actual, se ha elegido la fuente **Inter** para los títulos y secciones principales. Se trata de una tipografía gratuita y de código abierto, diseñada para ofrecer buena legibilidad en pantalla, que se ha descargado desde <https://fonts.google.com/specimen/Inter>.

En la figura [6.6] se pueden ver algunos ejemplos de los svg, la tipografía Inter y la

paleta de colores que se ha seguido en la implementación de la aplicación.

Logo de la plataforma

Para completar la identidad visual, se ha diseñado un logo para la plataforma, manteniendo la coherencia cromática con la paleta institucional. Este logo combina dos elementos clave: una granada, símbolo de la ciudad de Granada, y la silueta de un avión, que alude directamente a la idea de viaje y movilidad que caracteriza al programa Erasmus. Este logo aparece en la barra superior y en la landing, así como en menús laterales o formularios, funcionando como elemento identificador y reforzando la marca visual propia de la plataforma.

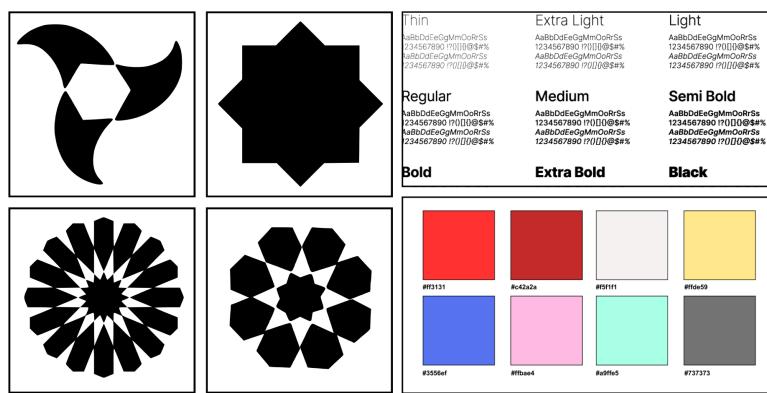


Figura 6.6: Gráficos SVG creados. Tipografía usada y paleta de colores de la plataforma



Figura 6.7: Logo de *ErasmusUGR*

6.3.3. Prototipado y herramientas

Antes de pasar al desarrollo de los componentes de la interfaz en código, fue fundamental definir cómo se organizarían visualmente las distintas secciones y componentes de la plataforma. Para ello, se optó por un proceso de prototipado ágil que permitiera experimentar rápidamente con el aspecto y disposición de los elementos clave.

En lugar de utilizar herramientas tradicionales de mockup o diseño de interfaz, se eligió **Canva** para esta fase inicial. Canva es una herramienta de diseño y publicación online, la cual ofrecía la flexibilidad de poder trabajar con elementos personalizados, permitiendo incorporar fácilmente los SVG propios diseñados para el proyecto, así como probar combinaciones de colores y estructuras en tiempo real. Esta herramienta facilitó la visualización de propuestas para la landing page, los menús laterales y los distintos formularios, acercando mucho el prototipo al aspecto que tendría el producto final.

Este enfoque ha permitido ajustar la composición, decidir la jerarquía de información y anticipar los posibles problemas de usabilidad antes de implementar los componentes en React.



Figura 6.8: Comparación entre el prototipado visual realizado en Canva y la versión final desarrollada de la sección de perfil.

6.3.4. Tecnologías empleadas

El principal framework utilizado para los estilos ha sido **Tailwind CSS** [52], una herramienta utility-first que permite aplicar clases directamente en los archivos de componentes (.jsx). Esto facilita la creación de interfaces coherentes, personalizables y, sobre todo, responsivas, ya que Tailwind integra un sistema de breakpoints y utilidades específicas para adaptar el diseño automáticamente a cualquier tamaño de pantalla. Así, la plataforma ofrece una experiencia óptima tanto en ordenadores como en dispositivos móviles, sin necesidad de escribir estilos CSS personalizados para cada caso.

Por otro lado, la interfaz se estructura en componentes React (ficheros .jsx), cada uno responsable de una parte concreta de la vista (dashboard, formularios, menús, etc.), lo que permite organizar el código de manera modular y reutilizable. Además, para la iconografía se ha utilizado la librería Lucide [5], que ofrece iconos escalables y modernos fácilmente integrables en los componentes de React. Gracias a esta combinación de React

y Tailwind, el desarrollo de la interfaz ha sido ágil, manteniendo la consistencia visual y la facilidad de mantenimiento.

6.3.5. Organización general, experiencia de usuario y diagramas de flujo

La plataforma se estructura en torno a un **dashboard central**, el cual se adapta según el rol del usuario, que actúa como punto de partida y eje principal de navegación para todos los perfiles. Desde este dashboard, cada usuario accede de manera intuitiva a las diferentes secciones y funcionalidades relevantes según su rol (estudiante, tutor o administrador). El dashboard no solo muestra información personalizada y accesos rápidos, sino que también guía el flujo de trabajo del usuario, facilitando el acceso a tareas habituales y notificaciones importantes.

Uno de los elementos clave de la interfaz es la **barra lateral de navegación**. Este menú lateral, que puede desplegarse o contraerse para lograr una experiencia más limpia y enfocada, permite el acceso directo a las principales secciones de la plataforma: perfil, destinos, acuerdos y, en el caso del administrador, áreas de gestión como usuarios o asignaturas. El contenido y los enlaces del menú se adaptan dinámicamente al rol del usuario, mostrando solo las opciones que le corresponden. Esto contribuye a una experiencia personalizada, evita la sobrecarga de información y refuerza la seguridad del sistema.

Elementos comunes como el **header** (encabezado superior) y el **footer** se reutilizan en la mayoría de vistas, manteniendo la coherencia visual y facilitando la orientación del usuario. El header, además de lucir la identidad visual (colores y patrones geométricos propios de la UGR), muestra el título y, opcionalmente, un subtítulo que indica el contexto actual, adaptándose automáticamente a cada sección de la plataforma. De esta forma, se logra una navegación fluida, moderna y centrada en el usuario, que se adapta a cualquier dispositivo y a las necesidades específicas de cada perfil.

Además, en la figura 6.9 se presentan los diagramas de flujo correspondientes a los perfiles de tutor, estudiante y administrador. Cada diagrama resume los pasos principales y la navegación típica según el rol, facilitando la comprensión del recorrido de usuario y de las acciones clave dentro de la plataforma.

Vistas principales de



Landing page

¿Por qué usar nuestra plataforma?

- Accede a destinos validados
- Consultaciones simplificadas
- Contacto directo con tu tutor
- Visualiza experiencias reales
- Documentación automática
- Filtros avanzados por aspiración

¿Cómo funciona?

- Regístrate
- Completa tu perfil
- Explora destinos
- Crea tu acuerdo
- Consulta experiencias

Te espera una experiencia inolvidable...

Viajar... Estudiar... Conocer gente!

¿Comenzamos la aventura?

Regístrate

Enlaces rápidos

- Sobre
- Destinos
- Destinos frecuentes

Legal

- Política de privacidad
- Aviso Legal
- Licencia del software

Contacto

- Universidad de Granada
- +34 958 24 30 30
- Oficina de Relaciones Internacionales
- relo@ugr.es

UNIVERSIDAD DE GRANADA

Plataforma de Aspiraciones Erasmus

Preguntas y respuestas

Preguntas Frecuentes

Buscar pregunta...

Todos Asunto Convocatorias Documentación General

¿Qué es el proceso Erasmus?

¿Qué requisitos debe cumplir para solicitar una beca Erasmus?

¿Dónde puedo consultar los destinos disponibles para mi titulación?

¿Qué documentación debe presentar antes de la lista de Erasmus?

¿Qué modificar si estoy de acuerdo con mis pedidos?

¿Qué información puedo consultar en el destino Erasmus?

Inicio de sesión

Hola de nuevo!

Email:

Contraseña:

¡No tienes cuenta? Regístrate

Elección de rol

Bienvenido

Alumno Tutor

Perfil del usuario

Mi Perfil

Información académica

Grado: Escuela Superior de Ingeniería Informática y de Telecomunicación

IIº Bach. Créditos: 120

Créditos superados: 120

Grado: Escuela Superior de Ingeniería Informática y de Telecomunicación

IIº Bach. Créditos: 120

Créditos superados: 120

Asignaturas Superiores

Grado: Escuela Superior de Ingeniería Informática y de Telecomunicación

IIº Bach. Créditos: 120

Créditos superados: 120

Grado: Escuela Superior de Ingeniería Informática y de Telecomunicación

IIº Bach. Créditos: 120

Créditos superados: 120

Másteres

Grado: Escuela Superior de Ingeniería Informática y de Telecomunicación

IIº Bach. Créditos: 60

Créditos superados: 60

Registro como nuevo usuario

Registro como estudiante

Nombre: Primer Apellido: El García

Segundo Apellido: El Pérez

Email: Contraseña:

Grado: Selecciona tu grado

Registrarse

Si ya tienes cuenta: Inicia sesión

Example of collection management panel

Gestión de Usuarios

Nombre	Email	Role	Acciones
Antón García	anavas@ugr.es	Estudiante	
Ana Ríu Pérez	ajg@ugr.es	Estudiante	
Maria Jiménez Rojas	mjg@ugr.es	Tutor	
Admin	soporte.its.ugr.es@gmail.com	Admin	
Pedro Domínguez Oller	pedro@ugr.es	Admin	

Dashboard del estudiante

¡Hola, Blanca!

Estás actualmente en acuerdo

Nombre completo: Blanca García

Destinos

- Mi Acceso
- Estado del Proceso
- Mi Destino
- Recursos extra

Notificaciones

Progress General

Completo: 40%

Dashboard del tutor

Bienvenido/a, María

Destinos asignados

- Alumno: Génova-Huelva (PL. ISARHON02)

Estudiantes asignados

Nombre	Email	Destino	Estado Acuerdo	Estado Proceso	Acciones
Blanca Gómez Ruiz	mgomezruiz@gmail.com	Alumno: Génova-Huelva	En Ejecución	Procesando	

Dashboard del administrador

Bienvenido/a, Admin

Gestión de Usuarios

Gestión de Destinos

Gestión de Asignaturas

Gestión de Grados

Notificaciones

Vista de destinos

Destinos Erasmus

Filtros

- Filtros básicos
- Destinos
- Mi acuerdo
- Destinos
- Destinos frecuentes
- Cursos existentes
- Todos los cursos
- Aspiraciones e intercambios
- Aspiraciones inteligente

Destinos más populares

- Rheinisch-Westfälische Technische Hochschule
- Instituto Superior Miguel de Cervantes
- Universidad de Studi di Milano
- Academia Gómez-Huelva

Ver detalles

Vista de detalle del destino

Rheinisch-Westfälische Technische Hochschule

Universidad

Aspiraciones

Experiencias

Detalles del universidad

Nombre: Rheinisch-Westfälische Technische Hochschule

Dirección: Universitätsstrasse 15, D-4422, Münster, Alemania

Facultades: Facultad de Ciencias Sociales y Humanas, Facultad de Ciencias Exactas y Naturales, Facultad de Ciencias Matemáticas y Físicas, Facultad de Ciencias Químicas, Facultad de Ciencias Biológicas, Facultad de Ciencias de la Salud, Facultad de Ciencias de la Ingeniería, Facultad de Ciencias Económicas y Empresariales, Facultad de Ciencias Políticas y Sociales, Facultad de Ciencias Jurídicas y Sociales, Facultad de Ciencias de la Educación, Facultad de Ciencias de la Comunicación y las Artes.

Aspiraciones:

- Alumnado: 10000
- Investigadores: 1000
- Personal: 1000
- Alumnos: 10000
- Visitas: 10000

Destinos:

- Universidad: 10000
- Aspiraciones: 10000
- Experiencias: 10000

Requisitos:

- Alumno: 10000
- Investigador: 10000
- Personal: 10000
- Alumno: 10000
- Visita: 10000

Información de contacto

Nombre: Rheinisch-Westfälische Technische Hochschule

Dirección: Universitätsstrasse 15, D-4422, Münster, Alemania

Teléfono: +49 251 83 30 00

Mapa:

Editor del acuerdo de estudios

Acuerdo de Estudios

Detalles del destinatario

Nombre: Blanca García

Apellido: Pérez

Sexo: Hembra

Fecha de nacimiento: 1990-01-01

Destino: Rheinisch-Westfälische Technische Hochschule

Nombre: Rheinisch-Westfälische Technische Hochschule

Dirección: Universitätsstrasse 15, D-4422, Münster, Alemania

Facultad: Facultad de Ciencias Sociales y Humanas

Programa: Programa de licenciatura en Administración de Empresas

Aspiraciones:

- Alumnado: 10000
- Investigadores: 10000
- Personal: 10000
- Alumnos: 10000
- Visitas: 10000

Destinos:

- Universidad: 10000
- Aspiraciones: 10000
- Experiencias: 10000

Requisitos:

- Alumno: 10000
- Investigador: 10000
- Personal: 10000
- Alumno: 10000
- Visita: 10000

Información de contacto

Nombre: Rheinisch-Westfälische Technische Hochschule

Dirección: Universitätsstrasse 15, D-4422, Münster, Alemania

Teléfono: +49 251 83 30 00

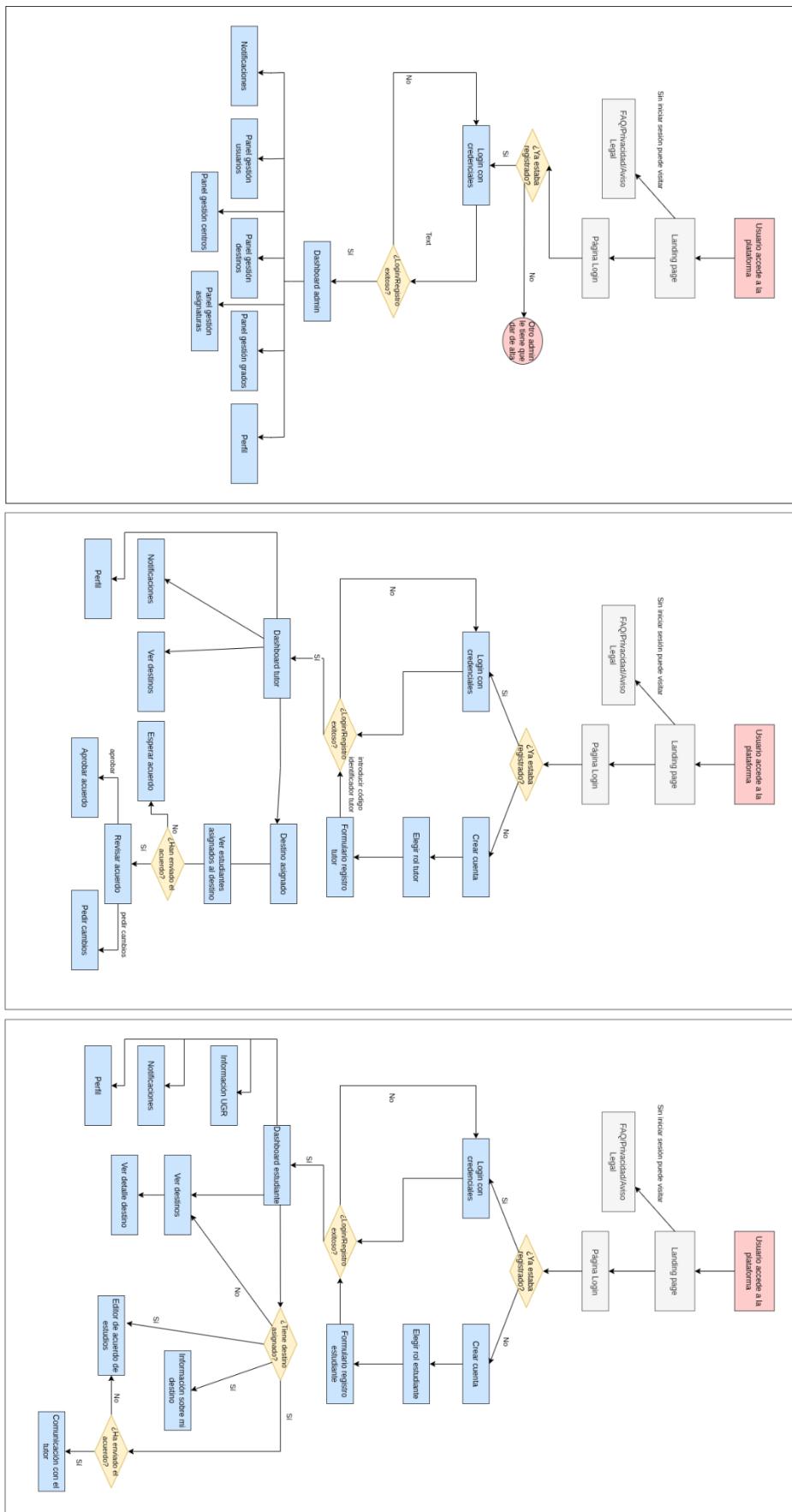


Figura 6.9: Diagrama de flujo de los distintos roles

Capítulo 7

Implementación

En este capítulo se explica, de manera detallada, cómo se ha construido técnicamente la plataforma. Más allá de enumerar las tecnologías empleadas, el objetivo es mostrar cómo cada decisión tomada ha ido aportando valor al sistema, permitiendo que sea funcional, seguro y fácil de mantener. Para ello, se recorre el proceso completo de implementación: desde el entorno de desarrollo y los lenguajes escogidos, hasta la forma en que se han materializado el backend y el frontend en una solución integrada y coherente.

7.1. Entorno y herramientas de desarrollo

En el capítulo [6] se introdujeron y justificaron las principales tecnologías usadas en el desarrollo. En esta sección, además de volver a enumerar el stack para tener una visión completa, se incluyen los lenguajes de programación, las herramientas y librerías adicionales que han sido necesarias para la implementación práctica.

7.1.1. Lenguajes de programación

La plataforma utiliza dos lenguajes principales, cada uno escogido por lo apropiados que son para la parte correspondiente del sistema:

- **Python (backend):** empleado en la construcción de la API. Se escogió por su sintaxis sencilla y legible, su gran ecosistema de librerías y su utilidad en el desarrollo web, ya que cuenta con marcos web como Flask pequeños y ligeros que proporcionan herramientas y funciones útiles para que la creación de este tipo de aplicaciones sea más fácil. Gracias a estas características, ha sido posible implementar funcionalidades clave como el registro y validación de usuarios, la gestión de acuerdos de estudios o la exportación de documentos en formato PDF. Su documentación se puede consultar en [45].

- **JavaScript + JSX (frontend):** JavaScript es un lenguaje de programación que se puede aplicar a documentos HTML y usarse para crear interactividad en sitios web. Es el lenguaje que entienden los navegadores. En este caso, se ha usado junto a React para construir la interfaz de usuario. La extensión JSX permite mezclar lógica con estructura HTML dentro de los componentes, lo que facilita el desarrollo de interfaces dinámicas y reutilizables. Esto resulta especialmente útil para vistas que cambian, por ejemplo, según el rol del usuario (estudiante, tutor o administrador). Su documentación se puede consultar en [42].

7.1.2. Frameworks. Bibliotecas utilizadas.

Backend

La parte del servidor se ha implementado con **Flask**, un microframework en Python que aporta ligereza y flexibilidad¹. Esta elección permite construir una API REST clara, manteniendo el control sobre la lógica sin depender de estructuras demasiado rígidas.

Las principales librerías complementarias empleadas en el backend son:

- **Werkzeug:** es una biblioteca de utilidades WSGI (Web Server Gateway Interface) para aplicaciones web en Python, que proporciona soporte avanzado en la gestión de peticiones HTTP, enrutamiento y depuración. Es la base sobre la que se construye Flask [6]. En esta plataforma se emplea principalmente para la *gestión segura de contraseñas*, mediante algoritmos hash que garantizan que las contraseñas no se almacenan en texto plano en la base de datos, preservando así la seguridad de los usuarios.
- **python-dotenv:** facilita la gestión de variables sensibles a través de archivos .env, como las claves secretas o el código necesario para el registro de tutores. Esto permite mejorar la seguridad y facilita el despliegue en diferentes entornos (desarrollo, pruebas y producción) sin necesidad de modificar el código.
- **ReportLab / WeasyPrint:** son herramientas destinadas a la generación automática de documentos PDF. Ambas permiten construir documentos a partir de código, siguiendo plantillas y estructuras predefinidas. En la plataforma se utilizan para producir los documentos de *Acuerdo de Estudios*, con un formato alineado a las exigencias y recomendaciones institucionales de la Universidad de Granada. Gracias a estas librerías se garantiza que los documentos generados sean homogéneos, presenten una estructura clara y se puedan descargar directamente desde la aplicación, evitando que los usuarios los tengan que crear manualmente [46].

¹Un *framework* es un entorno de trabajo predefinido que proporciona un conjunto de librerías, componentes y buenas prácticas para facilitar el desarrollo de software. En lugar de programar desde cero, el desarrollador se apoya en estas herramientas para ahorrar tiempo, reducir errores y mantener la coherencia en el proyecto.

Entorno virtual y gestión de dependencias. Para el desarrollo del backend se utilizó un **entorno virtual** creado con `venv`, lo que permitió aislar las dependencias del proyecto respecto al sistema global y garantizar la reproducibilidad en cualquier máquina.

Dentro del entorno virtual, las dependencias se instalaron mediante `pip` y se mantuvieron centralizadas en el archivo `requirements.txt`. Este fichero permite recrear el mismo entorno en otra máquina ejecutando un solo comando:

```
pip install -r requirements.txt
```

Esta práctica asegura:

- Aislamiento de librerías y versiones específicas del proyecto.
- Portabilidad a cualquier entorno de desarrollo o servidor.
- Control de versiones de dependencias para evitar incompatibilidades.

Frontend

El **frontend** de la plataforma —es decir, la parte visual con la que interactúan estudiantes, tutores y administradores— se ha construido con la biblioteca **React**. Esta tecnología es hoy en día una de las más utilizadas para el desarrollo de interfaces dinámicas porque se basa en el concepto de *componentes*: pequeñas piezas reutilizables que pueden combinarse para dar forma a pantallas completas. Esto permite que elementos comunes como el menú lateral, el encabezado o los formularios puedan desarrollarse una sola vez y reutilizarse en distintas secciones, reduciendo errores y mejorando la mantenibilidad.

Además, se integraron diversas herramientas complementarias que aportan funcionalidad y mejoran la experiencia tanto de desarrollo como de uso:

- **React.js + Vite**: esta combinación ofrece el corazón del frontend. React se encarga de gestionar los componentes y la lógica visual, mientras que Vite actúa como entorno de desarrollo y empaquetador, permitiendo recargar los cambios casi de forma instantánea y generar un código optimizado para producción. Esto acelera el trabajo del desarrollador y asegura un rendimiento fluido para el usuario final.
- **Tailwind CSS**: es un *framework* de utilidades CSS², lo que significa que en lugar de diseñar cada estilo desde cero, se utilizan clases predefinidas que controlan márgenes, colores, tamaños o tipografías. Gracias a ello, se ha logrado un diseño coherente, minimalista y moderno, que además es completamente responsive: la interfaz se adapta automáticamente a distintos dispositivos, desde ordenadores de escritorio hasta móviles.

²Un *framework de utilidades CSS* es un conjunto de clases ya definidas que permiten aplicar estilos directamente en el código HTML o JSX, evitando escribir largas hojas de estilo tradicionales.

- **Lucide-react:** librería de iconos vectoriales que aporta una iconografía clara y homogénea en toda la aplicación. Los iconos son escalables (no pierden calidad al cambiar de tamaño) y personalizables en color y forma.
- **React Router:** herramienta que gestiona la navegación entre las diferentes vistas de la aplicación. Al tratarse de una **SPA (Single Page Application)**, no se recarga toda la página cada vez que el usuario cambia de sección; en su lugar, React Router actualiza solo los componentes necesarios.
- **Fetch API:** utilizada para enviar y recibir datos entre el frontend y el backend a través de peticiones HTTP (GET, POST, PUT, DELETE). Por ejemplo, cuando un estudiante guarda su acuerdo de estudios o un tutor lo revisa, la petición se envía al backend con Fetch, y el resultado se devuelve en formato JSON para actualizar la interfaz en tiempo real.
- **LocalStorage:** un sistema de almacenamiento local en el navegador que permite guardar datos básicos, como la sesión activa del usuario o sus preferencias, sin necesidad de recurrir continuamente al servidor. Esto mejora la rapidez de la aplicación y hace posible que, aunque se cierre el navegador, el usuario mantenga su sesión al volver a entrar.

7.2. Desarrollo

7.2.1. Desarrollo del Backend

Tal y como se adelantó en el capítulo en el que se detallaba el diseño de la plataforma [6], el backend de la plataforma se ha implementado siguiendo una organización modular que favorece la claridad y el mantenimiento. La aplicación está dividida en **blueprints** (módulos reutilizables de Flask), lo que permite agrupar las rutas y la lógica de cada dominio (usuarios, destinos, centros, acuerdos, etc.) en ficheros independientes.

Estructura de ficheros

A alto nivel, la organización del código se basa en tres grandes bloques, con responsabilidades bien separadas:

- **models/:** Los archivos de esta carpeta definen cómo deben estructurarse los documentos que se almacenan en la base de datos. Cada fichero (`usuario.py`, `acuerdo.py`, `destino.py`, `asignatura.py`, etc) corresponde a una colección y expone funciones constructoras como `crear_usuario` o `crear_destino`, que devuelven diccionarios con la forma esperada. Esto asegura que todos los datos insertados en la base se ajusten a un formato coherente, inicializando campos opcionales con valores por defecto.

- **routes/**: Contiene los **endpoints REST** que son consumidos por el frontend. Cada fichero implementa un Blueprint de Flask. Por ejemplo:

- **usuarios.py** gestiona el registro, login, verificación de email, subida de fotos a Cloudinary y renuncias de destino.
- **acuerdos.py** coordina todo el ciclo de vida del Acuerdo de Estudios: creación de versiones, comentarios del tutor, envío a revisión, aprobación y generación del PDF.
- **destinos.py**, **centros.py**, **grados.py**, **asignaturas.py** ofrecen operaciones CRUD sobre las entidades académicas.

En cada endpoint se reciben peticiones HTTP(GET,POST,PUT,PATCH,DELETE), se valida la información, se consulta o actualiza la base de datos, y se devuelve una respuesta en formato JSON.

- **utils/**: En esta carpeta se almacenan los ficheros que centralizan la lógica auxiliar y los servicios externos: la autenticación con funciones para hacer hash y verificar contraseñas, la gestión del correo electrónico para que el sistema envíe los mensajes necesarios para la verificación o la resolución de dudas, o servicios externos como **Cloudinary** para la subida y el borrado de imágenes de usuarios y destinos.

Esta estructura modular evita la duplicación del código y permite evolucionar cada parte de forma independiente.

La mayor parte del backend consiste en operaciones CRUD (Create, Read, Update, Delete): crear (POST), leer (GET), actualizar (PUT/PATCH) y eliminar (DELETE) recursos. Sin embargo, otras funcionalidades clave han conllevado una implementación más específica, las cuales se repasarán a continuación:

Funcionalidades clave:

Acceso a datos y conexión única

El backend abre una única conexión a **MongoDB** y la comparte en todos los módulos: **db.py** crea el **MongoClient** y expone **db**, de modo que las rutas operan directamente sobre colecciones como **db.usuarios**, **db.destinos** o **db.acuerdos**. Este patrón simplifica el uso y aprovecha el *pool* de conexiones de PyMongo. **PyMongo** es el paquete oficial de controladores para trabajar con MongoDB desde Python, y permite realizar operaciones de consulta, inserción, actualización y agregación de forma eficiente.

Es importante distinguir responsabilidades: un **Blueprint** de Flask organiza rutas y lógica de negocio, pero *no* crea colecciones en MongoDB. Las colecciones se materializan automáticamente en la base de datos la primera vez que el backend realiza una operación de escritura sobre ellas. En otras palabras, el blueprint “**usuarios**” define los endpoints; cuando alguno de esos endpoints persiste un documento en “**usuarios**”, si la colección no existía, MongoDB la crea en ese momento.

Aunque MongoDB incluye de serie las bases `admin`, `local` y `config`, la plataforma trabaja sobre su propia base de datos (descrita en el capítulo [6]), donde residen las colecciones específicas de ErasmusUGR. “Añadir una colección” en esta arquitectura equivale, en la práctica, a incorporar el módulo de rutas que gestionará ese dominio y a realizar la primera inserción/actualización sobre la colección correspondiente; esa primera escritura la deja creada de forma implícita.

El flujo completo queda así: la SPA en React invoca un endpoint; el blueprint valida la petición (campos obligatorios, unicidad por combinaciones de campos, permisos y rol) y, sólo entonces, consulta o escribe en la colección adecuada usando la conexión única compartida. La respuesta se devuelve en JSON, con códigos HTTP coherentes (201 al crear, 409 si hay duplicados, 400 si faltan datos, 404 si no existe, 403 si no hay permisos o la cuenta no está verificada). Cuando se devuelven documentos completos, el identificador interno se serializa a texto para compatibilidad total con JSON. Con este enfoque, la relación *ruta* → *colección* queda clara, se evitan aperturas de conexión innecesarias y se mantiene la consistencia de los datos desde el propio backend.

Autenticación con hash de contraseña y verificación por correo

El proceso de autenticación se divide en tres momentos: *registro*, *verificación del correo* e *inicio de sesión*. La idea es sencilla: la contraseña nunca se guarda en claro (se almacena su hash), y la cuenta queda inactiva hasta que el usuario confirma su correo mediante un enlace único enviado por email. Así se protege el acceso y se evitan altas con correos no válidos.

Registro. Cuando el usuario se da de alta, el backend calcula el **hash** de la contraseña y guarda la cuenta marcada como **verificado = false**. Además, genera un **token** de verificación asociado al usuario y envía un **correo HTML** con un enlace del tipo `/verificar-email?token=....` Antes de crear la cuenta, se comprueba que no exista otra con el mismo email (y, en su caso, que no haya colisión por nombre y apellidos) y, si el rol solicitado es **tutor**, se valida un **código de tutor** definido en la configuración. Las credenciales del envío (usuario, clave, remitente) se gestionan por variables de entorno y el correo se envía mediante conexión segura (SSL).

Verificación del correo. Al abrir el enlace, el backend localiza el token y marca la cuenta como **verificado = true**; el token se invalida para que no pueda reutilizarse. Si el token no existe o ya fue usado, se informa con un error claro. Con esta confirmación se cierra el ciclo de alta y la cuenta queda lista para iniciar sesión.

Inicio de sesión. En el login, el backend comprueba que el email exista, verifica el **hash** de la contraseña y **exige** que la cuenta esté verificada; de lo contrario, deniega el acceso e informa explícitamente de que falta la verificación por correo. Los mensajes de error siguen la semántica HTTP para que el cliente los interprete de forma consistente: 401 si la contraseña no coincide, 403 si la cuenta no está verificada, 404 si el usuario no existe, 409 si se detectan duplicados en el registro. Todas las respuestas viajan en **JSON**, de modo que el frontend puede mostrar feedback inmediato y comprensible.

Recuperación de contraseña. Para dar soporte a usuarios que olvidan su clave, se im-

plementa un mecanismo de **reset seguro**. El flujo se inicia con un formulario donde se introduce el correo; si existe, el sistema genera un **token temporal** de un solo uso, lo asocia al usuario y envía un email con un enlace de recuperación (`/auth?resetToken=...`). Este enlace caduca a la hora y se invalida tras ser usado, lo que impide su reutilización. Al acceder, el usuario define una nueva contraseña que se almacena también mediante **hash**, sobrescribiendo la anterior. Por motivos de seguridad, la respuesta al formulario inicial es neutra (no confirma si el correo existe o no) y el backend elimina inmediatamente el token en cuanto se completa el cambio. Así se garantiza que el proceso sea claro para el usuario y a la vez resistente frente a ataques de enumeración de cuentas.

Sistema de notificaciones

El sistema de notificaciones asegura que las acciones críticas dentro de la plataforma (asignación o renuncia de destinos, envío y revisión de acuerdos, peticiones de cambio, nuevos comentarios, etc.) queden registradas y comunicadas al usuario de manera clara. De este modo, además de informar, se mantiene un historial consultable que aporta trazabilidad al proceso.

Cada notificación se almacena como un documento con campos como `usuario_email`, `titulo`, `mensaje`, `tipo` (informativa, alerta, cambio), `leida`, `enlace` y un sello temporal. Esta estructura ligera permite que el frontend muestre listas resumidas, contadores de “no leídas” y que, al hacer clic, el usuario acceda directamente al contexto del aviso (por ejemplo, un acuerdo concreto).

Las notificaciones se generan automáticamente a partir de eventos de negocio:

- **Gestión de destinos:** asignaciones y renuncias, notificando a estudiantes, tutores y administradores según corresponda.
- **Ciclo de acuerdos:** envío a revisión, solicitud de cambios, aprobación o mensajes en el hilo tutor–estudiante.
- **Eventos administrativos:** incidencias o comunicaciones globales, difundidas a los administradores mediante un helper específico.

El backend expone endpoints para listar notificaciones (ordenadas por fecha), marcarlas como leídas (individual o masivamente), crear avisos puntuales y limpiar el historial de un usuario. El cliente consulta periódicamente o tras acciones clave, manteniendo siempre actualizado el contador de pendientes.

Con este diseño, la comunicación es inmediata, se conserva un rastro persistente y la experiencia de usuario se optimiza al ofrecer un feedback claro y contextualizado en todo momento.

Acuerdo de Estudios

El acuerdo de estudios se gestiona como un documento versionado que centraliza la propuesta del estudiante y el seguimiento del tutor hasta su aprobación final. El backend

asegura que cada acción quede registrada y que el estudiante y el tutor reciban feedback inmediato a lo largo del proceso.

Cada acuerdo mantiene una **versión** numerada, el **estado** del proceso y un **historial** de eventos. Esto permite que el sistema nunca sobrescriba la información previa: cada cambio genera una nueva versión, lo que garantiza trazabilidad y transparencia. Además, el estudiante tiene un *estado compacto* en su perfil que refleja en todo momento si el acuerdo está en borrador, enviado, con cambios solicitados o aprobado.

Las transiciones se producen de forma controlada:

- **Envío:** el estudiante envía su acuerdo a revisión, se bloquea la edición y se notifica al tutor.
- **Solicitud de cambios:** el tutor añade comentarios generales o por bloques, el acuerdo pasa a estado **cambios_solicitados** y el estudiante recibe una notificación.
- **Aprobación:** el tutor valida el acuerdo, se marca la versión como **aprobada**, se actualiza el **estado_proceso** del estudiante y se notifica el resultado.

El acuerdo incluye también un **hilo de mensajería** tutor–estudiante y un **log de acciones** con fecha, autor y descripción, que conforman un expediente auditável del proceso. Una vez aprobado, la plataforma permite **exportar un PDF oficial** a partir de la última versión, listo para descarga o envío a la oficina internacional.

Con este diseño, el backend garantiza que cada versión, comentario y decisión queden registrados, que el flujo estudiante–tutor sea claro y que el resultado final se convierta en un documento formal, todo ello con una API consistente y sencilla de consumir desde el frontend.

Servicios externos integrados

Como se ha mencionado anteriormente, el backend se apoya en servicios externos que amplían sus funcionalidades y aportan eficiencia en la gestión de recursos.

Gestión de imágenes con Cloudinary. Las fotos de perfil de los usuarios y las imágenes de universidades se gestionan a través de **Cloudinary**, un servicio especializado en almacenamiento y optimización de imágenes. El backend emplea el **SDK oficial de Cloudinary para Python**, configurado mediante variables de entorno, y funciones auxiliares para subir o eliminar imágenes de forma segura. En lugar de almacenar los binarios en MongoDB, únicamente se guardan la **url** y el **public_id** de la imagen, lo que reduce el peso en la base de datos y mejora la velocidad de carga en el frontend. El flujo incluye la validación del tipo de archivo y la eliminación de la imagen anterior en caso de sustitución, garantizando siempre la coherencia de la información.

Envío de correos electrónicos. Para las comunicaciones transaccionales, como la verificación de cuentas de usuario, se utiliza la librería estándar de Python **smtplib**, junto con las clases **MIMEText** y **MIMEMultipart** para construir mensajes en formato HTML. El servidor establece la conexión con Gmail mediante **SMTP seguro (SSL)** en el puerto 465, autenticándose con las credenciales del remitente definidas en variables de entorno. En cada registro se genera un **token único** que se inserta en el enlace de verificación enviado al correo del usuario, de modo que la cuenta no queda activa hasta que se accede a dicho enlace. Con este mecanismo se garantiza la validez de la dirección de correo y se refuerza la seguridad del sistema.

En conjunto, la integración de **Cloudinary** para las imágenes y **smtplib/SMTP** para el correo electrónico proporciona una solución ligera, segura y escalable: las imágenes se sirven desde una CDN optimizada y las notificaciones llegan directamente al buzón del usuario, mejorando tanto la experiencia como la fiabilidad de la plataforma.



Figura 7.1: Logo de *Cloudinary*

7.2.2. Desarrollo del Frontend

Para lograr que la plataforma funcione como la **Single Page Application (SPA)** descrita en el capítulo [6] y que la navegación y las actualizaciones de contenido se realicen dentro del navegador, sin necesidad de recargar toda la página, se ha llevado a cabo la siguiente implementación:

Estructura de ficheros

La aplicación se organiza en la carpeta principal **src**, que contiene dos bloques fundamentales:

- **components/**: aquí se agrupan los distintos componentes de React, organizados por dominio o por funcionalidad. Existen componentes generales compartidos, como barras laterales o menús, y otros específicos según el rol del usuario (estudiante, tutor o administrador).
- **assets/**: contiene imágenes, iconos y recursos gráficos utilizados en la interfaz, como logotipos o elementos visuales de la landing page.

Además, en la raíz de **src** se encuentran los ficheros **main.jsx** y **App.jsx**. El primero es el punto de entrada de la aplicación, donde se monta la SPA en el DOM del navegador. El segundo define la estructura principal de rutas con **react-router-dom**, que determina qué componente debe renderizarse en función de la URL.

Gestión de estados y efectos

La lógica de los componentes se apoya en **hooks**³ de React:

- **useState** se utiliza para gestionar el estado local de cada componente (por ejemplo, la información cargada del perfil, la lista de destinos o si un formulario está en modo edición).
- **useEffect** permite ejecutar efectos secundarios cuando cambian determinadas variables: se usa para cargar datos desde el backend al montar el componente o al variar dependencias clave (como el código de grado del usuario para traer sus asignaturas).
- **useNavigate** facilita redirecciones internas en la aplicación tras ciertas acciones (como volver al dashboard después de editar un acuerdo).
- **localStorage** se utiliza para mantener la sesión del usuario entre recargas, almacenando información básica (email, rol) y accediendo a ella en cada renderizado.

Gracias a estos mecanismos, los componentes se mantienen sincronizados con el backend: cada vez que el usuario accede a su perfil o dashboard, un **useEffect** lanza una petición HTTP (**fetch**) a la API, y al llegar la respuesta se actualiza el estado del componente, que se re-renderiza de forma automática mostrando los datos actuales.

Componentes principales

Cada componente cumple una función precisa y, a su vez, puede estar formado por componentes más pequeños con funcionalidades concretas:

- Las **páginas** (o contenedores) gestionan los datos y la navegación. Son los encargados de lanzar peticiones al backend, manejar estados globales (como carga o error) y decidir qué subcomponentes mostrar en cada momento.
- Los **subcomponentes** se centran exclusivamente en la interfaz. Estos componentes renderizan los datos que reciben y ejecutan las acciones que se les pasan mediante propiedades, sin tener que preocuparse de la lógica de negocio.

Este reparto evita mezclar responsabilidades, facilita la reutilización de elementos y reduce el acoplamiento, haciendo que el código sea más fácil de mantener y extender.

Funcionalidades. Dentro de esta estructura, la plataforma cuenta con varias vistas clave, cuyos diseños ya se han visto en el capítulo anterior:

³Un hook en React es una función especial que permite a los componentes funcionales “engancharse” al estado y a otros ciclos de vida de React, como efectos secundarios o contexto, sin necesidad de usar clases.

- **Landing page:** sirve como carta de presentación de la plataforma. Se ha diseñado para ser visualmente más atractiva, incluyendo secciones informativas, llamadas a la acción y elementos gráficos personalizados que refuerzan la identidad con la UGR.
- **Dashboards por rol:** cada tipo de usuario (estudiante, tutor, administrador) accede a un panel específico que concentra la información y acciones que le corresponden. Estos dashboards integran subcomponentes como tarjetas de progreso, tablas de acuerdos o notificaciones.
- **Perfil de usuario:** vista dedicada a la gestión de datos personales y académicos, así como a la edición y previsualización de la foto de perfil.
- **Gestión de destinos:** páginas que permiten explorar, filtrar y consultar el detalle de las universidades de destino, con fichas que muestran requisitos, asignaturas y experiencias de otros estudiantes.
- **Gestión del acuerdo de estudios:** editor interactivo en el que el estudiante construye su propuesta de equivalencias, y el tutor la revisa o modifica.
- **Paneles de administración:** vistas reservadas para los administradores del sistema, desde donde se gestionan los contenidos de las colecciones y otras funcionalidades del backend.
- **Secciones de apoyo:** páginas estáticas como *Aviso legal*, *Política de privacidad* o *FAQ*, accesibles desde la navegación principal, que completan la experiencia de usuario con información relevante y de referencia.

Además, se aplicó un **diseño responsive** mediante las clases utilitarias de **Tailwind CSS**, utilizando los *breakpoints* estándar (`sm:`, `md:`, `lg:`) para garantizar que la interfaz se adapta correctamente a distintos dispositivos. Esto asegura que la plataforma pueda consultarse de forma cómoda tanto en ordenadores como en tablets o móviles, manteniendo siempre la coherencia visual y la usabilidad.

De esta manera, el frontend no solo ofrece las funcionalidades principales de gestión de movilidad, sino también una experiencia de usuario completa, coherente y accesible desde cualquier dispositivo.

Capítulo 8

Verificación y pruebas funcionales

El desarrollo de la plataforma no se dio por concluido hasta haber realizado una fase de verificación que asegurara el correcto funcionamiento de los módulos principales . Para ello, se optó por un enfoque práctico y ligero: utilizar **Postman** para probar la API y complementar con pruebas manuales en el navegador, simulando flujos de usuario reales.

8.1. ¿Qué es Postman?

Postman en sus inicios nace como una extensión que se podía utilizar en el navegador *Chrome*. Esta herramienta permite realizar peticiones de manera simple para testear APIs de tipo REST propias o de terceros. Gracias a los avances tecnológicos, Postman ha evolucionado y ha pasado de ser una extensión a una aplicación que dispone de herramientas nativas para varios sistemas operativos [44].

Sus principales ventajas son:

- Permite enviar peticiones HTTP (GET, POST, PATCH, DELETE, etc.) de manera sencilla.
- Organiza las peticiones en colecciones y carpetas, lo que permite tener un *plan de pruebas* estructurado.
- Facilita la definición de entornos con variables (por ejemplo, la URL base `http://localhost:5000`, correos de prueba o identificadores de grados), evitando modificar cada petición manualmente.
- Permite añadir pequeñas validaciones automáticas en JavaScript (tests) que verifican el código de estado de las respuestas (200, 201, 400, 404, 409, etc.) o reutilizan datos obtenidos en pasos anteriores.

En otras palabras, Postman no solo sirve para “probar” una ruta de la API, sino para simular recorridos completos y automatizar comprobaciones repetibles.



Figura 8.1: Logo de *Postman*

8.2. Diseño de la colección de pruebas

Para verificar el funcionamiento, se creó una colección llamada **Erasmus-Smoke**, siguiendo la estrategia de *smoke-testing*. Este enfoque consiste en ejecutar un conjunto reducido pero representativo de pruebas que verifican los **flujos críticos del sistema** de manera rápida. La idea es similar a comprobar que “no sale humo”: si estas pruebas básicas fallan, no merece la pena continuar con escenarios más avanzados o pruebas manuales porque la aplicación no es estable.

Las pruebas se organizaron en carpetas que reflejan las principales áreas del sistema, centrándose en operaciones CRUD que constituyen la base del backend:

- **Auth & Usuarios:** registro, login, consulta y actualización de perfiles.
- **Centros:** creación, listado, edición y borrado, incluyendo la comprobación de duplicados.
- **Grados:** operaciones CRUD similares a los centros.
- **Asignaturas:** gestión con su clave compuesta (código y grado), con pruebas tanto de casos válidos como de errores.
- **Destinos:** creación (validando restricciones), consulta filtrada y detección de duplicados.
- **Acuerdos:** simulación de un ciclo completo de vida (borrador → envío → cambios → reenvío → aprobación).

Cada carpeta incluye tanto escenarios positivos (p. ej. 201 `Created` al crear un recurso) como negativos (400 `Bad Request` si faltan datos, 404 `Not Found` si no existe, 409 `Conflict` en caso de duplicados). De esta manera, con un solo repaso de la colección se obtiene una visión clara de si el sistema funciona en sus aspectos esenciales.

8.2.1. Entorno de pruebas

Para no afectar a los datos reales, se configuró el backend para conectarse a una base de datos de pruebas llamada `erasmus_test`. En Postman, se definió un entorno “Erasmus-Local” con variables como:

- `baseUrl = http://localhost:5000`
- `email_estudiante, email_tutor, email_admin` (usuarios de prueba)
- `id_centro, id_grado, codigo_destino` (rellenados dinámicamente durante las ejecuciones)

Esto permitió lanzar todas las pruebas de colección de forma automática, sin riesgo de romper datos de producción. La colección usada se puede consultar en el repositorio del proyecto.

8.2.2. Ejecución de las pruebas

La ejecución se realizó con el *Collection Runner* de Postman. En unos minutos se recorren todos los endpoints principales, verificando que:

- Los recursos nuevos se crean correctamente (201 `Created`).
- No se permiten duplicados (409 `Conflict`).
- Los errores se gestionan adecuadamente (400 `Bad Request`, 404 `Not Found`).
- El ciclo del acuerdo de estudios sigue las transiciones de estado esperadas.

Además, estas pruebas automáticas se complementaron con verificaciones manuales en el navegador.

8.3. Pruebas de Navegación y Flujo del Usuario

Además de las pruebas automáticas realizadas sobre la API mediante Postman, se llevaron a cabo verificaciones manuales desde el navegador web para evaluar la experiencia de uso completa del sistema. Estas pruebas se centraron en reproducir los recorridos que se pueden observar en la Figura 6.9, con el objetivo de comprobar tanto la funcionalidad como la usabilidad de la plataforma.

8.3.1. Validación de formularios y acciones básicas

Se verificó el correcto funcionamiento de:

- El registro y acceso de usuarios con distintos roles (estudiante, tutor y administrador).
- La edición del perfil de usuario y actualización de datos.
- La correcta gestión de errores en caso de introducir credenciales incorrectas o datos incompletos.
- Los distintos formularios de la plataforma: envío de dudas, gestión de las colecciones.

8.3.2. Comprobaciones adicionales

Se prestó atención a aspectos transversales como:

- La accesibilidad a las secciones clave desde la barra de navegación y menús laterales, además de a páginas estáticas o el redireccionamiento a enlaces de la Universidad de Granada u otro tipo de recursos.
- La coherencia en la secuencia de páginas y estados (por ejemplo: *login* → *consulta de destinos* → *asignación de destino* → *acuerdo de estudios*).
- La correcta visibilidad de mensajes de error, avisos y confirmaciones al realizar acciones.
- El control de acceso a rutas restringidas, de modo que solo los roles autorizados puedan acceder a cada funcionalidad.
- La visualización adecuada de la interfaz en distintos tamaños de pantalla, comprobando la correcta aplicación del diseño adaptativo.

8.3.3. Pruebas de visualización en distintos dispositivos

Con el objetivo de garantizar la accesibilidad de la plataforma, se ha probado su diseño *responsive* en diferentes dispositivos. De este modo, la interfaz se adapta tanto a pantallas de ordenador como a tabletas y teléfonos móviles, reorganizando menús y columnas para facilitar la navegación. Este enfoque resulta especialmente útil en situaciones cotidianas: por ejemplo, un estudiante puede consultar desde el móvil los destinos disponibles mientras viaja en transporte público, o revisar el estado de su acuerdo de estudios en un momento libre sin necesidad de acceder a un ordenador. Gracias a esta adaptabilidad, la plataforma ofrece una experiencia coherente y cómoda en cualquier contexto de uso.

Ejemplos de vistas en dispositivos de distinto tamaño



iPhone:

This screenshot shows the mobile application's main dashboard. On the left, there's a sidebar with icons for 'Por qué usar nuestra plataforma?' (Why use our platform?), '¿Cómo funciona?' (How it works), and five numbered steps (1-5) for registration, profile completion, destination exploration, creating an agreement, and consulting experiences. The main content area features a large search bar at the top labeled 'Destinos Erasmus' with the sub-instruction 'Filtrá y encuentra tu mejor opción'. Below it are dropdown menus for 'Todos los países', 'Idioma de instrucción', 'Todos los idiomas', 'Curso académico', and 'Todos los cursos'. A section titled 'Asignaturas a Convalidar' includes a 'Búsqueda Inteligente' input field. A 'Búsqueda' button is at the bottom. The central part of the screen displays a card for 'Rheinisch Westfälische Technische Hochschule' (Germany) with a photo of the university building, course requirements (B1 Aleman, B2 Ingles), and a 'Ver detalles' button.

This screenshot shows the login and profile creation screen. It features a 'Log In' button at the top right. Below it is a '¡Hola de nuevo!' greeting and fields for 'Email' and 'Contraseña' (Password). A red 'Enviar' (Send) button is at the bottom. To the right, there's a 'Mi Perfil' section with a placeholder profile picture and the name 'Blanca Mihi Perez' with the role 'Estudiante'. A 'Sin destino' (No destination) button is shown. A 'Búsqueda' button is located at the bottom right.

This screenshot shows the user profile management screen. It displays a placeholder profile picture and the name 'Blanca Mihi Perez' with the role 'Estudiante'. Below this is a 'Sin destino' (No destination) button. A 'Búsqueda' button is at the bottom right. The central part of the screen includes sections for 'Información académica' (Academic information) and 'Asignaturas Superadas' (Completed subjects).

Ipad:

This screenshot shows the mobile application's main dashboard on an iPad. It features a large search bar at the top labeled 'Destinos Erasmus' with the sub-instruction 'Filtrá y encuentra tu mejor opción'. Below it are dropdown menus for 'País de destino', 'Todos los países', 'Idioma de instrucción', 'Todos los idiomas', 'Curso académico', and 'Todos los cursos'. A section titled 'Asignaturas a Convalidar' includes a 'Búsqueda Inteligente' input field. A 'Búsqueda' button is at the bottom. The central part of the screen displays cards for 'Rheinisch Westfälische Technische Hochschule' (Germany) and 'Instituto Superior Miguel Torga (ISMT)' (Portugal), each with a photo of the university building, course requirements, and a 'Ver detalles' button.

Ipad:

This screenshot shows the mobile application's main dashboard on an iPad. It features a sidebar with icons for 'Viajes...', 'Estudiar...', 'Conocer gente!', and 'Comenzamos la aventura?'. The main content area includes a 'Dashboard' button, 'Mi Perfil', 'Destinos', and 'Mi acuerdo'. A 'Cerrar sesión' (Logout) button is at the bottom. The central part of the screen displays a '¡Hola, Bianca!' greeting and a 'Estado actual: Sin destino asignado' (Current state: No destination assigned) message. Below this are six cards: 'Mi Acuerdo' (Review and manage study agreement), 'Estado del Proceso' (Check progress of your application), 'Mi Destino' (Information about your destination university), 'Recursos útiles' (Useful resources), 'Comunicación con Tutor' (Communication with tutor), and 'Mi Perfil' (Update personal information). A 'Notificaciones' (Notifications) section at the bottom indicates 'No tienes notificaciones por ahora.' (You don't have any notifications now).

This screenshot shows the mobile application's main dashboard on an iPad. It features a sidebar with icons for 'Universidad', 'Asignaturas', and 'Experiencias'. The main content area displays cards for two universities: 'Rheinisch Westfälische Technische Hochschule' (Germany) and 'Instituto Superior Miguel Torga (ISMT)' (Portugal). Each card includes a photo of the university building, course requirements, and a 'Ver detalles' button. The 'Asignaturas' tab is selected, showing a grid of subjects like 'Basic Techniques in Computer Graphics', 'Fundamentals on Business Process Management', 'Machine Learning', and 'Business Process Intelligence', each with details like 'Asignatura UGR:', 'Créditos:', 'Semestre:', and 'Último año de reconocimiento:'.

Capítulo 9

Conclusiones. Trabajo futuro

El propósito de este capítulo es hacer balance del camino recorrido, mostrando de forma global qué resultados se han obtenido y qué aprendizajes han surgido durante el proceso. El trabajo tuvo como meta principal analizar, diseñar e implementar una plataforma web capaz de dar soporte a la movilidad Erasmus en la Universidad de Granada, integrando en un mismo sistema la gestión de acuerdos de estudios, las equivalencias y la comunicación entre los distintos roles implicados.

En las siguientes páginas se revisa hasta qué punto se han alcanzado los objetivos definidos al inicio (Sección 1.3), se señalan las dificultades más relevantes que aparecieron por el camino y cómo se afrontaron, y se reflexiona sobre lo aprendido. Finalmente, se apuntan también algunas posibles líneas de mejora y evolución futura de la plataforma.

9.1. Consecución de objetivos. Resultados

En términos generales, puede afirmarse que el objetivo principal del trabajo se ha alcanzado: se ha desarrollado una plataforma web que da soporte a los procesos y aspectos esenciales de la movilidad Erasmus para la Universidad de Granada.

La mayoría de los objetivos específicos planteados en el primer capítulo se han cumplido satisfactoriamente, aunque en algunos casos determinadas funcionalidades se han dejado en una versión inicial o con margen de mejora. Este es el comportamiento natural de un proyecto de este tipo, en el que siempre existen elementos susceptibles de perfeccionamiento o ampliación.

En la siguiente tabla se presenta una síntesis del grado de consecución de cada objetivo, señalando los que se han cumplido, los que se encuentran parcialmente resueltos y aquellos que quedan abiertos para trabajos futuros.

Objetivo Específico	Resultado
OE1.1 Estudiar cómo se gestionan los procesos actualmente	Cumplido
OE1.2 Detectar dificultades y posibles mejoras	Cumplido
OE2.1 Analizar plataformas Erasmus de otras instituciones	Cumplido
OE2.2 Identificar funcionalidades útiles y aplicables a la UGR	Cumplido
OE2.3 Determinar carencias y limitaciones de las soluciones actuales	Cumplido
OE2.4 Estudiar tecnologías usadas en estas plataformas	Cumplido
OE3.1 Definir arquitectura y estructura de la aplicación	Cumplido
OE3.2 Crear prototipos centrados en la experiencia del usuario	Cumplido
OE3.3 Desarrollar la interfaz con React	Cumplido
OE4.1 Diseñar el modelo de datos	Cumplido
OE4.2 Implementar la base de datos en MongoDB	Cumplido
OE5.1 Desarrollar una API RESTful con Flask	Cumplido
OE5.2 Gestionar datos desde la API	Cumplido
OE5.3 Asegurar comunicación segura usando JSON	Cumplido
OE6.1 Diseñar el flujo de trabajo de solicitudes	Cumplido
OE6.2 Permitir envío de solicitudes por parte de estudiantes	Cumplido
OE6.3 Crear un panel para coordinadores/tutores	Cumplido
OE6.4 Incluir notificaciones sobre el estado de las solicitudes	Cumplido
OE7 Integrar interacción social entre estudiantes	Parcial
OE8.1 Investigar medidas básicas de seguridad	Cumplido
OE8.2 Implementar autenticación y autorización de usuarios	Cumplido

Cuadro 9.1: Resumen de la consecución de los objetivos específicos

Al revisar los objetivos planteados al inicio, el balance general es muy positivo. En primer lugar, los bloques orientados a entender cómo funciona actualmente la gestión Erasmus en la UGR y a analizar plataformas de referencia de otras universidades se completaron plenamente, sirviendo como base sólida para guiar el diseño de la aplicación.

En la parte técnica, se alcanzaron los hitos principales: se definió una arquitectura clara, se diseñó una base de datos flexible en *MongoDB* y se implementó una API en *Flask* que se comunica con el *frontend* en *React*. Se logró además poner en marcha el flujo completo de acuerdos y solicitudes de convalidación, lo que permitió que estudiantes y tutores interactúasen con el sistema de una forma muy próxima a la que se sigue en la práctica real. Este fue uno de los resultados más satisfactorios, al comprobar que la plataforma funciona de manera operativa y no solo como un prototipo conceptual.

El único objetivo que no se implementó tal y como se había previsto fue la interacción directa entre estudiantes. Finalmente, se resolvió de una forma más simple, permitiendo la consulta de experiencias de destino en lugar de habilitar un sistema de comunicación propio. Aunque quedó pendiente ese componente social, la decisión permitió centrar los esfuerzos en lo esencial: digitalizar y mejorar la gestión administrativa.

Respecto a la seguridad, se incluyeron medidas básicas como la autenticación de usuarios y el control de accesos por rol, suficientes para garantizar un uso seguro en un entorno de pruebas.

Las pruebas realizadas ofrecieron una visión clara sobre la robustez y la usabilidad de la plataforma. Las verificaciones automáticas proporcionaron rapidez y repetibilidad para comprobar que el sistema respondía de manera consistente tanto en casos habituales como en situaciones de error. Las pruebas manuales, por su parte, aportaron la perspectiva del usuario final, ayudando a validar la lógica de los flujos, la claridad de los mensajes y la coherencia visual de la interfaz. Gracias a ello, fue posible ajustar pequeños fallos visuales, mejorar la redacción de mensajes y confirmar que la experiencia era fluida para los distintos roles.

Como refuerzo de estos resultados, se ha elaborado un vídeo demostrativo que muestra la interacción de los tres roles principales (estudiante, tutor y administrador) en distintos escenarios de uso. Este material audiovisual ofrece una visión práctica del funcionamiento global del sistema y está disponible en el siguiente enlace:

Vídeo demostrativo de la plataforma.

En conjunto, puede afirmarse que la plataforma responde al objetivo general del proyecto. Si bien siempre quedan aspectos susceptibles de mejora o ampliación, el resultado es una herramienta funcional, útil y con un alto potencial de evolución.

9.2. Trabajo futuro

La plataforma desarrollada demuestra que existe un gran potencial para convertirse en una herramienta útil dentro de la gestión Erasmus. El siguiente paso lógico sería trabajar hacia un despliegue real en la Universidad de Granada, de forma que los estudiantes pudieran utilizarla en su día a día. Para ello, un aspecto clave sería integrar un sistema de autenticación institucional, como el servicio *OAuth* de la UGR, que garantizaría un acceso seguro y facilitaría su adopción en un entorno oficial.

Más allá de esta integración, se identifican varias líneas de mejora que pueden enriquecer y consolidar la plataforma:

- **Escalabilidad:** ampliar el sistema a más centros, grados y destinos, garantizando que el rendimiento se mantenga incluso con un volumen elevado de usuarios.
- **Mejora de la arquitectura:** refinar la organización del backend y del frontend para que el código sea más modular, mantenible y fácil de evolucionar.
- **Comunicación entre estudiante y tutor:** pasar de un modelo basado únicamente en comentarios a un sistema de interacción más directo y estructurado, que permita resolver dudas y avanzar en el proceso de forma más fluida.

- **Nuevas funcionalidades:** explorar la incorporación de apartados de intercambio de experiencias, herramientas de apoyo a la toma de decisiones o integración con otros servicios universitarios.

En definitiva, lo conseguido hasta ahora constituye una base sólida y funcional. El reto a futuro no es reinventar la plataforma, sino consolidarla, ampliarla y adaptarla a las necesidades reales de la comunidad universitaria, de manera que pueda convertirse en una herramienta práctica y sostenible en el tiempo.

Con estas posibles líneas de mejora, el proyecto no se concibe únicamente como el cierre de una etapa académica, sino también como un punto de partida sobre el que seguir construyendo. La plataforma que se ha desarrollado es ya funcional y coherente con los procesos Erasmus, pero su verdadero valor radica en que puede evolucionar y adaptarse hasta convertirse en una herramienta real de apoyo para estudiantes, tutores y administradores. Esta idea de continuidad refuerza la satisfacción por el trabajo realizado y abre la puerta a que lo aprendido en este TFG tenga un impacto más allá del ámbito estrictamente académico.

Bibliografía

- [1] María G. Alonso de Castro and Francisco J. García-Peñalvo. Erasmus+ educational projects on elearning and related methodologies: Data from erasmus+ project results platform. In Francisco J. García-Peñalvo, editor, *Information Technology Trends for a Global and Interdisciplinary Research Community*, pages 111–133. IGI Global, Hershey, PA, USA, 2021.
- [2] Amazon Web Services (AWS). Comparación entre arquitecturas monolíticas y de microservicios. <https://aws.amazon.com/es/compare/the-difference-between-monolithic-and-microservices-architecture/>, 2024. Accedido 10-05-2025.
- [3] Comisión Europea. Cómo obtener la erasmus+ app. <https://erasmus-plus.ec.europa.eu/european-student-card-initiative/erasmus-app/how-to-get>. Accedido: 05-05-2025.
- [4] Comisión Europea. Sobre la aplicación y su contenido. <https://erasmusapp.eu/about/content-app:es>. Accedido: 05-05-2025.
- [5] Lucide Contributors. Lucide icons. <https://lucide.dev/guide/>, 2024. Accedido 10-05-2025.
- [6] Cybrosys Technologies. What is werkzeug library in python? <https://www.cybrosys.com/blog/what-is-werkzeug-library-in-python>, 2024. Accedido 10-05-2025.
- [7] Educaweb. 35 años del programa erasmus en cifras. <https://www.educaweb.com/noticia/2022/06/21/35-anos-programa-erasmus-cifras-20957/>, 2022. Accedido el 5 de mayo de 2025.
- [8] Erasmus Student Alumni Association (ESAA). ESN as partner organisation. <https://www.esaa-eu.org/about-us/partner-organisations/esn>, 2024. Accedido 28-04-2025.
- [9] Erasmus Student Network. Erasmus without paper. <https://esn.org/erasmus-without-paper>, 2024. Accedido 1-05-2025.
- [10] Erasmus Student Network. Esncard - discounts for international students. <https://esncard.org/>, 2024. Accedido 1-05-2025.

- [11] Erasmus Student Network. ESN.org - plataforma de recursos para estudiantes. <https://esn.org/>, 2024. Accedido: 30-04-2024.
- [12] Erasmusu. Erasmusu linkedin profile. <https://www.linkedin.com/company/erasmusu/about/>, 2024. Accedido 10-05-2025.
- [13] Erasmusu. Te damos la bienvenida a erasmusu. <http://erasmusu.com/es/blog-erasmus/noticias-erasmus/te-damos-la-bienvenida-a-erasmusu-1297097>, 2024. Accedido 28-04-2025.
- [14] Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación. Reglamento sobre reconocimiento de créditos cursados en programas de movilidad internacional para estudiantes salientes. https://etsiit.ugr.es/sites/centros/etsiit/public/inline-files/Reglamento_Reconocimiento_Internacional_ESIIIT_2021.pdf, 2021. Accedido: 23-04-2025.
- [15] Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación, Universidad de Granada. Manual práctico para la elaboración, tramitación y modificación del acuerdo de estudios en la etsiit. https://etsiit.ugr.es/sites/centros/etsiit/public/ficheros/docmovilidad/ManualAcuerdoDeEstudios%20-%20editado_oct24.pdf, 2024. Accedido: 28-04-2025.
- [16] ESN France. Buddy system - esn france. <https://esnfrance.org/en/tools-and-actions/buddy-system/>, 2024. Accedido 28-04-2025.
- [17] ESN Spain. ¿qué es esn? <https://www.esn-spain.org/que-es-esn>, 2024. Accedido 28-04-2025.
- [18] European Commission. Learning agreement guidelines for traineeships (ka131) – erasmus+. <https://erasmus-plus.ec.europa.eu/resources-and-tools/mobility-and-learning-agreements/learning-agreements/traineeships-agreement-guidelines-ka131>. Accedido: 22-04-2025.
- [19] European Commission. Learning agreement guidelines for traineeships (ka171) – erasmus+. <https://erasmus-plus.ec.europa.eu/resources-and-tools/mobility-and-learning-agreements/learning-agreements/traineeships-agreement-guidelines-ka171>. Accedido: 20-02-2025.
- [20] European Commission. Comprehensive user guide for digital erasmus+ learning agreements. https://erasmus-plus.ec.europa.eu/sites/default/files/2023-10/User-Guide-Erasmus-Digital-LA_en.pdf, 2023. Accedido: 27-04-2025.
- [21] European Commission. About the ewp dashboard. <https://erasmus-plus.ec.europa.eu/european-student-card-initiative/ewp/dashboard>, 2024. Accedido 1-05-2025.

- [22] European Commission. Benefits of the erasmus+ app. <https://erasmus-plus.ec.europa.eu/es/european-student-card-initiative/erasmus-app/benefits>, 2024. Accedido 1-05-2025.
- [23] European Commission. Erasmus+ app. <https://erasmusapp.eu/>, 2024. Accedido: 30-04-2024.
- [24] European Commission. Erasmus+ project results platform. <https://erasmus-plus.ec.europa.eu/projects>, 2024. Accedido 5-05-2025.
- [25] European Commission. Erasmus without paper. <https://erasmus-plus.ec.europa.eu/european-student-card-initiative/ewp>, 2024. Accedido 1-05-2025.
- [26] European Commission. European student card initiative. <https://education.ec.europa.eu/education-levels/higher-education/european-student-card-initiative>, 2024. Accedido: 30-04-2024.
- [27] European Commission. European student card initiative. <https://education.ec.europa.eu/education-levels/higher-education/european-student-card-initiative>, 2024. Accedido 5-05-2025.
- [28] European Commission. Ewp dashboard - login. <https://www.erasmus-dashboard.eu/account/login>, 2024. Accedido 1-05-2025.
- [29] European Commission. How it works - european student card. <https://erasmus-plus.ec.europa.eu/es/european-student-card-initiative/card/how-it-works>, 2024. Accedido 5-05-2025.
- [30] European Union. Erasmus+ app (android). <https://play.google.com/store/apps/details?id=com.euf.project.erasmus&hl=es>, 2024. Versión Android, accedido 5-05-2025.
- [31] European University Foundation. Erasmus dashboard. [https://erasmus-dashbboard.eu/](https://erasmus-dashboard.eu/), 2023. Accedido: 30-04-2024.
- [32] European University Foundation. Erasmus without paper. <https://www.erasmuswithoutpaper.eu/>, 2023. Accedido: 30-04-2024.
- [33] European University Foundation. About the future of erasmus without paper. Technical report, European University Foundation, 2024. Accedido 28-04-2025.
- [34] European University Foundation. Erasmus without paper competence centre. <https://esci-sd.atlassian.net/wiki/spaces/EWP/pages/10879420/About>, 2024. Accedido: 30-04-2024.
- [35] European University Foundation. Faq — online learning agreement. <https://learning-agreement.eu/faq>, 2024. Accedido 26-04-2025.

- [36] Free Software Foundation. GNU General Public License. <http://www.gnu.org/licenses/gpl.html>.
- [37] Heroku. Heroku pricing. <https://www.heroku.com/pricing/>, 2024. Accedido 13-05-2025.
- [38] IBM. ¿qué es mongodb? <https://www.ibm.com/es-es/think/topics/mongodb>. Accedido: 18-08-2025.
- [39] Kandara. The client-server communication. <https://medium.com/@gamzekandara/the-client-server-communication-in-the-context-of-react-frontend-and-flask-backend-75c540911807>, 2024. Accedido 10-05-2025.
- [40] Lucid Software Inc. Diagrama de secuencia - lucidchart, 2024. Accedido 10-05-2025.
- [41] Jean-Noé Montagné and European University Foundation. Erasmus without paper: Digitalising student mobility. <https://www.eunis.org/wp-content/uploads/2019/07/3-eunis2019-mobility-slides-jmdEWP.pdf>, 2019. Presentación en EUNIS 2019, accedido 1-05-2025.
- [42] Mozilla Developer Network. Javascript reference. <https://developer.mozilla.org/en-US/docs/Web/JavaScript>, 2024. Accedido 10-05-2025.
- [43] Online Learning Agreement. Plataforma ola. <https://learning-agreement.eu/>, 2024. Accedido: 30-04-2024.
- [44] OpenWebinars. ¿qué es postman? <https://openwebinars.net/blog/que-es-postman/>, 2024. Accedido 10-05-2025.
- [45] Python Software Foundation. Python 3 documentation. <https://docs.python.org/3/>, 2024. Accedido 10-05-2025.
- [46] ReportLab Inc. Reportlab documentation. <https://docs.reportlab.com/>, 2024. Accedido 10-05-2025.
- [47] M. Vega Rodríguez. Casos de uso. <https://lsi2.ugr.es/~mvega/docis/casos%20de%20uso.pdf>, s.f. Departamento de Ciencias de la Computación e Inteligencia Artificial, Universidad de Granada.
- [48] Ian Sommerville. *Software Engineering*. Addison-Wesley, 9 edition, 2011.
- [49] Universidad de Granada. Texto consolidado del reglamento de movilidad internacional de estudiantes de la universidad de granada. https://www.ugr.es/sites/default/files/2023-09/Texto%20Consolidado%20de%20Reglamento%20de%20movilidad%20internacional%20de%20estudiantes%20de%20la%20Universidad%20de%20Granada_1.pdf, 2023. Accedido: 23-04-2025.
- [50] Universidad de Granada. *Manual de Identidad Visual Corporativa de la Universidad de Granada*, 2024. Accedido 10-05-2025.

- [51] Universidad de Granada. Movilidad internacional: Formalización inicial del acuerdo de estudios (grado). <https://sede.ugr.es/procs/Movilidad-internacional-Formalizacion-inicial-del-acuerdo-de-estudios-Grado/>, 2024. Accedido 10-05-2025.
- [52] Adam Wathan, Jonathan Reinink, David Hemphill, and Steve Schoger. Tailwind css. <https://tailwindcss.com/>, 2024. Accedido 10-05-2025.