Facultade de Informática
UNIVERSIDADE DA CORUÑA

TRABALLO FIN DE GRAO
GRAO EN ENXEÑARÍA INFORMÁTICA
MENCIÓN EN TECNOLOXÍAS DA INFORMACIÓN

Deseño e Implementación dunha Aplicación Web para Formar Grupos de Estudo e Compartir Recursos

Estudante: Javier Darriba González

Dirección: Juan Raposo Santiago

A Coruña, September de 2023.

Dedicado a mi familia y amigos

Agradecimientos

Quiero agradecer primero a mi familia, siempre presente para brindarme apoyo. A mis amigos, que considero fundamentales para poder seguir creciendo y aprendiendo todos juntos. Y en especial, a mi hermana, quien siempre me ofrece su ayuda y ánimo, incluso en este trabajo para discutir lo que considera importante entorno a la educación, dada tu implicación en este ámbito, sé que serás la mejor profesora.

Resumen

En este proyecto se lleva a cabo el diseño e implementación de una aplicación web enfocada en la búsqueda y participación en grupos de estudio dentro de una comunidad educativa. El propósito principal de la aplicación es facilitar la colaboración y el intercambio de recursos e incentivar la comunicación en persona entre estos grupos programando reuniones, además de ofrecer herramientas de aprendizaje en conjunto.

La aplicación ofrece a los usuarios la búsqueda de universidades de su interés y, dentro de ellas, explorar las asignaturas disponibles y los grupos relacionados, permitiéndoles unirse a estos últimos después de iniciar sesión.

Los grupos pueden ser configurados como públicos o privados y cuentan con administradores responsables de su moderación. Dentro de cada grupo, los miembros tienen la capacidad de interactuar a través de mensajes, compartir recursos y programar reuniones mediante un calendario compartido.

En lo que respecta a los recursos compartidos en los grupos, se permite la subida y descarga de archivos en formato PDF, imágenes y videos. Además, la aplicación ofrece la posibilidad de crear y compartir exámenes en formato tipo test, encuestas y flashcards, tarjetas de aprendizaje que presentan breves preguntas y respuestas que facilitan la comprensión de conceptos clave.

Por último, hay que destacar que la aplicación está disponible en tres idiomas: castellano, gallego e inglés.

Abstract

This project entails the design and implementation of a web application focused on searching for and participating in study groups within an educational community. The main purpose of the application is to facilitate collaboration and resource sharing, as well as encourage in-person communication among these groups by scheduling meetings, while also providing collaborative learning tools.

The application offers users the ability to search for universities of their interest and, within those universities, explore available subjects and related study groups. Users can join these study groups after logging in.

Study groups can be configured as public or private and are managed by administrators responsible for moderation. Within each group, members have the ability to interact through messages, share resources, and schedule meetings using a shared calendar.

Regarding shared resources within the groups, users can upload and download files in PDF format, images, and videos. Additionally, the application provides the option to create and share exams in multiple-choice format, surveys, and flashcards—learning cards that present brief questions and answers to facilitate the understanding of key concepts.

Lastly, it's important to highlight that the application is available in three languages: Spanish, Galician, and English.

Palabras clave:

- Grupos de estudio
- Reuniones
- Flashcard
- Exámenes
- Encuestas
- Universidades
- Aplicación web

Keywords:

- Study groups
- Meetings
- Flashcard
- Exams
- Surveys
- Universities
- Web application

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introdución

1.1 Motivación

La socialización durante la etapa escolar adquiere una importancia crucial en el crecimiento y desarrollo de una persona. Esto se acentúa aún más en la etapa universitaria, donde para muchos estudiantes, implica mudarse a una nueva ciudad o incluso a un país diferente. Este periodo brinda una oportunidad única para crecer a nivel personal, experimentar otras culturas y forjar amistades duraderas.

Además, la colaboración con los iguales aporta un valor inmenso para los estudios en sí. Establecer una comunicación efectiva, no solo dentro de una clase, sino entre distintos niveles académicos, crea un ecosistema de aprendizaje vibrante. Interactuar con estudiantes de diferentes orígenes y trayectorias académicas enriquece las discusiones y la comprensión. Estas interacciones allanan el camino para la mentoría, ya que estudiantes mayores pueden brindar perspectivas y orientación basadas en sus experiencias.

Por ello, esta aplicación quiere dar la oportunidad de formar dichos grupos de estudio simplemente buscando tu universidad o materias en la aplicación para poder ponerse en contacto con personas que tienen ese mismo interés y ahí poder compartir recursos y organizar quedadas.

1.2 Objetivos

Los objetivos que perseguidos al desarrollar esta aplicación son los siguientes:

- Desarrollar una aplicación extensible y escalable aplicando buenas técnicas y patrones de diseño.
- Utilizar un subconjunto de tecnologías Java estándar y frameworks open source para la implementación del backend de la aplicación.

- Desarrollar el frontend de la aplicación con el lenguaje de Dart y más concretamente con Flutter Web que también es un frameworks open source desarrollado y compatible con Google.
- Trabajar con un enfoque de desarrollo iterativo e incremental
- Desarrollar una aplicación cuyas principales funcionalidades sean las siguientes:
  - Autenticación y Registro: Los usuarios pueden registrarse en la plataforma proporcionando información básica para su perfil y tienen la opcion de actualizarlo en caso de ser necesario. Además, debe haber un sistema de inicio y cierre de sesión para los usuarios registrados.
  - Exploración de Universidades y Asignaturas: Los usuarios pueden buscar y explorar universidades de su interés y las asignaturas que contiene cada una con su correspondiente información.
  - Creación y Búsqueda de Grupos de Estudio: Los usuarios pueden buscar grupos de estudio relacionados con asignaturas específicas o crear los suyos propios. Para poder unirse a grupos deben estar registrados en la aplicación y tener la sesión iniciada.
  - Gestión de Grupos: El creador del grupo es el encargado de su moderación, pudiendo expulsar a otros usuarios y añadir o quitar permisos de moderación al resto de miembros. Los moderadores también son los encagados de actualizar la información del grupo y de establacer si es público o privado.
  - Comunicación y Programaciñon de Reuniones: Dentro del grupo cualquier miembro puede enviar mensajes por el chat grupal y los moderadores pueden establecer reuniones en un calendario común indicando su fecha y dirección.
  - Gestión de recursos: Los usuarios registrados pueden subir a la aplicación archivos PDF, de imágenes y videos, además de, poder crear a través de herramientas de la propia aplicación encuestas, exámenes y flashcards. Todos estos recursos se pueden compartir en los grupos que seleccione su creador.
  - Soporte multilingüe: La aplicación está disponible en tres idiomas: castellano, gallego e inglés. El usuario puede seleccionar el idioma deseado en la página de inicio a través de un botón.

1.3 Estructura de la memoria

1. Introducción: Capítulo actual dónde se expone la idea central del proyecto junto con los objetivos que guían su desarrollo.

2. Estado del arte: Exploración y comparación de varias aplicaciones diseñadas para encontrar grupos de estudio y compartir recursos en el ámbito educativo.

3. Herramientas: Listado de herramientas y tecnologías usadas para el desarrollo del proyecto, incluyendo hardware, lenguajes de programación, librerías y bibliotecas que los complementan y los programas específicos.

4. Metodología: Explicación de la metodología ágil seleccionada y cómo se realizó su adaptación e implementación al proyecto.

5. Análisis: Capítulo dónde se exponen los actores principales, las áreas funcionales de la aplicación, las historias de usuario que sirvieron para su desarrollo y los sprints en los que se dividieron las historias de usuario.

6. Diseño: Explicación de la estructura de la aplicación y los patrones utilizados.

7. Implementación: Exposición de parte del código realizado y de las pruebas ejecutadas.

8. Planificación y evaluación de costes: Explicación del tiempo de desarrollo invertido y estimación del coste del proyecto

9. Conclusiones: Capítulo donde se evalúa la concordancia con los objetivos iniciales, se comentan las lecciones aprendidas y se detallan posibles líneas futuras de trabajo para el proyecto.

Capítulo 2

Estado del arte

Dada su importancia en la presente era digital, especialmente en el período posterior a la pandemia, se han desarrollado varias aplicaciones diseñadas para encontrar grupos de estudio y compartir recursos. Por lo tanto, en este capítulo se explorarán algunas de las aplicaciones más destacadas en este campo, analizando sus características, funcionalidades y popularidad entre los estudiantes.

2.1 StudyBlue[1]

Una de las pioneras en este ámbito, que ha ganado popularidad por su enfoque en las herramientas de aprendizaje personalizado. Permite a los estudiantes crear, compartir y acceder a flashcards, facilitando la revisión rápida de conceptos clave. Además, su algoritmo de aprendizaje personalizado ayuda a los usuarios a identificar las áreas que requieren más atención.

2.2 Quizlet[2]

Ofrece juegos educativos, flashcards, exámenes y ejercicios interactivos para mejorar la retención del conocimiento. Los estudiantes pueden crear sets de estudio personalizados o acceder a sets existentes compartidos por otros usuarios.

2.3 Microsoft Teams[3]

Plataforma de colaboración integral diseñada para grupos, equipos y organizaciones. Ofrece una amplia gama de herramientas que permiten la comunicación, la colaboración, la compartición de archivos y la organización en un entorno digital. Aunque no se enfoca exclusivamente en grupos de estudio, puede utilizarse de manera efectiva para esta finalidad.

2.4 Conclusión y tabla comparativa

Las aplicaciones diseñadas para facilitar la colaboración y el aprendizaje entre estudiantes han evolucionado significativamente en los últimos años. Desde enfoques centrados en el intercambio de archivos y la colaboración en línea hasta plataformas de comunicación versátiles, estas herramientas se han convertido en aliados indispensables para aquellos que buscan maximizar su rendimiento académico a través del trabajo en equipo y la interacción colaborativa. Cada una de estas aplicaciones presenta características únicas que se ajustan a diversas preferencias y necesidades, lo que subraya la importancia de elegir la opción más adecuada para cada estudiante.

Por último, se presenta una tabla que compara las funcionalidades de las aplicaciones nombreadas. Para cada una, se evalúa en términos de características como flashcards, chat grupal, intercambio de recursos y más, brindando una visión general de las capacidades que ofrecen para mejorar la experiencia de estudio colaborativo.

Funcionalidad	**StudyBlue**	**Quizlet**	**Microsoft Teams**
Flashcards	Sí	Sí	No
Grupos de estudio	No	No	Sí
Aprendizaje personalizado	Sí	No	No
Chat grupal	No	No	Sí
Intercambio de recursos	No	No	Sí
Juegos educativos	No	Sí	No
Plataforma en línea	Sí	Sí	Sí

Tabla 2.1: Tabla Comparativa de Funcionalidades

Capítulo 3

Herramientas tecnológicas

Este capítulo detalla y expone tanto el hardware como el software que se empleó en el proyecto, describiendo los diversos lenguajes, librerías y programas utilizados. Cada uno de estos componentes se presenta junto con su función específica dentro del proyecto, destacando cómo contribuyen a distintas partes y propósitos del mismo.

3.1 Hardware

Para realizar este proyecto se ha utilizado mi equipo personal que tiene las siguientes especificaciones:

- Marca y modelo: Lenovo ideapad Gaming 3
- Procesador: AMD Ryzen 7 4800H
- Memoria RAM: 16GB
- Almacenamiento: 500GB
- Tarjeta Gráfica: AMD Radeon Graphisc Processor
- Sistema Operativo: Microsoft Windows 11 Home

3.2 Lenguajes de programación

3.2.1 SQL

Lenguaje especializado en la administración de bases de datos que desempeñó un papel esencial en el modelado del almacenamiento y la recuperación de datos. A través de SQL, se diseñaron las tablas requeridas para diversas entidades, y al establecer relaciones entre estas tablas, se logró estructurar la información de manera coherente. Además, se desarrolló un

script que facilitó la inserción de datos iniciales, como el perfil del administrador, universidades, asignaturas y grupos de ejemplo. Esta estrategia evitó la necesidad de añadir manualmente todos los elementos cada vez que construía nuevas tablas y se recreaba la base de datos desde cero.

3.2.2 Java

Único lenguaje en la construcción del backend del proyecto. Desde la definición de las entidades que representan los datos en la base hasta el desarrollo de los controladores REST que permiten la interacción con el frontend, Java fue versátil, útil y sólido en todas las capas del backend.

3.2.3 Dart[4]

Lenguaje utilizado en el frontend. Desde la creación de interfaces visuales hasta la gestión de interacciones dinámicas, Dart demostró su eficacia en cada aspecto del diseño de la interfaz de usuario y la experiencia del usuario. Con su enfoque en la programación reactiva y la facilidad para construir componentes interactivos, Dart permitió crear una experiencia de usuario fluida y atractiva.

3.3 Frameworks y librerías

3.3.1 JPA

API utilizada para simplificar la interacción con la base de datos, definiendo las entidades y sus relaciones de manera intuitiva gracias a la estructura basada en clases y anotaciones. Esto permitió enfocar el diseño de datos de manera más natural, independientemente de la base de datos relacional que se utilice en concreto. En resumen, facilitó la gestión de la persistencia y contribuyó a un código más limpio, mantenible y legible.

Hibernate

Implementación de JPA que se encargó de traducir las entidades y sus relaciones en consultas SQL, ocultando gran parte de la complejidad del acceso a la base de datos. Su capacidad para generar automáticamente el esquema de la base de datos basado en las entidades aceleró el proceso de desarrollo. Combinar JPA y Hibernate resultó en un manejo eficiente y simplificado de la persistencia, permitiéndome centrarme en la funcionalidad principal del proyecto.

3.3.2 Spring Framework

Framework utlizado para crear una base sólida y coherente en la aplicación a través de la inversión de control y la inyección de dependencias simplificando la gestión de componentes y promoviendo la modularidad.

3.3.3 Spring Data

Permitió utilizar interfaces de repositorio y consultas basadas en métodos, pudiendo realizar operaciones de CRUD y consultas personalizadas de manera intuitiva y segura.

3.3.4 Spring Boot

Aceleró el proceso de desarrollo proporcionando un marco de trabajo con el que se pudo crear rápidamente una aplicación lista para ser desplegada sin perder tiempo en configuraciones complejas. La facilidad de inicio rápido y la gestión de dependencias simplificada fueron aspectos que más aportaron.

3.3.5 Spring Security

A través de una configuración personalizada, permitió gestionar el acceso y la protección de los recursos de la aplicación de manera efectiva. Entre lo más destacado se encuentra establecer reglas de autorización y autenticación para cada ruta y método de la aplicación permitiendo o restringiendo el acceso basado en roles y permisos definidos, implementación un filtro de autenticación personalizado que gestionaba la autenticación a través de tokens JWT y, por último, para garantizar la comunicación segura entre el frontend y el backend, configuré el manejo de solicitudes de diferentes orígenes mediante CORS.

3.3.6 JUnit

Librería utilizada para hacer las pruebas unitarias del proyecto a través de sus anotaciones y métodos de aserción.

3.3.7 Flutter Web[5]

Flutter Web es una extensión de Flutter, un framework de código abierto desarrollado por Google que permite crear aplicaciones nativas para múltiples plataformas desde un solo código base. En el proyecto, fue utilizado para desarrollar el frontend de la aplicación. A continuación, se resaltarán diversas características que condujeron a la elección de este framework.

- Hot Reload en Tiempo Real: La funcionalidad que permite realizar cambios en tiempo real mientras se desarrolla la aplicación. Esto agilizó el proceso de desarrollo y ofreciéndo la capacidad de ver instantáneamente cómo evoluciona la aplicación con cada modificación.
- Widgets Adaptativos: Se ajustan a diferentes tamaños de pantalla y dispositivos, lo que garantiza una experiencia de usuario consistente en diversos entornos.
- Compatibilidad con Navegadores: Compatible con varios navegadores web líderes, como Chrome, Firefox, Safari y Edge. Esto amplía el alcance de la aplicación, permitiendo que llegue a una audiencia más amplia.
- Rendimiento Avanzado: Impulsado por la potencia de WebAssembly, logra un rendimiento excepcionalmente rápido y proporciona una experiencia fluida incluso en navegadores modernos.

Por último, comentar algunas de las bibliotecas y frameworks que se utilizaron junto a flutter web:

Provider

Biblioteca para la gestión de estado. Utilizada en el proyecto para administrar y propagar cambios en el estado a lo largo de la aplicación de forma eficiente. Se basa en un patrón de diseño llamado ”InheritedWidget” y es especialmente útil para evitar problemas de rendimiento y complejidad en la gestión de estados.

Get

Paquete que proporciona una forma eficiente de manejar la internacionalización de la aplicación haciendo uso de una clase abstracta llamada Translations.

Cupertino icons

Ofrece un conjunto de íconos de estilo Cupertino, que es el estilo de diseño utilizado en iOS, haciendo que la aplicación web tenga un aspecto nativo y coherente en dispositivos Apple.

Http

Biblioteca fundamental para interactuar con servicios web, obtener y enviar datos, y realizar operaciones como la autenticación y la recuperación de información en línea.

Table calendar

Biblioteca que permitió crear calendarios personalizables para cada grupo indicando que días hay reuniones y cuantas de ellas hay.

Cached network image

Biblioteca utilizada para la carga y el almacenamiento en caché de imágenes como, por ejemplo, las posibles imágenes en las flashcards.

3.4 Herramientas utilizadas para el desarrollo

3.4.1 MySQL

Sistema de gestión de bases de datos relacionales de código abierto desplegada en la aplicación web. Fue elegida al propio conocimiento y experiencia previa en su uso, aprovechando su capacidad para gestionar eficientemente bases de datos relacionales, se diseñó y construyó la estructura de la base de datos, implementando índices para mejorar la eficiencia y el rendimiento en el acceso y búsqueda de datos. Además, en el caso de que se llegase a ampliar la aplicación, el gestor ofrece una gran escalabilidad y flexibilidad.

3.4.2 IntelliJ IDEA

IntelliJ IDEA, creado por JetBrains, es un entorno de desarrollo que utilizado en el desarrollo del backend en Java. Este IDE destacó por su capacidad de personalización, lo que resultó en una experiencia de desarrollo muy cómoda y agradable. Gracias a su interfaz amigable y a la experiencia previa con la herramienta, fue posible comenzar rápidamente con la codificación del backend.

Además, cabe mencionar que tiene una gran variedad de plugins como, por ejemplo, Rainbow Brackets, para mejorar la legibilidad del código al resaltar las estructuras de paréntesis y corchetes, y Wakanda Time, que registró el tiempo que invertido en el proyecto, permitiéndo visualizarlo en un panel de control.

3.4.3 Apache Maven

Herramienta de gestión y construcción de proyectos software utilizada junto a Java para orquestar de manera eficiente las dependencias del proyecto y los módulos externos. Mediante el archivo POM, fue posible definir las dependencias necesarias y establecer la estructura del proyecto. Apache Maven, con su enfoque en la automatización y la coherencia, se encargó de

descargar y gestionar las bibliotecas externas requeridas, lo que agilizó el proceso de desarrollo y evitó problemas de compatibilidad.

3.4.4 Android studio

Es el entorno de desarrollo oficial de Google utilizado para desarrollar el frontend de la página web usando Flutter web. Al igual que IntelliJ, está equipado con una interfaz intuitiva y brinda una gama completa de herramientas esenciales para desarrolladores.

3.4.5 GitHub

Plataforma de desarrollo colaborativo que se basa en el sistema de control de versiones Git. Se utilizó esta plataforma como repositorio centralizado para alojar y gestionar el proyecto software. Entre sus características clave están el control de versiones, para rastrear y administrar cambios en el código, y los repositorios que permitieron almacenar archivos y gestionar diferentes ramas de desarrollo.

3.4.6 Postman

Plataforma que se utilizó para realizar pruebas a la API REST creada en el proyecto. A través de su interfaz intuitiva, se enviaron solicitudes HTTP a las diferentes rutas y endpoints de la API, lo que nos permitió verificar el comportamiento de los métodos GET, POST y PUT.

3.4.7 Balsamiq Wireframes[6]

Herramienta de diseño de prototipos utilizada en el proyecto para crear bocetos de la interfaz de usuario. Proporciona una forma rápida y sencilla de visualizar la disposición y estructura de las pantallas y elementos de la aplicación. Sus características incluyen una amplia variedad de componentes de interfaz predefinidos y una interfaz de usuario amigable que permite la creación de representaciones visuales de la aplicación antes de la implementación real.

Capítulo 4

Metodología

Dado el carácter dinámico, cambiante y colaborativo del mundo del desarrollo software, las metodologías ágiles han emergido como un enfoque revolucionario para abordar los desafios de este entorno. La aportación de estas metodologías consiste en centrarse en la adaptabilidad, la colaboración y la entrega continua de valor al cliente.

Son muchas las metodologías desarrolladas y estudiadas para estos entornos, pero dado las características de este proyecto, que es unipersonal, no existe un cliente real y no se pudo establecer un horario fijo durante el tiempo en el que se estuvo desarrollando el trabajo debido a diferentes situaciones personales, la metodología por la que se optó fue la Scrumban[7], que es una fusión de dos metodologías líderes, Scrum[8] y Kanban[9], de forma que pude aprovechar principios y técnicas de ambas de forma muy personalizada y adaptada al proyecto.

4.1 Metodología Scrumban[7]

Metodología híbrida que combina elementos de dos metodologías ágiles populares: Scrum[8] y Kanban[9]. Esta combinación tiene como objetivo aprovechar lo mejor de ambos enfoques para adaptarse a las necesidades específicas de un equipo o proyecto.

4.1.1 Scrum[8]

Se basa en la organización del trabajo en sprints, siendo estos periodos fijos en el tiempo, donde en cada iteración el equipo se compromete a entregar un conjunto de funcionalidades prioritarias de las que se encuentran en el backlog. Durante el sprint se realizan breves reuniones diarias para que el equipo informe sobre su avance, obstáculos y metas del día. Una vez finalizado el sprint, se realiza una revisión del mismo para demostrar el trabajo completado y recibir los comentarios del cliente o responsable, además de reflexionar sobre qué funcionó bien y cómo puede mejorarse el siguiente sprint. Establece tres roles importantes:

- Product Owner: Responsable de definir y priorizar la lista de funcionalidades y requisitos que el equipo de desarrollo debe implementar para considerar finalizado el producto. Este rol está para asegurarse de que el trabajo realizado proporcione el máximo valor al negocio y al cliente.
- Scrum Master: Actúa como facilitador y defensor del equipo Scrum. Su función principal es asegurarse de que se sigan correctamente las prácticas y principios de Scrum, además de fomentar la mejora continua ayudando a crear un entorno en el equipo en el que se pueda colaborar y trabajar eficientemente.
- Equipo de Desarrollo: Grupo de profesionales que lleva a cabo el trabajo de desarrollo de las tareas durante cada sprint. Puede incluir programadores, diseñadores, analistas y cualquier otro miembro necesario para crear el producto. El equipo de desarrollo es autoorganizado y se encarga de estimar, planificar y completar las tareas necesarias para entregar el trabajo durante el sprint.

Por último, mencionar varios artefactos característicos de esta metodología que se tuvo en cuenta:

- Product Backlog: Lista priorizada de todas las funcionalidades, características y requisitos que deben ser implementados en el producto para considerarse finalizado.
- Sprint Backlog: Lista de tareas y elementos del backlog del producto seleccionados para ser desarrollados durante el sprint actual.
- Historias de usuario: Descripciones concisas de una funcionalidad o característica desde la perspectiva del usuario final. Contiene información sobre quién es el usuario y qué necesita lograr.

4.1.2 Kanban[9]

Basada en la entrega continua de valor al cliente a través de la visualización de tareas en un determinado flujo de trabajo. Para ello hace uso de los siguientes artefactos:

- Tablero Kanban: Tablero visual dividido en columnas que representan diferentes etapas del flujo de trabajo. Cada columna puede representar desde tareas pendientes hasta tareas completadas.
- Tareas: Unidades individuales que describen un objetivo concreto y que se utilizan para mover de un lado a otro del tablero para indicar el progreso de la tarea.

- Limitación del Trabajo en Progreso: Límite de tareas que pueden estar en una columna del Tablero, previniendo así un exceso de trabajo en curso y ayudando a mantener un flujo de entrega de trabajo constante.
- Colaboración y Comunicación: El tablero Kanban proporciona de forma transparente información el progreso y estado de las tareas en tiempo real de todos los miembros del equipo.
- Enfoque en el Valor: Se priorizan las tareas en función del valor que aportan al cliente, lo que asegura que se aborden primero las tareas más importantes.

4.2 Adaptación de la metodología Scrumban a mi proyecto

A continuación, se ennumeran y explican las diferentes técnicas y principios que aplicados de esta combinación:

- Realización y priorización de historias de usuario: Durante una etapa de análisis inicial del proyecto se redactaron lo que serían las historias de usuario para tener en cuenta la perspectiva del cliente ficticio y se comunicaron al tutor para que pudiera revisarlas y comentar cualquier cambio necesario.
- Planificación temporal del trabajo en nueve sprints que abarcan todas las historias de usuario descritas.
- Creación de tareas a partir de las historias de usuario para, posteriormente, utilizarlas en el tablero.
- Tablero Kanban: Utilizado para visualizar las tareas en el flujo de trabajo divididas en cinco columnas:
  - PRODUCT BACKLOG: Reserva ordenada por prioridades de todas las tareas que faltan por hacer para terminar el proyecto.
  - SPRINT BACKLOG: Lista de tareas a terminar antes de pasar a otra parte del proyecto y, por lo tanto, añadir más tareas desde el 'product BACKLOG'.
  - IN PROGRESS: Tareas que están en ese intante en desarrollo.
  - TESTING: Tareas que están finalizadas, pero necesitan realizarse unos test unitarios para comprobar que funcionen correctamente.
  - DONE: Lista de tareas ya finalizadas.

- Limitación del trabajo en progreso: Al pasar las tareas del 'PRODUCT BACKLOG' a la lista de 'SPRINT BACKLOG', se estableció de forma lógica un número de elementos máximos para no acumular tareas por pendientes y, así, centrar la atención únicamente en las pocas tareas que a terminar para poder continuar.

Capítulo 5
Análisis

En este capítulo se exponen los actores principales que hacen uso de la aplicación, las áreas funcionales que la componen, las historias de usuario que sirvieron para su desarrollo y los sprints en los que se dividieron las historias de usuario.

5.1 Actores del sistema

5.1.1 Usuario anónimo

Se refiere a cualquier persona que accede a la aplicación web sin haber iniciado sesión. Las acciones disponibles para este tipo de usuario están limitadas a explorar la plataforma. Pueden realizar búsquedas de universidades y asignaturas, accediendo a la información detallada proporcionada sobre ellas. También pueden visualizar una lista de grupos asociados a dichas asignaturas. Sin embargo, la información visible en relación a los grupos se limita al nombre y el tipo de privacidad del grupo. Para acceder a funcionalidades adicionales y a los contenidos completos de los grupos, los usuarios anónimos deberán registrarse o iniciar sesión en caso de tener una cuenta previamente creada.

5.1.2 Usuario registrado

Este tipo de usuario, además de las capacidades disponibles para el usuario anónimo, puede unirse y participar activamente en grupos, enviar mensajes, ver reuniones planificadas, explorar asignaturas enlazadas, visualizar los diferentes usuarios del grupo, acceder a los recursos compartidos por otros miembros y compartir sus propios recursos.

A mayores de estas funciones, este usuario tiene la posibilidad de crear su propio grupo y asumir el rol de moderador. Y como tal, puede gestionar el grupo, incluyendo la expulsión de usuarios, asignación y destitución del rol de moderador a otros miembros, el establecimiento y eliminación de enlaces entre las asignaturas y el grupo y la creación de reuniones en el

calendario común.

Una característica adicional es la sección de recursos personales, donde este usuario puede diseñar exámenes tipo test, encuestas y flashcards, posteriormente compartiéndolos con otros miembros para que puedan visualizarlos y realizarlos. También tiene la opción de subir archivos en formato PDF, imágenes o videos alojados en la web, proporcionando la URL correspondiente. Estas acciones permiten al usuario registrado contribuir de manera activa y enriquecer la experiencia colaborativa en la aplicación.

5.1.3 Administrador de la aplicación web

Rol con mayores privilegios en la aplicación web. Además de tener acceso a todas las funciones disponibles para los usuarios con una sesión iniciada, el administrador tiene la capacidad adicional de gestionar la estructura educativa de la plataforma. Esto incluye la capacidad de añadir, modificar y eliminar universidades y asignaturas en la aplicación.

El administrador puede contribuir al enriquecimiento de la base de datos de la plataforma, asegurándose de que la información sobre las universidades y asignaturas esté actualizada. Esta función es esencial para garantizar que los usuarios tengan acceso a información relevante y actualizada sobre las instituciones educativas y las materias ofrecidas.

5.2 Subsistemas

Se han definido cuatro áreas distintas en la aplicación web, cada una de las cuales engloba funcionalidades específicas y que han permitido categorizar de manera más precisa las historias de usuario:

5.2.1 Gestión de Usuarios

Este subsistema se centra en las operaciones relacionadas con la gestión de usuarios en la plataforma. Comprende diversas funcionalidades, como el registro de nuevos usuarios, el inicio de sesión en cuentas existentes, la actualización de los perfiles de usuario, la finalización de la sesión y la eliminación de cuentas. Cada una de estas características contribuye a proporcionar a los usuarios un control total sobre sus cuentas y la información asociada.

5.2.2 Gestión de Universidades

Este subsistema se enfoca en la administración integral de las universidades y las asignaturas dentro de la plataforma. Proporciona al administrador la capacidad de realizar operaciones esenciales como la creación, actualización y eliminación de universidades y sus respectivas

asignaturas. Además, brinda a los usuarios la funcionalidad de búsqueda de universidades, lo que les permite explorar la oferta educativa disponible en la aplicación.

La gestión de universidades abarca la creación y mantenimiento de perfiles detallados de cada universidad, incluyendo información relevante como su nombre, ubicación y descripción. Asimismo, permite agregar, editar y eliminar asignaturas dentro de cada universidad, asegurando que la información sobre los programas de estudio esté siempre actualizada y precisa.

En este área, los usuarios pueden explorar las universidades y las asignaturas que estas ofrecen. Esto facilita la búsqueda de instituciones educativas y la identificación de las asignaturas que se alinean con los intereses académicos de los usuarios.

5.2.3 Gestión de Grupos

Este subsistema está dedicado a la gestión completa de grupos dentro de la plataforma. Los usuarios tienen la capacidad de crear sus propios grupos, así como unirse a otros existentes a través de la búsqueda basada en universidades y asignaturas. Además, pueden actualizar su membresía en los grupos, lo que incluye unirse, salir o incluso asumir roles de moderador en un grupo.

La funcionalidad de gestión de grupos brinda a los usuarios la oportunidad de colaborar y comunicarse de manera efectiva con los demás miembros del grupo. En este contexto, pueden compartir mensajes, acceder a reuniones planificadas, ver la lista completa de miembros y explorar los recursos compartidos. Además, tienen la capacidad de compartir sus propios recursos y eliminar aquellos que hayan compartido previamente. Los usuarios con roles de moderador también ejercen autoridad sobre el grupo, pudiendo expulsar miembros, gestionar roles de moderador, vincular asignaturas, programar reuniones y actualizar la información y privacidad del grupo según sea necesario.

Este área fomenta la interacción y la colaboración entre los usuarios, brindándoles un espacio en el cual puedan trabajar juntos en proyectos, intercambiar información y compartir recursos.

5.2.4 Gestión de Recursos

Abarca el área dónde los usuarios registrados tienen la capacidad de crear diversos tipos de recursos, como exámenes, encuestas y flashcards, para enriquecer y poner a prueba el aprendizaje de sus compañeros compartiéndolos con otros usuarios, fomentando la colaboración y el intercambio de conocimientos.

Además, los usuarios también pueden subir archivos en formatos como PDF, imágenes o videos, los cuales se almacenan en la aplicación y pueden ser accedidos posteriormente. Esta

funcionalidad facilita el acceso a materiales de estudio y recursos educativos, contribuyendo aún más a un proceso de aprendizaje.

Asimismo, la gestión de recursos abarca la administración de los recursos compartidos. Los usuarios pueden editar y actualizar sus recursos existentes, así como eliminarlos si ya no son relevantes. Esta característica brinda flexibilidad y control sobre el contenido compartido, asegurando que siempre se disponga de información actualizada y pertinente.

5.3 Historias de usuario

A continuación, se presenta una tabla que resume los requisitos funcionales de la aplicación a través de historias de usuario. Estas historias están identificadas y categorizadas según las áreas de usuarios (US), universidades (UN), grupos (GR) y recursos (RC).

Tabla 5.1: Tabla compuesta de las historias de usuario

ID	Área	Funcionalidad
1	US	Como usuario anónimo, quiero registrarme proporcionando mi nombre, apellidos, correo y ubicación, ésta última de manera opcional.
2	US	Como usuario anónimo en posesión de una cuenta, quiero autenticarme en la aplicación.
3	US	Como usuario registrado, quiero cerrar sesión en la aplicación.
4	US	Como usuario registrado, quiero modificar la información de registro.
5	US	Como usuario registrado, quiero poder eliminar mi cuenta una vez no quiera seguir usándola.
6	UN	Como administrador, quiero añadir universidades indicando su nombre y localización.
7	UN	Como administrador, quiero poder modificar el nombre y localización de una universidad.

. . . . . *(continúa en la página siguiente)* . . . . . .

Tabla 5.1 – *(viene de la página anterior)*

ID	Área	Funcionalidad
8	UN	Como administrador de la aplicación quiero poder eliminar una universidad, lo que implicaría también eliminar todas las asignaturas que le pertenecen y desligar los grupos a dichas asignaturas.
9	UN	Como administrador, quiero añadir asignaturas indicando su nombre, código, facultad y curso.
10	UN	Como administrador, quiero poder modificar el nombre, código, facultad y curso de las asignaturas.
11	UN	Como administrador, quiero poder eliminar una asignatura lo que implicaría desligar los grupos asociados a ella.
12	UN	Como usuario registrado, quiero buscar por su nombre las diferentes universidades dadas de alta en la aplicación
13	UN	Como usuario registrado, quiero buscar a través de la localización las diferentes universidades dadas de alta en la aplicación.
14	UN	Como usuario registrado, quiero buscar las diferentes asignaturas dentro de una universidad.
15	GR	Como usuario registrado, quiero poder crear un grupo indicando su nombre, privacidad, y, opcionalmente, descripción.
16	GR	Como usuario registrado, quiero buscar los grupos existen en cada asignatura.
17	GR	Como usuario registrado, quiero poder unirme a un grupo que haya encontrado en una universidad y asignatura concreta
18	GR	Como usuario registrado que es parte de un grupo quiero poder salir de él.

. . . . . *(continúa en la página siguiente)* . . . . . .

Tabla 5.1 – *(viene de la página anterior)*

ID	Área	Funcionalidad
19	GR	Como moderador de un grupo, quiero poder eliminar a los participantes de un grupo que no se comporten.
20	GR	Como moderador de un grupo, quiero convertir en moderadores a los usuarios que considere.
21	GR	Como moderador de un grupo, quiero poder actualizar el nombre, descripción, privacidad y contraseña del grupo.
22	GR	Como moderador de un grupo, quiero poder asociar mi grupo a una o más asignaturas indicando el año y cuatrimestre para el que se liga dicho grupo a la asignatura.
23	GR	Como moderador de un grupo, quiero poder desasociar mi grupo a una o más asignaturas.
24	GR	Como moderador de un grupo, quiero crear reuniones indicando su descripción, fecha y lugar.
25	GR	Como moderador de un grupo, quiero poder modificar la descripción, fecha y dirección de una reunión.
26	GR	Como moderador de un grupo, quiero poder cancelar una reunión.
27	GR	Como usuario que forma parte de un grupo en el que se organizó al menos una reunión, quiero poder ver una lista de las reuniones programadas.
28	GR	Cómo usuario que forma parte de un grupo en el que se organizó una reunión, quiero poder confirmar si asisto o no.
29	GR	Como usuario que forma parte de un grupo, quiero enviar mensajes a través de un grupo.

. . . . . *(continúa en la página siguiente)* . . . . . .

Tabla 5.1 – *(viene de la página anterior)*

ID	Área	Funcionalidad
30	GR	Como usuario que forma parte de un grupo, quiero leer los mensajes que envían el resto de los participantes del grupo.
31	RC	Como usuario registrado, quiero subir al servidor un PDF, imagen o vídeo indicando su nombre y descripción.
32	RC	Como usuario registrado, quiero crear una flashcard proporcionando su nombre, pregunta, respuesta y, opcionalmente, una explicación e imagen.
33	RC	Como usuario registrado, quiero crear un examen proporcionando su nombre, tiempo límite, número máximo de intentos, las preguntas y posibles repuestas, indicando si la respuesta es la correcta o no.
34	RC	Como usuario registrado, quiero crear una encuesta indicando la pregunta y las posibles opciones.
35	RC	Como usuario creador de una flashcard quiero poder modificar su nombre, pregunta, respuesta y explicación.
36	RC	Como usuario que forma parte de un grupo en el que un participante compartió un PDF, imagen o video, quiero poder descargarlos.
37	RC	Como usuario que forma parte de un grupo en el que un participante compartió un PDF, imagen, video o flashcard, quiero poder visualizarlo.
38	RC	Como usuario que forma parte de un grupo en el que un participante compartió una encuesta, quiero poder responderla.
39	RC	Como usuario que forma parte de un grupo en el que un participante compartió una encuesta, quiero poder ver sus resultados actuales.

. . . . . *(continúa en la página siguiente)* . . . . . .

Tabla 5.1 – *(viene de la página anterior)*

ID	Área	Funcionalidad
40	RC	Como usuario que respondió a una encuesta, quiero poder cambiar la selección que realicé.
41	RC	Como usuario que forma parte de un grupo en el que un participante compartió un examen, quiero poder responderlo y conocer la nota resultante.
42	RC	Como usuario que realizó un examen compartido en un grupo, quiero poder revisar mis intentos en dicho examen.
43	RC	Como usuario creador de un examen compartido a un grupo en el que otros usuarios realizaron el examen, quiero poder acceder a los diferentes intentos de cada usuario visualizando su nota correspondiente y las respuestas que seleccionaron para cada pregunta.

5.4 Sprints

La planificación de los sprints para el desarrollo se llevó a cabo con la intención de optimizar la eficiencia y funcionalidad del producto final. Uno de los factores clave en esta planificación fue la priorización de tareas, que permitiría construir una base sólida para desarrollar funcionalidades más complejas de manera coherente.

En el proceso de priorización, se identificaron las tareas necesarias para poder luego implementar otras. Un ejemplo claro de esto fue la decisión de priorizar el área de ”Gestión de Usuarios” ante las demás, ya que para poder hacer cualquier otra acción en la aplicación web necesitaba poder tener un usuario identificado, como es el caso del administrador de la aplicación para cuyo rol me permitiría luego añadir, actualizar y eliminar las universidades y asignaturas del catálogo. Este mismo motivo también justifica la elección de priorizar la ”Gestión de Universidades” antes que la ”Gestión de Grupos” debido a que la capacidad de buscar universidades y asignaturas era fundamental para permitir a los usuarios acceder a la información académica relevante y posteriormente poder unirse a grupos que estuvieran asociados a esas asignaturas. Sin esta funcionalidad, los usuarios no podrían encontrar los grupos en los cuales desearían participar.

Además, considerando la naturaleza social de la aplicación, se optó por dar prioridad al área de ”Grupos” antes que al área de ”Recursos”, debido a que el intercambio de información

y la comunicación entre los usuarios en grupos eran elementos esenciales para el éxito de la plataforma. La idea de crear, unirse y participar en grupos es el aspecto sobre el que se basa la aplicación en sí.

Por último, para asegurar un enfoque coherente y eficiente, se optó agrupar las funcionalidades por temáticas. Esto posibilitó la concentración en tareas relacionadas y evitar perder tiempo al cambiar constantemente de ámbito maximizando así la productividad del sprint.

5.4.1 Sprint 1

En este sprint se consideraron todas las funcionalidades relacionadas con la gestión de usuarios, ya que se consideraron esenciales para poder construir el resto de la aplicación, siendo en total las cinco que se presentan a continuación:

ID	Área	Funcionalidad
1	US	Como usuario anónimo, quiero registrarme proporcionando mi nombre, apellidos, correo y ubicación, ésta última de manera opcional.
2	US	Como usuario anónimo en posesión de una cuenta, quiero autenticarme en la aplicación.
3	US	Como usuario registrado, quiero cerrar sesión en la aplicación.
4	US	Como usuario registrado, quiero modificar la información de registro.
5	US	Como usuario registrado, quiero poder eliminar mi cuenta una vez no quiera seguir usándola.

Tabla 5.2: Sprint 1 - Gestión de Usuarios

5.4.2 Sprint 2

En este segundo sprint, la atención se centró exclusivamente en el módulo de universidades. El objetivo principal fue permitir al administrador realizar operaciones fundamentales, como la creación, modificación y eliminación de universidades. Además, se trabajó en la implementación de la funcionalidad de búsqueda, que permitía a los usuarios buscar universidades por su nombre o ubicación directamente desde la pantalla inicial de la aplicación.

ID	Área	Funcionalidad
6	UN	Como administrador, quiero añadir universidades indicando su nombre y localización.
7	UN	Como administrador, quiero poder modificar el nombre y localización de una universidad.
8	UN	Como administrador de la aplicación quiero poder eliminar una universidad, lo que implicaría también eliminar todas las asignaturas que le pertenecen y desligar los grupos a dichas asignaturas.
12	UN	Como usuario registrado, quiero buscar por su nombre las diferentes universidades dadas de alta en la aplicación
13	UN	Como usuario registrado, quiero por la localización las diferentes universidades dadas de alta en la aplicación.

Tabla 5.3: Sprint 2 - Gestión de Universidades

5.4.3 Sprint 3

El tercer sprint significó la conclusión del área de ”Gestión de Universidades” y se centró en las funcionalidades relacionadas con las asignaturas. Se trabajó en permitir que el administrador pudiera añadir, modificar y eliminar las asignaturas de la aplicación.

Además, se implementó la capacidad para que cualquier usuario, al acceder a una universidad, pudiera visualizar todas las asignaturas asociadas a esa institución. Se incorporó un sistema de filtros que permitía a los usuarios buscar asignaturas por diversos parámetros, como nombre, código, curso y facultad.

ID	Área	Funcionalidad
9	UN	Como administrador, quiero añadir asignaturas indicando su nombre, código, facultad y curso.
10	UN	Como administrador, quiero poder modificar el nombre, código, facultad y curso de las asignaturas.
11	UN	Como administrador, quiero poder eliminar una asignatura lo que implicaría desligar los grupos asociados a ella.
14	UN	Como usuario registrado, quiero buscar las diferentes asignaturas dentro de una universidad.

Tabla 5.4: Sprint 3 - Gestión de Asignaturas

5.4.4 Sprint 4

El cuarto sprint se centró en la implementación de las funcionalidades relacionadas con ”Gestión de Grupos” en la plataforma. Durante esta iteración, se permitió a los usuarios registrados crear grupos, proporcionando información como el nombre y la privacidad del grupo. Además, se habilitó la capacidad de modificar la información de los grupos existentes.

Además, como el creador del grupo está directamente designado como su moderador, también implementé la capacidad de asociar o desligar del grupo las asignaturas deseadas y, por lo tanto, también su visualización y búsqueda desde dichas asignaturas por parte de cualquier usuario.

ID	Área	Funcionalidad
15	GR	Como usuario registrado, quiero poder crear un grupo indicando su nombre, privacidad, y, opcionalmente, descripción.
16	GR	Como usuario registrado, quiero buscar los grupos existen en cada asignatura.
21	GR	Como moderador de un grupo, quiero poder actualizar el nombre, descripción, privacidad y contraseña del grupo.
22	GR	Como moderador de un grupo, quiero poder asociar mi grupo a una o más asignaturas indicando el año y cuatrimestre para el que se liga dicho grupo a la asignatura.
23	GR	Como moderador de un grupo, quiero poder desasociar mi grupo a una o más asignaturas.

Tabla 5.5: Sprint 4 - Gestión de Grupos

5.4.5 Sprint 5

En el quinto sprint, la intención fue enriquecer las funcionalidades de los grupos permitiendo a los usuarios unirse y salir de ellos, además de poder enviar mensajes para que pudieran comunicarse. Y a la vez que se pudo tener varios participantes formando el grupo se les dio a todos la opción de ver la lista de perfiles que lo conforman. Por último, a los administradores se les otorgó la capacidad de eliminar a los usuarios que ellos consideren, y se les permitió asignar o retirar el rol de moderador a los demás usuarios del grupo.

ID	Área	Funcionalidad
17	GR	Como usuario registrado, quiero poder unirme a un grupo que haya encontrado en una universidad y asignatura concreta
18	GR	Como usuario registrado que es parte de un grupo quiero poder salir de él.
19	GR	Como moderador de un grupo, quiero poder eliminar a los participantes de un grupo que no se comporten.
20	GR	Como moderador de un grupo, quiero convertir en moderadores a los usuarios que considere.
29	GR	Como usuario que forma parte de un grupo, quiero enviar mensajes a través de un grupo.
30	GR	Como usuario que forma parte de un grupo, quiero leer los mensajes que envían el resto de los participantes del grupo.

Tabla 5.6: Sprint 5 - Gestión de Usuarios en los Grupos

5.4.6 Sprint 6

En el sexto sprint, la focalización estuvo en la gestión de las reuniones dentro de los grupos. Se permitió a los moderadores del grupo la capacidad de crear, modificar y cancelar reuniones. A su vez, se brindó a todos los usuarios pertenecientes al grupo la posibilidad de visualizar estas reuniones, acompañadas de un calendario grupal donde se señalan las fechas en las que están programadas.

ID	Área	Funcionalidad
24	GR	Como moderador de un grupo, quiero crear reuniones indicando su descripción, fecha y lugar.
25	GR	Como moderador de un grupo, quiero poder modificar su descripción, fecha y dirección de una reunión.
26	GR	Como moderador de un grupo, quiero poder cancelar una reunión.
27	GR	Como usuario que forma parte de un grupo en el que se organizó al menos una reunión, quiero poder ver una lista de las reuniones programadas.
28	GR	Como usuario que forma parte de un grupo en el que se organizó una reunión, quiero poder confirmar si asisto o no.

Tabla 5.7: Sprint 6 - Gestión de Reuniones

5.4.7 Sprint 7

Durante el séptimo sprint, se inició la labor en la gestión de recursos, lo que posibilitó a cualquier usuario registrado subir archivos PDF, imágenes y videos a la aplicación web, ofreciendo así un espacio para alojar diferentes tipos de contenido. Además, también se incorporó la funcionalidad de crear y editar flashcards.

En este mismo sprint, se implementó la opción de compartir estos recursos con los grupos a los que pertenecían el usuario. Habilitándose la capacidad de visualizar los PDFs, imágenes, videos y flashcards compartidos, y se brindó la opción de descargar los PDFs, imágenes y videos.

ID	Área	Funcionalidad
31	RC	Como usuario registrado, quiero subir al servidor un PDF, imagen o vídeo indicando su nombre y descripción.
32	RC	Como usuario registrado, quiero crear una flashcard proporcionando su nombre, pregunta, respuesta y, opcionalmente, una explicación e imagen.
35	RC	Como usuario creador de una flashcard quiero poder modificar su nombre, pregunta, respuesta y explicación.
36	RC	Como usuario que forma parte de un grupo en el que un participante compartió un PDF, imagen o video, quiero poder descargarlos.
37	RC	Como usuario que forma parte de un grupo en el que un participante compartió un PDF, imagen, video o flashcard, quiero poder visualizarlo.

Tabla 5.8: Sprint 7 - Gestión de Recursos Web y Flashcards

5.4.8 Sprint 8

Durante el octavo sprint, se avanzó hacia la gestión de encuestas, brindando a todos los usuarios registrados la capacidad de crear sus propias encuestas. Esta función permite a los usuarios formular preguntas y definir las posibles respuestas para cada una.

Además, se implementó la opción de compartirlas en los grupos a los que pertenece el usuario, lo que permite a los demás participantes del grupo votar en ellas. Los usuarios tienen la flexibilidad de cambiar su voto en cualquier momento durante y de visualizar los resultados actuales de la encuesta en tiempo real, lo que proporciona a los usuarios una visión clara de cómo están evolucionando las respuestas.

ID	Área	Funcionalidad
34	RC	Como usuario registrado, quiero crear una encuesta indicando la pregunta y las posibles opciones.
38	RC	Como usuario que forma parte de un grupo en el que un participante compartió una encuesta, quiero poder responderla.
39	RC	Como usuario que forma parte de un grupo en el que un participante compartió una encuesta, quiero poder ver sus resultados actuales.
40	RC	Como usuario que respondió a una encuesta, quiero poder cambiar la selección que realicé.

Tabla 5.9: Sprint 8 - Gestión de Encuestas

5.4.9 Sprint 9

En el noveno y último sprint, se culminó la implementación de la aplicación con la incorporación de exámenes tipo test. Esto permitió a cualquier usuario registrado crear exámenes, configurando las preguntas, las posibles respuestas y especificando la respuesta correcta. Además, se agregó la opción de establecer un número máximo de intentos y un límite de tiempo para completar el examen.

Una vez creado el examen, los usuarios tienen la capacidad de compartirlo con grupos específicos, dando la oportunidad a otros miembros del grupo de realizar el examen y conocer su calificación inmediatamente después de finalizarlo. Además, se habilitó la opción de revisar los intentos tanto para el propio usuario (que realizó el examen) como para el creador del documento, quien puede ver los intentos de todos los usuarios que participaron en su examen.

ID	Área	Funcionalidad
33	RC	Como usuario registrado, quiero crear un examen proporcionando su nombre, tiempo límite, número máximo de intentos, las preguntas y posibles repuestas, indicando si la respuesta es la correcta o no.
41	RC	Como usuario que forma parte de un grupo en el que un participante compartió un examen, quiero poder responderlo y conocer la nota resultante.
42	RC	Como usuario que realizó un examen compartido en un grupo, quiero poder revisar mis intentos en dicho examen.
43	RC	Como usuario creador de un examen compartido a un grupo en el que otros usuarios realizaron el examen, quiero poder acceder a los diferentes intentos de cada usuario visualizando su nota correspondiente y las respuestas que seleccionaron para cada pregunta.

Tabla 5.10: Sprint 9 - Gestión de Exámenes

Capítulo 6

Diseño

Tras realizar una fase de análisis, se inció una etapa de diseño en la cual se determinaron los patrones fundamentales que guiaron el desarrollo y se estableció la arquitectura del proyecto, tanto la de la base de datos y backend como la disposición en el frontend.

6.1 Patrón de diseño utilizados

Un patrón de diseño de software es una solución probada para problemas comunes. Ofrecen enfoques reutilizables y optimizados para resolver situaciones similares, mejorando la eficiencia y coherencia del desarrollo.

6.1.1 Patrón DAO

Centrado en separar la lógica de acceso a los datos de la lógica de negocio en una aplicación. Proporciona una abstracción en la forma en que los datos se almacenan y recuperan, esto permite que los cambios en la base de datos no afecten directamente la lógica de negocio, ayudando a que el código de la lógica de negocio sea más limpio y legible. Además, al utilizar DAO, puedes centralizar la lógica de acceso a los datos en un lugar y así mejorar la reutilización, la consistencia y la escalabilidad del código.

6.1.2 Paginación

Sirve para dividir grandes conjuntos de datos en segmentos más pequeños (páginas) que tienen un números limitado de elementos, pudiendo navegar a través de ellos accediendo así a las diferentes secciones del conjunto de datos y reduciendo la carga en la interfaz de usuario y en los servidores. Sobre todo, se tuvo en cuenta para mostrar listas de objetos al usuario de forma que le fuese rádida y cómoda.

6.1.3 Inversión del Control

Basado en que en lugar de que un componente controle directamente la ejecución de otros componentes, la inversión del control invierte este flujo, permitiendo que un framework controle cuándo y cómo se invocan ciertas funciones. Promueve el desacoplamiento entre los diferentes componentes de una aplicación al eliminar las dependencias directas y facilita la reutilización de componentes y la adaptación a diferentes contextos. Un ejemplo sería que, en lugar de que un componente de una aplicación cree directamente una instancia de otro componente utilizando la palabra clave ”new”, la inversión del control podría emplearse mediante un contenedor o framework que administre la creación y la gestión de objetos. Esto permite que la aplicación obtenga instancias de componentes necesarios sin tener que conocer detalles internos de su construcción.

Inyección de Dependencias

Es una forma de aplicar el principio de inversión del control. El patrón opera como un intermediario que se encarga de crear las piezas necesarias que una clase utiliza y luego se las proporciona. Esto significa que la clase no se encarga directamente de crear sus propias partes, sino que las recibe de otra fuente. Esto evita que la clase tenga que mencionar y crear explícitamente las partes que necesita dentro de su propio código.

6.1.4 Patrón Fachada

Patrón de diseño arquitectónico que se enfoca en proporcionar una interfaz simplificada y unificada para un conjunto de subsistemas más complejos dentro de la aplicación. Su objetivo principal es ocultar la complejidad interna y proporcionar a otros componentes o módulos de la aplicación un punto de entrada único y fácil de usar. Este patrón es especialmente útil al diseñar una aplicación que consta de múltiples subsistemas con funcionalidades diversas y complejas. Al utilizar la Fachada, se logra una mayor claridad en el diseño y la estructura del código, lo que facilita el mantenimiento y la escalabilidad de la aplicación.

6.1.5 DTO

Un DTO es una estructura de datos simple que agrupa diferentes campos o propiedades para almacenar información, se utilizan para transmitir datos entre la capa de presentación (frontend) y la capa de servicio (backend), minimizando la cantidad de información transmitida. Además, ayudan a mantener el desacoplamiento entre las diferentes capas de una aplicación.

6.1.6 MultiProvider

Patrón que emplea la arquitectura de administración de estado mediante varios proveedores, en contraposición a la aproximación convencional que utiliza un único proveedor para abarcar todo el estado de la aplicación, este enfoque propone la utilización de múltiples proveedores especializados que gestionan segmentos específicos del estado. En cuanto a sus características más importantes, se pueden destacar:

- Separación de Responsabilidades y Modularidad: Capacidad para promover la separación de responsabilidades. En lugar de tener un proveedor monolítico que abarque todo el estado, se crean múltiples proveedores, cada uno centrado en una porción concreta de la aplicación. Por ejemplo, en este proyecto se creo un provider para cada subsistema definido en el apartado 5.2. Esto aumenta la claridad y la legibilidad del código al dividir el estado y la lógica asociada en unidades manejables.

- Optimización y Eficiencia: La fragmentación del estado en varios proveedores especializados tiene beneficios directos en términos de eficiencia y rendimiento. Cuando una porción particular del estado cambia, solo los proveedores relacionados con esa área necesitan actualizarse, lo que evita la sobrecarga de actualizar componentes que no están directamente afectados.

- Reusabilidad y Consistencia: El uso de múltiples proveedores especializados fomenta la reutilización de código. Los mismos proveedores pueden ser utilizados en diferentes partes de la aplicación, lo que simplifica la gestión y la consistencia del estado en toda la aplicación. Además, al tener una abstracción más específica de los diferentes aspectos del estado, se facilita la implementación de características nuevas o cambios sin impactar a otros componentes.

6.2 Arquitectura Global

La aplicación utiliza una base de datos relacional que se comunica con el servidor de la aplicación a través de una capa de acceso a datos. Una vez que los datos se recuperan, pasan por la lógica de negocio necesaria y luego se transmiten al frontend mediante una API implementada en la capa de servicios. En cuanto a la estructura del frontend, consta de dos capas: una que se encarga de comunicarse con la capa de servicios del servidor y de gestionar el estado de la aplicación, y otra que permite a los usuarios interactuar con la interfaz que puede presentarse en diferentes idiomas.

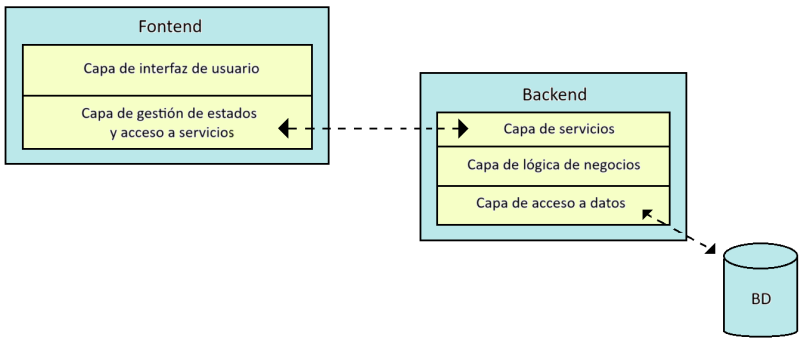

Figura 6.1: Aquitectura Global de la Aplicación Web

6.3 Estructura del backend

6.3.1 Diagrama UML y Explicación de sus Entidades

Para empezar, se diseñó un diagrama de entidades UML que representa visualmente la estructura de datos que hay que almacenar de forma persistente en la base de datos para el correcto funcionamiento de la aplicación.

Este diagrama ofrece una vista clara y concisa de las entidades, sus atributos y las relaciones entre ellas en el contexto del sistema de manera representándose como se explica a continuación.

- Entidades:Represendas como rectángulos. Cada entidad corresponde a una tabla o clase en la base de datos o en el modelo del sistema.
- Atributos: Dentro de cada entidad, es una lista que describen las propiedades o características de esa entidad. Estos atributos se enumeran en la parte superior del cuadro de la entidad. Cada atributo se presenta en el formato siguiente: ”- nombreDelAtributo: tipoDeDato”. El signo ”-” indica que es un atributo privado, mientras que el signo ”+” indicaría un atributo público.
- Relaciones: Las relaciones entre las entidades se representan mediante flechas que indican la dirección y el sentido. Si una flecha tiene una punta en un extremo, esto indica una relación unidireccional y, en cambio, si no tiene una punta indica una relación bidireccional. Tambien indicar que en el extremo de cada flecha, se coloca un número para indicar la cardinalidad de la relación.

A continuacion, se explican las entidades que forman el diagrama y el significado de cada una:

- User: Cuenta de usuario en la aplicación, incluyendo igualmente al administrador de la página como a los usuarios convencionales.
- University: Información sobre una universidad dada de alta en la aplicación.
- Subject: Información sobre una asignatura perteneciente a una universidad presente en la aplicación.
- Team: Representa un grupo creado en la aplicación.
- SubjectTeam: Entidad que representa la relación entre un grupo y una asignatura.
- UserTeam: Entidad que representa la relación entre un usuario y un equipo, indicando la pertenencia del usuario a un equipo específico.
- Meeting: Reunión creada en un determinado equipo por un usuario moderador.
- MeetingResponse: Respuesta individual de asistencia a una reunión planificada en un grupo.
- Message: Mensaje enviado por un usuario al chat de un grupo.
- WebResource: PDF, imagen o video subido a la aplicación web por un usuario en concreto.
- WebResourceTeam: PDF, imagen o video compartido a un grupo por el usuario que subió el recurso a la aplicación web.
- Flashcard: Tarjeta de información creada por un usuario en la propia aplicación web.
- FlashcardTeam: Entidad que representa una flashcard compartida a un grupo por el creador de la misma.
- Exam: Examen creado por un usuario en la propia aplicación web.
- ExamTeam: Entidad que representa un examen compartido a un grupo por el creador del mismo.
- Question: Pregunta perteneciente a un examen en concreto.
- AnswerRespuesta: perteneciente a una pregunta que, a su vez, pertenece a un examen creado en la aplicación web.
- Attemp: Intento de un examen creado en la aplicación web por parte de un usuario.

- AnswerSelected: Respuesta seleccionada por un usuario en el contexto de un intento de examen.
- Survey: Encuesta creada por un usuario en la propia aplicación web.
- SurveyTeam: Entidad que representa una encuesta compartida a un grupo por el creador de la misma.
- SurveyOption: Opción creada para una encuesta en concreto.
- OptionSelection: Selección individual de una de las opciones de una encuesta por parte de un usuario.

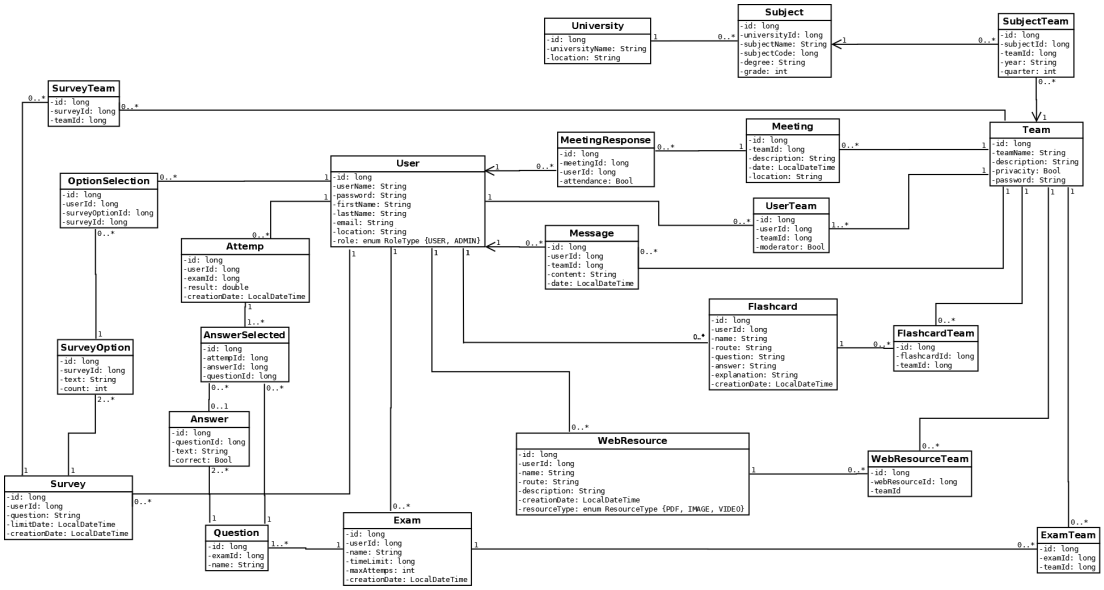

Figura 6.2: Diagrama de entidades UML

6.3.2 Capa de Acceso a Datos

Actúa como un puente entre la lógica de negocio de la aplicación y la persistencia de los datos en la base de datos. Las entidades presentes en esta capa actúan como representaciones de las tablas existentes en la base de datos. Estas entidades encapsulan los atributos y relaciones propias de la estructura de datos, asegurando que las operaciones de lectura y escritura se lleven a cabo de manera eficiente y coherente, sin comprometer la integridad de los datos.

Para lograr esta conexión sólida y organizada, se ha implementado estratégicamente el patrón DAO, donde cada entidad en la base de datos cuenta con su propio DAO dedicado, y la técnica de paginación dentro de cada uno.

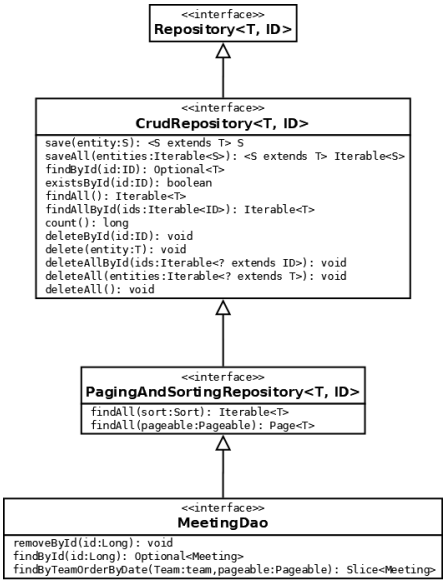

Figura 6.3: Diseño utilizando el patrón DAO con la entidad Meeting

6.3.3 Capa de Lógica de Negocio

Para cada subsistema de la aplicación, se diseñó una interfaz de servicio junto con su respectiva implementación, además de, un servicio común, que recopila y centraliza el código redundante que pueda surgir en diferentes implementaciones de los servicios. Todo esto resultó en una organización lógica y modular de la capa y un código limpio y mantenible, que evita duplicaciones innecesarias y fomenta la coherencia en el desarrollo.

Adicionalmente, se crearon un conjunto de excepciones que abarcan las limitaciones establecidas en la lógica de negocio de la aplicación. Estas excepciones fueron diseñadas para proporcionar un manejo preciso y claro de estas situaciones, asegurando que los usuarios estén informados de manera personalizada sobre las acciones permitidas y los posibles errores que puedan ocurrir.

En esta capa se utilizaron principalmente dos patrones de diseño para estructurar y simplificar el desarrollo. El primero fue la Inversión de Control en combinación con la técnica de Inyección de Dependencias, que permitió una conexión controlada y eficiente entre la clase servicio y los DAOs de la capa de acceso a datos. El segundo fue el patrón Fachada, que expuso de manera sencilla a otra capa del backend diversas funcionalidades de cada subsistema a través de una interfaz, su implementación y las diversas interfaces que utiliza.

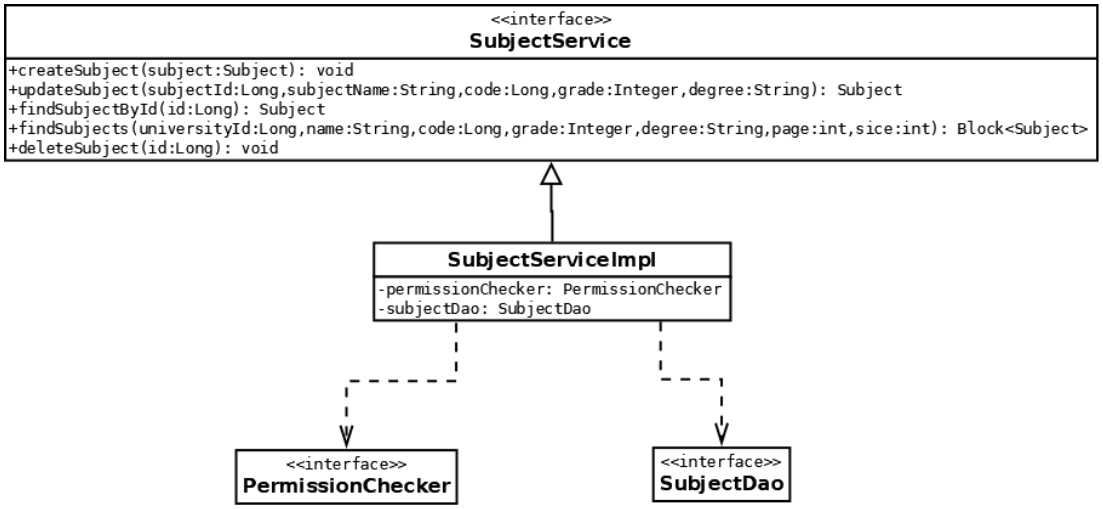

Figura 6.4: Diseño de un sevicio

6.3.4 Capa de Servicios

En esta capa se desplegó una API REST para interactuar de manera fluida con el frontend. Y para asegurar la seguridad y confidencialidad de los datos se implementaron enfoques estratégicos y patrones de diseño esenciales.

En primera instancia, se utilizó el patrón DTO para cada entidad, permitiendo la creación de versiones filtradas de las mismas eliminando la información sensible que no debía salir del servidor. Además, se adaptaron los formatos de los datos para que fueran manejables en un fichero JSON, asegurando una comunicación efectiva con el frontend.

A continuación, se realizó la creación de un controlador REST para cada subsistema del proyecto, exponiendo todas las operaciones CRUD necesarias a través de solicitudes POST, PUT, GET y POST Overloaded.

Adicionalmente, se implementó un sistema de seguridad con ayuda de una política de creación de sesiones, filtros JWT y la autorización basada en roles. Además, cada ruta de la API se definió con precision, especificando qué roles de usuario tienen acceso a cada recurso.

Por último, se presenta una tabla que describe el controlador creado para gestionar las funcionalidades relacionadas con la gestión de universidades y asignaturas. La tabla incluye detalles como la URL de acceso, el método HTTP asociado, los parámetros de entrada y salida, el cuerpo de la petición de entrada y las excepciones que pueden ocurrir en cada operación. Además, se proporciona una representación de los dos DTOs utilizados por el controlador, incluyendo los atributos que son expuestos al cliente y la relación entre ellos.

Operación	URL	Método HTTP	Parámetros	Cuerpo	Salida	Excepciones
createUniversity	/institutions/universities/create	POST		UniversityDto	ResponseEntity<UniversityDto>	DuplicateInstanceException PermissionException
updateUniversity	/institutions/universities/{universityId}	PUT	Long universityId	UniversityDto	UniversityDto	InstanceNotFoundException PermissionException
findUniversities	/institutions/universities	GET	String universityName String location Integer page		BlockDto<UniversitySummaryDto>	
removeUniversity	/institutions/universities /{universityId}/remove	POST	Long universityId		void	PermissionException InstanceNotFoundException
createSubject	/institutions/universities/ {universityId}/subjects/create	POST		SubjectDto	ResponseEntity<SubjectDto>	InstanceNotFoundException PermissionException DuplicateInstanceException
updateSubject	/institutions/universities/ {universityId}/subjects/{subjectId}	PUT	Long subjectId	SubjectDto	SubjectDto	InstanceNotFoundException PermissionException
findSubjects	/institutions/universities/ {universityId}/subjects	GET	Long universityId String name Long code Integer grade String degree int page		Block<SubjectDto>	
removeSubject	/institutions/universities/{universityId}/ subjects/{subjectId}/remove	POST	Long subjectId		void	PermissionException InstanceNotFoundException

Tabla 6.1: Diseño API de Gestión de Universidades

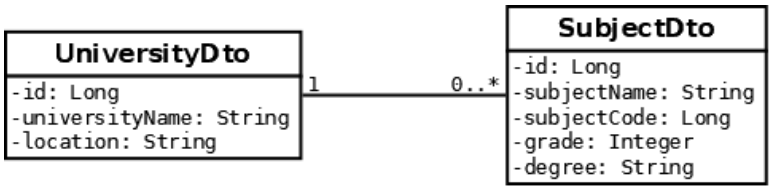

Figura 6.5: DTOs utilizados por el controlador de la gestión de universidades

6.4 Estructura del frontend

6.4.1 Capa de Gestión de Estado y Acceso a Servicios

Esta capa está compuesta de un conjunto de clases que utilizan el patrón MultiProvider para gestionar y proporcionar el estado a diferentes componentes de la interfaz de usuario. Estas mismas, también hacen uso de unas clases DTO pensadas a partir de las diversas entidades del proyecto.

Las clases que implementan el patrón MultiProvider son responsables de preparar, llevar a cabo y procesar las solicitudes hacia el servidor REST, actuando como intermediarias entre la interfaz de usuario y el servidor, asegurando que la información se envíe y reciba de manera adecuada y en el formato correcto.

Adicionalmente, esta capa se dedica a la internacionalización de la aplicación lograda mediante la creación de una clase que extiende la clase abstracta 'Translations' dónde se definen las traducciones para los diferentes idiomas.

A continuación, se muestra una representación de una de las clases que emplea el patrón MultiProvider. Esta clase se encarga de administrar cierta información sobre el estado de la aplicación del usuario. Además, expone métodos que permiten la comunicación con el backend y la gestión del estado interno de la aplicación.

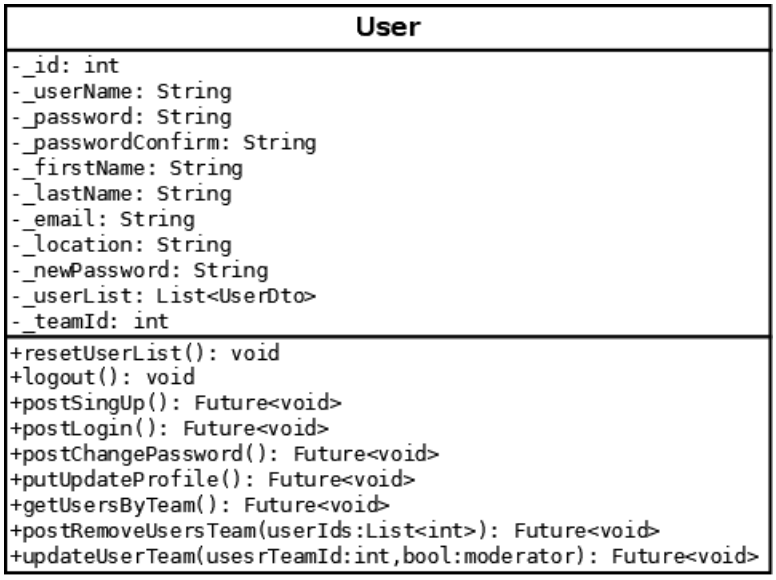

Figura 6.6: Diseño UML de la clase provider para la gestión de usuarios

6.4.2 Capa de Interfaz de Usuario

Parte de la aplicación con la que los usuarios interactúan directamente para realizar las acciones que desean, modificando los datos haciendo uso de las funcionalidades proporcionadas por la capa de gestión de estado y la capa de acceso a datos en el backend.

Dado que el proyecto consiste en una SPA esta capa toma forma a través de un único conjunto de componentes que se cargan y actualizan dinámicamente a medida que el usuario navega por la aplicación. Se enfoca en la presentación de información y la interacción con el usuario. Los elementos de diseño, los formularios, los botones, las transiciones y otras características visuales se implementan de manera que sean intuitivos y atractivos para el usuario.

A continuación, se muestra un wireframe sobre el diseño de la página inicial de la aplicación con el propósito de destacar y ejemplificar varios de los componentes dinámicos. Estos ejemplos incluyen elementos como el botón desplegable que facilita la selección entre búsqueda por nombre o por ubicación, el propio campo de entrada de texto y la lista interactiva de universidades que se muestra debajo del campo de búsqueda.

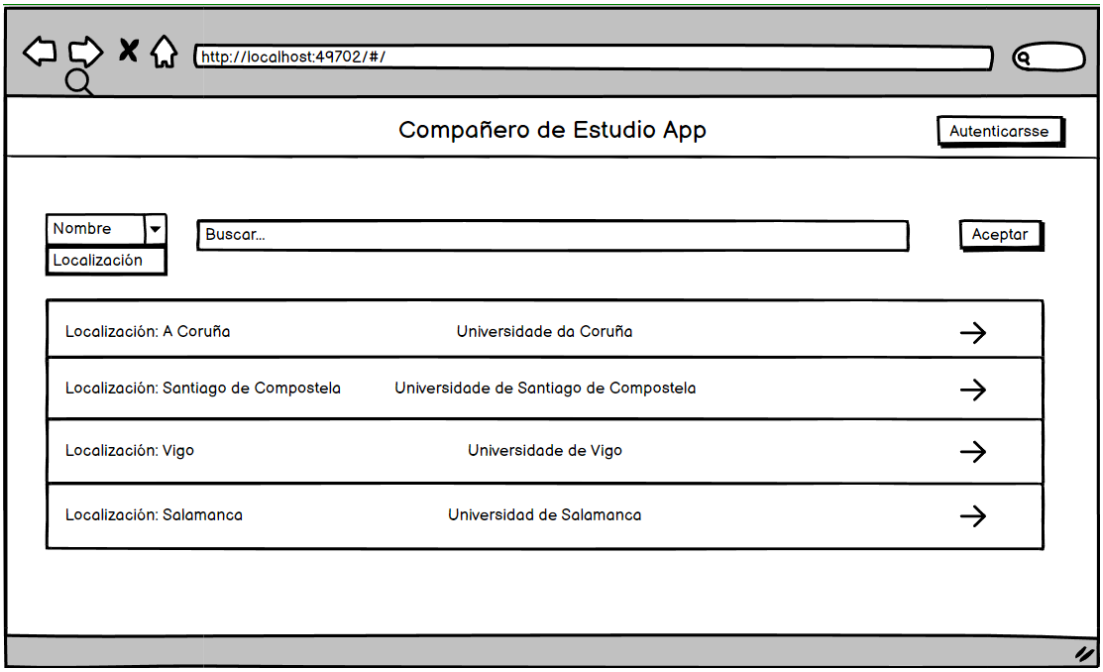

Figura 6.7: Wireframe de la Página de Inicio de la Aplicación Web

Capítulo 7

Implementación

En este capítulo, se expone cómo se ha realizado la implementación de la aplicación siguiendo el diseño establecido previamente y empleando los lenguajes y librerías mencionados en ”Herramientas Tecnológicas” (Capítulo 3). La estructura del capítulo comienza por la creación de las base de datos, y continuando con el backend y el frontend, detallando las capas correspondientes en cada uno. Al final del capítulo, se describe la implementación de las pruebas efectuadas para confirmar el correcto funcionamiento de la aplicación web.

7.1 Base de Datos MySQL

7.1.1 Creación de Tablas

Para la implementación de las tablas en la base de datos de nuestro proyecto, se utilizó el lenguaje SQL y el comando CREATE TABLE para definir cada entidad. Cada tabla creada siguiendo las especificaciones del diseño previo estableciendo las relaciones entre las entidades a través de claves primarias y foráneas para garantizar la integridad y coherencia de los datos.

Tomando como ejemplo la entidad ”Subject”, el siguiente código SQL fue empleado para crear su tabla correspondiente:

```
1 CREATE TABLE Subject
2 (
3     id           BIGINT                          NOT NULL
    AUTO_INCREMENT,
4     universityId BIGINT                          NOT NULL,
5     subjectName  VARCHAR(60) COLLATE latin1_bin NOT NULL,
6     subjectCode  BIGINT                          NOT NULL,
7     grade        INT                             NOT NULL,
8     degree       VARCHAR(60)                     NOT NULL,
9     CONSTRAINT SubjectPK PRIMARY KEY (id),
```

```
    CONSTRAINT SubjectUniversityFK FOREIGN KEY (universityId)
    REFERENCES University (id),
    CONSTRAINT CodeUniqueKey UNIQUE (subjectCode)
) ENGINE = InnoDB;
```

Este fragmento de código comienza con la declaración de la tabla ”Subject”. Las columnas se definen con sus respectivos nombres y tipos de datos, junto con restricciones que aseguran la integridad de los datos, como la clave primaria, las claves foráneas y la unicidad del campo ”SubjectCode”. La cláusula ENGINE = InnoDB se utiliza para especificar el motor de almacenamiento que se utilizará para la tabla, en este caso, InnoDB que es compatible con MYSQL.

De esta manera, se crearon todas las tablas del proyecto siguiendo un enfoque similar, y además, al principio del scrpit se establecieron, por un orden que no comprometiese las relaciones entre las tablas, los comandos DROP TABLE IF EXIST necesarios para borrar todas las tablas existentes y crear la base de datos desde cero.

```
[...]
DROP TABLE IF EXISTS SelectedAnswer;
DROP TABLE IF EXISTS Answer;
DROP TABLE IF EXISTS Question;
DROP TABLE IF EXISTS Attemp;
DROP TABLE IF EXISTS Exam;
[...]
DROP TABLE IF EXISTS Team;
DROP TABLE IF EXISTS User;
```

7.1.2 Inserción de Información

Para facilitar la inserción inicial de datos cada vez que se ejecutaba el script anterior, se implementó otro script que emplea el comando INSERT INTO (...) VALUES (...). Estos registros iniciales son valiosos para verificar el funcionamiento de la aplicación en un entorno más realista y permiten realizar pruebas sin tener que crear manualmente todos los datos en cada iteración del desarrollo.

A continuación, se presenta un fragmento de este script:

```
INSERT INTO User (userName, password, firstName, lastName, email,
    location, role) VALUES ("admin",
    "$2a$10$bcE3UJO62Dqyx.U3kkAeeu2AhVUKDcrnDZFa2kXFPIvkKzcZ8jSxS",
    "Adminez", "Darriba", "admin@gmail.com", "Adminlandia", 0);

INSERT INTO University (universityName, location) VALUES ("UDC", "A
    Coruna");
```

```
4 INSERT INTO University (universityName, location) VALUES ("USC",
      "Santiago");
5
6 INSERT INTO Subject (universityId, subjectName, subjectCode, grade,
      mention) VALUES (1, "FC", 123, 1, "Ingeniería Informática");
7 INSERT INTO Subject (universityId, subjectName, subjectCode, grade,
      mention) VALUES (1, "AR", 17951, 4, "Ingeniería Informática");
8
9 INSERT INTO Team (teamName, description, privacity, link) VALUES
      ("Pandilla patatilla", "Primer grupo de la aplicación", true,
      "C0nTr4s3n4");
10
11 INSERT INTO SubjectTeam (subjectId, teamId, year, quarter) VALUES
      (1, 1, "2022/2023", 2);
12
13 INSERT INTO UserTeam (userId, teamId, moderator) VALUES (1, 1,
      true);
```

En este fragmento, se ilustra cómo se insertan el usuario administrador, las universidades de Coruña y Santiago, dos asignaturas asociadas a la Univesidad de Coruña, y se simula la creación de un equipo por parte del administrador ligándolo a la asignatura llamada ”FC”.

7.2 Implementación del Backend

Para el servidor se optó por utilizar el lenguaje de programación Java en conjunto con el framework Spring. Esta combinación proporcionó un entorno robusto y eficiente para desarrollar las diversas funcionalidades y componentes que conforman la lógica del backend. A continuación, se describen los aspectos más relevantes de esta implementación en cada capa.

7.2.1 Capa de Acceso a Datos

En esta capa, se llevó a cabo la implementación de las entidades que representan las tablas de la base de datos, manteniendo la misma nomenclatura y atributos definidos en la creación de las tablas. Para cada entidad, se creó un constructor vacío, que es requerido por el framework Spring, otro con todos los parámetros necesarios para crear la entidad (exceptuando el campo de identificación, ya que este se genera automáticamente), y un tercer constructor que acepta los datos provenientes del DTO, lo que facilita su creación desde el conversor en la capa de servicios.

En la implementación de cada entidad, se utilizaron las etiquetas de anotación proporcionadas por JPA para indicar su mapeo con la base de datos y las relaciones entre ellas. Las etiquetas utilizadas fueron las siguientes:

- @Entity: Marca la clase como una entidad persistente que se mapea a una tabla en la base de datos.
- @Id: Indica que el campo al que se aplica es la clave primaria de la entidad.
- @GeneratedValue(strategy = GenerationType.IDENTITY): Define la estrategia de generación de valores para la clave primaria, en este caso, se utiliza una estrategia de incremento automático.
- @ManyToOne(optional = false, fetch = FetchType.LAZY): Indica una relación muchos a uno entre entidades, especificando que es requerida y utilizando un comportamiento de carga perezosa, lo que significa que los datos relacionados no se recuperan inmediatamente de la base de datos cuando se solicita una entidad, sino que se obtienen solo cuando se accede explícitamente a ellos.
- @JoinColumn(name = ”teamId”): Establece la columna de unión para la relación, en este caso, con la entidad ”Team”.
- @OneToMany(mappedBy = ”meeting”): Indica una relación uno a muchos con otra entidad, la parte mappedBy = ”meeting” indica que el campo de la otra entidad que establece la relación es llamado ”meeting”.

A continuación, se puede observar un fragmento donde se puede ver el uso de todos los constructores y anotaciones comentados:

```
1  @Entity
2  public class Meeting {
3      private Long id;
4      private Team team;
5      private String name;
6      private String description;
7      private LocalDateTime date;
8      private String direccion;
9      private Set<MeetingResponse> meetingResponses = new HashSet<>();
10
11     public Meeting() {
12     }
13
14     public Meeting(Team team, String name, String description,
       LocalDateTime date, String direccion) {
15         this.name = name;
16         this.team = team;
17         this.description = description;
18         this.date = date;
19         this.direccion = direccion;
```

```
20      }
21
22      public Meeting(String name, String description, LocalDateTime
        date, String direccion) {
23          this.name = name;
24          this.description = description;
25          this.date = date;
26          this.direccion = direccion;
27      }
28
29      @Id
30      @GeneratedValue(strategy = GenerationType.IDENTITY)
31      public Long getId() {
32          return id;
33      }
34
35      ...
36
37      @ManyToOne(optional = false, fetch = FetchType.LAZY)
38      @JoinColumn(name = "teamId")
39      public Team getTeam() {
40          return team;
41      }
42
43      ...
44
45      @OneToMany(mappedBy = "meeting")
46      public Set<MeetingResponse> getMeetingResponses() {
47          return meetingResponses;
48      }
49  }
```

En esta capa también se encuentran los DAO de estas entidades que se han implementado junto a la técnica de paginación comentada anteriormente.

```
1  public interface UserDao extends PagingAndSortingRepository<User,
       Long> {
2      boolean existsByUserName(String userName);
3
4      Optional<User> findByUserName(String userName);
5
6      void removeUserById(Long userId);
7  }
```

Como se puede observar en este fragmento se encarga de gestionar las operaciones relacionadas con la entidad "User". A través de la interfaz PagingAndSortingRepository, se definen

métodos como existsByUserName, findByUserName y removeUserById, que permiten verificar la existencia y buscar un usuario por su nombre de usuario y eliminar un usuario por su identificador, respectivamente. Por último, comentar que la interfaz DAO también ofrece funciones como ”save”, permitiendo crear o actualizar un registro en la base de datos, y ”delete”, que se utiliza para eliminar un registro de la base de datos.

7.2.2 Capa de Lógica de negocio

En esta capa se encuentran las interfaces de los servicios, sus implementaciones y las excepciones personalizadas creadas para manejar situaciones específicas en la aplicación.

Interfaces de servicios

Definición de interfaces que exponen las operaciones de lógica de negocio que serán utilizadas por los controladores REST. Estas interfaces proporcionan una abstracción de alto nivel de las operaciones que pueden ser realizadas en el sistema. Este fragmento de la interfaz de servicio de la gestión de grupos, contiene los métodos para crear grupos, actualizar su información y obtener las respuestas a una determinada reunión:

```
1 public interface TeamService {
2
3     void createTeam(Long userId, Team team) throws
      DuplicateInstanceException, InstanceNotFoundException;
4
5     Team updateTeam(Long teamId, String name, String description,
      Boolean privacity, String link) throws InstanceNotFoundException;
6
7     Block<MeetingResponse> getMeetingResponseByMeeting(Meeting
      meeting, int page, int size);
8
9 }
```

Implementación de los servicios

Las implementaciones de los servicios se anotan con @Service para que Spring las reconozca como componentes de servicio. La anotación @Transactional se utiliza para definir el ámbito transaccional en torno a los métodos del servicio. La inyección de dependencias se realiza mediante @Autowired.

Dentro de las implementaciones de los servicios, también se puede observar el uso de componentes como PermissionChecker que encapsulan lógica común, ayudando a mantener un código limpio y modular.

Por ejemplo, la implementación del método createTeam en TeamServiceImpl luce así:

```
@Service
@Transactional
public class TeamServiceImpl implements TeamService {

    @Autowired
    private TeamDao teamDao;

    @Autowired
    private UserDao userDao;

    @Autowired
    private UserTeamDao userTeamDao;

    @Autowired
    private PermissionChecker permissionChecker;

    @Override
    public void createTeam(Long userId, Team team) throws
    DuplicateInstanceException, InstanceNotFoundException {

        if (teamDao.existsByTeamName(team.getTeamName()))
            throw new
    DuplicateInstanceException("project.entities.team",
    team.getTeamName());

        teamDao.save(team);

        User user = permissionChecker.checkUser(userId);
        UserTeam newUserTeam = new UserTeam(user, team, true);
        team.addUser(newUserTeam);
        user.addTeam(newUserTeam);

        userTeamDao.save(newUserTeam);
    }

    // Otros métodos implementados
}
```

La lógica seguida en este método es primero verificar si el nombre del equipo ya existe, crear el equipo y se establecer la relación entre el equipo y el usuario.

Excepciones Personalizadas

Las excepciones personalizadas, como DuplicateInstanceException e InstanceNotFoundException, se utilizan para capturar situaciones específicas, como en el caso anterior, que

salta si el equipo ya existe o si no se encuentra en la base de datos el usuario que quiere crear el equipo, y proporcionar información detallada sobre el error. Estas excepciones extienden de excepciones genéricas personalizándolas con mensajes significativos para los usuarios y desarrolladores.

7.2.3 Capa de Servicios

En esta capa, se encuentran los DTO, los conversores entre los DTO y las entidades, los controladores REST y las clases de configuración de seguridad del sistema.

DTO

Clases compuestas únicamente de sus atributos, un contructor vacío, otro con todos los atributos y métodos get y set para cada uno, representan de manera simplificada la información que se desea mostrar en el frontend o comunicar al backend. Un ejemplo de esto es la clase FlashcardDto, donde podemos observar que en lugar de pasar el objeto de usuario completo, se utiliza el nombre de usuario para identificar al creador de la tarjeta. Además, se eligió representar la fecha de creación como un valor numérico long en lugar de utilizar LocalDateTime debido a que trabajar con valores de tiempo en milisegundos proporciona un formato más manejable y sencillo para su transferencia y manipulación.

```
1  public class FlashcardDto {
2
3      private Long id;
4      private String userName;
5      private String name;
6      private String route;
7      private String question;
8      private String answer;
9      private String explanation;
10     private long creationDate;
11
12     public FlashcardDto() {
13     }
14
15     public FlashcardDto(Long id, String userName, String name,
       String route, String question, String answer,
16                         String explanation, long creationDate) {
17         this.id = id;
18         this.userName = userName;
19         this.name = name;
20         this.route = route;
21         this.question = question;
22         this.answer = answer;
```

```
        this.explanation = explanation;
        this.creationDate = creationDate;
    }

    public Long getId() {
        return id;
    }

    public void setId(Long id) {
        this.id = id;
    }

   //Resto de métodos get y set
}
```

Conversores entre DTO y Entidades

Clases encargadas de transformar los objetos de las entidades en objetos DTO y viceversa. Por ejemplo, la función toFlashcardDto convierte un objeto Flashcard en un FlashcardDto al mapear los campos relevantes de la entidad a los campos correspondientes del DTO, incluyendo también la función toFlashcard, que realiza la conversión inversa. Además, la función toFlashcardDtos permite convertir una lista de objetos Flashcard en una lista de objetos FlashcardDto utilizando el método map y Collectors.toList().

Es importante destacar que estas operaciones de conversión optimizan la manipulación de datos y facilitan la coherencia entre las capas del sistema.

La función privada toMillis dentro del conversor se utiliza para transformar un objeto LocalDateTime al sistema para la descripción de instantes de tiempo Unix, que son los segundos transcurridos desde la medianoche UTC del 1 de enero de 1970. Esto es útil para representar fechas en un formato más sencillo para el transporte y la manipulación.

```
public class FlashcardConversor {

    public FlashcardConversor() {
    }

    public final static FlashcardDto toFlashcardDto(Flashcard
    flashcard){
        return new FlashcardDto(flashcard.getId(),
    flashcard.getUser().getUserName(), flashcard.getName(),
                flashcard.getRoute(), flashcard.getQuestion(),
    flashcard.getAnswer(), flashcard.getExplanation(),
                toMillis(flashcard.getCreationDate()));
    }
```

```
11
12      public final static Flashcard toFlashcard(FlashcardDto
        flashcardDto){
13          return new Flashcard(flashcardDto.getName(),
        flashcardDto.getRoute(), flashcardDto.getQuestion(),
14                  flashcardDto.getAnswer(),
        flashcardDto.getExplanation());
15      }
16
17      public final static List<FlashcardDto>
        toFlashcardDtos(List<Flashcard> flashcards) {
18          return
        flashcards.stream().map(FlashcardConversor::toFlashcardDto)
19          .collect(Collectors.toList());
20      }
21
22      private static long toMillis(LocalDateTime date) {
23          return
        date.truncatedTo(ChronoUnit.MINUTES).atZone(ZoneOffset
24          .systemDefault()).toInstant().toEpochMilli();
25      }
26  }
```

Controladores REST

Desempeñan el papel de exponer las funcionalidades del servidor al frontend, permitiendo que las solicitudes y respuestas se gestionen a través de la comunicación HTTP.

Para establecer una configuración inicial se emplearon las etiquetas @RestController, que indica que la clase es un controlador REST y que los métodos en ella responderán a solicitudes HTTP, y @RequestMapping utilizado para establecer la ruta base para todos los endpoints dentro del controlador. Además, se utilizó de nuevo la inyección de dependencias con la etiqueta @Autowired para acceder a los servicios de la capa de lógica de negocio.

```
1  @RestController
2  @RequestMapping("/teams")
3  public class TeamController {
4
5      @Autowired
6      private TeamService teamService;
7
8      @Autowired
9      private PermissionChecker permissionChecker;
10
11     // Métodos del controlador...
```

```
}
```

Para los métodos, encargados de gestionar las operaciones CRUD relacionadas con los equipos, se anotó con las etiquetas @PostMapping, @PutMapping o @GetMapping, según el tipo de operación HTTP que corresponda y utilizando la técnica de Post Overoaled para los borrados en vez de @DeleteMapping. Además los parámetros de estos métodos también incluyen etiquetas, como @PathVariable, para extraer valores de variables de ruta, y @RequestBody para vincular automáticamente el cuerpo de una solicitud HTTP a un objeto Java.

```
@PostMapping("/create")
    public ResponseEntity<TeamDto> createTeam(@RequestParam Long
    userId, @Validated({TeamDto.AllValidations.class}) @RequestBody
    TeamDto teamDto)
            throws DuplicateInstanceException,
    InstanceNotFoundException {

        Team team = TeamConversor.toTeam(teamDto);
        teamService.createTeam(userId, team);

        URI location = ServletUriComponentsBuilder
                .fromCurrentRequest().path("/{teamId}")
                .buildAndExpand(team.getId()).toUri();

        return
    ResponseEntity.created(location).body(toTeamDto(team));
    }
```

Configuración de la seguridad

Establece cómo se gestiona la autenticación y la autorización en la aplicación web. El código define una configuración de seguridad personalizada a través de la clase SecurityConfig, que está anotada con @Configuration, que indica que la clase contiene configuraciones que se utilizarán en la aplicación Spring, y @EnableWebSecurity para habilitar dicha configuración. Las acciones que hace esta configuración son:

- Se deshabilita la protección contra ataques CSRF y se establece la creación de sesiones en política sin estado (STATELESS), lo que significa que no se mantendrá información de sesión en el servidor.
- Se agrega un filtro (JwtFilter) para la autenticación basada en JWT.
- Se definen las reglas de autorización para diferentes rutas y métodos HTTP utilizando authorizeRequests(). Cada llamada a antMatchers() establece permisos para rutas específicas y se especifica qué roles pueden acceder a esas rutas.

- Se utiliza el método denyAll() para denegar el acceso a cualquier otra solicitud que no haya sido explícitamente permitida.
- Se configura la política de manejo de CORS a través de la implementación de corsConfigurationSource(). Se permite el acceso desde cualquier origen (*), se habilitan todos los métodos HTTP y se permiten todas las cabeceras.

```
@Configuration
@EnableWebSecurity
public class SecurityConfig extends WebSecurityConfigurerAdapter {

    @Autowired
    private JwtGenerator jwtGenerator;

    @Override
    protected void configure(HttpSecurity http) throws Exception {
        http.cors().and().csrf().disable()
            .sessionManagement().sessionCreationPolicy(
            SessionCreationPolicy.STATELESS).and()
            .addFilter(new JwtFilter(authenticationManager(),
    jwtGenerator))
            .authorizeRequests()
            .antMatchers(HttpMethod.POST,
    "/users/signUp").permitAll()
            .antMatchers(HttpMethod.POST,
    "/users/login").permitAll()
            .antMatchers(HttpMethod.POST,
    "/users/loginFromServiceToken").permitAll()
            .antMatchers(HttpMethod.PUT, "/users/*").hasRole("USER")
            // ... Otras reglas de autorización ...
            .anyRequest().denyAll();
    }

    @Bean
    public CorsConfigurationSource corsConfigurationSource() {
        CorsConfiguration config = new CorsConfiguration();
        UrlBasedCorsConfigurationSource source = new
    UrlBasedCorsConfigurationSource();
        config.setAllowCredentials(true);
        config.setAllowedOriginPatterns(Arrays.asList("*"));
        config.addAllowedHeader("*");
        config.addAllowedMethod("*");
        source.registerCorsConfiguration("/**", config);
        return source;
    }}
```

7.3 Implementación del Frontend

En la programación de las vistas en el frontend, se tomó la decisión de utilizar el lenguaje de programación Dart[4] en combinación con el framework Flutter Web[5]. A continuación, se detallan los aspectos más destacados de esta implementación en cada capa.

7.3.1 Capa de Gestión de Estado y Acceso a Servicios

Esta capa incluye clases proveedoras que son responsables de manejar el estado de la aplicación y ofrecer acceso a los servicios del backend a través de la biblioteca HTTP. También se crearon clases DTO para almacenar y estructurar la información proveniente del servidor o para comunicarsela, además, se implementó la localización de la aplicación en tres idiomas.

Clases Provider y DTO

Clases compuestas por los atributos necesarios para poder realizar las llamadas al servidor y de listas de los DTOs necesarios para procesar las llamadas GET, además de los getters y setters necesarios para acceder a la información. En el siguiente fragmento se puede observar el caso en el que se llama al servidor para crear un Equipo a través de una petición POST, pasándole como cuerpo de la petición los parámetros necesarios para crearlo, y una petición GET para recuperar todos los equipos en los que participa un usuario, procesando luego la respuesta en una lista de DTOs de los equipos.

```
class Team {
  late int _id;
  late int _userId;
  String _teamName = "";
  String _description = "";
  String _password = "";
  bool _privacity = true;
  List<TeamDto> _listGrupos = [];
  ...

  Future<void> postCreateTeam() async {
    Map<String, Object> data = {
      "teamName": _teamName,
      "description": _description,
      "privacity": _privacity,
      "password": _password
    };

    var response = await http.post(
```

```
20
      Uri.parse("http://localhost:8080/teams/create?userId=$_userId"),
21      headers: {
22        'Content-Type': 'application/json',
23      },
24      body: json.encode(data),
25    );
26
27    if (response.statusCode != 201) {
28      throw Exception('errorCreateTeam'.tr);
29    }
30  }
31
32  Future<void> getTeams() async {
33    Uri url =
      Uri.parse("http://localhost:8080/teams?userId=$_userId");
34    http.Response response = await http.get(url);
35
36    if (response.statusCode == 200) {
37      var jsonData = json.decode(response.body);
38
39      List<dynamic> teamsData = jsonData["items"];
40
41      _listGrupos = teamsData.map((teamData) {
42        bool moderator = false;
43        var userTeamList = teamData["userTeam"];
44        if (userTeamList is List && userTeamList.isNotEmpty) {
45          moderator = userTeamList[0]["moderator"] ?? false;
46        }
47
48        return TeamDto(
49          id: teamData["id"],
50          teamName: teamData["teamName"],
51          description: teamData["description"],
52          privacity: teamData["privacity"],
53          password: teamData["password"],
54          moderator: moderator,
55        );
56      }).toList();
57
58
59    } else {
60      throw Exception('errorGetTeams'.tr);
61    }
62  }
63 }
```

```
64
65 class TeamDto {
66   final int id;
67   final String teamName;
68   final String description;
69   final bool privacity;
70   final String password;
71   late bool moderator;
72
73   TeamDto(
74       {required this.id,
75       required this.teamName,
76       required this.description,
77       required this.privacity,
78       required this.password,
79       required this.moderator});
80 }
```

Localización

Al iniciar la aplicación se realiza una configuración inicial mediante GetMaterialApp. En sus parámetros, se especifica la clase que contiene las definiciones de traducciones, en este caso, LocalString(), y se elige una localización disponible de manera arbitraria. Luego, para utilizar estas traducciones en toda la aplicación, simplemente usamos el método tr.

A continuación, se muestra la clase que almacena las traducciones junto con las definiciones para las diferentes localizaciones disponibles, incluyendo las traducciones para el título de la aplicación y la palabra ”nombre”. Finalmente, se muestra cómo se utiliza el método tr para traducir el título de la aplicación.

```
1  class LocalString extends Translations {
2      @override
3      Map<String, Map<String, String>> get keys => {
4          'es_ES': {
5              'titulo': 'Compañero de Estudio App',
6              'nombre': 'Nombre',
7              ...
8          },
9          'gl_ES': {
10             'titulo': 'Compañeiro de Estudo App',
11             'nombre': 'Nome',
12             ...
13          },
14         'en': {
15             'titulo': 'Study Buddy App',
```

```
            'nombre': 'Name',
            ...
        }
    };
}

class Home extends StatelessWidget {
    ...
    return Scaffold(
        appBar: AppBar(
          title: Text('titulo'.tr),
    ...
}
```

7.3.2 Capa de Interfaz de Usuario

Parte visual y tangible de la aplicación, donde los usuarios interactúan con la funcionalidad proporcionada por las capas anteriores. En esta capa, se utilizan widgets y elementos para componer y distribuir la disposición de la interfaz, y se responden a las interacciones de los usuarios.

El fragmento que se muestra a continuación pertenece a la vista dedicada a crear un equipo utilizando un formulario.

```
class CreateTeam extends StatefulWidget {
  const CreateTeam({Key? key}) : super(key: key);

  @override
  State<CreateTeam> createState() => _CreateTeamState();
}

class _CreateTeamState extends State<CreateTeam> {
  final _formKey = GlobalKey<FormState>();

  @override
  Widget build(BuildContext context) {
    final grupo = Provider.of<Team>(context);
    final data = Provider.of<Data>(context);

    // Métodos para construir componentes reutilizables
    // teamName(), description(), DropdownButtonPrivacidad(),
    aceptar()

    return Scaffold(
      backgroundColor: Colors.white,
      appBar: AppBar(
```

```
        title: Text('titulo'.tr),
      ),
      body: Center(
        child: Padding(
          padding: EdgeInsets.all(MediaQuery.of(context).size.width
    / 30),
          child: Form(
            key: _formKey,
            child: Column(
              mainAxisAlignment: MainAxisAlignment.spaceAround,
              children: <Widget>[
                Text('crearGrupo'.tr),
                teamtName(),
                description(),
                Text('botonPrivacidad'.tr),
                DropdownButtonPrivacidad(),
                aceptar(),
              ],
            ),
          ),
        ),
      ),
    );
  }
}
```

Podemos observar los siguientes elementos:

- CreateTeam: Widget de tipo StatefulWidget que permite construir la interfaz de creación de equipos. Define el estado mutable _CreateTeamState.
- _formKey: Clave global que identifica el formulario y se utiliza para validar y guardar los campos.
- Provider.of<Team>(context) y Provider.of<Data>(context): Llamadas para acceder a instancias de las clases Team y Data respectivamente, proporcionadas por el patrón MultiProvider.
- teamtName(), description(), DropdownButtonPrivacidad() y aceptar(): Métodos que definen widgets reutilizables para diferentes partes del formulario, como el nombre del equipo, la descripción, el botón desplegable que permite eligir entre ”Público” o ”Privado” y el botón ”Aceptar”.
- Scaffold: Widget que proporciona la estructura básica de la página, incluyendo la barra de navegación y el área de contenido.

- AppBar: Definición de la barra de navegación con un título.
- Form: Widget que agrupa los campos de entrada y permite realizar validaciones y guardar datos cuando se envía el formulario.
- Column: Widget que organiza los elementos en una columna vertical.
- Text: Muestra textos traducidos en diferentes idiomas haciendo uso del método tr.

7.4 Pruebas

Para poder comprobar el correcto funcionamiento de la implementación explicada se realizaron una serie de pruebas en los diferentes niveles que componen la aplicación.

7.4.1 Pruebas de Integración

Una vez implementadas las funcionalidades en la capa de lógica de negocio, de cada sprint, se realizaron pruebas de integración utilizando la librería JUnit, que además de comprobrar que la lógica de negocio esté bien implementada, también se prueba la comunicación con la base de datos creando las entidades necesarias y recuperando los datos solicitados de manera predecible.

En cuanto a la configuración de las clases encargadas de realizar las pruebas destacar la utilización de las siguientes anotaciones:

- @SpringBootTest: Anotación que indica la realización pruebas de Spring Boot y carga el contexto de la aplicación. Esto permite que todos los componentes de Spring estén disponibles durante las pruebas.
- @ActiveProfiles(”test”): Establece el perfil activo de la aplicación durante las pruebas, en este caso, se utiliza el perfil ”test”, lo que implica configuraciones específicas.
- @Transactional: Asegura que cada método de prueba se ejecute en una transacción y que se revierta una vez que se complete el método. Esto ayuda a mantener la base de datos en un estado consistente entre las pruebas.
- @Test: forma en la que JUnit marca un método como una prueba a ejecutar, permitiendo verificar automáticamente si el código cumple con las expectativas definidas en las aserciones.

En el framento utilizado a continuación se muestran las pruebas de integración de la actualización de una asignatura y de la creación incorrecta de una asignatura debido a que está duplicada y resultando en la excepción correspondiente a ese caso.

```
@SpringBootTest
@ActiveProfiles("test")
@Transactional
public class SubjectServiceTest {

    @Autowired
    private SubjectService subjectService;

    @Test
    public void updateSubject() throws PermissionException,
    DuplicateInstanceException, InstanceNotFoundException {
        Subject subject = createSubject();
        subjectService.createSubject(subject);

        subject.setSubjectName("Updated" +
    subject.getSubjectName());
        subject.setSubjectCode(987654321L);
        subject.setGrade(2);
        subject.setMention("Updated" + subject.getMention());

        subjectService.updateSubject(subject.getId(), "Updated" +
    subject.getSubjectName(),
                98765431L, 2, "Updated" + subject.getMention());

        Subject updatedSubject =
    subjectService.findSubjectById(subject.getId());

        assertEquals(subject, updatedSubject);
    }

        @Test
    public void testCreateDuplicatedSubject() throws
    PermissionException, DuplicateInstanceException {
    Subject subject = createSubject();

        subjectService.createSubject(subject);
        assertThrows(DuplicateInstanceException.class, () ->
    subjectService.createSubject(subject));
    }
}
```

7.4.2 Pruebas de la API REST

Después de agregar nuevas características a los controladores de la API REST y ajustar la configuración de seguridad para permitir estas acciones, se llevaron a cabo pruebas utilizando el cliente Postman. Estas pruebas tenían como objetivo verificar que las nuevas funciones afectaran la base de datos según lo previsto y que las respuestas de la API fueran las adecuadas.

En la primera captura de pantalla se simuló la creación de un registro para la Universidad de Salamanca en la aplicación y en la segunda captura se realizó una solicitud para buscar universidades cuyas direcciones contuvieran la letra "a". Con ambas se confirmó que la API devolvía los códigos, mensajes y datos esperados en la respuesta.

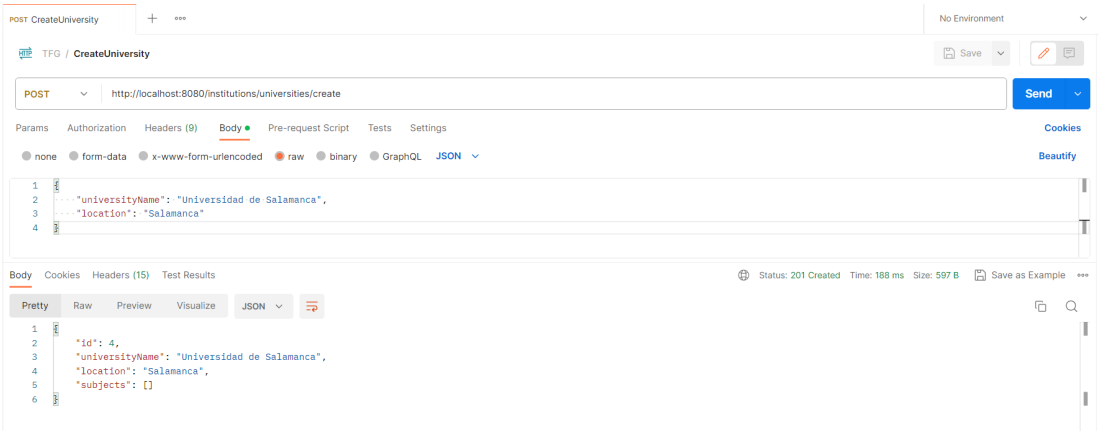

Figura 7.1: Prueba en Postman para la Creación de una Universidad

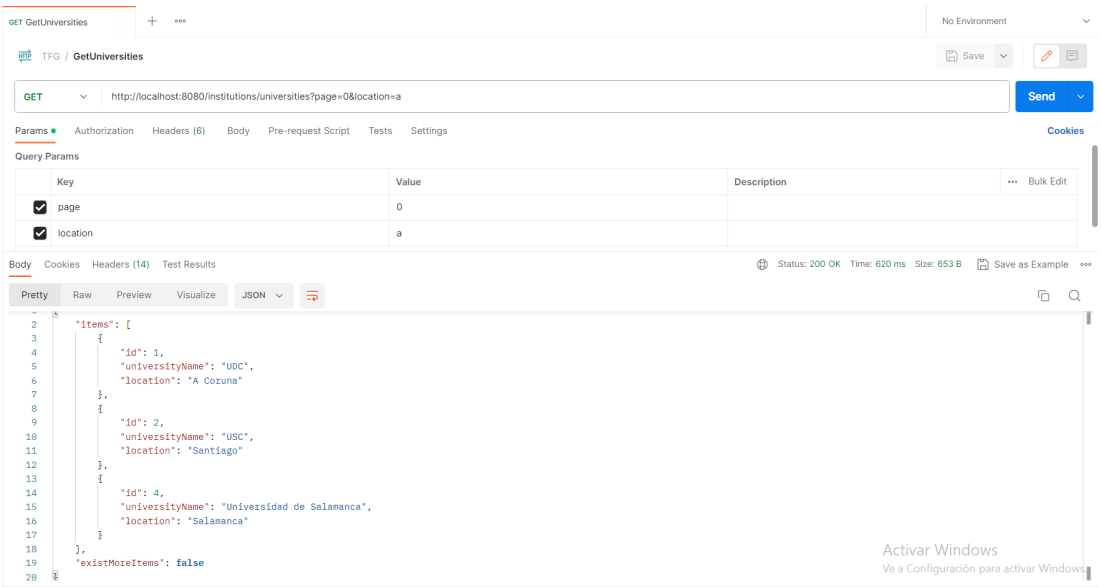

Figura 7.2: Prueba en Postman de la Recuperación de Universidades

7.4.3 Pruebas Funcionales

Al concluir cada sprint, se llevaron a cabo este tipo de pruebas para asegurarse de que la aplicación cumpliera con los requerimientos establecidos y que sus todos los componentes estuvieran operativos de manera adecuada. Estas pruebas se enfocaron en evaluar la funcionalidad del software tal como lo haría el usuario final. Se trataba de verificar que todas las historias de usuario mencionadas anteriormente se pudieran completar, adoptando la perspectiva de un usuario de la aplicación y explorando las diversas rutas de acceso y abarcando todos los posibles casos de uso.

Capítulo 8

Planificación y Costes

En este capítulo se explicará el desarrollo del proyecto en cuanto a tiempos, periodos de desarrollo por iteración y la asignación de tiempo a cada una. Además, se realizará una estimación del coste del proyecto basada en las horas trabajadas.

8.1 Planificación temporal

El desarrollo del proyecto comenzó a finales de enero de 2023 con la conceptualización de la idea, ya que el anteproyecto otorga una gran libertad creativa. Las fases de diseño y análisis del proyecto se iniciaron en esa etapa. Sin embargo, debido al comienzo de prácticas de empresa a mediados de marzo y otros motivos personales, el proyecto quedó mayoritariamente en pausa hasta que fue retomado a mediados de julio, por lo que los siguientes sprints se extendieron hasta finales de agosto, lo que representa un total de aproximadamente 14 semanas de trabajo.

Para poder valorar mejor el esfuerzo y tiempo dedicado se tendrán en cuenta 3 roles diferenciados:

- Analista: Responsable de las tareas de conceptualización, análisis y diseño del proyecto.
- Programador: Encargado de implementar la aplicación y realizar las pruebas de la aplicación web.
- Jefe coordinador del proyecto: Persona encargada de dirigir al resto de roles, en este caso, es el papel realizado por el tutor del TFG.

A continuación se muestra de forma más detallada cuanto tiempo se dedico a cada tarea y sprint.

- Conceptualización, análisis y diseño inicial: Significó las dos primeras semanas de febrero, que al no ser dedicadas mayoritariamente al proyecto, acumularon unas 30 horas de trabajo antes de comenzar con el primer sprint.
- Sprint 1 - Gestión de Usuarios: Al ser la primera toma de contacto con la implementación del proyecto supuso más esfuerzo y tiempo que los sprints posteriores pese a que la carga de trabajo está distribuida de manera equitativa entre los diferentes sprints. Finalmente, abarcó unas 45 horas de trabajo, divididas entre las dos últimas semanas de febrero.
- Sprint 2 - Gestión de Universidades: Fue la última iteración realizada antes de pausar el proyecto, se extendió temporalmente entre finales de febrero hasta mediados de marzo y cómo todavía no había ganado la sufiente soltura en el desarrollo ocupó unas 40 horas de trabajo.
- Sprint 3 - Gestión de Asignaturas: Retomando a mediados de julio el proyecto con la determinación de presentarlo en la convotoría de septiembre y debido al mayor tiempo libre disponible en verano esta iteración se completo en una semana con otras 40 horas de trabajo.
- Sprint 4 - Gestión de Grupos: Continuando con el ritmo de trabajo adquirido ésta iteración se completó menos de una semana acumulando 30 horas de trabajo.
- Sprint 5 - Gestión de Usuarios en los Grupos: Siendo este sprint el único que tiene mayor carga de trabajo que el resto, llevo algo más de una semana completarlo pese a dedicarle bastante tiempo diario, llegando a acumular 40 horas de trabajo
- Sprint 6 - Gestión de Reuniones: continuando con los horarios ya establecidos se completo a lo largo de una semana dedicándole 30 horas.
- Sprint 7 - Gestión de Recursos Web y Flashcards: Con bastante más soltura y ánimo debido a los avances hasta el momento este sprint fue implementado en varios días de trabajo, acumulando 25 horas.
- Sprint 8 - Gestión de Encuestas: Igual que en caso anterior, y teniendo en cuenta que es algo menos carga de trabajo, supuso de nuevo unas 25 horas de trabajo realizadas en unos días.
- Sprint 9 - Gestión de Exámenes: Para acabar, este sprint supuso algo más de complejidad que el resto pero debido a la experiencia adquirida a lo largo del recorrido fue resuelto en una semana sumando 35 horas de trabajo.

Periodo	Objetivo	Horas
Conceptualización	Análisis y Diseño Inicial	30
Sprint 1	Gestión de Usuarios	45
Sprint 2	Gestión de Universidades	40
Sprint 3	Gestión de Asignaturas	40
Sprint 4	Gestión de Grupos	30
Sprint 5	Gestión de Usuarios en los Grupos	40
Sprint 6	Gestión de Reuniones	30
Sprint 7	Gestión de Rescursos Web y Flashcards	25
Sprint 8	Gestión de Encuestas	25
Sprint 9	Gestión de Exámenes	35
Total		340

Tabla 8.1: Resumen de la Planificación Temporal

Calculando la suma de todas las horas dedicadas dan un total de 340 horas de las cuales se estiman que 70 fueron dedicadas a las tareas descritas en el rol de analista y por lo tanto resultan en 270 horas realizadas por el programador.

8.2 Cálculo de coste del proyecto

Para el cálculo de costes de proyecto se ha considerado el salario medio según la página Talent [10] para cada rol de los comentados anteriormente y se han estimado unas 25 horas de dedicación al proyecto por parte del tutor.

- Analista software: 14,62€/h * 70h = 1203,4€
- Programador Junior en Java: 11,54€/h * 270h = 3115,8€
- Jefe de proyecto software: 24,36€/h * 25h = 609€

Por último, se debe sumar un 15% para los gastos generales, que incluyen suministros, y aplicar un IVA del 21% a todo el costo. Esto da como resultado el siguiente cálculo final:

Resursos humanos: 1203,4 € + 3115,8 € + 609 € = 4928,2 €

Gastos generales: 0.15 * 4928,2 € = 739,23 €

Coste total sin IVA: 4928,2 € + 739,23 € = 5.667,43 €

IVA (21%): 0.21 * 5.667,43 € = 1.190,16 €

Costo total con IVA: 5.667,43 € + 1.190,16 € = 6.857,59 €

Capítulo 9

Conclusiones

Último capítulo de la memoria, donde se revisan y evalúan los objetivos inciales respecto a los alcanzados, las lecciones aprendidas junto a la relación de las competencias adquiridas en la titulación y las posibles lineas futuras para la aplicación web.

9.1 Concordancia con objetivos iniciales

Una vez finalizado este proyecto, a continuación se realiza un análisis exhaustivo de cómo los resultados obtenidos se alinean con los objetivos establecidos en la etapa inicial con el propósito realizar un repaso crítico de cada uno de esos objetivos y valorar en qué medida han sido alcanzados.

- Respecto a la idea de desarrollar una aplicación extensible y escalable aplicando buenas técnicas y patrones de diseño, se destaca que este enfoque se mantuvo presente a lo largo de todo el proceso de desarrollo. Se aplicaron exitosamente los patrones DAO, DTO y Fachada, así como la técnica de Paginación, garantizando una estructura de datos eficiente y una separación clara entre capas. Además, el patrón de inversión de control, junto con la implementación de la inyección de dependencias, contribuyó a una arquitectura más modular y adaptable. Finalmente, el uso del patrón MultiProvider añadió una capa de flexibilidad al sistema. En conjunto, estas elecciones de diseño respaldaron la extensibilidad y escalabilidad de la aplicación de manera coherente con los objetivos iniciales.

- En relación al requisito de utilizar un subconjunto de tecnologías Java estándar y frameworks open source para la implementación del backend de la aplicación, se puede afirmar que este objetivo fue satisfactoriamente cumplido. Para la implementación, se seleccionaron tecnologías y frameworks como Java, JPA Hibernate, Spring, Spring Data, Spring Boot, Spring Security y JUnit. Estas opciones, todas ellas cumplen la descripción

del objetivo y forman un entorno completo, robusto y escalable en el backend.

- En relación al objetivo de desarrollar el frontend de la aplicación con el lenguaje Dart[4], específicamente utilizando Flutter Web[5], un framework open source desarrollado y compatible con Google, se confirma que este objetivo se cumplió exitosamente y que se llevaron a cabo exploraciones exhaustivas de diversas bibliotecas de Flutter Web, mencionadas en la sección 3.3.7, lo que enriqueció la experiencia de desarrollo y contribuyó a la creación de una interfaz de usuario rica y altamente funcional.

- Con respecto al objetivo de trabajar con un enfoque de desarrollo iterativo e incremental, es importante reconocer que este objetivo no se cumplió en su totalidad debido a las circunstancias particulares del proyecto. A pesar de la división de la carga de trabajo en iteraciones, como se detalló en la sección 5.4, se debe destacar que las iteraciones no se realizaron en sprints de tiempo fijo, como lo dicta la metodología Scrum[8]. En lugar de eso, cada iteración fue asignada un tiempo determinado basado en la disponibilidad de horas libres. Aunque si bien no se siguieron los plazos estables, el enfoque iterativo permitió una evolución gradual del proyecto con cada iteración agregando nuevas funcionalidades y mejoras.

- Por último, el objetivo de desarrollar las funcionalidades de la aplicación web se ha alcanzado en su totalidad. Este cumplimiento se ha confirmado mediante pruebas funcionales detalladas en la sección 7.4.3, validando con éxito cada característica implementada asegurando su correcto funcionamiento.

9.2 Lecciones aprendidas

A continuación, se presentan las lecciones más significativas derivadas del proceso de desarrollo de este proyecto, estableciendo su vinculación con las competencias correspondientes a la titulación y la mención de Tecnologías de la Información cursada por el alumno.

La primera lección reside en la adquisición y dominio de nuevas herramientas tecnológicas, habiendo tenido experiencia previa con Java, Dart[4] y SQL en diversas asignaturas de la carrera, como ”Deseño Software”, ”Interfaces Persoa Máquina” e ”Integración de Aplicacións”. A lo largo de este proyecto, fue esencial adentrarse en el uso efectivo de bibliotecas y frameworks de estos lenguajes que no conocía.

La familiarización con tecnologías como Spring Boot, Spring Data y Spring Security abrió nuevas perspectivas en el desarrollo de aplicaciones, permitiendo una integración ágil y segura de los componentes del backend. La comprensión y aplicación de JPA y Hibernate, si bien basadas en tecnologías familiares, revelaron capas más profundas de optimización y gestión de datos. Por otro lado, en el entorno frontend, destacar la introducción a Cached Network

Image en Flutter que me proporcionó herramientas esenciales para el manejo eficiente de imágenes, mejorando la calidad visual y la experiencia del usuario.

La segunda radica en el conocimiento adquirido sobre las metodologías de desarrollo de software. Aunque ya se había recibido formación en asignaturas como ”Proceso de Software” y ”Calidad en la Gestión TIC”, donde se exploraron diversas metodologías, como Scrum[8], la verdadera aplicación práctica y adaptación al proyecto aportaron un nivel de comprensión más profundo.

A pesar de la base teórica previa, la necesidad de fusionar elementos de Scrum con Kanban[9] para adaptar esta combinación a las necesidades específicas del proyecto resultó en un desafío a la vez que en una experienza muy enriquecedora.

Por último, se destaca la lección relacionada con la profundización en patrones ya conocidos, que previamente se habían abordado en asignaturas como ”Diseño de Software” e ”Internet y Sistemas Distribuidos”. Entre estos patrones se encuentran el DAO, el patrón de Inversión de Control y el uso de DTOs. Sin embargo, esta ocasión permitió una comprensión más profunda a través de su aplicación práctica en un proyecto real.

Además de explorar nuevas técnicas y patrones que no había utilizado previamente como la implementación de técnicas de paginación en la interfaz de usuario, la aplicación de inyección de dependencias para mejorar la modularidad y la adopción del patrón Multiprovider enriquecieron la caja de herramientas de desarrollo.

9.3 Futuras líneas de trabajo

En cuanto a las futuras líneas de trabajo, a pesar de haber cumplido con los objetivos establecidos en el proyecto, se identifican varias áreas donde aún es posible mejorar y agregar valor adicional a la aplicación, tal como se pudo observar en el Estado del Arte en el capítulo 2:

1. Diseño e Implementación de Juegos para el Aprendizaje: Explorar la integración de diferentes tipos de juegos educativos que podrían proporcionar a los usuarios una nueva forma interactiva y atractiva de mejorar su aprendizaje.

2. Capacidad de Convocar Reuniones Online con Videollamadas: Ampliar las capacidades de socialización de la aplicación, permitiendo que los grupos organicen reuniones en línea y realicen videollamadas dentro de la plataforma.

3. Repositorio de Recursos Públicos con Algoritmo de Recomendación: Implementar un repositorio central de recursos educativos, donde los usuarios puedan explorar y acceder a materiales compartidos por otros usuarios y agregar un algoritmo de recomendación que sugiera recursos relevantes según los intereses y estudios del usuario.

4. Integración con Google Maps para Visualización de Ubicaciones: En la función de búsqueda de universidades y grupos, incorporar la integración con Google Maps para mostrar las ubicaciones de manera visual, facilitando la identificación de opciones cercanas al usuario.

5. Expansión de la audiencia de la aplicación: Expandir la disponibilidad de la aplicación al desarrollar versiones tanto para escritorio como para dispositivos móviles. Aprovechar esta oportunidad para añadir más idiomas, lo que permitiría que la aplicación llegue a una audiencia más amplia y diversa.

Lista de acrónimos

API Application Programming Interface. iii, 7, 11, 35, 40, 63

CORS Cross Origin Resource Sharing. 8, 55

CRUD CREATE, READ, UPDATE and DELETE. 8, 40, 54

CSRF Cross-Site Request Forgery. 54

DAO Data Access Object. ii, 33, 38, 39, 48, 49, 69, 71

DTO Data Transfer Object. iii, 34, 40, 41, 46, 51, 52, 56, 69, 71

HTTP Hypertext Transfer Protocol. 11, 40, 53–56

IDE Integrated Development Environment. 10

iOS iPhone Operating System. 9

JPA Java Persistence AP. i, 7, 46, 69, 70

JSON JavaScript Object Notation. 40

JWT Json Web Token. 8, 40, 54

PDF Portable Document Format. 2, 17, 18, 22, 30, 37

POM Project Object Model. 10

REST REpresentational State Transfer. iii, 7, 11, 40, 41, 49, 51, 53, 63

SPA Single Page Application. 42

SQL Structured Query Language. i, 6, 7, 44

UML Unified Modeling Language. iii, 36

URL Uniform Resource Locator. 17

UTC Universal Time Coordinated. 52

Bibliografía

[1] "Study blue." [En línea]. Disponible en: https://www.studyblue.com

[2] "Quizlet." [En línea]. Disponible en: https://quizlet.com/es

[3] "Microsoft teams." [En línea]. Disponible en: https://www.microsoft.com/es-es/microsoft-teams/group-chat-software

[4] "Dart." [En línea]. Disponible en: https://dart.dev/overview

[5] "A fonfo con flutter en la web," 2023, consultado el 8 de septiembre de 2023. [En línea]. Disponible en: https://www.codemotion.com/magazine/es/frontend-es/desarrollo-web/a-fondo-con-flutter-en-la-web/

[6] "Balsamiq wireframes." [En línea]. Disponible en: https://balsamiq.com/wireframes/

[7] S. Laoyan, "Scrumban: lo mejor de dos metodologías ágiles," 2022, consultado el 8 de septiembre de 2023. [En línea]. Disponible en: https://asana.com/es/resources/scrumban

[8] J. S. Hurtado, "Cómo funciona la metodología scrum: Qué es y cómo utilizarla," 2021, consultado el 8 de septiembre de 2023. [En línea]. Disponible en: https://www.iebschool.com/blog/metodologia-scrum-agile-scrum/

[9] L. C. Lendínez, "Kanban. metodología para aumentar la eficiencia de los procesos," 2019, consultado el 8 de septiembre de 2023. [En línea]. Disponible en: https://www.3ciencias.com/wp-content/uploads/2019/03/ART.-2-TECNO-Ed.-29_Vol.-8_n%C2%BA-1-1.pdf

[10] "Talent: Salarios en españa 2023." [En línea]. Disponible en: https://es.talent.com/salary