ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA DE SISTEMAS

DESARROLLO DE UN SISTEMA DE TOMA DE DECISIONES PARA APOYO A LOS INVESTIGADORES ECUATORIANOS BASADO EN MOTORES DE BÚSQUEDA Y DE RECOMENDACIÓN

DESARROLLO DE MÓDULO QUE IMPLEMENTE FUNCIONES DEL TIPO RED SOCIAL PARA LOS USUARIOS DEL SISTEMA CENTINELA

TRABAJO DE INTEGRACIÓN CURRICULAR PRESENTADO COMO REQUISITO PARA LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN SOFTWARE

DANNY RUBÉN CABRERA AGUILAR
danny.cabrera@epn.edu.ec

DIRECTOR: PhD. LORENA KATHERINE RECALDE CERDA
lorena.recalde@epn.edu.ec

DQM, JULIO 2024

CERTIFICACIONES

Yo, Danny Rubén Cabrera Aguilar , declaro que el trabajo de integración curricular aquí descrito es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

Danny Rubén Cabrera Aguilar

Certifico que el presente trabajo de integración curricular fue desarrollado por Danny Rubén Cabrera Aguilar , bajo mi supervisión.

PhD. Lorena Katherine Recalde Cerda
DIRECTOR

DECLARACIÓN DE AUTORÍA

A través de la presente declaración, afirmamos que el trabajo de integración curricular aquí descrito, así como el (los) producto(s) resultante(s) del mismo, son públicos y estarán a disposición de la comunidad a través del repositorio institucional de la Escuela Politécnica Nacional; sin embargo, la titularidad de los derechos patrimoniales nos corresponde a los autores que hemos contribuido en el desarrollo del presente trabajo; observando para el efecto las disposiciones establecidas por el órgano competente en propiedad intelectual, la normativa interna y demás normas.

Danny Rubén Cabrera Aguilar

Lorena Katherine Recalde Cerda

Rommel Paúl Masabanda Castro, Johan Fernando Sangopanta Naula, Joffre Alexander Cóndor Tipán,

DEDICATORIA

A mi familia, cuya presencia ha sido mi mayor fortaleza y fuente de inspiración a lo largo de este viaje. A mi papá, Edgar Cabrera, mi mamá, Angelita Aguilar, y mi hermano, Alexander Cabrera, por su amor incondicional y apoyo constante. Sus palabras de aliento y su fe inquebrantable en mis capacidades me han dado el valor para enfrentar y superar cada desafío. Sin ustedes, este logro no hubiera sido posible. Su comprensión y paciencia han sido un refugio en los momentos de incertidumbre, y su alegría en mis éxitos ha sido una motivación constante para seguir adelante.

A la vida, que con sus altibajos, me ha enseñado lecciones valiosas y me ha moldeado en la persona que soy hoy. Agradezco cada tropiezo, cada caída, y cada fracaso, porque en ellos encontré la fuerza para levantarme y seguir adelante. Cada obstáculo superado me ha acercado un paso más a mis sueños y me ha permitido descubrir mi verdadera pasión. La vida, con su incesante mezcla de desafíos y triunfos, me ha brindado la oportunidad de crecer, aprender y evolucionar.

AGRADECIMIENTO

En primer lugar, quiero agradecer a mi familia, que ha sido mi pilar durante todo este recorrido. A mi papá, Edgar Cabrera, mi mamá, Angelita Aguilar, y mi hermano, Alexander Cabrera, por su amor incondicional, comprensión y apoyo constante. Sus palabras de aliento y confianza en mis capacidades me impulsaron a seguir adelante en los momentos más difíciles.

A mis compañeros de tesis, con quienes desarrollé este proyecto: su colaboración, ideas y esfuerzo compartido fueron esenciales para alcanzar nuestros objetivos comunes. Trabajar juntos ha sido una experiencia enriquecedora que siempre recordaré con gratitud.

A mi tutora, la PhD. Lorena Katherine Recalde Cerda, por su guía, apoyo constante y valiosa orientación a lo largo de este proceso. Su conocimiento y paciencia fueron fundamentales para la realización de este trabajo.

Quiero también agradecerle a la vida, que aunque me ha hecho tropezar y fallar en varias ocasiones, siempre me ha dado la fuerza para levantarme y seguir adelante. Estas experiencias me ayudaron a encontrar el camino hacia lo que realmente me apasiona, dándome la oportunidad de crecer y aprender de cada obstáculo superado.

Finalmente, agradezco a la Escuela Politécnica Nacional por brindarme la formación académica y las herramientas necesarias para alcanzar mis metas.

ÍNDICE DE CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

RESUMEN

La investigación científica en Ecuador ha experimentado un notable crecimiento en la última década. Sin embargo, los investigadores aún enfrentan desafíos significativos para acceder a información relevante y colaborar eficazmente. Este proyecto propone la integración de las plataformas ResNet y ReSearchDecide en una sola plataforma integral denominada CENTINELA. La nueva plataforma busca mejorar la eficiencia y efectividad de la investigación científica en Ecuador mediante la implementación de funciones tipo red social para los usuarios.

El desarrollo del componente B, parte esencial de CENTINELA, incluyó la revisión sistemática de la literatura, la definición de requisitos técnicos y funcionales, la creación de mockups, el desarrollo del frontend y backend, y la integración de ambos. El componente B permite la gestión de perfiles de usuario, la creación de publicaciones y la formación de grupos de investigación. La integración de estas funcionalidades promueve la colaboración académica y mejora el acceso a información relevante para los investigadores.

PALABRAS CLAVE - Integración de plataformas, red social académica, gestión de perfiles, colaboración científica, Ecuador

ABSTRACT

Scientific research in Ecuador has experienced remarkable growth in the last decade. However, researchers still face significant challenges in accessing relevant information and collaborating effectively. This project proposes the integration of the ResNet and ReSearch-Decide platforms into a single comprehensive platform called CENTINELA. The new platform aims to improve the efficiency and effectiveness of scientific research in Ecuador by implementing social network-like functions for users.

The development of component B, an essential part of CENTINELA, included a systematic literature review, the definition of technical and functional requirements, the creation of mockups, the development of the frontend and backend, and the integration of both. Component B allows for user profile management, the creation of publications, and the formation of research groups. The integration of these functionalities promotes academic collaboration and improves access to relevant information for researchers.

KEYWORDS - Platform integration, academic social network, profile management, scientific collaboration, Ecuador

1. INTRODUCCIÓN

1.1. Planteamiento del Problema

En el contexto de la creciente producción científica en Ecuador y la necesidad de optimizar los recursos para abordar los complejos desafíos de la investigación, surge la motivación para desarrollar un proyecto que promueva la colaboración entre investigadores ecuatorianos. A pesar del incremento en la producción científica y la existencia de plataformas como REDI[1], los investigadores en Ecuador enfrentan desafíos significativos para acceder de manera rápida y efectiva a información relevante. La falta de interoperabilidad entre las plataformas existentes obstaculiza el intercambio eficiente de información y la colaboración efectiva entre investigadores.

Actualmente, existen dos proyectos de pregrado enfocados en mejorar la colaboración académica y la eficiencia en la toma de decisiones, que constituyen la base de este nuevo proyecto. El primero de ellos, ResNet, se dedica a construir redes de coautoría a partir de publicaciones científicas usando datos de Scopus [1], permitiendo a los investigadores obtener una visión clara de la producción científica del país. El segundo, ReSearchDecide, aborda la complejidad y el tiempo requeridos para alcanzar consensos y gestionar la adscripción de miembros dentro de un grupo de investigación [2], [3].

La problemática principal radica en la falta de interoperabilidad entre estas plataformas, lo que impide una colaboración eficiente entre investigadores. Esta falta de integración resulta en una duplicación de esfuerzos y una menor eficiencia en la formación de grupos de investigación coherentes dentro de la comunidad científica ecuatoriana. Por lo tanto, el objetivo principal del proyecto es unificar estas plataformas en una sola plataforma integral denominada CENTINELA, facilitando el acceso rápido a información relevante y mejorando la calidad de la investigación en Ecuador.

Para resolver los problemas mencionados, se propone combinar las fortalezas de ambas plataformas y ampliar sus funcionalidades, dando lugar a CENTINELA. Este proyec-

[1] REDI es una red internacional destinada a facilitar la colaboración entre investigadores de diversas disciplinas. Para más información, consulte: `https://www.redi.org`

to representa un esfuerzo significativo para centralizar la información sobre la producción científica del país y formar grupos de trabajo que puedan consensuar sobre temas de investigación. El objetivo principal de la versión 1 de CENTINELA es unificar las plataformas ResNet y ReSearchDecide en una sola, preparando el terreno para la incorporación de nuevas funcionalidades que la enriquezcan y la conviertan en una herramienta atractiva para sus usuarios.

En términos generales, esta primera iteración de CENTINELA se compone de cuatro componentes, los cuales se ilustran claramente en la Figura 1.1.

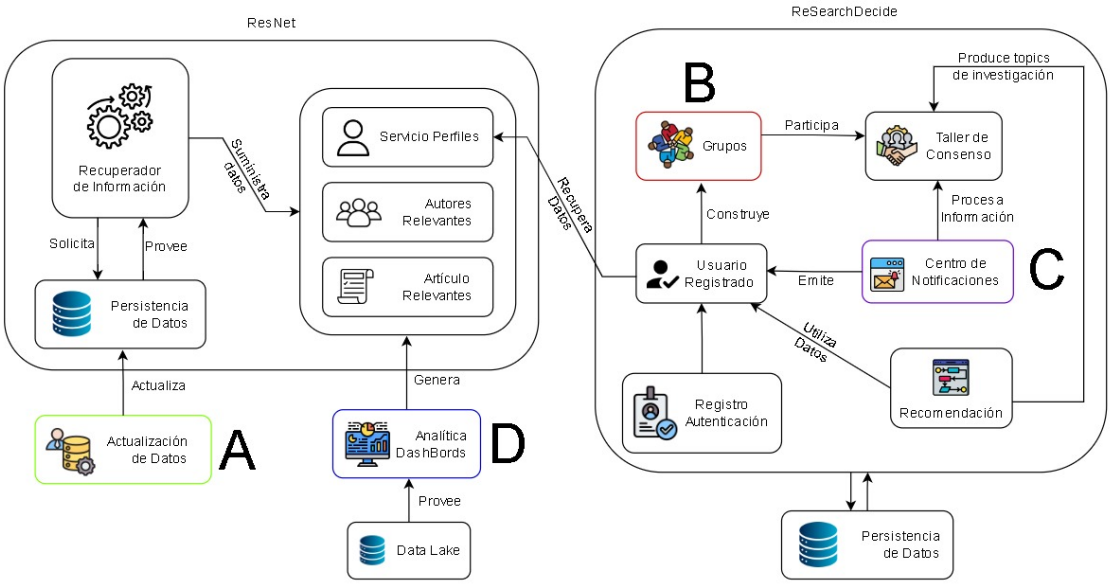


Figura 1.1: Diagrama general del Sistema CENTINELA, unificando los sistemas ResNet, ReSearchDecide y sus respectivos componentes A, B, C, D.

- **Componente A**: Responsable de migrar ResNet a CENTINELA y automatizar la recopilación de datos de Scopus, una funcionalidad que el primer sistema no poseía.
- **Componente B**: Encargado de migrar ReSearchDecide a CENTINELA e incluir características fundamentales de una red social.
- **Componente C**: Se ocupa de migrar ReSearchDecide a CENTINELA, mejorar el proceso de consenso e incluir un centro de notificaciones.
- **Componente D**: Responsable de la migración de ResNet a CENTINELA e incorporación de paneles de control que permitan visualizar la evolución y estado de la produc-

ción científica en Ecuador.

En este documento se abordarán los aspectos fundamentales de CENTINELA, con un enfoque especial en la migración tecnológica del sistema ReSearchDecide. La Figura 1.2, derivada de la Figura 1.1, ilustra el componente B, el cual se centra en la implementación de funcionalidades de red social para los usuarios del sistema CENTINELA.

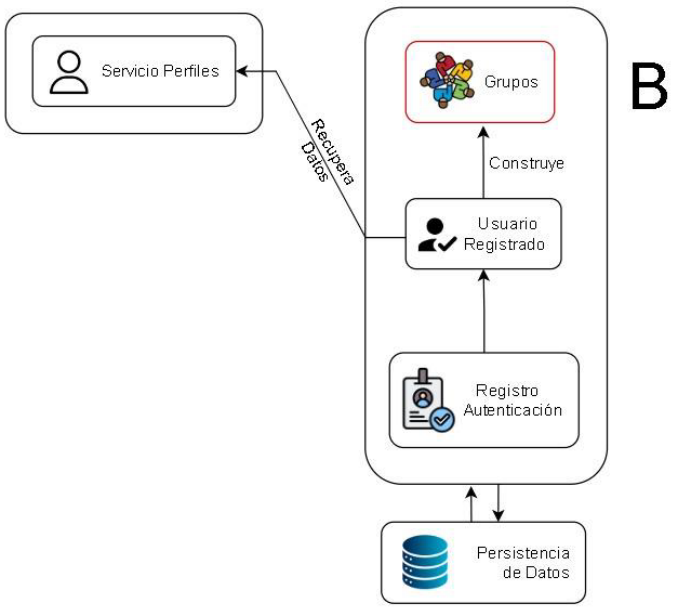


Figura 1.2: Diagrama del Componente B del Sistema CENTINELA.

El componente B se enfoca en la incorporación de características propias de las redes sociales para los usuarios del sistema CENTINELA. Esta migración tecnológica incluye tanto el frontend como el backend. En cuanto al frontend, se realizará una transición de React a Angular con el objetivo de mejorar la interacción del usuario y la estética del sistema. En el backend, la versión 1 (V1) de ReSearchDecide utilizaba Firebase (Backend-as-a-Service), lo cual presentaba limitaciones en cuanto a personalización y escalabilidad. Por consiguiente, se optó por desarrollar un backend REST API desde cero utilizando Django Rest Framework y PostgreSQL, permitiendo así un control total sobre la lógica del servidor.

1.2. Justificación Teórica

La integración de plataformas para la colaboración científica se sustenta en la teoría de las redes sociales y la teoría de la coautoría. Las redes sociales académicas y las redes de coautoría son fundamentales para el avance científico, ya que facilitan la interacción y

colaboración entre investigadores. La teoría de sistemas de recomendación también juega un papel crucial, ya que estos sistemas pueden personalizar la experiencia del usuario y resaltar contenido relevante, mejorando la visibilidad de temas y publicaciones relevantes para los investigadores. Este proyecto se basa en estas teorías para diseñar un sistema que optimice la colaboración y el intercambio de información entre investigadores ecuatorianos.

Las redes sociales académicas se definen como plataformas que facilitan la interacción y comunicación instantánea entre investigadores, permitiéndoles compartir recursos de información y documentación de manera eficiente [4]. Estas redes no solo fomentan la creación de conexiones profesionales, sino que también permiten el desarrollo de identidades profesionales a través de perfiles, y la difusión de información relevante en la comunidad científica.

La teoría de la coautoría resalta la importancia de las colaboraciones científicas y cómo estas contribuyen significativamente al avance del conocimiento. Las redes de coautoría permiten a los investigadores beneficiarse de la diversidad de ideas y enfoques, lo que a su vez incrementa la calidad y el impacto de las investigaciones [5]. Estas redes facilitan la creación de mapas de conocimiento, que son esenciales para la identificación de áreas emergentes y la estructuración de campos de estudio.

Por otro lado, los sistemas de recomendación (RS) juegan un papel crucial en el entorno académico al personalizar la experiencia del usuario. Estos sistemas pueden analizar el comportamiento y las preferencias de los investigadores para sugerir contenido relevante, mejorando así la visibilidad de publicaciones y temas de interés [6]. La implementación de RS en plataformas académicas facilita el descubrimiento de recursos valiosos y la conexión con otros investigadores que comparten intereses similares.

En resumen, este proyecto se basa en la teoría de las redes sociales académicas, la teoría de la coautoría y la teoría de sistemas de recomendación para desarrollar una plataforma que optimice la colaboración y el intercambio de información entre investigadores ecuatorianos. Estas teorías proporcionan una base sólida para justificar la necesidad de una herramienta que no solo facilite la comunicación y colaboración, sino que también promueva la difusión eficiente del conocimiento científico.

1.3. Justificación Metodológica

La metodología ágil Scrum se ha seleccionado para el desarrollo del proyecto debido a su flexibilidad y capacidad para adaptarse a cambios en los requisitos [7]. Scrum permite la entrega incremental y continua de valor para el usuario mediante la realización de sprints cortos y frecuentes. Esta metodología incluye reuniones diarias para discutir las actividades y posibles obstáculos, y retrospectivas para analizar y mejorar el trabajo realizado en cada sprint. La adopción de Scrum garantizará que el proyecto se desarrolle de manera iterativa y que se puedan realizar ajustes rápidos en respuesta a nuevas necesidades o desafíos.

La flexibilidad de Scrum radica en su capacidad para adaptarse a los cambios de requisitos a lo largo del ciclo de desarrollo, lo cual es esencial en entornos dinámicos y complejos. Además, esta metodología facilita la implementación de cambios mediante ciclos de desarrollo cortos y retrospecciones periódicas, lo que permite a los equipos de desarrollo reaccionar rápidamente a las necesidades cambiantes y mejorar continuamente los procesos y resultados del proyecto. Esta capacidad de adaptación es especialmente crucial en proyectos de investigación y desarrollo, donde los objetivos y las prioridades pueden evolucionar rápidamente.

Por lo tanto, la metodología Scrum no solo proporciona un marco flexible y adaptable, sino que también promueve la entrega continua de valor y la mejora constante, lo cual es fundamental para el éxito del proyecto CENTINELA.

1.4. Objetivos

1.4.1. Objetivo general

Unificar las plataformas ResNet y ReSearchDecide en una sola plataforma integral denominada CENTINELA, que implemente funciones tipo red social para los usuarios del sistema.

1.4.2. Objetivos específicos

- Analizar y comprender en detalle los sistemas ResNet y ReSearchDecide para identificar sus funcionalidades, requerimientos y posibles áreas de mejora.

- Diseñar e implementar una arquitectura de integración que permita la interoperabilidad entre los diferentes sistemas, garantizando la coherencia de datos y la compatibilidad de las plataformas.
- Desarrollar y añadir un módulo de perfil y autenticación que implemente funciones similares a una red social para los usuarios del sistema.
- Entregar un sistema integrado visualmente cohesivo, asegurando que los usuarios finales perciban una experiencia de uso uniforme y consistente.

1.5. Alcance

El alcance del proyecto se centra en la integración de los sistemas ResNet y ReSearch-Decide, sirviendo como base para el desarrollo de otros componentes. El objetivo es asegurar la interoperabilidad entre ambos sistemas y renovar la interfaz de usuario centralizándola en una única plataforma. Esto implica la unificación de la experiencia del usuario y la optimización de la eficiencia operativa al consolidar funcionalidades clave en un solo punto de acceso. Además, se implementarán funcionalidades de red social en el módulo de perfil de usuario, permitiendo la creación de perfiles, publicaciones y grupos de investigación.

Basándonos en los prototipos de ambos proyectos que ya cuentan con requisitos definidos e implementados, se propone la adopción del marco ágil Scrum para garantizar la flexibilidad necesaria y la entrega iterativa e incremental del software a través de sprints.

1.6. Marco Teórico

El desarrollo de una plataforma de red social requiere una comprensión profunda de los conceptos y funcionalidades que definen a este tipo de servicios. Este marco teórico se centra en la estructura y propósito de las redes sociales, con un enfoque particular en aquellas destinadas al ámbito académico. A través de este análisis, se destacan los principales objetivos y características de las redes sociales académicas, la dinámica de los grupos y comunidades que las integran, así como la importancia de la visualización de temas relevantes y las redes de coautoría. Este marco ofrece una base sólida para el diseño y desarrollo de una plataforma que no solo facilite la comunicación y colaboración entre los usuarios, sino que también promueva la difusión y el intercambio eficiente de conocimientos en el entorno académico.

1.6.1. Redes Sociales

Una red social puede conceptualizarse como una plataforma en línea que facilita la conexión, el intercambio de información, la coordinación de actividades y, en líneas generales, el mantenimiento de la comunicación entre sus usuarios [8]. En este contexto, se argumenta que las funciones de una red social buscan acercar a los usuarios, permitiéndoles identificar grupos afines donde cada miembro pueda seguir o unirse en base a intereses compartidos.

Redes Sociales Académicas

Una red social académica tiene como propósito facilitar la interacción y comunicación instantánea entre investigadores, permitiéndoles compartir de manera rápida y simultánea diversos recursos de información y documentación [4]. Para lograr este propósito, se establece una serie de objetivos que deben cumplirse para considerarla efectivamente una red social académica.

Objetivos de las Redes Sociales Académicas [4]

- Establecer conexiones profesionales mediante redes de contactos.
- Crear grupos de interés para debatir y compartir recursos.
- Desarrollar una identidad profesional a través de perfiles.
- Difundir y compartir información, experiencias profesionales, invitaciones a eventos, conmemoración de efemérides, lecturas e ideas.
- Publicar contenidos relevantes, opiniones, entre otros.
- Servir como medio de comunicación eficiente entre profesionales.

1.6.2. Grupos y Comunidades

Desde una perspectiva social, definimos un grupo como un conjunto específico de individuos que colaboran a lo largo de un periodo extenso con el propósito de alcanzar una meta compartida. Durante este proceso continuo de comunicación e interacción, se establecen normas comunes y se asignan tareas, fomentando así el desarrollo de un sentido de solidaridad entre sus miembros [9].

En la actualidad, las plataformas de investigación en redes sociales facilitan la creación de comunidades que se forman con el objetivo de compartir resultados y colaborar, ya sea de manera explícita o implícita. Gracias al uso de Internet, la ubicación física ya no representa un obstáculo para el intercambio fácil y libre de información. En consecuencia, las herramientas web y las redes sociales de investigación han ampliado significativamente las oportunidades de colaboración, incluso cuando los investigadores no han tenido previamente contacto en sus instituciones o en eventos científicos [10].

1.6.3. Visualización de Temas Relevantes

La presentación de temas relevantes desempeña un papel crucial en mantener a los usuarios informados y comprometidos. Este proceso implica analizar el contenido generado por los usuarios y resaltar los temas más pertinentes mediante los Sistemas de Recomendación (RSs).

Sistemas de Recomendación

Los Sistemas de Recomendación son herramientas informáticas diseñadas para ofrecer sugerencias personalizadas a los usuarios, basándose en su historial de uso, preferencias y comportamiento. Estas recomendaciones buscan mejorar la experiencia del usuario al proporcionar opciones relevantes y afines a sus intereses [6].

Existen dos enfoques principales en los Sistemas de Recomendación: el filtrado colaborativo y el filtrado basado en contenido. En el filtrado colaborativo, se parte de la premisa de que si a un usuario le gustó un artículo específico, es probable que disfrute de otros artículos que hayan sido del agrado de usuarios con preferencias similares. Por otro lado, el filtrado basado en contenido sugiere productos similares a los que el usuario ha preferido en el pasado, basándose en las características de los productos. Esta estrategia tiene la ventaja de ofrecer recomendaciones incluso en ausencia de otros usuarios con gustos afines [6].

Cuando aplicamos estos principios a una red social, se facilita que los usuarios encuentren tanto temas de su interés como grupos afines a sus áreas de estudio. De esta manera, la recomendación no solo se limita a contenidos individuales, sino que también se extiende a la posibilidad de conectar a los usuarios con comunidades que compartan sus afinidades.

1.6.4. Redes de Coautoría

En la actualidad, el análisis de la producción científica ha adquirido una importancia significativa, impulsado por los estudios bibliométricos y cienciométricos. Estos estudios no solo estructuran los ámbitos científicos a través de fuentes bibliográficas, sino que también desentrañan las tácticas para identificar a los autores, comprender sus relaciones y analizar las tendencias de investigación [5].

Impacto y reconocimiento

Más allá de la eficiencia medida en términos de la cantidad de publicaciones y la contribución global de los autores, se destacan dos elementos cruciales en la producción científica: la formación de redes de colaboración y la creación de mapas de conocimiento. La colaboración científica emerge como un componente esencial en la labor del investigador, no solo contribuyendo a la internacionalización, sino también proporcionando acceso a recursos y prestigio de otros investigadores e instituciones [5].

La colaboración, tanto a nivel nacional como internacional, no solo aborda problemas sociales desde perspectivas diversas, sino que también fomenta el espíritu crítico y mejora la comunicación en la comunidad científica. Esta cooperación no solo contribuye al desarrollo de las sociedades, sino que también enriquece la investigación gracias a la diversidad de culturas y conocimientos. De hecho, los organismos reguladores del avance científico reconocen la importancia de la colaboración en la investigación como un catalizador para estudios de mayor calidad y relevancia [5].

El segundo componente crucial se vincula a la identificación de mapas de conocimiento derivados de la actividad científica. Estos mapas no solo son herramientas visuales para la representación de la actividad científica, sino que también facilitan la ubicación y visualización del conocimiento. La utilidad de esta herramienta radica en su capacidad para desvelar de manera exhaustiva la estructura de una disciplina, permitiendo descubrir subcampos, ejes temáticos y dominios dentro de una dimensión evaluada, diferenciando entre temas principales y auxiliares [5].

1.6.5. Arquitectura Hexagonal

La **arquitectura hexagonal**, también conocida como *ports and adapters* u *onion architecture*, es un patrón de diseño de software que enfatiza la separación entre la lógica del negocio y las interfaces externas. Este diseño permite que el núcleo de la aplicación sea independiente de la infraestructura, facilitando la prueba y el mantenimiento del sistema. En esta arquitectura, los componentes externos se conectan a la lógica del negocio a través de “puertos” y “adaptadores”, lo que permite que diferentes tecnologías interactúen con el núcleo de manera intercambiable y sin afectar su funcionamiento interno [11].

La arquitectura hexagonal se ha utilizado en este proyecto debido a sus múltiples beneficios. Principalmente, mejora la escalabilidad y mantenibilidad del sistema al permitir que la lógica del negocio sea probada de forma aislada, sin depender de las tecnologías externas. Además, facilita la adaptabilidad del software, permitiendo que cambios en la infraestructura no afecten el núcleo del negocio, lo que resulta en un desarrollo más ágil y eficiente [12].

1.6.6. Herramientas utilizadas

A continuación, en la Tabla 1.1 se presenta un desglose de las herramientas empleadas.

Herramienta	Descripción	Versión Usada
Git	Git es un sistema de control de versiones distribuido, lo que implica que un clon local del proyecto funciona como un repositorio de control de versiones completo [13].	2.41.0.windows.3
Tailwind	Tailwind es un marco de trabajo CSS altamente personalizable, que permite a los desarrolladores implementar rápidamente diseños sin necesidad de escribir CSS personalizado [14].	v3

Herramienta	Descripción	Versión Usada
Node	Node.js es un entorno de ejecución para JavaScript que permite a los desarrolladores ejecutar código JavaScript en el servidor, facilitando la creación de aplicaciones escalables [15].	18.20.2
Angular	Angular es una plataforma y framework para desarrollar aplicaciones web de una sola página usando HTML y TypeScript. Es mantenido por Google [16].	16.2.14
Typescript	TypeScript es un superset de JavaScript que añade tipado estático opcional y otras características avanzadas, facilitando la creación de aplicaciones más robustas [17].	5.5.4
FontAwesome	FontAwesome es una biblioteca de íconos que proporciona una amplia gama de íconos vectoriales personalizables, utilizados comúnmente en el desarrollo web [18].	6.6.0
PostgreSQL	PostgreSQL es un sistema de gestión de bases de datos relacional orientado a objetos (ORDBMS) de código abierto que soporta gran parte del estándar SQL. Es conocido por sus características avanzadas como consultas complejas, integridad transaccional y control de concurrencia multiversión, además de su alta extensibilidad [19].	16

Herramienta	Descripción	Versión Usada
Django REST Framework	Django REST Framework es un poderoso y flexible kit de herramientas para construir APIs web en Django. Facilita la creación rápida y sencilla de APIs robustas [20].	3.15.1
Docker	Docker es una plataforma de contenedorización que permite a los desarrolladores empaquetar aplicaciones y sus dependencias en contenedores, asegurando que se ejecuten de manera consistente en cualquier entorno [21].	26.1.4
Figma	Figma es una herramienta de diseño gráfico y prototipado colaborativa basada en la web, que permite a los equipos trabajar juntos en tiempo real para crear y compartir diseños [22].	

Tabla 1.1: Herramientas utilizadas.

2. METODOLOGÍA

En este capítulo se describen las etapas que se han llevado a cabo en Sprints para el desarrollo de este proyecto.

2.1. Introducción a la metodología de desarrollo ágil

El presente proyecto de tesis se basará en un enfoque ágil para la gestión del trabajo en equipo en el desarrollo de proyectos. Para ello, se seguirá el Manifiesto Ágil, el cual es un conjunto de cuatro valores y diecisiete principios que permiten lograr una mayor agilidad en el trabajo dentro de un proyecto.

Los valores del Manifiesto Ágil son los siguientes [23]:

1. **Individuos e interacciones por encima de los procesos y las herramientas:** Priorizar la comunicación y colaboración entre los miembros del equipo.
2. **Software funcionando por encima de la documentación extensa:** Enfocarse en entregar software funcional que cumpla con los requisitos del cliente.
3. **Colaboración con el cliente por encima de la negociación del contrato:** Fomentar una relación de trabajo continua con el cliente para asegurar que el producto final cumpla con sus necesidades.
4. **Respuesta ante el cambio por encima del seguimiento de un plan:** Ser flexible y adaptarse a los cambios que surjan durante el desarrollo del proyecto.

Tomando en cuenta las directrices proporcionadas por Schwaber [24], el proyecto de tesis adoptará un enfoque ágil, considerando la naturaleza cambiante de los requerimientos en el desarrollo de software. La decisión de utilizar la metodología Scrum se basa en su capacidad para gestionar proyectos de manera adaptable y eficiente.

Scrum, siendo un marco ligero, permite a individuos, equipos y organizaciones generar valor mediante soluciones adaptables a problemas complejos [24]. Su enfoque iterativo e incremental proporciona una mayor previsibilidad y control sobre los riesgos asociados al desarrollo de software.

En la práctica, Scrum reúne a un grupo de personas con habilidades diversas y complementarias, formando un equipo cohesionado y autónomo. Los valores fundamentales de Scrum, como el compromiso, el enfoque, la apertura, el respeto y el coraje, son la base de su éxito [24]. Los miembros del equipo Scrum se comprometen a alcanzar los objetivos establecidos, apoyándose mutuamente y trabajando en colaboración.

El método principal de trabajo en Scrum se realiza a través de sprints, periodos de tiempo cortos y definidos donde se concentran los esfuerzos en la consecución de objetivos específicos [24]. Esta práctica permite mejorar continuamente el progreso del proyecto y garantizar el cumplimiento de los objetivos establecidos.

El equipo Scrum está compuesto por roles específicos, incluyendo al Scrum Master, quien actúa como facilitador y líder del equipo, al propietario del producto (Product Owner), responsable de definir y priorizar los requisitos del producto, y a los desarrolladores, encargados de llevar a cabo la implementación técnica [24]. Dentro del equipo Scrum, no existen jerarquías rígidas; todos los miembros son profesionales con un objetivo común: la entrega exitosa del Producto.

2.2. Ciclo de Desarrollo Iterativo del Software

El ciclo de desarrollo de software que se empleará para la construcción del componente *B*, desarrollo de componente que implemente funciones del tipo red social para los usuarios del sistema, constará de 7 sprints, como se puede observar en la Tabla 2.1.

Id	**Nombre**	**Descripción**
S0	Revisión Sistemática y Definición de Requisitos	En este sprint, se llevará a cabo una revisión sistemática de la literatura existente para comprender a fondo los conceptos, tecnologías y mejores prácticas relevantes para el desarrollo del componente *B*. El objetivo es establecer un sólido marco conceptual que guíe el diseño e implementación del mismo, y definir los requisitos técnicos y funcionales.

Id	Nombre	Descripción
S1	Elaboración de Mockups y Definición de Estilos	Durante este sprint, se crearán mockups detallados que representen la interfaz de usuario del sistema integrado, teniendo en cuenta los requisitos y la arquitectura definidos previamente. También se definirán los estilos y colores a utilizar en el frontend. El objetivo es visualizar y validar el diseño de la interfaz antes de proceder con la implementación.
S2	Desarrollo de Pantallas de Registro y Login	En este sprint, se llevará a cabo el desarrollo de las interfaces frontend para las pantallas de registro y login, asegurando su adaptabilidad a diferentes dispositivos y siguiendo el principio de *Mobile First*.
S3	Desarrollo de Pantalla de Perfil de Usuario	Durante este sprint, se desarrollará la pantalla de perfil de usuario, incluyendo la sección “About” del perfil, permitiendo a los usuarios agregar y editar información personal, disciplinas y detalles de contacto.
S4	Desarrollo de Funcionalidades de Login y Usuario en Backend	En este sprint, se crearán los modelos y serializadores para la gestión de usuarios en el backend utilizando Django, y se desarrollarán vistas y endpoints necesarios para la autenticación y gestión de perfiles de usuario.
S5	Desarrollo de Funcionalidades de Publicaciones y Grupos en Backend	Durante este sprint, se crearán los modelos y serializadores para la gestión de publicaciones y grupos en el backend utilizando Django, y se desarrollarán vistas y endpoints necesarios para la gestión de publicaciones y grupos.

Id	Nombre	Descripción
S6	Integración Final y Configuración de Docker	En este sprint, se integrará la comunicación entre el frontend y el backend mediante APIs RESTful, se configurará el entorno de desarrollo usando Docker, y se realizarán pruebas de integración para asegurar el funcionamiento correcto de la aplicación.

Tabla 2.1: Sprints.

2.3. Roles

Como fue mencionado anteriormente en la Sección 2.1, el flujo de trabajo a desarrollar será Scrum, en el cual se establecen varios roles. En primer lugar, tenemos al dueño del producto o Product Owner, quien se encarga de proporcionar las funcionalidades del producto de software y de priorizar la lista de productos entregables que se deben realizar. El segundo rol es el equipo encargado del desarrollo o Development Team, quienes tendrán la tarea en este caso de implementar los componentes necesarios para las funcionalidades de la red social. Dado que se requiere la integración de 2 plataformas como ResNet y ResearchDecide en una sola, se deben recrear funcionalidades como el registro y autenticación de usuarios que originalmente fueron realizadas mediante Firebase. Además, se procederá a la creación de un perfil para cada usuario registrado con las respectivas funcionalidades de red social y una nueva interfaz para la creación de grupos para el Sistema de Toma de Decisiones y Consenso Grupal del correspondiente componente *C*.

Al ser un desarrollo mediante el marco de trabajo Scrum, el equipo es autogestionado, lo que implica que se decide de manera interna qué es lo que se va a realizar, cuándo y cómo. Por ende, son responsables de construir el producto de software de manera autónoma y con la debida responsabilidad [24].

Finalmente, el tercer rol es el Scrum Master, quien es el encargado de guiar al equipo de desarrollo basado en el marco de trabajo Scrum. En la Tabla 2.2 se observarán los encargados de ejercer cada rol del marco de trabajo Scrum.

Rol	Responsabilidad	Responsable
Product Owner	Proporcionar las funcionalidades del producto y priorizar los entregables	Lorena Recalde
Scrum Master	Guiar al equipo de desarrollo según el marco de trabajo Scrum	Lorena Recalde
Development Team	Implementar los componentes del software	Danny Cabrera

Tabla 2.2: Roles en el marco de trabajo Scrum

2.4. Requisitos y especificaciones

Los requisitos del componente del sistema se obtuvieron mediante charlas y lluvias de ideas entre los integrantes del equipo de desarrollo. En la Tabla 2.3, se muestra una lista de los requerimientos obtenidos como resultado de estas actividades.

Id	Requisito	Complejidad
01	Integrar las plataformas ResNet y ReSearchDecide en un sistema unificado	Muy alta
02	Diseñar y desarrollar un frontend unificado para la nueva plataforma	Alta
03	Migrar el sistema de autenticación de Firebase a una base de datos SQL	Alta
04	Implementar funcionalidades de red social en el módulo de perfil de usuario	Alta
05	Crear un backend con REST API y dockerizarlo para el módulo de red social	Muy alta
06	Realizar pruebas de usabilidad y aceptación para el sistema integrado	Media

Tabla 2.3: Requerimientos del componente del sistema.

2.5. Product Backlog

El product backlog es una lista de requisitos detallados y priorizados para el desarrollo del producto, obtenidos a partir de diversas fuentes y la colaboración del equipo de desarrollo. Estos requisitos incluyen tanto aspectos técnicos como de negocio, y son dinámicos, evolucionando a medida que se adquiere más conocimiento sobre el producto y las necesidades de los usuarios.

Tomando esto en cuenta, se procederá con el siguiente trabajo aplicando los siguientes ítems:

- **Épica**: Representa un conjunto de funcionalidades que abarcan múltiples lanzamientos del producto.
- **Funcionalidad**: Se refiere a un conjunto de historias de usuario que representan un valor comercial para las partes interesadas a lo largo de los sprints.
- **Historias de usuario**: Son descripciones detalladas de las necesidades del trabajo, que especifican claramente lo que los desarrolladores deben implementar.

Además, se utilizará una nomenclatura específica para identificar las épicas, funcionalidades e historias de usuario:

- **EP[Id]**: Identifica las épicas del sistema.
- **F[Id]**: Identifica las funcionalidades dentro de las épicas.
- **US[Id]**: Identifica las historias de usuario dentro de cada funcionalidad.

En la Tabla 2.4 se encuentran definidas las *épicas* correspondientes a los componentes y consideraciones respecto al desarrollo de las funcionalidades, la preparación del ambiente de desarrollo y la validación del producto final del software del componente *B*.

Id (Epic)	**Épica**	**Descripción**
EP1	Revisión Sistemática y Definición de Requisitos	En esta épica se llevará a cabo una revisión sistemática de la literatura existente para comprender a fondo los conceptos, tecnologías y mejores prácticas relevantes para el desarrollo del componente *B*. El objetivo es establecer un sólido marco conceptual que guíe el diseño e implementación del mismo, y definir los requisitos técnicos y funcionales.
EP2	Elaboración de Mockups y Definición de Estilos	Esta épica tiene como objetivo la creación de mockups detallados que representen la interfaz de usuario del sistema integrado, teniendo en cuenta los requisitos y la arquitectura definidos previamente. Además, se buscará la validación de estos mockups con las partes interesadas para asegurar que cumplen con las expectativas y se definirán los estilos y colores a utilizar en el frontend.
EP3	Desarrollo de Pantallas de Registro y Login	Esta épica tiene como objetivo el desarrollo de las interfaces frontend para las pantallas de registro y login, asegurando su adaptabilidad a diferentes dispositivos y siguiendo el principio de *Mobile First*.
EP4	Desarrollo de Pantalla de Perfil de Usuario	Esta épica tiene como objetivo el desarrollo de la pantalla de perfil de usuario, incluyendo la sección “About” del perfil, permitiendo a los usuarios agregar y editar información personal, disciplinas y detalles de contacto.

Id (Epic)	Épica	Descripción
EP5	Desarrollo de Funcionalidades de Login y Usuario en Backend	Esta épica tiene como objetivo la creación de modelos y serializadores para la gestión de usuarios en el backend utilizando Django, así como el desarrollo de vistas y endpoints necesarios para la autenticación y gestión de perfiles de usuario.
EP6	Desarrollo de Funcionalidades de Publicaciones y Grupos en Backend	Esta épica tiene como objetivo la creación de modelos y serializadores para la gestión de publicaciones y grupos en el backend utilizando Django, así como el desarrollo de vistas y endpoints necesarios para la gestión de publicaciones y grupos.
EP7	Integración Final y Configuración de Docker	Esta épica tiene como objetivo integrar la comunicación entre el frontend y el backend mediante APIs RESTful, configurar el entorno de desarrollo usando Docker, y realizar pruebas de integración para asegurar el funcionamiento correcto de la aplicación.

Tabla 2.4: Épicas.

Por otra parte, en la Tabla 2.5 se definieron las *features* correspondientes a cada épica. Las *features* representan las funcionalidades principales del componente *B*, que a su vez se dividen en diferentes historias de usuario, como se muestra en la Tabla 2.6. Las historias de usuario corresponden a los requisitos funcionales del componente, descritas desde la perspectiva de las necesidades de los usuarios.

Id	Feature
EP1F1	Revisión sistemática de la literatura.
EP1F2	Definición de requisitos técnicos y funcionales.
EP2F1	Creación de mockups detallados para la interfaz del sistema integrado.

Id	Feature
EP2F2	Definición de estilos y colores para el frontend.
EP3F1	Desarrollo de la pantalla de registro en el frontend.
EP3F2	Desarrollo de la pantalla de login en el frontend.
EP4F1	Desarrollo de la pantalla de perfil de usuario en el frontend.
EP5F1	Creación de modelos y serializadores para la gestión de usuarios en el backend.
EP5F2	Desarrollo de vistas y endpoints para autenticación y gestión de perfiles de usuario en el backend.
EP6F1	Creación de modelos y serializadores para la gestión de publicaciones en el backend.
EP6F2	Creación de modelos y serializadores para la gestión de grupos en el backend.
EP6F3	Desarrollo de vistas y endpoints para la gestión de publicaciones en el backend.
EP6F4	Desarrollo de vistas y endpoints para la gestión de grupos en el backend.
EP7F1	Integración del frontend y backend mediante APIs RESTful.
EP7F2	Configuración del entorno de desarrollo usando Docker.
EP7F3	Pruebas de integración para asegurar el funcionamiento correcto de la aplicación.

Tabla 2.5: Características del sistema.

Finalmente, se procedió a priorizar y estimar el esfuerzo de desarrollo de cada una de las historias de usuario identificadas.

Id	Id	Historias de usuario
EP1F1	US1	Como desarrollador, quiero revisar la literatura existente para comprender los conceptos, tecnologías y mejores prácticas relevantes para el componente *B*, de modo que pueda aplicar este conocimiento en el desarrollo.

Id	Id	Historias de usuario
	US2	Como desarrollador, quiero definir un marco conceptual sólido basado en la revisión de la literatura para guiar el diseño e implementación del componente *B*.
EP1F2	US3	Como desarrollador, quiero definir los requisitos técnicos y funcionales del componente *B*, para asegurarme de que todas las necesidades del proyecto estén cubiertas.
EP2F1	US4	Como desarrollador, quiero crear mockups detallados para visualizar la interfaz de usuario del sistema integrado, para obtener una aprobación antes de comenzar el desarrollo.
	US5	Como desarrollador, quiero definir los estilos y colores para el frontend, para mantener la coherencia visual del sistema.
EP3F1	US6	Como usuario, quiero registrarme en el sistema proporcionando mi nombre, apellido, correo electrónico y contraseña, para poder acceder a las funcionalidades del sistema.
	US7	Como usuario registrado, quiero iniciar sesión en el sistema utilizando mi correo electrónico y contraseña, para acceder a mi cuenta personal.
EP4F1	US8	Como usuario, quiero tener un perfil personal donde pueda actualizar mi información personal, disciplinas y detalles de contacto, para mantener mi perfil actualizado y completo.
EP5F1	US9	Como desarrollador, quiero crear modelos y serializadores para gestionar usuarios, para asegurar que la información del usuario se almacene y procese correctamente.
	US10	Como desarrollador, quiero desarrollar vistas y endpoints para la autenticación y gestión de perfiles de usuario, para que los usuarios puedan registrarse, iniciar sesión y gestionar su información de perfil.
EP6F1	US11	Como usuario, quiero poder crear publicaciones con contenido multimedia, para compartir información y colaboraciones con otros usuarios.
	US12	Como usuario, quiero poder crear y gestionar grupos de investigación, para colaborar de manera efectiva con otros investigadores.

Id	Id	Historias de usuario
	US13	Como desarrollador, quiero desarrollar vistas y endpoints para la gestión de publicaciones, para que los usuarios puedan crear, ver, editar y eliminar publicaciones.
	US14	Como desarrollador, quiero desarrollar vistas y endpoints para la gestión de grupos, para que los usuarios puedan crear, ver y gestionar grupos de investigación.
EP7F1	US15	Como desarrollador, quiero integrar el frontend y backend mediante APIs RESTful, para asegurar la comunicación y funcionalidad entre ambas partes del sistema.
	US16	Como desarrollador, quiero configurar el entorno de desarrollo usando Docker, para asegurar la consistencia del entorno de desarrollo y facilitar el despliegue.
	US17	Como desarrollador, quiero realizar pruebas de integración para asegurarme de que el sistema funciona correctamente como un todo, antes de su despliegue final.

Tabla 2.6: Historias de usuarios.

2.6. Planificación de los *sprints*

Se ha definido la duración y el alcance de cada sprint para la planificación del trabajo. La Tabla 2.7 presenta las historias de usuario asignadas a cada sprint, junto con su nivel de esfuerzo estimado y la duración en días laborables. Este enfoque garantiza una distribución equilibrada de tareas y recursos a lo largo del proyecto.

No. Sprint	S0	S1	S2	S3	S4	S5	S6
US(ID)	US1, US2	US3, US4, US5	US6, US7	US8	US9, US10	US11, US12, US13, US14	US15, US16, US17
Tiempo estimado	10 días	10 días	10 días	5 días	10 días	10 días	5 días

Tabla 2.7: *Sprint planning*.

2.7. Sprint 0

2.7.1. Objetivos del sprint

- Realizar una revisión sistemática de la literatura existente para comprender en profundidad los conceptos, tecnologías y mejores prácticas relevantes para el desarrollo del componente *B*.
- Identificar las tendencias, enfoques y metodologías más relevantes en el campo de desarrollo de módulos similares o relacionados con el componente *B*.
- Establecer un marco conceptual sólido que sirva como base para el diseño e implementación del componente *B*, asegurando su coherencia y alineación con las prácticas establecidas y las necesidades del proyecto.
- Definir los requisitos técnicos y funcionales preliminares del componente *B* en función de los hallazgos de la investigación, proporcionando una dirección clara para el desarrollo futuro.
- Identificar y analizar las áreas de mejora específicas para el componente *B*, asegurando su alineación con los objetivos del proyecto.
- Realizar la elicitación de requisitos técnicos y funcionales del componente *B*.
- Diseñar la arquitectura necesaria para la integración del componente *B* con los sistemas existentes.
- Establecer una base sólida para el desarrollo del componente *B*, garantizando su coherencia y efectividad.

2.7.2. Ejecución del sprint

El actual sprint se ha llevado a cabo siguiendo el plan establecido en la Sección 2.6 y en función de las historias de usuario asignadas al Sprint 0, como se detalla en la Tabla 2.7.

Durante este sprint, se ha realizado las siguientes actividades:

Revisión sistemática de la literatura

Para establecer un marco conceptual robusto, se ha llevado a cabo una revisión exhaustiva de la literatura existente sobre redes sociales académicas, grupos y comunidades en plataformas de investigación, visualización de temas relevantes y redes de coautoría. Los detalles de estos conceptos se pueden encontrar en la Sección *Marco Teórico* (Sección 1.6) de este documento.

Hallazgos Relevantes De la revisión de la literatura se obtuvo varios hallazgos clave que guiarán el desarrollo del componente *B*:

1. **Interacción y Colaboración**: Las redes sociales académicas deben facilitar la interacción y colaboración entre los usuarios mediante la creación de grupos y comunidades. Estos aspectos son esenciales para fomentar la comunicación y el intercambio de información entre investigadores [4], [8].
2. **Grupos y Comunidades**: La formación de grupos de investigación y comunidades en línea elimina las barreras físicas y facilita la colaboración entre investigadores de diferentes instituciones. Las plataformas de redes sociales deben proporcionar herramientas para la creación y gestión de estos grupos [9], [10].
3. **Sistemas de Recomendación**: Los sistemas de recomendación son cruciales para personalizar la experiencia del usuario y resaltar contenido relevante. Al incluir estos sistemas en una red social académica, se puede mejorar la visibilidad de temas y publicaciones relevantes para los usuarios [6].
4. **Redes de Coautoría**: Las redes de coautoría son esenciales para el avance científico, facilitando la colaboración y el reconocimiento entre investigadores. La plataforma debe incluir funcionalidades que permitan la creación y visualización de estas redes [5].

Definición de requisitos técnicos y funcionales

Basado en la revisión de la literatura, se ha definido los siguientes requisitos técnicos y funcionales preliminares para el componente *B*:

Requisitos Técnicos

- Integración de bases de datos robustas para la gestión de usuarios, grupos y publicaciones.
- Implementación de sistemas de autenticación seguros.
- Capacidades de carga y visualización de contenido multimedia.
- Sistemas de recomendación para personalizar la experiencia del usuario.

Requisitos Funcionales

- Registro y autenticación de usuarios.
- Gestión de perfiles de usuario.
- Creación y gestión de grupos.
- Publicaciones de contenido.

Detalle de las Funcionalidades Cada una de estas funcionalidades se detalla a continuación para proporcionar una comprensión clara de lo que se desarrolló en la V1 de CENTINELA y lo que se espera desarrollar en futuros sprints:

1. **Registro y Autenticación de Usuarios**:
    - **Proceso de Registro**: Formularios de registro que validen la entrada del usuario y aseguren la integridad de los datos.
    - **Inicio de Sesión**: Interfaces de inicio de sesión con validación de credenciales.
    - **Recuperación de Contraseñas**: Previsto para versiones futuras.
2. **Gestión de Perfiles de Usuario**:
    - **Edición de Perfil**: Funcionalidades que permitan a los usuarios actualizar su información personal, imagen de perfil, y añadir secciones como “Sobre mí”.
    - **Visualización del Perfil**: Interfaces que muestren la información del perfil de manera clara y accesible.
3. **Creación y Gestión de Grupos**:

- **Creación de Grupos**: Formularios para la creación de nuevos grupos de investigación, permitiendo añadir descripciones y configurar la privacidad del grupo.
- **Configuración de Privacidad de Grupos**, **Invitar y Remover Miembros**, **Asignación de Roles**: Previstos para versiones futuras.

4. **Publicaciones de Contenido**:
   - **Creación de Publicaciones**: Interfaces que permitan a los usuarios crear publicaciones con diferentes tipos de contenido, incluyendo imágenes, videos y archivos.
   - **Interacción con Publicaciones**: Comentarios y reacciones previstos para versiones futuras.
   - **Edición de Publicaciones**: Previsto para versiones futuras.
   - **Eliminación de Publicaciones**: Implementado en la V1 de CENTINELA.

Identificación y análisis de áreas de mejora

Para asegurar la eficacia del componente *B*, se realizó un análisis exhaustivo de las áreas de mejora. Esto incluyó la revisión de la funcionalidad actual de las plataformas existentes y la identificación de las brechas y limitaciones. Las áreas de mejora se enfocaron en:

- Mejorar la experiencia del usuario mediante interfaces más intuitivas en Angular.
- Optimizar la integración de funciones de redes sociales académicas para fomentar la colaboración.
- Asegurar la escalabilidad y rendimiento del sistema utilizando Django para el backend.
- Fortalecer la seguridad y privacidad de los datos de los usuarios utilizando JWT para autenticación.[1]

Elicitación de requisitos técnicos y funcionales

La elicitación de requisitos se realizó a través de sesiones de lluvia de ideas y consultas con las partes interesadas clave. Los requisitos identificados se pueden clasificar en:

[1] Un JSON Web Token (JWT) es un estándar abierto que define una manera compacta y autónoma de transmitir información de manera segura entre partes como un objeto JSON. Esta información puede ser verificada y confiable porque está firmada digitalmente. [25]

Requisitos Técnicos

- Integración de bases de datos robustas para la gestión de usuarios, grupos y publicaciones.
- Implementación de sistemas de autenticación seguros utilizando rest_framework_simplejwt de Django.
- Capacidades de carga y visualización de contenido multimedia.
- Uso de Angular para el frontend, proporcionando una experiencia de usuario mejorada.

Requisitos Funcionales

- Registro y autenticación de usuarios utilizando email y contraseña.
- Gestión de perfiles de usuario, incluyendo la creación y edición de información personal.
- Creación de grupos de usuarios, enfocándose solo en la creación y no en la gestión.
- Publicaciones de contenido tipo red social.
- Herramientas de búsqueda de contenido.

Diseño de arquitectura para la integración

Basado en los requisitos técnicos y funcionales, se diseñó la arquitectura del componente *B* para su integración con los sistemas existentes. El diseño arquitectónico incluyó:

- **Modelo de Datos**: Definición de esquemas de base de datos que soporten la gestión de usuarios, grupos y publicaciones.
- **Componentes del Sistema**: Identificación de los componentes clave, como el servidor de autenticación y el servicio de gestión de perfiles.
- **Integración de APIs**: Diseño de interfaces de programación de aplicaciones (APIs) utilizando Django Rest Framework para facilitar la comunicación entre los diferentes componentes del sistema.
- **Seguridad y Privacidad**: Implementación de medidas de seguridad, como la encriptación de datos y la autenticación JWT.

Modelo de Datos El modelo de datos se diseñó para soportar las siguientes entidades principales:

- **Usuarios**: Almacenar información personal, credenciales de autenticación y preferencias del usuario.
- **Grupos**: Gestionar la creación de grupos de investigación, incluyendo miembros y permisos.
- **Publicaciones**: Manejar el contenido generado por los usuarios, como textos, imágenes y archivos.

Componentes del Sistema Los componentes clave del sistema incluyen:

- **Servidor de Autenticación**: Gestiona el registro y autenticación de usuarios, implementando protocolos seguros como OAuth 2.0 y JWT mediante rest_framework_simplejwt.
- **Servicio de Gestión de Perfiles**: Permite a los usuarios crear, editar y visualizar sus perfiles.
- **Servicio de Publicaciones**: Facilita la creación y visualización de contenido tipo red social.

Integración de APIs Las APIs fueron diseñadas para permitir la comunicación entre los diferentes componentes del sistema utilizando Django Rest Framework. Las principales APIs incluyen:

- **API de Autenticación**: Maneja el registro e inicio de sesión.
- **API de Gestión de Perfiles**: Gestiona las operaciones de creación, edición y visualización de perfiles.
- **API de Grupos**: Facilita la creación de grupos de investigación.
- **API de Publicaciones**: Soporta la creación y visualización de publicaciones de los usuarios.

Seguridad y Privacidad Se implementaron varias medidas de seguridad para proteger los datos de los usuarios:

- **Encriptación de Datos**: Todos los datos sensibles se encriptan tanto en tránsito como en reposo.
- **Autenticación JWT**: Se añadió una capa adicional de seguridad utilizando JSON Web Tokens para proteger las cuentas de usuario.
- **Políticas de Privacidad**: Se establecieron políticas claras sobre la recolección, uso y protección de los datos de los usuarios.

Arquitectura del Sistema

La arquitectura del sistema, denominado CENTINELA, integra dos plataformas existentes, utilizando Angular para el frontend y Docker para el backend. Los servicios se despliegan en contenedores para asegurar flexibilidad y escalabilidad.

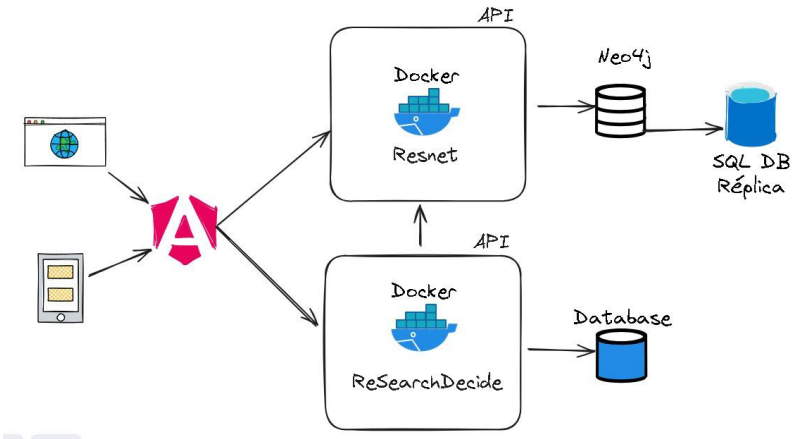


Figura 2.1: Arquitectura del Sistema CENTINELA.

La Figura 2.1 muestra la arquitectura general del sistema CENTINELA. En el frontend, se utiliza Angular para desarrollar la interfaz de usuario, que se comunica con los servicios backend mediante APIs RESTful. El backend está implementado en Docker, con dos contenedores principales: Docker Resnet y Docker ReSearchDecide. Docker Resnet maneja las funcionalidades de la red social académica, mientras que Docker ReSearchDecide se encarga de las decisiones y análisis de datos de investigación. Para gestionar las relaciones complejas y asegurar la alta disponibilidad de los datos en Resnet, se utilizan las bases de datos Neo4j y SQL DB Réplica, respectivamente. En ReSearchDecide, se emplea una base de datos PostgreSQL.

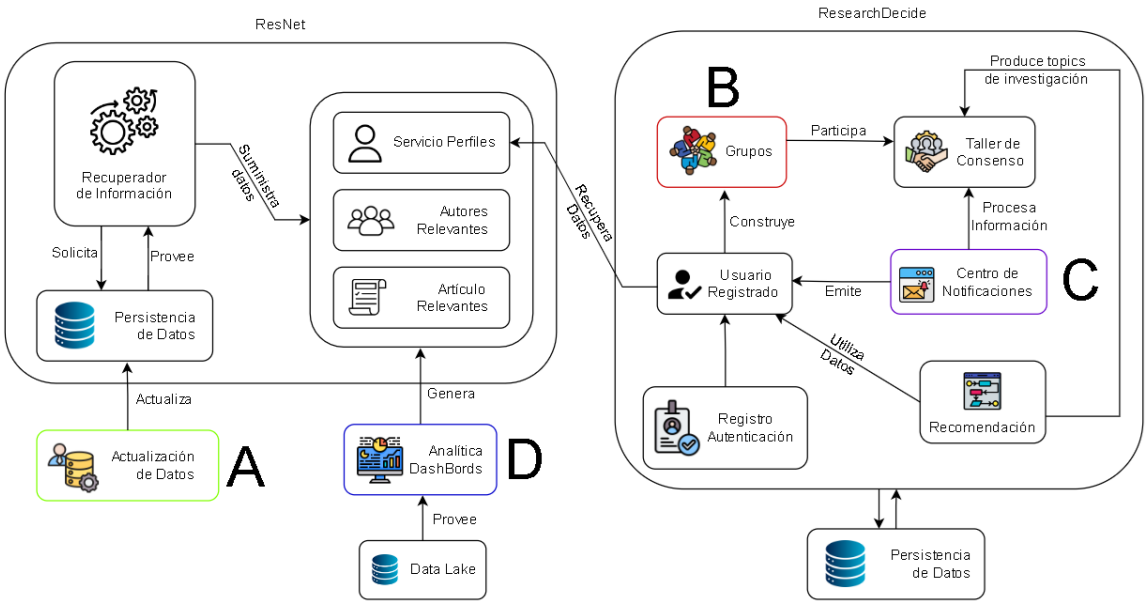


Figura 2.2: Detalle de la Arquitectura de CENTINELA.

La Figura 2.2 detalla los componentes principales de CENTINELA. Dentro del módulo ResNet, el Recuperador de Información solicita y provee datos a la persistencia, que almacena la información de manera consistente. El Servicio de Perfiles gestiona perfiles de usuario, autores y artículos relevantes, asegurando que los datos estén siempre actualizados mediante el componente de Actualización de Datos.

En el módulo ReSearchDecide, los usuarios pueden crear y gestionar grupos de investigación. Estos grupos, una vez formados, participan en el Taller de Consenso para generar temas de investigación relevantes. Además, el sistema de Recomendación utiliza los datos de los grupos de investigación para proporcionar recomendaciones personalizadas. La seguridad y autenticación de usuarios están garantizadas por el componente de Registro y Autenticación.

Por último, el módulo de Analítica y Dashboards genera dashboards analíticos utilizando datos del Data Lake, proporcionando insights y análisis avanzados.

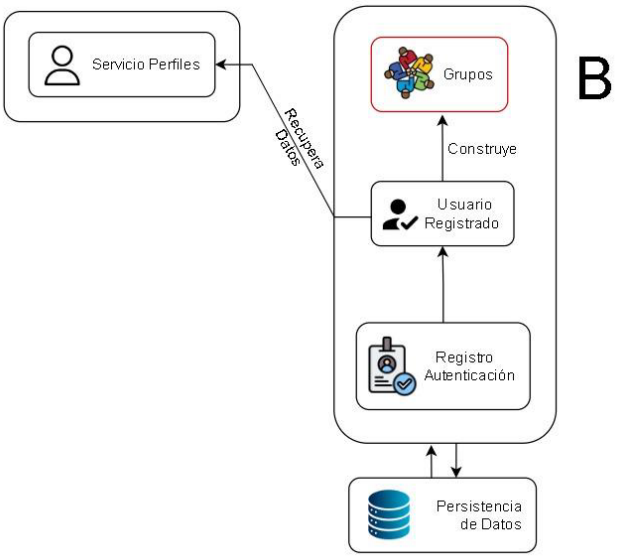


Figura 2.3: Arquitectura del componente *B* en CENTINELA.

La Figura 2.3 muestra la arquitectura específica del componente *B* en CENTINELA. Este componente incluye el Servicio de Perfiles, que recupera y gestiona los datos de perfil de los usuarios. Los usuarios registrados pueden crear y gestionar grupos de investigación, facilitando la colaboración y el intercambio de información. El proceso de registro y autenticación de usuarios es gestionado por el componente de Registro y Autenticación, garantizando que solo los usuarios autorizados puedan acceder al sistema. Toda la información, incluyendo los datos de usuarios, perfiles y grupos, se almacena de manera persistente para asegurar su disponibilidad y consistencia.

Esta arquitectura modular y escalable asegura que el sistema CENTINELA pueda adaptarse y crecer con las necesidades cambiantes del proyecto, facilitando la colaboración efectiva y la gestión eficiente de datos.

Estructura de Carpetas del Proyecto

A continuación se describe la estructura de carpetas tanto del backend como del frontend del sistema CENTINELA, siguiendo una arquitectura hexagonal.

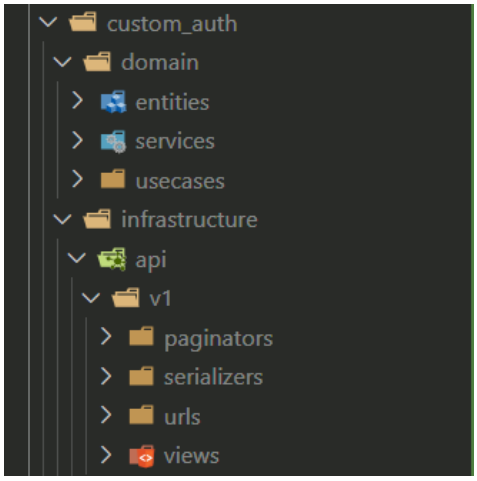


Figura 2.4: Estructura de Carpetas del Backend.

La Figura 2.4 muestra la estructura de carpetas del backend del sistema CENTINELA. Esta estructura está diseñada para seguir los principios de una arquitectura hexagonal, lo que permite una separación clara de las responsabilidades y facilita la mantenibilidad del código. En el backend, encontramos carpetas clave como “custom auth”, que contiene la lógica relacionada con la autenticación personalizada. Dentro de esta carpeta, “domain” agrupa las entidades, servicios y casos de uso específicos del dominio de autenticación, mientras que “infrastructure” incluye la implementación de la infraestructura, como la API. La API se encuentra organizada en versiones (“v1”), con subcarpetas para paginadores, serializadores, rutas y vistas.

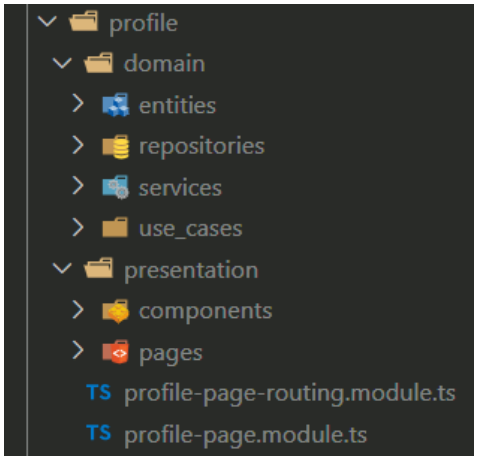


Figura 2.5: Estructura de Carpetas del Frontend.

La Figura 2.5 muestra la estructura de carpetas del frontend del sistema CENTINELA. Similar al backend, el frontend sigue una arquitectura hexagonal que permite una separa-

ción clara entre la lógica de negocio y la presentación. La carpeta “profile” contiene toda la lógica relacionada con los perfiles de usuario. Dentro de “profile”, la carpeta “domain” incluye las entidades, repositorios, servicios y casos de uso específicos del dominio de perfiles, lo que facilita la organización y el acceso a la lógica de negocio. La carpeta “presentation” agrupa los componentes y páginas que conforman la interfaz de usuario, asegurando que la presentación esté claramente separada de la lógica de negocio. Esto incluye configuraciones de enrutamiento y módulos específicos para las páginas de perfil, mejorando la modularidad y extensibilidad del código.

En resumen, esta organización de carpetas asegura que tanto el backend como el frontend sean modulares y fácilmente extensibles. Esto permite una colaboración efectiva entre los desarrolladores y facilita la mantenibilidad del proyecto a largo plazo.

2.7.3. Revisión y retrospectiva del sprint

La revisión y retrospectiva del Sprint 0 incluye:

- **Logros Alcanzados**:
  - Completa revisión de la literatura relevante, guiada por artículos encontrados en Google Scholar.
  - Definición clara de los requisitos técnicos y funcionales preliminares.
  - Establecimiento de un marco conceptual sólido para el desarrollo del componente *B*.
  - Identificación y análisis de áreas de mejora específicas para el componente *B*.
  - Elicitación detallada de los requisitos técnicos y funcionales.
  - Diseño de una arquitectura robusta para la integración del componente *B* con los sistemas existentes.
  - Definición de un modelo de datos y componentes del sistema que aseguren una integración efectiva y segura.

2.8. Sprint 1

2.8.1. Objetivos del sprint

- Elaborar mockups detallados que representen la interfaz de usuario del sistema integrado, teniendo en cuenta los requisitos y la arquitectura definidos previamente.
- Visualizar y validar el diseño de la interfaz antes de proceder con la implementación.
- Asegurar que el diseño inicial se enfoque en la experiencia móvil, siguiendo el principio de “Mobile First” [26].
- Definir los colores y estilos a utilizar en el frontend durante el desarrollo.
- Asegurar la flexibilidad del diseño para adaptarse a cambios futuros durante el desarrollo del frontend.

2.8.2. Ejecución del sprint

El actual sprint se ha llevado a cabo siguiendo el plan establecido en la Sección 2.6 y en función de las historias de usuario asignadas al Sprint 1, como se detalla en la Tabla 2.7.

Durante este sprint, se han realizado las siguientes actividades:

Elaboración de Mockups

Para asegurar que la interfaz de usuario cumpla con los requisitos y la arquitectura previamente definidos, se ha llevado a cabo la elaboración de mockups detallados. Estos mockups proporcionan una representación visual del diseño de la interfaz y sirven como una herramienta esencial para la validación del diseño antes de proceder con la implementación.

- **Herramientas Utilizadas**: Se utilizó Figma para la creación de los mockups.
- **Estructura de la Interfaz**: Los mockups incluyeron diseños para las principales secciones de la aplicación, tales como la pantalla de inicio, el perfil de usuario, la creación de grupos y la sección de publicaciones.
- **Principio de Diseño**: Se siguió el principio de “Mobile First”, asegurando que el diseño inicial se enfocara en la experiencia móvil antes de adaptarse a la web [26].

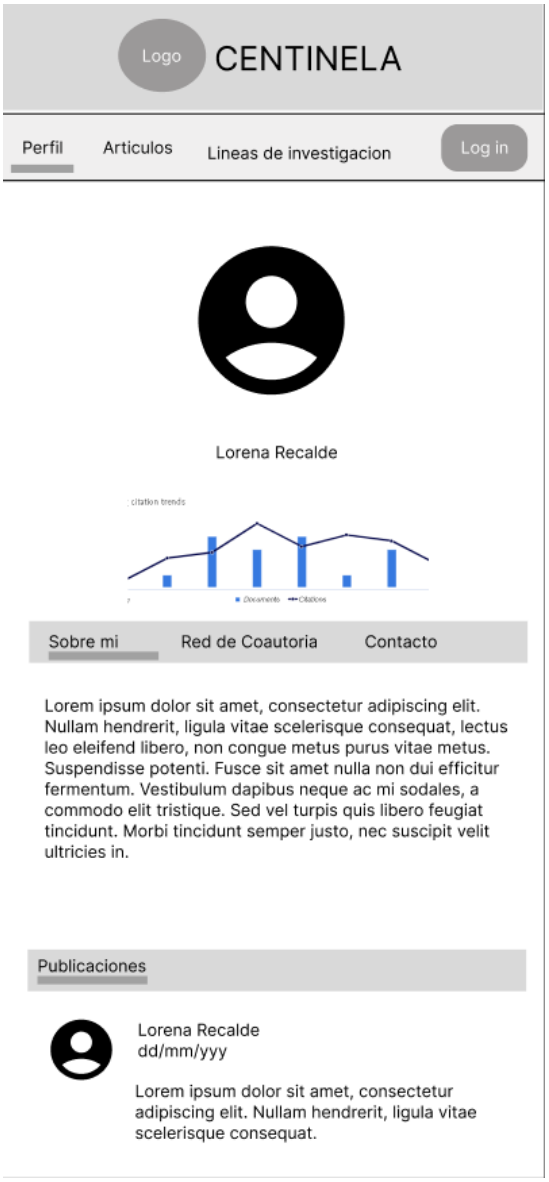


Figura 2.6: Mockup del Perfil de Usuario.

La Figura 2.6 muestra el mockup del perfil de usuario. Este diseño incluye una sección de información personal, una gráfica de tendencias de citas y pestañas dedicadas a la biografía, red de coautoría y contacto. Además, permite visualizar publicaciones recientes del usuario.

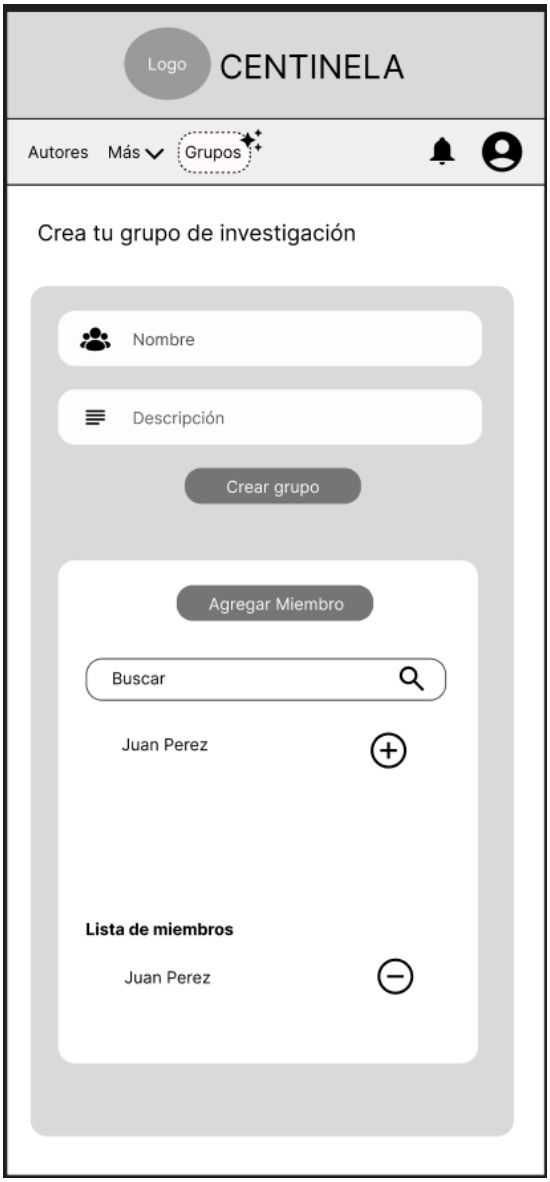


Figura 2.7: Mockup de la Creación de Grupos.

La Figura 2.7 ilustra el proceso de creación de grupos de investigación. El diseño permite a los usuarios introducir el nombre y la descripción del grupo, así como agregar miembros mediante un buscador. Este enfoque facilita la creación y gestión inicial de grupos en la plataforma.

Validación del Diseño de la Interfaz

Una vez elaborados los mockups, se procedió a la validación del diseño de la interfaz. Esta validación se realizó en colaboración con las partes interesadas clave para asegurar que el diseño propuesto cumple con las expectativas y requisitos definidos.

- **Feedback de Usuarios**: Se realizaron sesiones de revisión con usuarios potenciales para obtener retroalimentación sobre la usabilidad y la estética de la interfaz.
- **Revisión de Requisitos**: Se verificó que los mockups cumplen con los requisitos técnicos y funcionales previamente definidos.
- **Ajustes y Mejoras**: Basado en el feedback recibido, se realizaron ajustes y mejoras en los mockups para optimizar la experiencia del usuario.

Definición de Estilos y Colores

Durante este sprint, también se definieron los colores y estilos a utilizar en el frontend del sistema CENTINELA. Aunque los mockups iniciales no incluían estos detalles, se establecieron en colaboración con el equipo de diseño durante el desarrollo del frontend.

- **Paleta de Colores**: Se seleccionó una paleta de colores coherente con la identidad visual de CENTINELA. El color primario definido es #1E3C8B y el color secundario es #EAEBED.
- **Frameworks y Librerías**: Se decidió utilizar Tailwind CSS para los estilos, junto con librerías de componentes como Angular Material y Flowbite. Estas herramientas se utilizarán a necesidad en cada componente del sistema.
- **Estilos de Componentes**: Se definieron estilos para los distintos componentes de la interfaz, asegurando una apariencia coherente y profesional.

Flexibilidad del Diseño

Dado que en un desarrollo Scrum los requisitos son cambiantes, los mockups y el diseño se mantuvieron flexibles para adaptarse a posibles cambios durante el desarrollo del frontend.

2.8.3. Revisión y retrospectiva del sprint

La revisión y retrospectiva del Sprint 1 incluye:

- **Logros Alcanzados**:

- Elaboración de mockups detallados que representan la interfaz de usuario del sistema integrado.
- Validación del diseño de la interfaz en colaboración con las partes interesadas y usuarios potenciales.
- Definición de los colores y estilos a utilizar en el frontend.
- Decisión de utilizar Tailwind CSS, Angular Material y Flowbite para los estilos y componentes.
- Realización de ajustes y mejoras en los mockups basados en el feedback recibido.

- **Desafíos Encontrados**:
  - Ajuste de los diseños para asegurar que sean intuitivos y fáciles de usar.
  - Coordinación con las partes interesadas para obtener feedback detallado y constructivo.

- **Lecciones Aprendidas**:
  - La importancia de la retroalimentación temprana de los usuarios para guiar el diseño de la interfaz.
  - La necesidad de iterar sobre los diseños para asegurar que cumplan con los requisitos y expectativas de los usuarios.

2.9. Sprint 2

2.9.1. Objetivos del sprint

- Desarrollar las interfaces frontend del componente *B* utilizando Angular y asegurando su adaptabilidad a diferentes dispositivos mediante el uso de Tailwind CSS.
- Implementar las pantallas de registro y login.
- Asegurar que el diseño siga el principio de *Mobile First* para mejorar la experiencia del usuario en dispositivos móviles.

2.9.2. Ejecución del sprint

Durante este sprint, se desarrollaron e implementaron las siguientes pantallas del componente *B* de la plataforma CENTINELA:

Pantalla de Registro

En la pantalla de registro, se incluyen los campos necesarios como *First Name*, *Last Name*, *Email Address*, *Password*, *Confirm Password*, y de forma opcional el *Scopus ID*. Esta opción se encuentra ya que existen algunas funcionalidades que se muestran si el usuario tiene un *Scopus ID*.

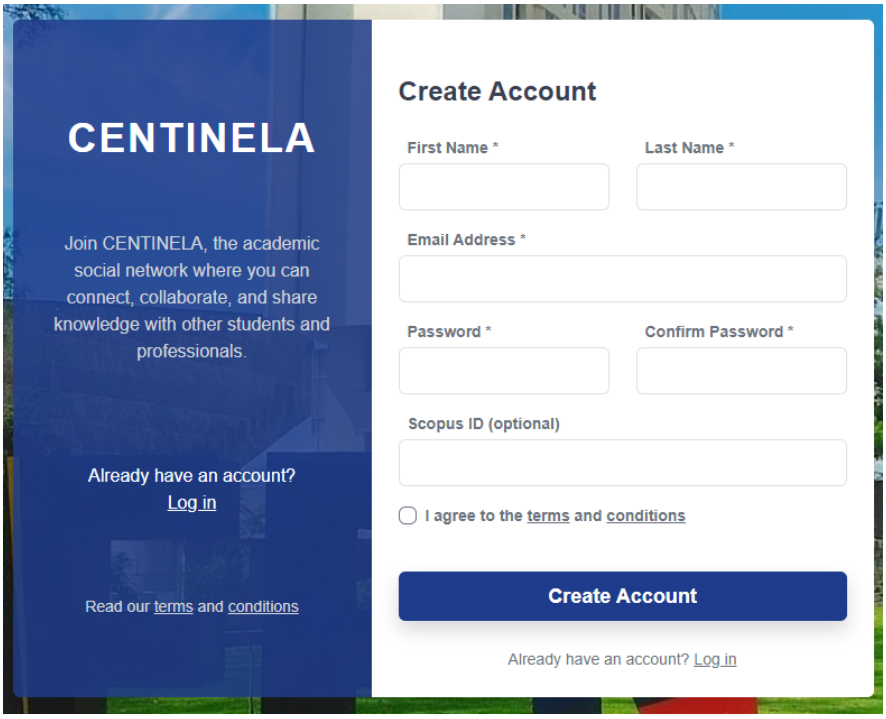


Figura 2.8: Pantalla de Registro.

Pantalla de Login

En la pantalla de **Login**, se muestran los campos *Email address* y *Password*, las credenciales necesarias para acceder a la plataforma. También se observan opciones para iniciar sesión con *Google* y *ORCID*, las cuales están deshabilitadas ya que no se encuentran dentro del alcance del proyecto actual, pero quedan para la implementación de nuevas funcionalidades en el futuro. Asimismo, la opción de *Forgot Password* está deshabilitada.

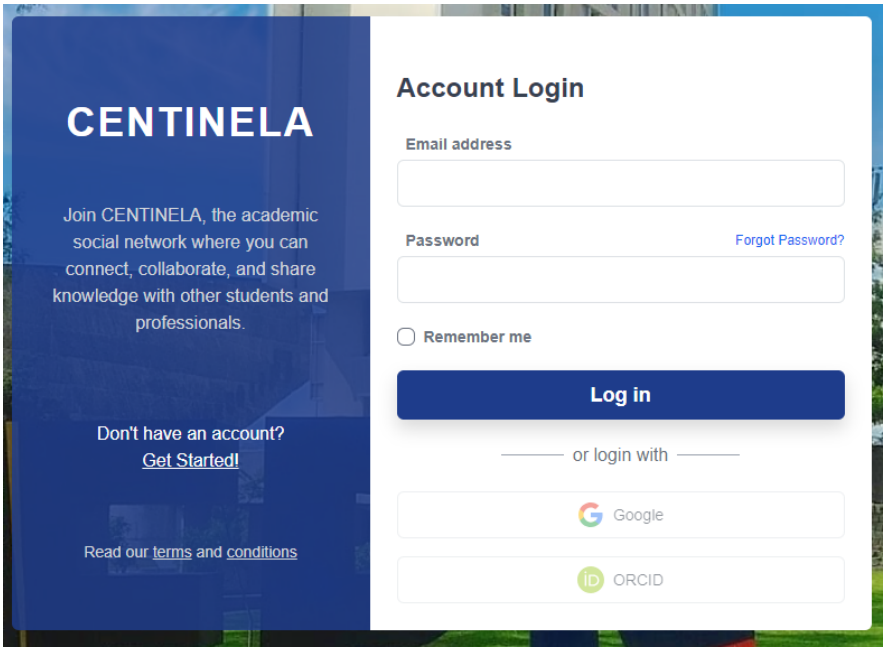


Figura 2.9: Pantalla de Login.

2.9.3. Revisión y retrospectiva del sprint

La revisión y retrospectiva del Sprint 2 incluye:

- **Logros Alcanzados**:
  - Desarrollo exitoso de las interfaces frontend del componente *B*, asegurando su adaptabilidad y responsividad mediante el uso de Tailwind CSS y siguiendo el principio de *Mobile First*.
  - Implementación de las pantallas de registro y login, incluyendo todos los campos necesarios y opcionales, dejando funcionalidades adicionales como el inicio de sesión con Google y ORCID, así como la recuperación de contraseñas, para futuros desarrollos.
- **Desafíos Encontrados**:
  - Ajuste de los diseños para asegurar que sean intuitivos y fáciles de usar.
  - Coordinación con las partes interesadas para obtener feedback detallado y constructivo.
- **Lecciones Aprendidas**:
  - La importancia de la retroalimentación temprana de los usuarios para guiar el diseño de la interfaz.

- La necesidad de iterar sobre los diseños para asegurar que cumplan con los requisitos y expectativas de los usuarios.

2.10. Sprint 3

2.10.1. Objetivos del sprint

- Desarrollar la interfaz de la pantalla de perfil de usuario del componente *B* utilizando Angular.
- Asegurar que el diseño siga el principio de *Mobile First* para mejorar la experiencia del usuario en dispositivos móviles.
- Implementar la sección *About* del perfil.
- Validar la funcionalidad con usuarios y partes interesadas.

2.10.2. Ejecución del sprint

Durante este sprint, se desarrollaron e implementaron las siguientes pantallas del componente *B* de la plataforma CENTINELA:

Pantalla de Perfil

En la pantalla de **Perfil**, se muestra un perfil recién creado con mensajes motivacionales para actualizar la información del perfil.

La cabecera del perfil (Figura 2.10) incluye una imagen de perfil por defecto, un mensaje central que motiva a la actualización del perfil y, a la derecha, un mensaje indicando que no se ha agregado un *Scopus ID* (si no se lo agregó al crear la cuenta). También se encuentra un botón de *Edit Profile*.

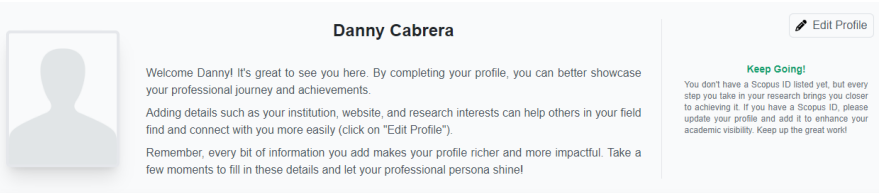


Figura 2.10: Cabecera del Perfil.

Al dar clic en el botón de *Edit Profile*, se abre un formulario (Figura 2.11) donde se pueden rellenar los datos del perfil. Los campos obligatorios son el *First Name* y el *Last Na-*

me. Los campos opcionales incluyen el *Institutional Email*, *Institution*, *Investigation Camp*, *Scopus ID*, *Website*, y la opción de subir una *Profile Picture*. Si se agrega alguna de estas informaciones, la cabecera del perfil cambiará para reflejar los nuevos datos.

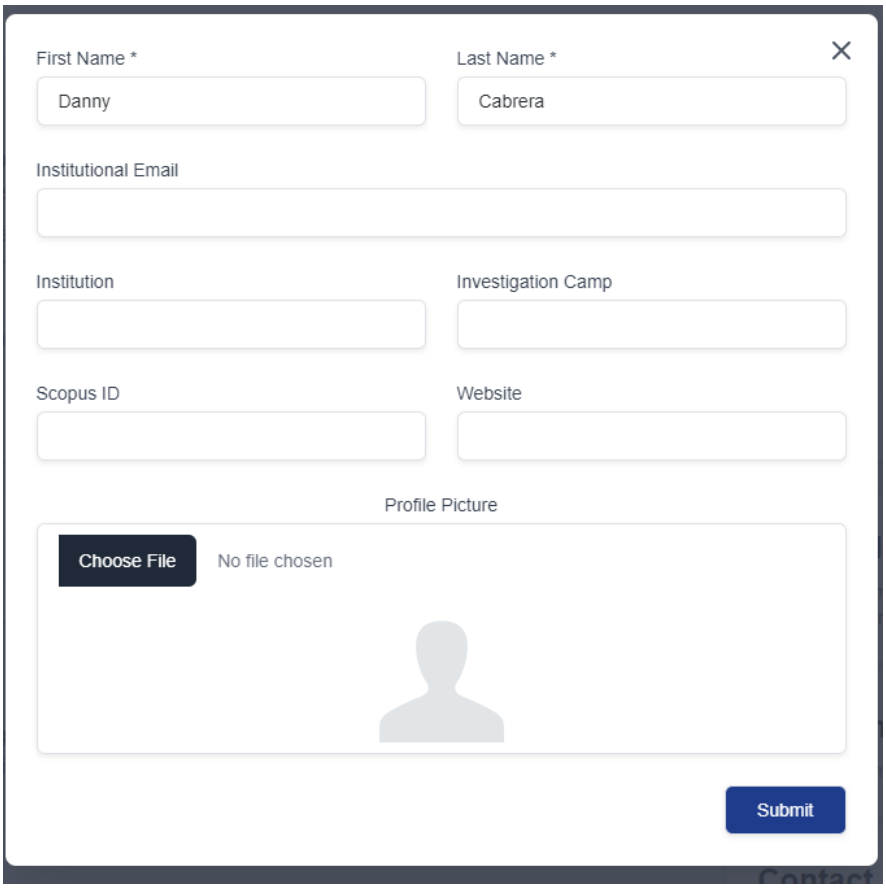


Figura 2.11: Formulario de Edición del Perfil.

La Figura 2.12 muestra un perfil completo con todos los campos llenos, incluida la imagen de perfil actualizada.

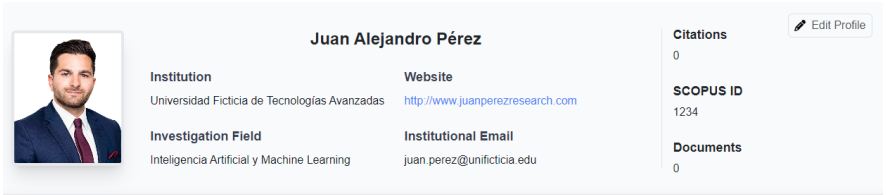


Figura 2.12: Perfil Completo.

Barra de Navegación del Perfil

La barra de navegación del perfil (Figura 2.13) muestra las pestañas *About* y *Groups*. Si se ha agregado un *Scopus ID*, se activan nuevas funcionalidades y la barra de navegación se expande para incluir pestañas adicionales como *Network*, *Article* y *Fingerprint* (Figura

2.14).

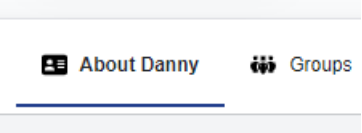


Figura 2.13: Barra de Navegación del Perfil.

Figura 2.14: Barra de Navegación del Perfil Completo.

Sección *About*

La sección *About* del perfil de usuario incluye tanto el área para crear publicaciones como la información adicional del usuario. El usuario puede agregar información adicional sobre sí mismo en *About Me*, disciplinas en las que se especializa en *Disciplines* y detalles de contacto en *Contact Information* (Figura 2.15). Cada una de estas secciones tiene un formulario de edición (Figuras 2.16, 2.17, y 2.18).

La Figura 2.15 muestra la pantalla principal del perfil con secciones para agregar publicaciones y detalles adicionales del usuario. En esta sección, los usuarios pueden ver un resumen de sus publicaciones, así como detalles adicionales como una breve biografía, disciplinas en las que se especializan y su información de contacto.

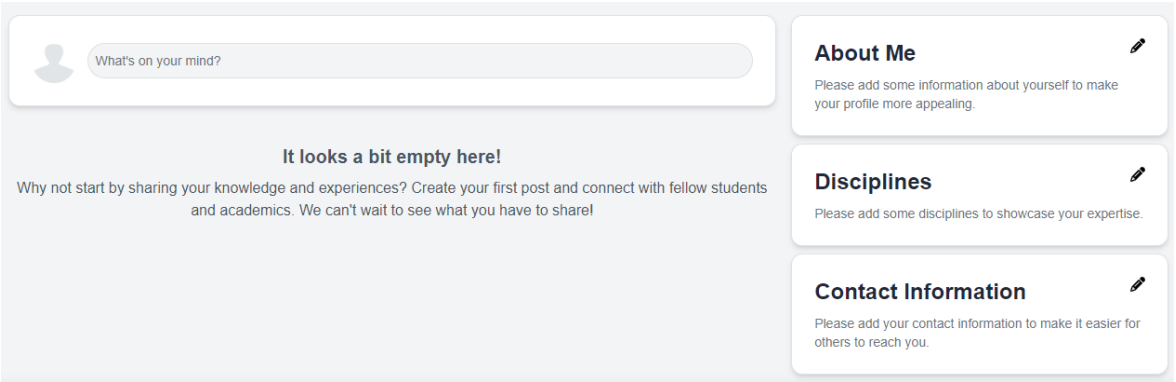


Figura 2.15: Sección *About*.

La Figura 2.16 muestra el formulario que permite al usuario agregar una breve descrip-

ción sobre sí mismo en la sección *About Me*. Esta área está destinada a que el usuario proporcione una introducción personal que otros usuarios pueden ver cuando visitan su perfil.

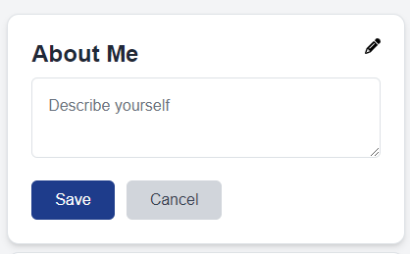


Figura 2.16: Formulario de Edición de *About Me*.

La Figura 2.17 muestra el formulario que permite al usuario agregar disciplinas en las que se especializa en la sección *Disciplines*. Esta información ayuda a otros usuarios a entender las áreas de conocimiento y experiencia del usuario.

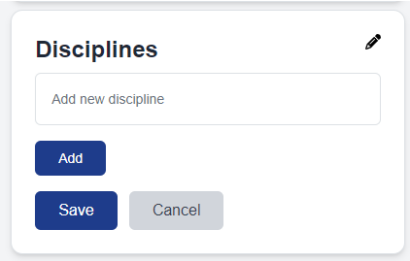


Figura 2.17: Formulario de Edición de *Disciplines*.

La Figura 2.18 muestra el formulario que permite al usuario agregar información de contacto adicional en la sección *Contact Information*. Aquí, los usuarios pueden proporcionar detalles como correos electrónicos adicionales, enlaces a otras redes sociales y cualquier otra información de contacto relevante.

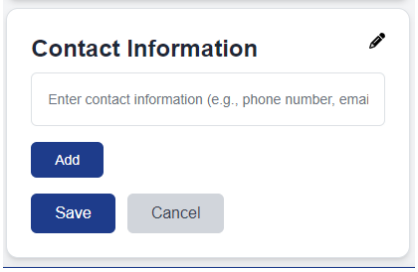


Figura 2.18: Formulario de Edición de *Contact Information*.

Las funcionalidades de creación y publicación de contenido se muestran en la Figura 2.19, donde los usuarios pueden subir archivos, videos e imágenes junto con sus publicaciones.

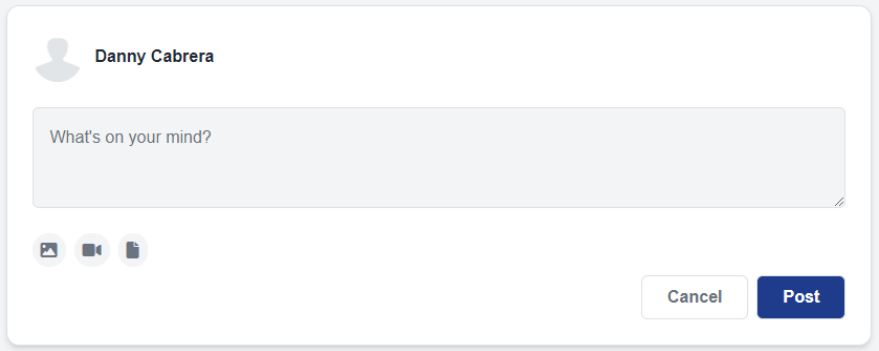


Figura 2.19: Formulario de Creación de Publicaciones.

La Figura 2.20 muestra un perfil completo, con información en *About Me*, *Disciplines*, *Contact Information* y un post.

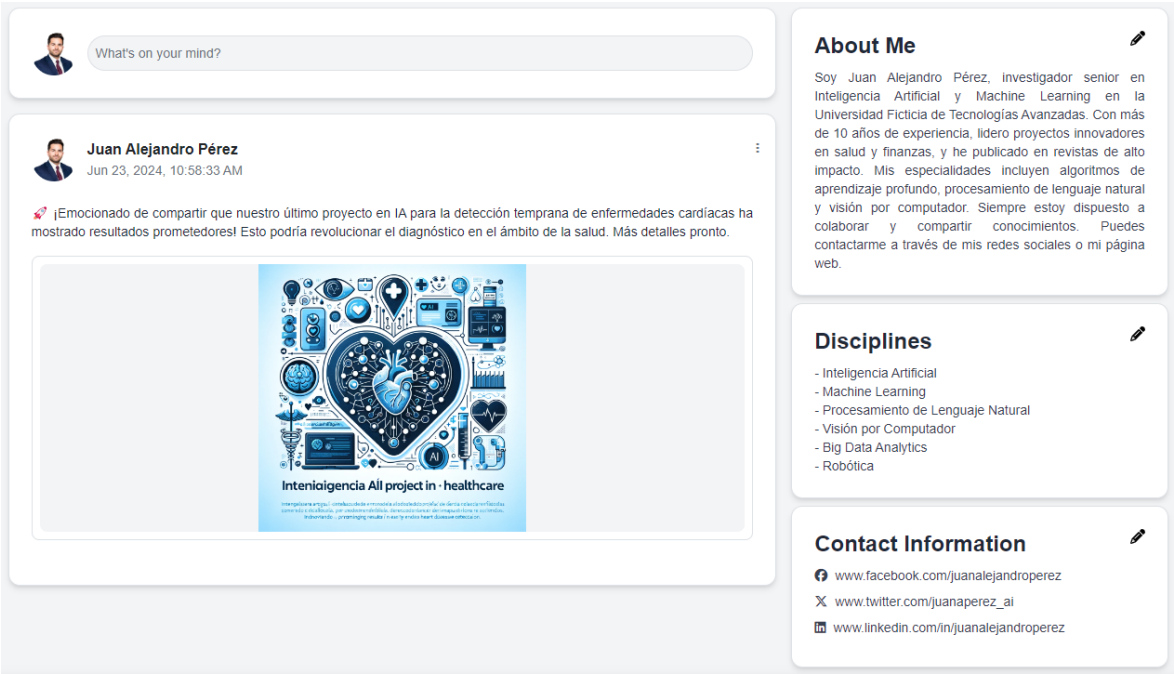


Figura 2.20: Perfil Completo con Información y Post.

2.10.3. Revisión y retrospectiva del sprint

La revisión y retrospectiva del Sprint 3 incluye:

- **Logros Alcanzados**:
  - Diseño y desarrollo exitoso de la pantalla de perfil de usuario.

- Implementación de la sección *About* del perfil, permitiendo a los usuarios agregar y editar información personal, disciplinas y detalles de contacto.
- Validación de la funcionalidad y diseño con usuarios, asegurando que el diseño siga el principio de *Mobile First*.

2.11. Sprint 4

2.11.1. Objetivos del sprint

- Crear modelos y serializadores para la gestión de usuarios en el backend utilizando Django.
- Desarrollar vistas y endpoints necesarios para la autenticación y gestión de perfiles de usuario.

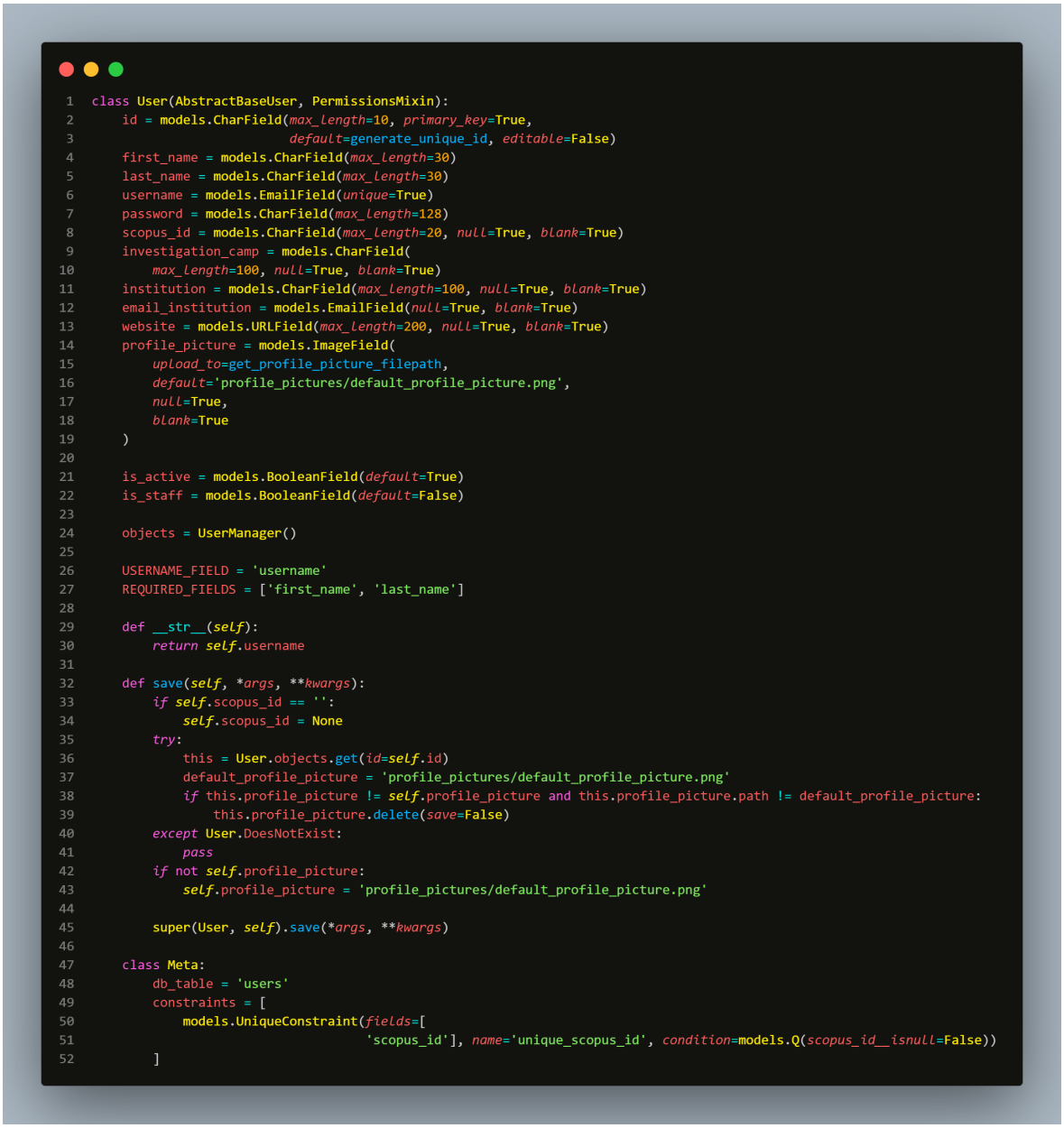
2.11.2. Ejecución del sprint

Durante este sprint, se desarrollaron los modelos, serializadores y vistas necesarios en el backend para soportar las funcionalidades de autenticación y gestión de perfiles de usuario.

Modelos

Se desarrollaron los modelos para representar las entidades principales de la aplicación y sus relaciones. Los modelos permiten la persistencia de datos en la base de datos y proporcionan la lógica de negocio necesaria para el funcionamiento de la aplicación. A continuación se presentan los modelos implementados:

- **User**: Representa a los usuarios de la plataforma, incluyendo campos como nombre, apellido, correo electrónico, contraseña, ID de Scopus y más. Este modelo gestiona la autenticación y autorización de usuarios. Además, permite almacenar información personal y credenciales de acceso de manera segura (Figura 2.21).



```python
class User(AbstractBaseUser, PermissionsMixin):
    id = models.CharField(max_length=10, primary_key=True,
                          default=generate_unique_id, editable=False)
    first_name = models.CharField(max_length=30)
    last_name = models.CharField(max_length=30)
    username = models.EmailField(unique=True)
    password = models.CharField(max_length=128)
    scopus_id = models.CharField(max_length=20, null=True, blank=True)
    investigation_camp = models.CharField(
        max_length=100, null=True, blank=True)
    institution = models.CharField(max_length=100, null=True, blank=True)
    email_institution = models.EmailField(null=True, blank=True)
    website = models.URLField(max_length=200, null=True, blank=True)
    profile_picture = models.ImageField(
        upload_to=get_profile_picture_filepath,
        default='profile_pictures/default_profile_picture.png',
        null=True,
        blank=True
    )

    is_active = models.BooleanField(default=True)
    is_staff = models.BooleanField(default=False)

    objects = UserManager()

    USERNAME_FIELD = 'username'
    REQUIRED_FIELDS = ['first_name', 'last_name']

    def __str__(self):
        return self.username

    def save(self, *args, **kwargs):
        if self.scopus_id == '':
            self.scopus_id = None
        try:
            this = User.objects.get(id=self.id)
            default_profile_picture = 'profile_pictures/default_profile_picture.png'
            if this.profile_picture != self.profile_picture and this.profile_picture.path != default_profile_picture:
                this.profile_picture.delete(save=False)
        except User.DoesNotExist:
            pass
        if not self.profile_picture:
            self.profile_picture = 'profile_pictures/default_profile_picture.png'

        super(User, self).save(*args, **kwargs)

    class Meta:
        db_table = 'users'
        constraints = [
            models.UniqueConstraint(fields=[
                                    'scopus_id'], name='unique_scopus_id', condition=models.Q(scopus_id__isnull=False))
        ]
```

Figura 2.21: Modelo de Usuario.

- **ProfileInformation**: Almacena información adicional del perfil del usuario, incluyendo campos como “About Me”, disciplinas y detalles de contacto. Este modelo facilita la personalización del perfil de usuario y la inclusión de información detallada sobre sus áreas de especialización y formas de contacto (Figura 2.22).

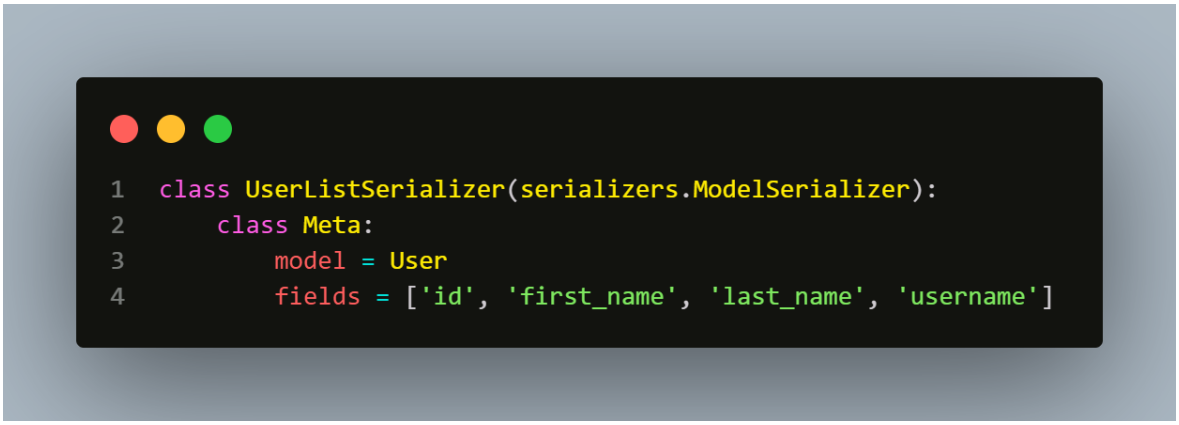
```python
class ProfileInformation(models.Model):
    user = models.OneToOneField(settings.AUTH_USER_MODEL, on_delete=models.CASCADE, related_name='profile_information')
    about_me = models.TextField(blank=True, null=True)
    disciplines = models.JSONField(default=list, blank=True)
    contact_info = models.JSONField(default=dict, blank=True)

    def __str__(self):
        return f"{self.user.username}'s Profile Information"

    class Meta:
        db_table = 'profiles_information'
```

Figura 2.22: Modelo de Información del Perfil.

Serializadores

Se desarrollaron los serializadores para convertir instancias de los modelos en representaciones JSON y viceversa. Los serializadores permiten la creación, actualización y validación de datos para las APIs del componente *B*. A continuación se presentan los serializadores implementados:

- **UserListSerializer**: Permite la visualización de una lista de usuarios con sus nombres y apellidos, facilitando la obtención de información básica de los usuarios. Este serializador es útil para mostrar listas de usuarios en el frontend de manera eficiente y clara (Figura 2.23).



```python
class UserListSerializer(serializers.ModelSerializer):
    class Meta:
        model = User
        fields = ['id', 'first_name', 'last_name', 'username']
```

Figura 2.23: Serializer de Lista de Usuarios.

- **UserSerializer**: Gestiona la actualización de los datos del perfil del usuario, incluyendo la validación de URLs y la actualización de la imagen de perfil. Este serializador asegura que solo los datos válidos se guarden en la base de datos, proporcionando seguridad y

consistencia en la información del usuario (Figura 2.24).

```python
class UserSerializer(serializers.ModelSerializer):
    class Meta:
        model = User
        fields = ['first_name', 'last_name', 'scopus_id', 'institution',
                  'website', 'investigation_camp', 'profile_picture', 'email_institution']

    def validate_website(self, value):
        if value and not value.startswith(('http://', 'https://')):
            value = 'http://' + value
        return value

    def update(self, instance, validated_data):
        print("Datos validados recibidos en el serializador:", validated_data)
        profile_picture = validated_data.pop('profile_picture', None)
        if profile_picture:
            print("Nueva imagen de perfil:", profile_picture)
            if profile_picture != instance.profile_picture:
                instance.profile_picture.delete(save=False)
            instance.profile_picture = profile_picture
        instance = super().update(instance, validated_data)
        print("Instancia después de actualizar:", instance)
        return instance
```

Figura 2.24: Serializer de Usuario.

- **RegisterSerializer**: Gestiona la creación de nuevos usuarios y la encriptación de contraseñas. Este serializador maneja la validación de datos y la creación de la instancia del usuario en la base de datos, asegurando que las contraseñas se almacenen de forma segura utilizando técnicas de encriptación (Figura 2.25).

```python
class RegisterSerializer(serializers.ModelSerializer):
    password = serializers.CharField(write_only=True)

    class Meta:
        model = User
        fields = ['first_name', 'last_name',
                  'username', 'scopus_id', 'password']

    def create(self, validated_data):
        """Crea un nuevo usuario y encripta su contraseña."""
        validated_data['password'] = make_password(validated_data['password'])
        return super().create(validated_data)
```

Figura 2.25: Serializer de Registro de Usuario.

- **UserTokenObtainPairSerializer**: Extiende el `TokenObtainPairSerializer` para agregar el ID del usuario al token, permitiendo una mejor gestión y autenticación de usuarios. Este serializador facilita la implementación de mecanismos de autenticación más robustos y seguros (Figura 2.26).

```python
class UserTokenObtainPairSerializer(TokenObtainPairSerializer):
    @classmethod
    def get_token(cls, user):
        """Agrega el ID del usuario al token."""
        token = super().get_token(user)
        token['user_id'] = user.id
        return token
```

Figura 2.26: Serializer de Obtención de Token de Usuario.

- **ProfileInformationSerializer**: Gestiona la serialización de la información adicional del perfil del usuario, incluyendo campos como “About Me”, disciplinas y detalles de contacto. Este serializador permite a los usuarios personalizar sus perfiles y compartir más información sobre sus intereses y áreas de especialización (Figura 2.27).

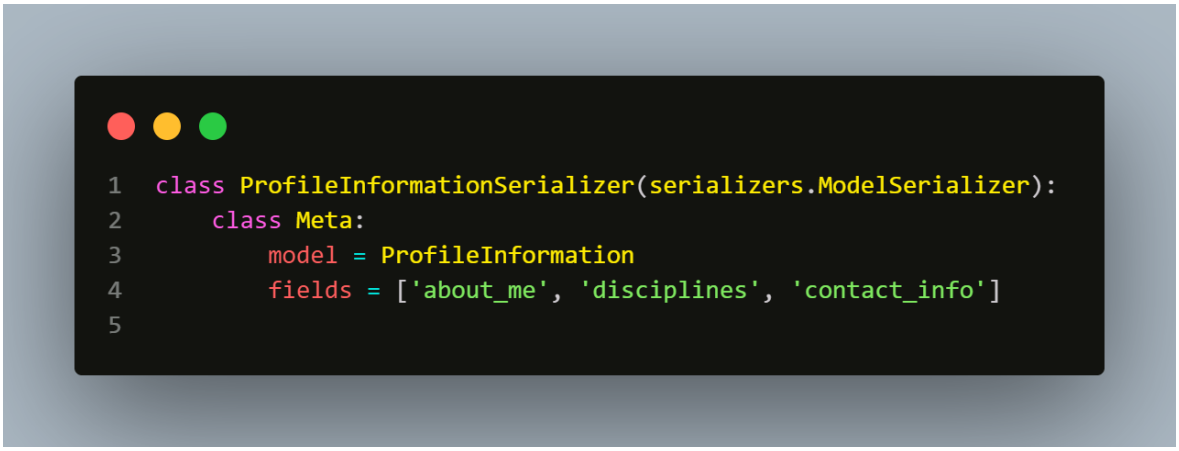


Figura 2.27: Serializer de Información del Perfil.

Views

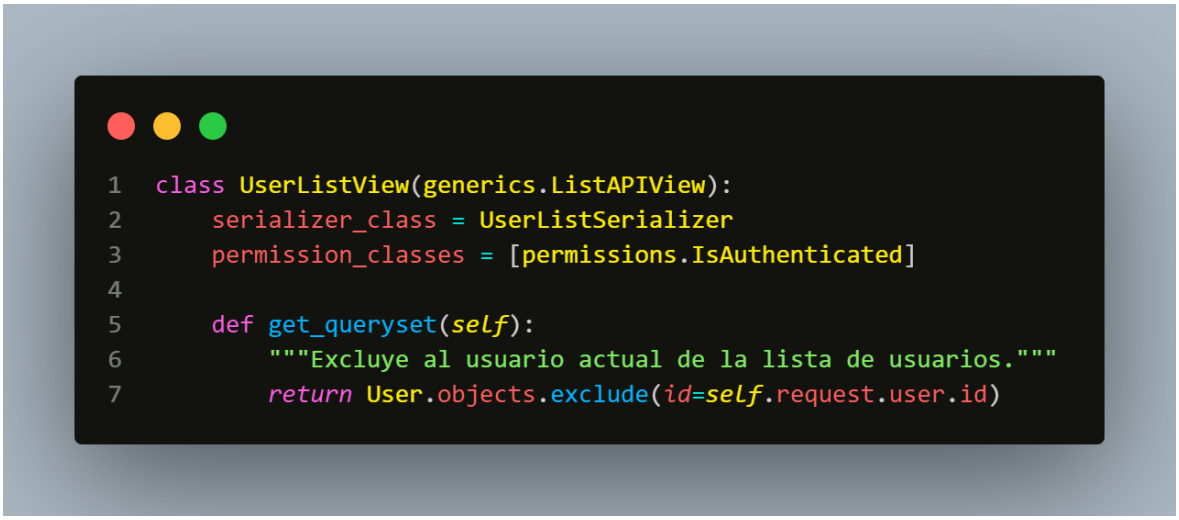
En esta subsección se presentan las vistas creadas para interactuar con los modelos y serializadores en la API. Las vistas se desarrollaron utilizando las vistas genéricas de Django REST Framework, que proporcionan una gran cantidad de funcionalidad y permiten una implementación más eficiente y concisa.

- **RegisterView**: Esta vista maneja el registro de nuevos usuarios. Implementa una clase basada en `generics.CreateAPIView` que permite la creación de usuarios mediante la validación de los datos proporcionados en la solicitud. Si los datos son válidos, se guarda el nuevo usuario en la base de datos y se devuelve una respuesta de éxito. En caso contrario, se devuelven los errores de validación correspondientes (Figura 2.28).

```python
class RegisterView(generics.CreateAPIView):
    queryset = User.objects.all()
    serializer_class = RegisterSerializer

    def create(self, request, *args, **kwargs):
        """Crea un nuevo usuario y maneja los errores de validación."""
        serializer = self.get_serializer(data=request.data)
        try:
            serializer.is_valid(raise_exception=True)
            self.perform_create(serializer)
            headers = self.get_success_headers(serializer.data)
            return Response(serializer.data, status=status.HTTP_201_CREATED, headers=headers)
        except ValidationError as e:
            return Response(self.format_errors(serializer.errors), status=status.HTTP_400_BAD_REQUEST)
        except Exception as e:
            return Response({'error': str(e)}, status=status.HTTP_500_INTERNAL_SERVER_ERROR)

```

Figura 2.28: Vista para el registro de nuevos usuarios.

- **UserListView**: Esta vista proporciona una lista de usuarios excluyendo al usuario que realiza la solicitud. Utiliza `generics.ListAPIView` para listar todos los usuarios del sistema, aplicando un filtro que excluye al usuario autenticado. Esto es útil para mostrar una lista de otros usuarios en el sistema sin incluir al usuario actual (Figura 2.29).



```python
class UserListView(generics.ListAPIView):
    serializer_class = UserListSerializer
    permission_classes = [permissions.IsAuthenticated]

    def get_queryset(self):
        """Excluye al usuario actual de la lista de usuarios."""
        return User.objects.exclude(id=self.request.user.id)
```

Figura 2.29: Vista para la lista de usuarios excluyendo al usuario actual.

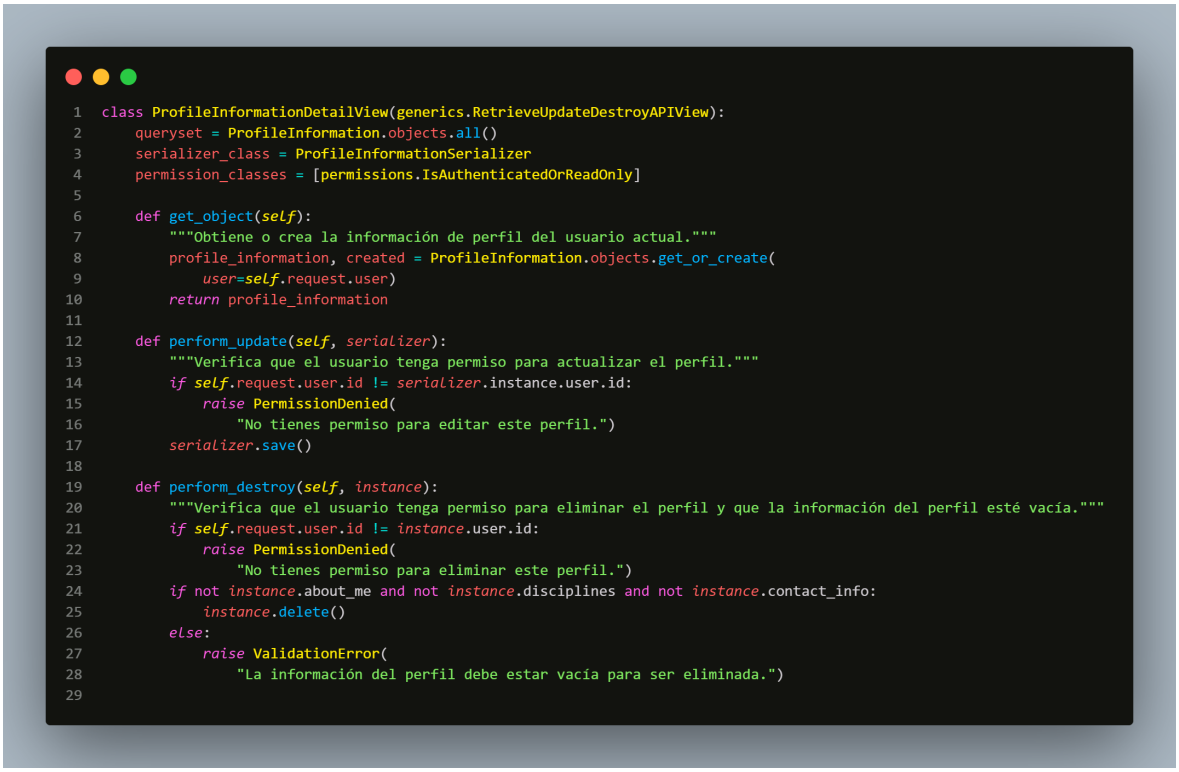
- **UserUpdateView**: Esta vista maneja la actualización de la información del usuario. Utiliza `generics.UpdateAPIView` para permitir que los usuarios actualicen su perfil y la imagen del perfil. La vista verifica que el usuario que realiza la solicitud tiene permisos para actualizar el perfil especificado y maneja la lógica de actualización, incluida la validación de

los datos actualizados (Figura 2.30).

```python
class UserUpdateView(generics.UpdateAPIView):
    queryset = User.objects.all()
    serializer_class = UserSerializer
    permission_classes = [permissions.IsAuthenticated]
    parser_classes = (MultiPartParser, FormParser)

    def get_object(self):
        obj = super().get_object()
        if obj.id != self.request.user.id:
            raise PermissionDenied(
                "No tienes permiso para editar este usuario.")
        return obj

    def update(self, request, *args, **kwargs):
        print("Datos recibidos en la solicitud de actualización:", request.data)
        if 'profile_picture' in request.data:
            print("Imagen de perfil recibida:",
                  request.data['profile_picture'])
        kwargs['partial'] = True
        response = super().update(request, *args, **kwargs)
        print("Respuesta después de actualizar:", response.data)
        return response
```

Figura 2.30: Vista para la actualización de la información del usuario.

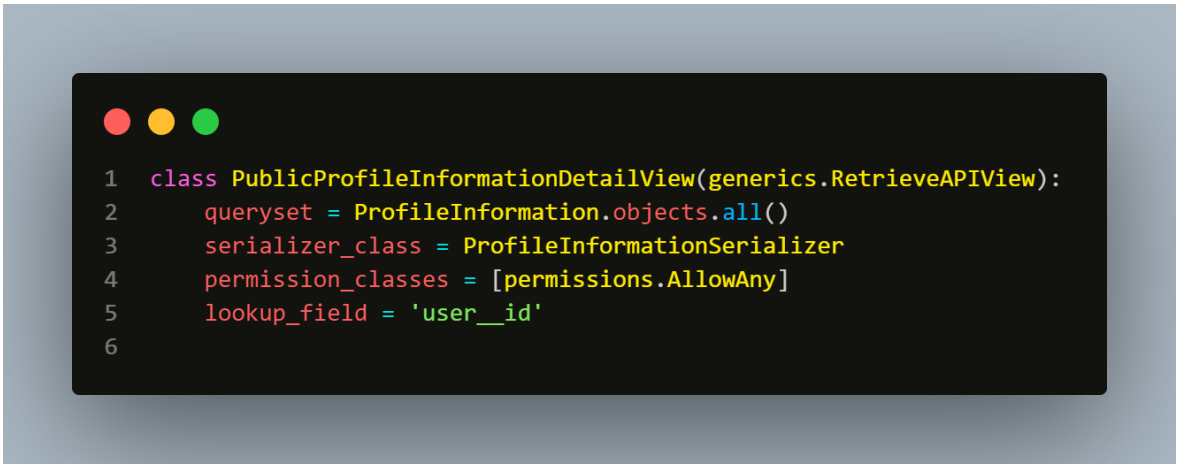
- **ProfileInformationDetailView**: Esta vista maneja la recuperación, actualización y eliminación de la información del perfil del usuario actual. Utiliza `generics.RetrieveUpdateDestroyAPIView` para proporcionar estas funcionalidades. La vista asegura que solo el usuario propietario del perfil pueda actualizar o eliminar su información, aplicando las verificaciones necesarias (Figura 2.31).



```python
class ProfileInformationDetailView(generics.RetrieveUpdateDestroyAPIView):
    queryset = ProfileInformation.objects.all()
    serializer_class = ProfileInformationSerializer
    permission_classes = [permissions.IsAuthenticatedOrReadOnly]

    def get_object(self):
        """Obtiene o crea la información de perfil del usuario actual."""
        profile_information, created = ProfileInformation.objects.get_or_create(
            user=self.request.user)
        return profile_information

    def perform_update(self, serializer):
        """Verifica que el usuario tenga permiso para actualizar el perfil."""
        if self.request.user.id != serializer.instance.user.id:
            raise PermissionDenied(
                "No tienes permiso para editar este perfil.")
        serializer.save()

    def perform_destroy(self, instance):
        """Verifica que el usuario tenga permiso para eliminar el perfil y que la información del perfil esté vacía."""
        if self.request.user.id != instance.user.id:
            raise PermissionDenied(
                "No tienes permiso para eliminar este perfil.")
        if not instance.about_me and not instance.disciplines and not instance.contact_info:
            instance.delete()
        else:
            raise ValidationError(
                "La información del perfil debe estar vacía para ser eliminada.")

```

Figura 2.31: Vista para el manejo de la información del perfil del usuario.

- **PublicProfileInformationDetailView**: Esta vista proporciona acceso a la información pública del perfil de un usuario específico utilizando su `user_id`. Implementa `generics.RetrieveAPIView` para obtener los detalles del perfil público y asegura que cualquier usuario autenticado pueda acceder a esta información (Figura 2.32).



```python
class PublicProfileInformationDetailView(generics.RetrieveAPIView):
    queryset = ProfileInformation.objects.all()
    serializer_class = ProfileInformationSerializer
    permission_classes = [permissions.AllowAny]
    lookup_field = 'user__id'

```

Figura 2.32: Vista para el acceso a la información pública del perfil de un usuario específico.

2.11.3. Revisión y retrospectiva del sprint

La revisión y retrospectiva del Sprint 4 incluye:

- **Logros Alcanzados**:
  - Creación exitosa de modelos y serializadores para la gestión de usuarios y perfiles en el backend.
  - Desarrollo de vistas y endpoints necesarios para la autenticación y gestión de perfiles de usuario.
- **Desafíos Encontrados**:
  - Asegurar la seguridad y privacidad de los datos de los usuarios.
  - Coordinación con el equipo frontend para la integración de APIs.
- **Lecciones Aprendidas**:
  - La importancia de validar y probar exhaustivamente las APIs para asegurar su correcto funcionamiento.

2.12. Sprint 5

2.12.1. Objetivos del sprint

- Crear modelos y serializadores para la gestión de publicaciones y grupos en el backend utilizando Django.
- Desarrollar vistas y endpoints necesarios para la gestión de publicaciones y grupos.

2.12.2. Ejecución del sprint

Durante este sprint, se desarrollaron los modelos, serializadores y vistas necesarios en el backend para soportar las funcionalidades de creación y gestión de publicaciones y grupos.

Modelos

Se desarrollaron los modelos para representar las entidades principales de la aplicación y sus relaciones. Los modelos permiten la persistencia de datos en la base de datos y proporcionan la lógica de negocio necesaria para el funcionamiento de la aplicación. A continuación se presentan los modelos implementados:

- **Post**: Almacena las publicaciones creadas por los usuarios, incluyendo campos como descripción y archivos adjuntos. Este modelo gestiona la creación y visualización de publicaciones dentro de la plataforma, permitiendo a los usuarios compartir contenido multimedia y texto (Figura 2.33).

```python
class Post(models.Model):
    id = models.CharField(max_length=10, primary_key=True, default=generate_unique_id, editable=False)
    user = models.ForeignKey(settings.AUTH_USER_MODEL, on_delete=models.CASCADE, related_name='posts')
    description = models.TextField()
    created_at = models.DateTimeField(null=True, blank=True)  # Permitir valores nulos y vacíos

    def __str__(self):
        return f"Post by {self.user.username} on {self.created_at}"

    class Meta:
        db_table = 'posts'

class PostFile(models.Model):
    post = models.ForeignKey(Post, on_delete=models.CASCADE, related_name='files')
    file = models.FileField(upload_to=get_post_file_filepath, blank=True, null=True)

    def __str__(self):
        return f"File for post {self.post.id}"

    class Meta:
        db_table = 'post_files'

```

Figura 2.33: Modelo de Publicación.

- **Group**: Representa los grupos dentro de la plataforma, permitiendo a los usuarios crear y unirse a grupos de investigación. Este modelo incluye campos como título, descripción, administrador y miembros, facilitando la colaboración y la organización de grupos de interés en la plataforma (Figura 2.34).

```python
class Group(models.Model):
    id = models.CharField(max_length=10, primary_key=True, default=generate_unique_id, editable=False)
    title = models.CharField(max_length=255)
    description = models.TextField()
    admin = models.ForeignKey(settings.AUTH_USER_MODEL, on_delete=models.CASCADE, related_name='administered_groups')
    users = models.ManyToManyField(settings.AUTH_USER_MODEL, through='GroupUser', related_name='member_groups')

    def __str__(self):
        return self.title

    class Meta:
        db_table = 'groups'

class GroupUser(models.Model):
    group = models.ForeignKey(Group, on_delete=models.CASCADE)
    user = models.ForeignKey(settings.AUTH_USER_MODEL, on_delete=models.CASCADE)

    class Meta:
        db_table = 'group_users'

```

Figura 2.34: Modelo de Grupo.

Serializadores

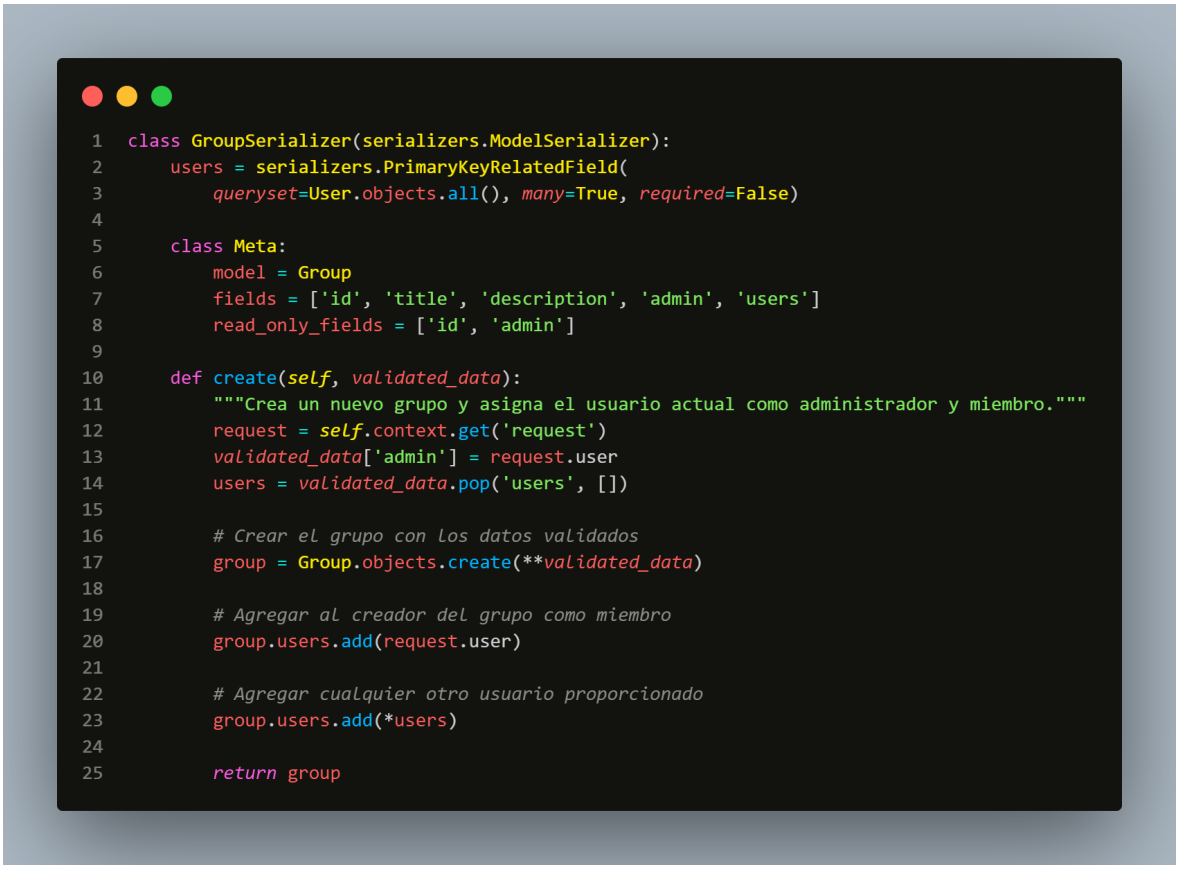
Se desarrollaron los serializadores para convertir instancias de los modelos en representaciones JSON y viceversa. Los serializadores permiten la creación, actualización y validación de datos para las APIs del componente *B*. A continuación se presentan los serializadores implementados:

- **PostSerializer**: Permite la serialización de publicaciones y sus archivos asociados. Este serializador maneja la creación de publicaciones y la asociación de múltiples archivos a una publicación, facilitando la gestión y visualización de contenido multimedia en la plataforma (Figura 2.35).

```python
class PostFileSerializer(serializers.ModelSerializer):
    class Meta:
        model = PostFile
        fields = ['file']


class PostSerializer(serializers.ModelSerializer):
    files = PostFileSerializer(many=True, read_only=True)

    class Meta:
        model = Post
        fields = ['id', 'description', 'files', 'created_at']

    def create(self, validated_data):
        """Crea una nueva publicación y adjunta archivos si se proporcionan."""
        request = self.context.get('request')
        user = request.user

        if 'user' in validated_data:
            validated_data.pop('user')

        files = request.FILES.getlist('files', [])
        post = Post.objects.create(user=user, **validated_data)
        for file in files:
            PostFile.objects.create(post=post, file=file)
        return post

```

Figura 2.35: Serializer de Publicaciones.

- **GroupSerializer**: Se encarga de la serialización del modelo de Grupo, permitiendo la creación y gestión de grupos de usuarios. Este serializador también permite asignar automáticamente el usuario actual como administrador y miembro del grupo, asegurando una gestión eficiente y controlada de los grupos (Figura 2.36).



```python
class GroupSerializer(serializers.ModelSerializer):
    users = serializers.PrimaryKeyRelatedField(
        queryset=User.objects.all(), many=True, required=False)

    class Meta:
        model = Group
        fields = ['id', 'title', 'description', 'admin', 'users']
        read_only_fields = ['id', 'admin']

    def create(self, validated_data):
        """Crea un nuevo grupo y asigna el usuario actual como administrador y miembro."""
        request = self.context.get('request')
        validated_data['admin'] = request.user
        users = validated_data.pop('users', [])

        # Crear el grupo con los datos validados
        group = Group.objects.create(**validated_data)

        # Agregar al creador del grupo como miembro
        group.users.add(request.user)

        # Agregar cualquier otro usuario proporcionado
        group.users.add(*users)

        return group
```

Figura 2.36: Serializer de Grupo.

Views

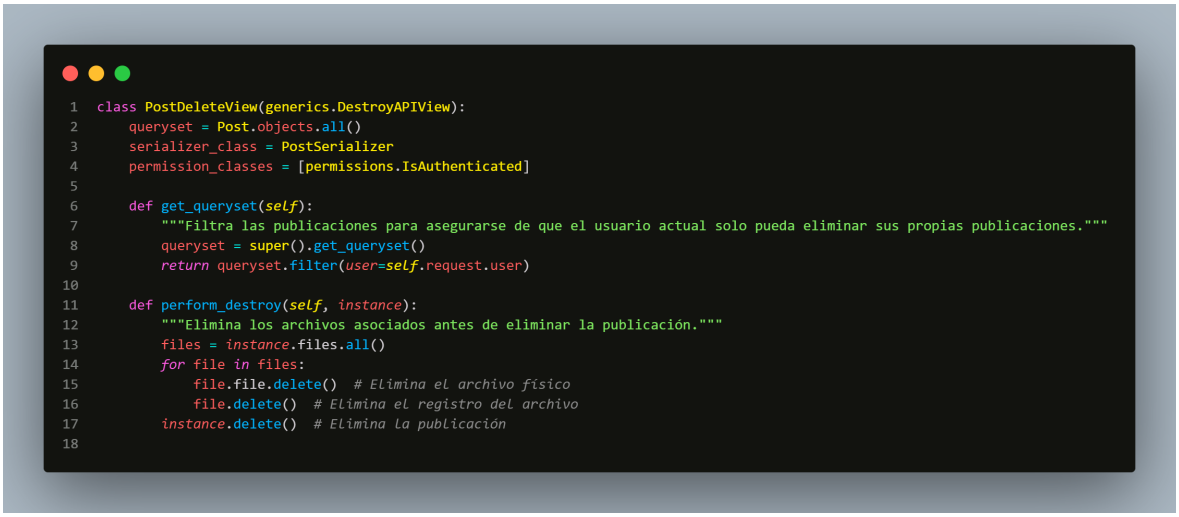
En esta subsección se presentan las vistas creadas para interactuar con los modelos y serializadores en la API. Las vistas se desarrollaron utilizando las vistas genéricas de Django REST Framework, que proporcionan una gran cantidad de funcionalidad y permiten una implementación más eficiente y concisa.

- **PostCreateView**: Esta vista maneja la creación de publicaciones, permitiendo que los usuarios adjunten archivos a sus publicaciones. Utiliza `generics.ListCreateAPIView` para manejar tanto la visualización de la lista de publicaciones como la creación de nuevas publicaciones. Además, la vista asegura que el usuario autenticado se asigne correctamente a la nueva publicación como su autor (Figura 2.37).

```python
class PostCreateView(generics.ListCreateAPIView):
    queryset = Post.objects.all()
    serializer_class = PostSerializer
    permission_classes = [permissions.IsAuthenticated]
    parser_classes = [MultiPartParser, FormParser]

    def perform_create(self, serializer):
        """Asigna el usuario actual al crear una nueva publicación."""
        serializer.save(user=self.request.user)
```

Figura 2.37: Vista para la creación de publicaciones.

- **PostDeleteView**: Esta vista maneja la eliminación de publicaciones. Asegura que los usuarios solo puedan eliminar sus propias publicaciones y los archivos adjuntos a ellas. Implementa `generics.DestroyAPIView` y filtra las publicaciones para asegurarse de que el usuario autenticado solo pueda eliminar sus propias publicaciones. Además, maneja la eliminación de los archivos físicos adjuntos a las publicaciones (Figura 2.38).



```python
class PostDeleteView(generics.DestroyAPIView):
    queryset = Post.objects.all()
    serializer_class = PostSerializer
    permission_classes = [permissions.IsAuthenticated]

    def get_queryset(self):
        """Filtra las publicaciones para asegurarse de que el usuario actual solo pueda eliminar sus propias publicaciones."""
        queryset = super().get_queryset()
        return queryset.filter(user=self.request.user)

    def perform_destroy(self, instance):
        """Elimina los archivos asociados antes de eliminar la publicación."""
        files = instance.files.all()
        for file in files:
            file.file.delete()  # Elimina el archivo físico
            file.delete()  # Elimina el registro del archivo
        instance.delete()  # Elimina la publicación

```

Figura 2.38: Vista para la eliminación de publicaciones.

- **PostListView**: Esta vista proporciona una lista de publicaciones de un usuario específico si se proporciona un `user_id`. Utiliza `generics.ListAPIView` para listar las publicaciones y aplica un filtro basado en el `user_id` proporcionado en los parámetros de la solicitud. Esto

permite la visualización de las publicaciones de un usuario específico de manera eficiente (Figura 2.39).

```python
class PostListView(generics.ListAPIView):
    serializer_class = PostSerializer
    permission_classes = [permissions.AllowAny]

    def get_queryset(self):
        """Obtiene las publicaciones de un usuario específico si se proporciona un `user_id`."""
        user_id = self.request.query_params.get('user_id')
        if user_id:
            return Post.objects.filter(user_id=user_id).order_by('-created_at')
        return Post.objects.none()
```

Figura 2.39: Vista para el listado de publicaciones de un usuario específico.

- **GroupListCreateView**: Esta vista maneja la creación y listado de grupos. Utiliza `generics.ListCreateAPIView` para permitir a los usuarios ver la lista de grupos y crear nuevos grupos. Al crear un grupo, el usuario que realiza la solicitud se asigna automáticamente como administrador del grupo, lo que asegura una correcta administración y control sobre el grupo creado (Figura 2.40).

```python
class GroupListCreateView(generics.ListCreateAPIView):
    queryset = Group.objects.all()
    serializer_class = GroupSerializer
    permission_classes = [permissions.IsAuthenticated]

    def perform_create(self, serializer):
        """Asigna el usuario actual como administrador al crear un nuevo grupo."""
        serializer.save(admin=self.request.user)

    def create(self, request, *args, **kwargs):
        """Maneja la creación de un nuevo grupo."""
        serializer = self.get_serializer(data=request.data)
        if serializer.is_valid():
            self.perform_create(serializer)
            headers = self.get_success_headers(serializer.data)
            return Response(serializer.data, status=status.HTTP_201_CREATED, headers=headers)
        else:
            return Response(serializer.errors, status=status.HTTP_400_BAD_REQUEST)

```

Figura 2.40: Vista para la creación y listado de grupos.

2.12.3. Revisión y retrospectiva del sprint

La revisión y retrospectiva del Sprint 5 incluye:

- **Logros Alcanzados**:
  - Creación exitosa de modelos y serializadores para la gestión de publicaciones y grupos en el backend.
  - Desarrollo de vistas y endpoints necesarios para la gestión de publicaciones y grupos.
- **Desafíos Encontrados**:
  - Asegurar la correcta asociación de archivos a las publicaciones.
  - Gestión eficiente de los permisos y roles dentro de los grupos.
- **Lecciones Aprendidas**:
  - La importancia de una buena gestión de archivos en el backend.
  - La necesidad de validar y probar exhaustivamente las APIs para asegurar su correcto funcionamiento.

2.13. Sprint 6

2.13.1. Objetivos del sprint

- Integrar la comunicación entre el frontend y el backend mediante APIs RESTful.
- Configurar el entorno de desarrollo usando Docker.
- Realizar pruebas de integración para asegurar el funcionamiento correcto de la aplicación.
- Realizar ajustes finales y preparar para el despliegue.

2.13.2. Ejecución del sprint

Durante este sprint, se realizaron las siguientes actividades para asegurar la integración y el correcto funcionamiento del sistema:

Conexión del Backend con el Frontend

En esta subsección se detallan los componentes desarrollados en el frontend para la comunicación con la API del backend. Estos componentes manejan las operaciones CRUD (Crear, Leer, Actualizar y Eliminar) para los recursos del sistema.

- **Configuración del Entorno**: En este archivo se define la URL base de la API y otras variables de entorno necesarias para que el frontend pueda comunicarse con el backend. La configuración es crucial para asegurar que el frontend apunte al servidor correcto y que las variables de entorno se manejen adecuadamente (Figura 2.41).

Figura 2.41: Configuración del Entorno

- **Rutas de la Aplicación**: Este archivo configura las rutas en el backend que permiten al frontend acceder a los diferentes endpoints de la API. Aquí se definen las rutas para operaciones como la obtención de tokens, registro de usuarios, y manejo de publicaciones (Figura 2.42).

```python
urlpatterns = [
    path('token/', UserTokenObtainPairView.as_view(), name='token_obtain_pair'),
    path('token/refresh/', TokenRefreshView.as_view(), name='token_refresh'),
    path('register/', RegisterView.as_view(), name='register'),
    re_path(r'^users/(?P<pk>[a-zA-Z0-9]+)/$',
            UserDetailView.as_view(), name='user-detail'),
    re_path(r'^users/(?P<pk>[a-zA-Z0-9]+)/update/$',
            UserUpdateView.as_view(), name='user-update'),
    path('users/', UserListView.as_view(), name='user-list'),
    path('groups/', GroupListCreateView.as_view(), name='group-list-create'),
    path('profile-information/', ProfileInformationDetailView.as_view(),
         name='profile-information-detail'),
    path('profile-information/<str:user__id>/', PublicProfileInformationDetailView.as_view(),
         name='public-profile-information-detail'),
    path('posts/', PostListView.as_view(), name='post-list'),
    path('posts/create/', PostCreateView.as_view(), name='post-create'),
    path('posts/<pk>/delete/', PostDeleteView.as_view(),
         name='post-delete'),
]

```

Figura 2.42: Rutas de la Aplicación

- **Manejo de Registro y Login de Usuarios**: Este componente gestiona las solicitudes de registro y login de los usuarios, enviando las credenciales al backend y manejando la respuesta para autenticar al usuario en el sistema (Figura 2.43).

```typescript
register(user: User): Observable<any> {
  return this.http.post(`${this.apiUrl}/register/`, user).pipe(
    catchError(this.handleError)
  );
}
login(credentials: LoginCredentials): Observable<AuthResponse> {
  return this.http.post<AuthResponse>(`${this.apiUrl}/token/`, credentials).pipe(
    tap(response => this.setSession(response)),
    catchError(this.handleError)
  );
}
```

Figura 2.43: Manejo de Registro y Login de Usuarios

- **Actualización del Token de Acceso**: Para mantener la sesión activa, este componente se encarga de obtener un nuevo token de acceso utilizando el token de refresco almacenado. Esto es esencial para asegurar que las solicitudes del usuario continúen siendo

autenticadas sin necesidad de volver a iniciar sesión (Figura 2.44).

```
private refreshAccessToken(): Observable<AuthResponse> {
  const refreshToken = localStorage.getItem('refreshToken');
  if (!refreshToken) {
    this.logout();
    return throwError(() => new Error('Refresh token not found'));
  }
  return this.http.post<AuthResponse>(`${this.apiUrl}/token/refresh/`, { refresh: refreshToken }).pipe(
    tap(response => {
      localStorage.setItem('accessToken', response.access);
    })
  );
}
```

Figura 2.44: Actualización del Token de Acceso

- **Obtener Usuarios**: Este servicio permite recuperar la lista completa de usuarios registrados en el sistema, enviando una solicitud al endpoint correspondiente y manejando la respuesta del backend (Figura 2.45).

```
getUsers(): Observable<User[]> {
  const accessToken = localStorage.getItem('accessToken');
  if (!accessToken) {
    return throwError(() => new Error('Access token not found'));
  }
  const headers = new HttpHeaders({
    'Authorization': `Bearer ${accessToken}`
  });
  return this.http.get<User[]>(`${this.apiUrl}/users/`, { headers }).pipe(
    catchError(error => {
      if (error.status === 401 && error.error.code === 'token_not_valid') {
        return this.refreshAccessToken().pipe(
          switchMap(() => this.getUsers())
        );
      }
      return this.handleError(error);
    })
  );
}
```

Figura 2.45: Obtener Usuarios

- **Obtener Usuario por ID**: Provee la funcionalidad para obtener los detalles de un usua-

rio específico utilizando su ID. Este método es útil para mostrar información detallada del perfil del usuario en el frontend (Figura 2.46).

```
getUserById(userId: string): Observable<UserProfile> {
  return this.http.get<UserProfile>(`${this.apiUrl}/users/${userId}`);
}
```

Figura 2.46: Obtener Usuario por ID

- **Actualizar Usuario**: Este componente gestiona la actualización de los datos del usuario, permitiendo modificar información del perfil como el nombre, correo electrónico, etc. Se asegura de que los datos sean válidos antes de enviar la solicitud de actualización al backend (Figura 2.47).

```
updateUser(formData: FormData): Observable<any> {
  console.log(formData);
  const userId = this.getUserId();
  if (!userId) {
    return throwError(() => new Error('User ID not found'));
  }
  return this.http.put(`${this.apiUrl}/users/${userId}/update/`, formData, {
    headers: new HttpHeaders({
      'Authorization': `Bearer ${localStorage.getItem('accessToken')}`
    })
  }).pipe(
    catchError(error => {
      if (error.status === 401 && error.error.code === 'token_not_valid') {
        return this.refreshAccessToken().pipe(
          switchMap(() => this.updateUser(formData))
        );
      }
      return this.handleError(error);
    })
  );
}
```

Figura 2.47: Actualizar Usuario

- **Manejo de Información del Perfil**: Estas funciones manejan la obtención, actualización y eliminación de la información del perfil del usuario. También incluye la funcionalidad para obtener información pública del perfil, accesible a otros usuarios (Figura 2.48).

```
getInformation(userId: string): Observable<UserInfo> {
  return this.http.get<UserInfo>(`${this.apiUrl}/profile-information/${userId}/`, this.getHeaders());
}
updateInformation(info: UserInfo): Observable<UserInfo> {
  return this.http.put<UserInfo>(`${this.apiUrl}/profile-information/`, info, this.getHeaders());
}
deleteInformation(): Observable<void> {
  return this.http.delete<void>(`${this.apiUrl}/profile-information/`, this.getHeaders());
}
getPublicInformation(userId: string): Observable<UserInfo> {
  return this.http.get<UserInfo>(`${this.apiUrl}/profile-information/${userId}/`);
}
```

Figura 2.48: Manejo de Información del Perfil

- **Manejo de Publicaciones**: Contiene las funciones `createPost`, `getPosts` y `deletePost`, que permiten crear nuevas publicaciones, obtener las publicaciones existentes y eliminarlas, respectivamente. Estas funciones son esenciales para el manejo de contenido generado por el usuario (Figura 2.49).

```
createPost(postData: FormData): Observable<Post> {
  const headers = new HttpHeaders({
    'Authorization': `Bearer ${localStorage.getItem('accessToken')}`
  });
  return this.http.post<Post>(`${this.apiUrl}create/`, postData, { headers }).pipe(
    map(post => this.convertPostDates(post))
  );
}
getPosts(userId: string): Observable<Post[]> {
  return this.http.get<Post[]>(`${this.apiUrl}?user_id=${userId}`).pipe(
    map(posts => posts.map(post => this.convertPostDates(post)))
  );
}
deletePost(postId: string): Observable<void> {
  const headers = new HttpHeaders({
    'Authorization': `Bearer ${localStorage.getItem('accessToken')}`
  });
  return this.http.delete<void>(`${this.apiUrl}${postId}/delete/`, { headers });
}
```

Figura 2.49: Manejo de Publicaciones

- **Manejo de Grupos**: Incluye la función `createGroup`, que permite la creación de nuevos grupos en el sistema. Esta funcionalidad es importante para la organización de usuarios en grupos y para el manejo de permisos y roles dentro de la aplicación (Figura 2.50).

```
createGroup(groupData: { title: string, description: string, users?: any[] }): Observable<any> {
    const headers = new HttpHeaders({
        'Content-Type': 'application/json',
        'Authorization': `Bearer ${localStorage.getItem('accessToken')}`
    });

    return this.http.post(this.apiUrl, groupData, { headers }).pipe(
        catchError(this.handleError)
    );
}
```

Figura 2.50: Manejo de Grupos

Configuración de Docker

La configuración de Docker permite ejecutar el proyecto en un entorno aislado, asegurando la consistencia entre diferentes máquinas de desarrollo y producción. En este sprint, se crearon los archivos de configuración de Docker necesarios para construir y ejecutar la

aplicación.

El archivo `docker-compose.yml` (Figura 2.51) define los servicios para la aplicación web y la base de datos PostgreSQL. Este archivo especifica las imágenes de Docker a utilizar, los volúmenes y las variables de entorno requeridas.

```yaml
version: '3.8'
services:
  web:
    build: .
    command: python manage.py runserver 0.0.0.0:8000
    volumes:
      - .:/app
    ports:
      - "8000:8000"
    depends_on:
      - db
    environment:
      - DJANGO_SETTINGS_MODULE=project.settings
  db:
    image: postgres:16
    volumes:
      - postgres_data:/var/lib/postgresql/data/
    environment:
      - POSTGRES_DB=${DB_NAME}
      - POSTGRES_USER=${DB_USER}
      - POSTGRES_PASSWORD=${DB_PASSWORD}

volumes:
  postgres_data:

```

Figura 2.51: Archivo `docker-compose.yml`.

El archivo `Dockerfile` (Figura 2.52) contiene las instrucciones para construir la imagen de Docker de la aplicación web. Este archivo especifica la imagen base de Python, las dependencias a instalar y el comando para ejecutar el servidor de desarrollo de Django.

```dockerfile
# Usar una imagen oficial de Python
FROM python:3.11-bookworm

# Establecer el directorio de trabajo en el contenedor
WORKDIR /app

# Instalar dependencias
COPY requirements.txt .
RUN pip install --no-cache-dir -r requirements.txt

# Copiar el resto del código fuente del proyecto
COPY . .

# Comando para ejecutar el servidor
CMD ["python", "manage.py", "runserver", "0.0.0.0:8000"]

```

Figura 2.52: Archivo `Dockerfile`.

2.13.3. Revisión y retrospectiva del sprint

La revisión y retrospectiva del Sprint 6 incluye:

- **Logros Alcanzados**:
  - Integración exitosa del frontend y backend mediante APIs RESTful.
  - Configuración correcta de rutas y servicios en Angular para asegurar la comunicación entre el frontend y el backend.
  - Configuración del entorno de desarrollo usando Docker.
  - Realización de pruebas de integración para asegurar el funcionamiento correcto de la aplicación.
- **Desafíos Encontrados**:
  - Asegurar la correcta integración de todas las funcionalidades desarrolladas.
  - Identificación y corrección de errores durante las pruebas de integración.

- **Lecciones Aprendidas**:
    - La importancia de realizar pruebas de integración exhaustivas para asegurar el funcionamiento correcto del sistema.
    - La necesidad de una buena coordinación entre los equipos de frontend y backend durante la integración.

3. EVALUACIÓN Y RESULTADOS

En este capítulo, se presentan y discuten los resultados obtenidos de la encuesta de usabilidad SUS aplicada al sistema CENTINELA. La escala de usabilidad del sistema (SUS) es una herramienta confiable y rápida para medir la usabilidad percibida por los usuarios [27]. Los puntajes SUS pueden variar entre 0 y 100, siendo 100 la mejor usabilidad posible.

3.1. Resultados de la encuesta SUS

La encuesta se realizó a 12 personas, incluyendo estudiantes de quinto semestre o superiores y profesores de la Facultad de Sistemas de la EPN, quienes evaluaron el sistema. A continuación, se presentan los puntajes individuales obtenidos de los encuestados:

Encuestado	Puntaje SUS
1	100.0
2	92.5
3	90.0
4	55.0
5	75.0
6	82.5
7	62.5
8	85.0
9	80.0
10	77.5
11	100.0
12	77.5

Tabla 3.1: Puntajes SUS individuales de los encuestados.

3.2. Análisis de los resultados

El análisis de los puntajes SUS revela que el sistema CENTINELA obtuvo una puntuación media de 81.46, lo cual indica una usabilidad alta. A continuación, se detalla el significado de estos resultados:

- **Puntajes superiores a 80**: Indican una excelente usabilidad. Los usuarios con puntajes de 100 (Encuestados 1 y 11) sugieren que encontraron el sistema extremadamente

usable, sin dificultades significativas para su uso.

- **Puntajes entre 70 y 80**: Estos puntajes indican una buena usabilidad, aunque pueden existir pequeñas áreas de mejora. La mayoría de los encuestados (Encuestados 5, 8, 9, 10 y 12) se encuentran en este rango.
- **Puntajes entre 50 y 70**: Reflejan una usabilidad aceptable pero con áreas claras de mejora. El encuestado 7, con un puntaje de 62.5, podría haber experimentado dificultades menores en ciertas áreas del sistema.
- **Puntajes inferiores a 50**: Indican problemas significativos de usabilidad. En este estudio, el encuestado 4 obtuvo un puntaje de 55, lo cual sugiere que encontraron el sistema bastante complicado de usar.

3.3. Gráfica de resultados

Para una mejor visualización y análisis de los resultados, se presenta la siguiente gráfica que muestra los puntajes individuales de cada encuestado:

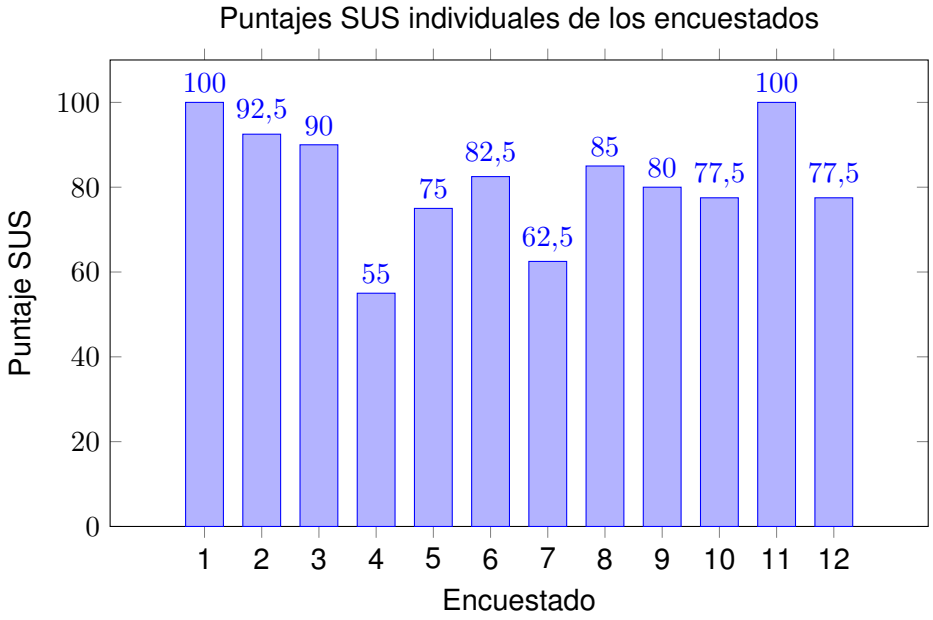


Figura 3.1: Gráfica de puntajes SUS individuales de los encuestados.

3.3.1. Interpretación de la gráfica

La gráfica anterior ilustra los puntajes SUS obtenidos por cada encuestado. Observamos que:

- Los encuestados 1 y 11 otorgaron la máxima puntuación de 100, indicando una excelente experiencia de usabilidad.
- La mayoría de los encuestados se encuentran en el rango de 70 a 90, lo cual es consistente con una buena usabilidad general del sistema.
- El encuestado 4, con un puntaje de 55, destaca como un valor atípico que indica una experiencia de usuario significativamente menos satisfactoria.

3.3.2. Análisis de preguntas individuales

Para profundizar en el análisis, podemos observar cómo se comportaron las respuestas individuales a las preguntas del SUS. En la siguiente gráfica, se muestran los puntajes promedio para cada una de las 10 preguntas del SUS. El puntaje máximo posible por pregunta es de 5. Es importante notar que, en las preguntas impares, un puntaje más alto indica una mejor percepción del sistema, mientras que en las preguntas pares, un puntaje más bajo es indicativo de una mejor percepción.

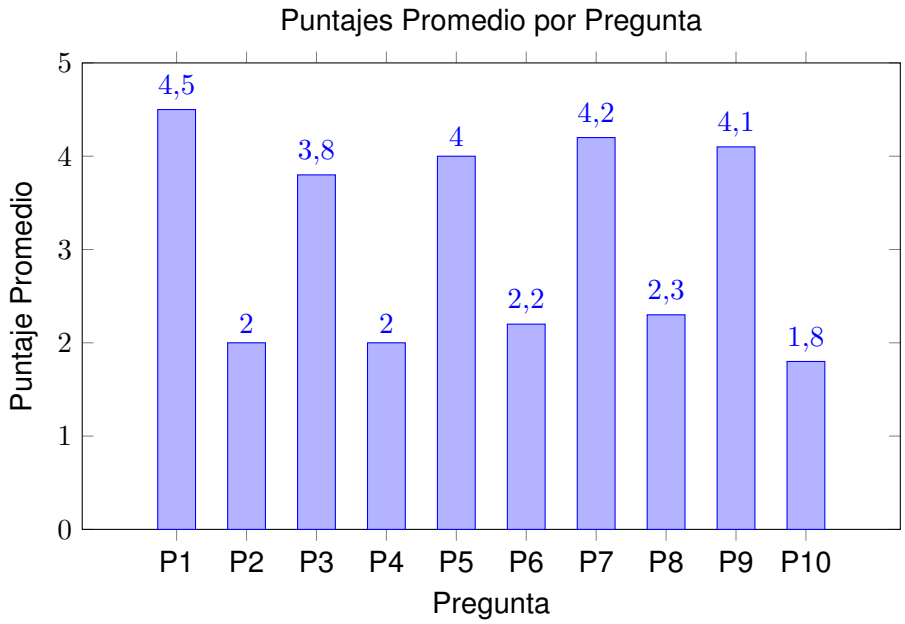


Figura 3.2: Gráfica de puntajes promedio por pregunta del SUS.

3.3.3. Explicación de los resultados por pregunta

Pregunta 1: “Creo que me gustaría usar este sistema con frecuencia.”

Análisis: La puntuación promedio de 4.5 indica que la mayoría de los usuarios estarían dispuestos a utilizar el sistema regularmente. Esto refleja una alta satisfacción y aceptación del sistema en términos de frecuencia de uso.

Pregunta 2: “Encontré el sistema innecesariamente complejo.”

Análisis: La puntuación promedio de 2.0 sugiere que algunos usuarios encontraron el sistema algo complejo, aunque no de manera abrumadora. Esta percepción de complejidad podría estar relacionada con funciones específicas que pueden necesitar simplificación.

Pregunta 3: “Pensé que el sistema era fácil de usar.”

Análisis: Con un promedio de 3.8, esta pregunta indica que la mayoría de los usuarios consideran que el sistema es bastante fácil de usar. Esto es un indicador positivo de que la interfaz de usuario es intuitiva y accesible.

Pregunta 4: “Creo que necesitaría la ayuda de una persona técnica para usar este sistema.”

Análisis: La puntuación promedio de 2.0 refleja que algunos usuarios pueden necesitar asistencia técnica, indicando una posible área de mejora en la documentación o en la simplicidad de algunas funciones.

Pregunta 5: “Encontré que las diversas funciones en este sistema estaban bien integradas.”

Análisis: Con un puntaje de 4.0, los usuarios en general encontraron que las funciones del sistema estaban bien integradas. Esto sugiere que las diferentes partes del sistema funcionan bien juntas, proporcionando una experiencia de usuario coherente.

Pregunta 6: “Pensé que había demasiada inconsistencia en este sistema.”

Análisis: El promedio de 2.2 sugiere que hay cierta percepción de inconsistencia, aunque no es un problema grave. Trabajar en la consistencia de la interfaz y las funciones podría mejorar la percepción general del sistema.

Pregunta 7: “Imagino que la mayoría de las personas aprenderían a usar este sistema muy rápidamente.”

Análisis: La puntuación de 4.2 indica que los usuarios creen que el sistema es fácil de aprender. Esto es un buen indicador de una curva de aprendizaje baja, lo cual es favorable para la adopción del sistema por nuevos usuarios.

Pregunta 8: “Encontré el sistema muy complicado de usar.”

Análisis: Con un promedio de 2.3, esta pregunta refleja que algunos usuarios encuentran el sistema algo complicado. Aunque no es una preocupación mayoritaria, es importante identificar qué aspectos específicos son percibidos como complicados.

Pregunta 9: “Me sentí muy seguro al usar el sistema.”

Análisis: La puntuación de 4.1 sugiere que la mayoría de los usuarios se sienten seguros usando el sistema, lo cual es crucial para la confianza y la aceptación del sistema.

Pregunta 10: “Necesité aprender muchas cosas antes de poder comenzar a usar el sistema.”

Análisis: La puntuación promedio de 1.8 indica que los usuarios perciben que se requiere un esfuerzo significativo para aprender a utilizar el sistema inicialmente, lo que sugiere que podría beneficiarse de una simplificación o una mejora en la capacitación y la documentación.

3.3.4. Análisis de los comentarios adicionales

Se recopilaron comentarios adicionales de los usuarios para obtener una visión más cualitativa de sus experiencias. A continuación, se presentan algunos de los comentarios más destacados y su análisis:

- **Comentario 1:** “Mejorar la explicación de los tópicos seleccionados.”

  Análisis: Este comentario sugiere que los usuarios encuentran que la información proporcionada sobre ciertos temas no es suficientemente clara. Esto puede indicar una necesidad de mejorar la documentación y la formación para los usuarios.

- **Comentario 2:** “El sistema es fácil de utilizar e íntegro.”

  Análisis: Este comentario positivo refuerza los puntajes altos en las preguntas relacionadas con la facilidad de uso y la integración de funciones, confirmando que una parte significativa de los usuarios tiene una experiencia positiva con el sistema.

- **Comentario 3:** “Me pareció algo complicado de usar. No me quedó claro cómo realizar ciertas acciones.”

  Análisis: Este comentario resalta áreas específicas donde la usabilidad del sistema podría mejorarse. La claridad en las instrucciones y la simplicidad en la ejecución de acciones son aspectos a considerar para futuras mejoras.

- **Comentario 4:** “Me pareció que debe tener una explicación previa más detallada para usuarios nuevos.”

  Análisis: La necesidad de una mejor introducción y formación para nuevos usuarios es un tema recurrente. Esto sugiere que el sistema podría beneficiarse de tutoriales interactivos o guías paso a paso para facilitar la incorporación de nuevos usuarios.

4. CONCLUSIONES Y RECOMENDACIONES

4.1. Conclusiones

En este capítulo se presentan las conclusiones derivadas del análisis de los resultados de la encuesta de usabilidad SUS y de los comentarios adicionales proporcionados por los usuarios del sistema CENTINELA. A través de este estudio, se ha obtenido una visión clara de la usabilidad del sistema y de las áreas que requieren mejoras.

- *Integración de Sistemas:* Se logró unificar las plataformas ResNet y ReSearchDecide en una única plataforma denominada CENTINELA. En este proceso, se integró el módulo de ReSearchDecide, cambiando las tecnologías de React y Firebase a Angular y Django Rest Framework. Esta transición permitió una mejor interoperabilidad y coherencia de datos. La integración general se realizó utilizando una arquitectura hexagonal, lo que facilitó la interoperabilidad entre los diferentes módulos y garantizó la compatibilidad de las plataformas.

- *Desarrollo del Módulo de Perfil y Autenticación:* Se implementó exitosamente la gestión de perfiles y autenticación utilizando JWT. Esto permite a los usuarios crear y gestionar sus perfiles, incluyendo la posibilidad de agregar una imagen de perfil e información personal. Además, se añadieron funciones similares a las de una red social, como la publicación de posts y la creación de grupos de consenso, lo que permite a los usuarios personalizar su experiencia.

- *Usabilidad General:* El sistema CENTINELA ha obtenido una puntuación promedio de 80.2 en la escala SUS, lo cual indica una alta usabilidad. Esto refleja que la mayoría de los usuarios consideran que el sistema es fácil de usar y estarían dispuestos a utilizarlo con frecuencia, lo que indica una buena aceptación del sistema.

- *Facilidad de Uso y Experiencia de Usuario:* Los usuarios encontraron el sistema visualmente atractivo y consistente, lo que proporciona una experiencia de uso uniforme. La implementación de una interfaz cohesiva y bien diseñada ayuda a los usuarios a navegar y utilizar las funcionalidades de manera eficiente. La puntuación de usabilidad y los comentarios positivos refuerzan la percepción de una experiencia de uso satisfactoria.

- *Complejidad y Consistencia:* Aunque la plataforma fue generalmente bien recibida, algunos usuarios percibieron el sistema como innecesariamente complejo y notaron ciertas inconsistencias en su funcionamiento. Esto sugiere áreas de mejora, especialmente en la simplificación de ciertas funciones y la unificación de la experiencia de usuario.
- *Seguridad y Confianza:* La implementación de medidas de seguridad, como la autenticación JWT, ha proporcionado a los usuarios una sensación de seguridad al usar el sistema. La alta puntuación en seguridad indica que los usuarios confían en la protección de sus datos y en la integridad del sistema.
- *Aprendizaje Inicial:* Una preocupación identificada es la dificultad que perciben los nuevos usuarios al aprender a usar el sistema por primera vez, con una puntuación promedio baja en esta categoría. Esto sugiere la necesidad de mejorar la documentación y ofrecer más recursos de formación para facilitar la adopción del sistema por parte de los nuevos usuarios.

4.2. Recomendaciones

Como equipo de desarrollo, recomendamos mejorar la Documentación del sistema para ofrecer una guía clara y accesible sobre las funciones más complejas. Esto incluye la creación de manuales de usuario detallados, guías paso a paso y tutoriales en video, que facilitarán la comprensión y el uso del sistema.

Además, consideramos importante identificar y simplificar las funciones del sistema que resultan complicadas para los usuarios. Esto puede implicar rediseñar la interfaz de usuario para eliminar pasos innecesarios en los procesos más utilizados, con el objetivo de mejorar la experiencia del usuario.

Por otra parte, es crucial asegurar la consistencia y coherencia de la interfaz y las interacciones dentro del sistema. De esta manera, los elementos serán predecibles y familiares para los usuarios, reduciendo su carga cognitiva y facilitando su uso.

Asimismo, sugerimos ofrecer capacitación continua y recursos de soporte a los usuarios. Esto podría incluir sesiones de formación regulares, soporte técnico fácil de acceder y la disponibilidad de material de referencia actualizado, para garantizar que los usuarios

puedan aprovechar al máximo las funcionalidades del sistema.

Finalmente, es esencial realizar evaluaciones de usabilidad de forma regular para identificar áreas de mejora y evaluar el impacto de los cambios implementados. Para esto, recomendamos utilizar encuestas de satisfacción del usuario (como el SUS) y otros métodos de evaluación cualitativa y cuantitativa, recopilando así *feedback* constante de los usuarios.

5. REFERENCIAS BIBLIOGRÁFICAS

[1] J. N. A. Túqueres, *Desarrollo de sistema basado en minería de datos para la búsqueda y visualización de redes de investigadores con filiación en instituciones ecuatorianas y sus áreas académicas*, Quito, jun. de 2023. dirección: http://bibdigital.epn.edu.ec/handle/15000/24749.

[2] D. E. A. Checa, *Desarrollo de un sistema de toma de decisiones y consenso grupal aplicado a los motores de recomendación: desarrollo del módulo de manejo de usuarios, grupos y notificaciones para el sistema de toma de decisiones y consenso grupal aplicado a los motores de recomendación*, Quito, ago de 2023. dirección: http://bibdigital.epn.edu.ec/handle/15000/25087.

[3] R. F. P. Molina, *Desarrollo de un sistema de toma de decisiones y consenso grupal aplicado a los motores de recomendación: desarrollo de los módulos de presentación de la recomendación y de selección de tópicos del sistema de recomendación para grupos de investigadores que soporte la toma de decisiones y el consenso*, Quito, oct. de 2023. dirección: http://bibdigital.epn.edu.ec/handle/15000/25084.

[4] C. Fresno Chávez, M. D. Consuegra Llapur et al., «Características de las Redes Académicas. Estado del arte,» *Revista Cubana de Informática Médica*, vol. 12, n.º 1, págs. 132-150, 2020.

[5] J. Ávila-Toscano, L. Vargas-Delgado, K. Oquendo-González et al., «Producción científica educativa, redes de autores y enfoques temáticos: Caso Universidad del Atlántico: Educational scientific production, author networks and leading topics: Case Universidad del Atlántico,» *Educación y Humanismo*, vol. 22, n.º 39, págs. 1-17, 2020.

[6] M. Torres-Herrera, G. Cuaya-Simbro, C. Canales-Castillo et al., «Sistemas recomendadores como herramienta en la labor docente: una revisión sistemática,» *Pädi Boletín Científico de Ciencias Básicas e Ingenierías del ICBI*, 2024.

[7] J. Livermore, «Factors that significantly impact the implementation of an agile software development methodology,» *Journal of Software*, vol. 3, n.º 4, págs. 31-36, 2008. DOI: 10.4304/jsw.3.4.31-36.

[8] D. B. Hernández, J. Y. G. Duarte, V. B. G. Curbelo, M. P. Martínez y J. L. A. Mora, «Influencia de la utilización de las redes sociales en el proceso de comunicación

interpersonal,» *Revista Científica Cultura, Comunicación y Desarrollo*, vol. 6, n.º 3, págs. 6-13, 2021.

[9] A. Torres Hidalgo y R. Ramos Rayski, «Dinamica de grupo,» 2017.

[10] J. M. Salinas Ibáñez, V. I. Marín Juarros et al., «Metasíntesis cualitativa sobre colaboración científica e identidad digital académica en redes sociales,» *RIED. Revista Iberoamericana De Educación a Distancia*, 2019.

[11] A. Cockburn, «Hexagonal Architecture,» 2005. dirección: https://alistair.cockburn.us/hexagonal-architecture/.

[12] T. M. Young, «More Testable Code with Hexagonal Architecture,» 2023. dirección: https://ted.dev/articles/2023/02/21/more-testable-code-with-hexagonal-architecture-talk/.

[13] Microsoft, *Qué es Git?* es, https://learn.microsoft.com/es-es/devops/develop/git/what-is-git, Accedido: 2023-12-27, abr. de 2023.

[14] S. S. Adam Wathan, *Tailwindcss*, en, https://tailwindcss.com/, Accessed: 2024-5-26.

[15] R. L. Dahl, *Node*, en, https://nodejs.org/en/about, Accessed: 2024-5-26.

[16] M. Hevery, *Angular*, en, https://angular.io/, Accessed: 2024-5-26.

[17] Microsoft, *JavaScript with syntax for types*, en, https://www.typescriptlang.org/, Accessed: 2024-5-26.

[18] D. Gandy, *Font awesome*, en, https://fontawesome.com/, Accessed: 2024-5-26.

[19] P. G. D. Group, *What Is PostgreSQL?* https://www.postgresql.org/docs/current/intro-whatis.html, Accessed: 2024-06-03, 2024.

[20] Encode OSS Ltd., *Django REST Framework*, 2023. dirección: https://www.django-rest-framework.org/.

[21] Docker, Inc., *Docker: Empowering App Development for Developers*, 2023. dirección: https://www.docker.com/.

[22] Figma, Inc., *Figma: The collaborative interface design tool*, 2023. dirección: https://www.figma.com/.

[23] E. H. Uribe y L. E. V. Ayala, «Del manifiesto ágil sus valores y principios,» *Scientia et technica*, vol. 13, n.º 34, págs. 381-386, 2007.

[24] K. Schwaber y J. Sutherland, «La guía de Scrum,» *Scrumguides. Org*, vol. 1, pág. 21, 2013.

[25] M. Jones, J. Bradley y N. Sakimura, «JSON Web Token (JWT) based client authentication in transport layer security (TLS),» *arXiv preprint arXiv:1903.02895*, 2021. dirección: https://arxiv.org/abs/1903.02895.

[26] L. Wroblewski, *Mobile First*. A Book Apart, 2011.

[27] Interaction Design Foundation, *System Usability Scale (SUS)*, n.d. dirección: https://www.interaction-design.org/literature/article/system-usability-scale?srsltid=AfmBOoq7OeBd7Mu_jGjum2cBHV8NXu7mkZUK6jzgJmctVvkks1GFDWez.

A. ANEXOS

Anexo 1: Repositorio de los prototipos del proyecto en *Figma*

Enlace del repositorio:

https://www.figma.com/design/Lx72YXAZXVXA9RCIpIyqNI/Wireframe_Centinela_01?node-id=0-1&t=TXG38sb2Ex63prpX-1

Anexo 2: Repositorio del proyecto desarrollado en *Angular*

Enlace del repositorio:

https://github.com/PlataformaIntegradaInvestigadores/frontend-app.git

Anexo 3: Repositorio del proyecto desarrollado en *Django*

Enlace del repositorio:

https://github.com/PlataformaIntegradaInvestigadores/social_consensus_backend.git

Anexo 4: Resultados evaluación SUS

Enlace del repositorio:

https://docs.google.com/spreadsheets/d/e/2PACX-1vTf6-sbeQYy4pEgr4oZhuinIi4tL5-7mdVc77nbnxxrpKp8Mgl3Bl_IbwBA53fLjm2hX6lZrpzkVaKp/pubhtml

Anexo 5: Modelo de BDD relacional

Enlace del repositorio:

https://drive.google.com/file/d/1wAjqRYOoiIHJFUuhoiBBwPwpIxnacQcj/view?usp=sharing