TRABAJO DE GRADO
Opción Investigación o Proyecto de Grado

Desarrollo de un prototipo de software para la gestión de proyectos de investigación mediante la integración de inteligencia artificial

Corporación Universitaria Remington
Facultad de Ingeniería
Programa de Ingeniería de Sistemas

Autores:
Jhon Bairo Hernández Gómez
Jorge Mario Ramírez Barrera

Tutor:
Alex David Morales Acosta

Proyecto de grado
Sincelejo, 2026

Dedicatoria

A mis padres, Heydi Luz Gómez Ortega y German Jose Hernández Nisperuza, y a mi abuela, por ser los pilares fundamentales en mi crianza y por su apoyo incondicional desde mis primeros años escolares hasta mi formación profesional. A mi abuelo, Rafael Segundo Badel Marchan, cuyo respaldo constante y generosidad hicieron posible contar con los recursos necesarios para culminar este camino. A mis hermanos, Jose German Hernández Gómez, por su guía fundamental en los procesos administrativos y de crédito, y a María Jose Hernández Gómez, por su cariño y compañía constante. Todo este esfuerzo es para ustedes.

– Jhon Bairo Hernández Gómez

Primeramente, a Dios. porque todo lo que soy se lo debo a el, el es digno de toda gloria y honra, a mis padres, Enrique Leon Ramirez jaramillo y Iscela Victoria Barrera Bohorquez por su amor incondicional y su dedicación inquebrantable y su apoyo, en cada paso en este camino nunca se apartó, a mi hermosa esposa Keren Elvira Caraballo Cardenas, por su perseverancia en mantenerse siempre motivado y llenarme de fuerzas cuando las mías flaqueaban, a mi hijo que es la razón de mi esfuerzo y el motor, a ellos gracias, a ellos son la razón de hacer esto posible. Y finalmente a mí mismo, fuiste perseverante y resiliente aunque todo fuera mal conseguiste tus objetivos. Gracias a todos ustedes este logro se los dedico a ustedes, a Dios sea la Gloria

– Jorge Mario Ramírez Barrera

Agradecimientos

En primer lugar, queremos expresar nuestro más sincero agradecimiento a la Corporación Universitaria Remington y al programa de Ingeniería de Sistemas, por brindarnos las herramientas técnicas y el espacio académico necesario para nuestra formación integral como ingenieros.

Un agradecimiento profundo y especial a nuestro tutor, Alex David Morales Acosta. Su guía constante, paciencia y orientación metodológica fueron piezas clave en el desarrollo de este proyecto. Más allá de su labor como tutor de grado, valoramos inmensamente su compromiso con nuestra formación profesional, ayudándonos a elevar nuestro nivel académico y visión de carrera. Sus consejos fueron fundamentales para llevar a cabo esta investigación con éxito.

Asimismo, extendemos nuestro reconocimiento a los profesores de la facultad que compartieron sus conocimientos a lo largo de este camino. De manera personal, yo, Jhon Bairo Hernández Gómez, deseo expresar una gratitud especial a la profesora Diana Margarita Salgado Polo, ya que su presencia fue simbólica y fundamental en mi paso por la universidad, habiendo sido quien me recibió en mi primera clase y quien me acompañó hasta la última etapa, además su apoyo y calidez académica dejaron una huella imborrable en mi proceso.

De igual manera, agradezco a mi familia y amigos por su paciencia durante las largas horas de estudios y por ser mi red de apoyo emocional en los momentos de mayor exigencia.

Por mi parte, yo, Jorge Mario Ramírez Barrera, agradezco a Dios, sin su gracia y misericordia no fuese posible este logro, y un agradecimiento muy especial a nuestro tutor Alex David Morales Acosta. Su influencia tanto en conocimiento como en eje humano, que gracias a su guía, sabiduría nos encaminó a ser más excelentes, sus consejos, su amistad, su capacidad resiliente nos motivó profundamente, también extiendo mi reconocimiento a doctora Laura Patricia Paternina, una docente muy especial, su alegría, carisma, profesionalismo y su cariño nos permitió abrirnos en esta bella carrera y fue una pieza fundamental para nuestro conocimiento y calidad humana inquebrantable y muy especial fue la chispa de nuestra pasión por esta carrera

Gracias infinitamente a toda mi familia por su comprensión, ánimo y apoyo incondicional durante este proceso me permitió poder dividirme entre lo personal, familiar y académico, ya que sin ellos no podría hacerse realidad y gracias a mis seres queridos por ser el motor que me impulsó a culminar esta etapa profesional.

Finalmente, nos agradecemos mutuamente como equipo de trabajo, nuestro compromiso, dedicación en las horas de trasnocho y la excelente sinergia que logramos fueron los motores que permitieron materializar hoy este trabajo y el desarrollo tecnológico logrado.

Tabla de Contenidos

Lista de tablas

Lista de figuras

Resumen

La elaboración de proyectos de investigación en el contexto universitario colombiano, específicamente en el departamento de Sucre, enfrenta desafíos significativos debido a la falta de herramientas tecnológicas integradas que optimicen la gestión y redacción científica. El presente trabajo tiene como objetivo desarrollar un prototipo funcional de software (nivel de madurez TRL4) para la gestión del ciclo de vida de proyectos de investigación, integrando un asistente de inteligencia artificial (IA) responsable para mejorar la eficiencia operativa y la toma de decisiones. El método utilizado se basó en un enfoque mixto. Para el desarrollo tecnológico, se adaptaron los principios del marco de trabajo ágil SCRUM, organizando el proceso mediante iteraciones (Sprints) que integraron de forma concurrente el análisis, diseño, codificación y pruebas continuas. Tecnológicamente, se empleó Vue.js para el Frontend, FastAPI para el Backend, PostgreSQL como base de datos y la API de Gemini para el módulo de IA. Los resultados incluyen la arquitectura completa del sistema, el diseño de interfaces (UI/UX) y la codificación de módulos clave como la gestión de usuarios, el control documental y el asistente inteligente capaz de sugerir redacción y gestionar referencias bibliográficas. Se concluye que la integración de la IA en herramientas de gestión investigativa ofrece una solución pertinente para cerrar brechas tecnológicas, facilitando la labor de estudiantes y docentes, y promoviendo una cultura de investigación más productiva y de alta calidad en la región.

Palabras clave: Inteligencia Artificial, Gestión de Proyectos, Desarrollo de Software, Metodología de la Investigación, Automatización, Innovación Académica.

Abstract

The development of research projects within the Colombian university context, specifically in the department of Sucre, faces significant challenges due to the lack of integrated technological tools that optimize management and scientific writing. This work aims to develop a functional software Prototype (TRL4) for managing the life cycle of research projects, integrating a responsible artificial intelligence (AI) assistant to improve operational efficiency and decision-making. The methodology followed a mixed approach. For the technological development, principles of the SCRUM agile framework, were adapted, organizing the process through iterations (Sprints) that concurrently integrated analysis, design, coding, and continuous testing. Regarding technology, Vue.js was used for the Frontend, FastAPI for the Backend, PostgreSQL for the database, and the Gemini API for the AI module. The results include the complete system architecture, the user interface design (UI/UX), and the coding of key modules such as user management, document control, and an intelligent assistant capable of suggesting writing improvements and managing bibliographic references. It is concluded that integrating AI into research management tools offers a relevant solution to bridge technological gaps, facilitating the work of students and teachers while promoting a more productive and high-quality research culture in the region.

Keywords: Artificial Intelligence, Project Management, Software Development, Research Methodology, Automation, Academic Innovation.

Introducción

La investigación científica y académica constituye el motor fundamental para la generación de conocimiento y el desarrollo social, tecnológico y económico de cualquier país. En la era digital actual, las instituciones de educación superior enfrentan el desafío de modernizar sus procesos para mantener altos estándares de calidad y competitividad. En este contexto, la irrupción de las Tecnologías de la Información y las Comunicaciones (TIC), y de manera más específica, de la Inteligencia Artificial (IA), ha abierto un abanico de posibilidades sin precedentes para transformar la manera en que se gestiona, estructura y redacta el quehacer investigativo.

A pesar de estos avances tecnológicos, los investigadores, docentes y estudiantes —particularmente en contextos regionales como el departamento de Sucre en Colombia— continúan enfrentando barreras significativas en su labor diaria. La gestión del ciclo de vida de un proyecto de investigación suele ser un proceso fragmentado y complejo, caracterizado por la sobrecarga administrativa, la dificultad en la organización de la literatura y la carencia de herramientas integradas que asistan en el diseño metodológico. Esta situación evidencia una necesidad latente de contar con soluciones tecnológicas centralizadas que optimicen el esfuerzo humano y reduzcan los tiempos invertidos en tareas operativas.

Para dar respuesta a esta problemática, el presente trabajo de grado expone el diseño y desarrollo de un prototipo de plataforma web innovadora orientada a la gestión integral de proyectos de investigación, potenciada por un asistente de Inteligencia Artificial mediante la API de Gemini. Este software busca no solo facilitar la administración de tareas, fases y documentos mediante herramientas visuales, sino también proveer un acompañamiento inteligente, ético y contextualizado. De esta forma, el sistema asiste al usuario en la ideación, revisión bibliográfica y

redacción académica, promoviendo una cultura de investigación más eficiente, accesible y rigurosa.

Para una adecuada comprensión del proyecto, el presente documento se estructura de la siguiente manera: en primera instancia, el Marco Teórico expone los fundamentos conceptuales sobre la integración de la IA en la educación y la optimización de procesos investigativos. Seguidamente, el Planteamiento del problema y la Justificación detallan las barreras tecnológicas actuales en la formulación de proyectos y la pertinencia académica y social de la solución propuesta. A continuación, se enuncian los Objetivos generales y específicos que trazaron la ruta del trabajo, seguidos del Marco metodológico, el cual describe la adaptación del marco de trabajo SCRUM y las tecnologías aplicadas. Posteriormente, la sección de Resultados detalla el análisis de requerimientos, la matriz de riesgos, el diseño arquitectónico, la codificación de los módulos y la validación del sistema mediante testing. Finalmente, se exponen las Conclusiones extraídas del proceso de desarrollo, cerrando con las referencias bibliográficas y los anexos que complementan el alcance técnico del software

Marco Teórico

1.1. Introducción Desafíos en la Elaboración de Proyectos de Investigación y el Potencial de la IA

La elaboración de proyectos de investigación representa un proceso fundamental dentro del ámbito académico, siendo a su vez una tarea compleja y demandante. Actualmente, muchos estudiantes y docentes enfrentan dificultades significativas al momento de desarrollar estos documentos, debido a la falta de orientación clara, el acceso limitado a herramientas adecuadas, y las complicaciones en la búsqueda de información confiable, redacción académica y estructuración metodológica. Estas barreras convierten el proceso en una tarea tediosa, prolongada y, en muchos casos abrumadora, lo que justifica la necesidad de soluciones innovadoras.

De hecho, Uribe Rodríguez (2024) señala que "gradualmente se ha incrementado el uso Inteligencia artificial en diferentes sectores de trabajo, estudio y hogar" (p.1). La capacidad de la IA para procesar grandes volúmenes de datos, identificar patrones, automatizar tareas y ofrecer asistencia inteligente se presenta como una solución prometedora para optimizar los procesos investigativos.

En este sentido, Chen Cheng et al. (2023) concluyen que la inteligencia artificial está revolucionando la industria de la ingeniería al ofrecer "una amplia gama de posibilidades para mejorar la eficiencia, precisión y productividad de los procesos" (p. 37). Al respecto, Uribe Rodríguez (2024) destaca la importancia de "establecer indicadores clave de desempeño (KPI) que permitan medir el impacto de la IA en la eficiencia operativa y la toma de decisiones estratégicas" (p. 8), un principio aplicable a la gestión de proyectos.

1.2. La Inteligencia Artificial en la Optimización de la Gestión y Elaboración de Proyectos Investigativos

La aplicación de la IA en la gestión de Tecnologías de la Información (TI) ha demostrado ser un factor determinante para potenciar el rendimiento y la sostenibilidad organizacional, optimizando procesos, automatizando tareas repetitivas y mejorando la toma de decisiones estratégicas (Uribe Rodríguez, 2024, p. 2). Estos principios son extrapolables a la gestión de proyectos de investigación, donde la IA puede facilitar la planificación, el seguimiento de tareas, la gestión de recursos y la generación de informes, funcionalidades centrales del software a desarrollar.

Ademas Uribe Rodríguez (2024) afirma que:

> la adopción de la IA en la gestión de Tl puede transformar por completa la forma en que operan las empresas, permitiéndoles adaptarse rápidamente a los cambios del entorno empresarial y anticiparse a las necesidades del mercado. Esto se traduce en ventajas significativas en términos de innovación, crecimiento y rentabilidad (p. 4).

Considerando el vertiginoso avance tecnológico actual, es innegable que la Inteligencia Artificial se presenta como una fuerza transformadora, con un potencial significativo para impactar diversas esferas, y la educación superior no es la excepción.

La integración de la IA en la educación superior ofrece un amplio abanico de oportunidades para mejorar la enseñanza y el aprendizaje, así como para optimizar la gestión institucional (Vera, 2023, p. 18), por tanto, esta tecnología no solo promete optimizar procesos existentes, sino que también abre caminos para enfoques pedagógicos más innovadores.

Uno de los principales beneficios es la personalización del aprendizaje. La IA permite adaptar el contenido educativo y las estrategias de aprendizaje a las necesidades individuales de

nuestros estudiantes, lo que puede mejorar la eficacia del proceso formativo y aumentar la motivación y el compromiso de nuestros estudiantes (Vera, 2023, p. 19).

No obstante, a pesar de estas significativas oportunidades, la integración de la IA en la educación superior también conlleva importantes desafíos que deben ser abordados cuidadosamente como resultado de la recopilación masiva de datos y el uso de algoritmos para el análisis y la toma de decisiones.

Otro desafío importante es la ética y la privacidad en la integración de la IA en la educación superior (Vera, 2023), así mismo, existen preocupaciones sobre la brecha de acceso y equidad, ya que no todos tienen igual acceso a la tecnología necesaria para aprovechar plenamente los beneficios, además, la toma de decisiones automatizada basada en algoritmos de IA plantea cuestionamientos éticos sobre la equidad, la transparencia y la responsabilidad, por lo que es fundamental establecer marcos éticos sólidos

Asimismo, la inteligencia artificial (IA) está permitiendo una mayor precisión en el análisis predictivo y en la toma de decisiones dentro de la industria de la ingeniería, ya que posibilita el análisis de grandes volúmenes de datos para optimizar procesos y reducir costos.

En este sentido, Chen Cheng et al. (2023) señalan que la integración de la IA en la educación y formación de ingenieros es una tendencia que está “transformando la manera en que se enseña y se aprende en el ámbito de la ingeniería” (p. 34), lo cual contribuye a preparar a los futuros profesionales para el desarrollo de soluciones innovadoras, permitiendo que dicho potencial pueda aprovecharse también en el ámbito de la investigación para identificar tendencias emergentes o anticipar posibles obstáculos en el desarrollo de un proyecto

La automatización de tareas repetitivas, como la gestión de inventarios o la asignación de recursos, “permite a los empleados centrarse en tareas más estratégicas” (Uribe Rodríguez, 2024,

p. 8), lo cual, en el contexto investigativo, se traduce en la liberación de tiempo y recursos cognitivos que pueden destinarse al análisis profundo de la información, al diseño metodológico y a la interpretación crítica de resultados, fortaleciendo así los procesos de generación de conocimiento y favoreciendo una mayor calidad y rigurosidad en la investigación

1.3. Integración de Asistentes de IA: Oportunidades y Consideraciones Éticas

Según Vera (2023), las herramientas de inteligencia artificial están redefiniendo el ámbito universitario mediante innovaciones que, a través de la personalización, la retroalimentación y el análisis de datos, logran optimizar el proceso educativo y mejorar sustancialmente los resultados académicos del alumnado.

En la formulación de proyectos de investigación, el asistente de IA del software propuesto ayudará en la generación de ideas, la elaboración de borradores de secciones o la búsqueda asistida de información bibliográfica.

Tal como advierte Vera (2023), “en este contexto, uno de los principales desafíos de la integración de IA en la educación superior es la brecha digital y la desigualdad de acceso a la tecnología” (p. 18), situación que afecta directamente el proceso formativo al limitar el acceso a herramientas de apoyo investigativo; ante esta realidad, el software desarrollado adopta un enfoque web orientado a mitigar dichas barreras, ya que requiere únicamente de un navegador para su ejecución y garantiza la accesibilidad desde múltiples plataformas, incluidos los dispositivos móviles.

Otro desafío importante es la ética y la privacidad en la integración de la IA en la educación superior (Vera, 2023, p. 19), frente a este escenario el desarrollo del software propone una integración responsable de estas tecnologías mediante el establecimiento de pautas claras para su uso, el cuidado de la información tratada y el fortalecimiento de la integridad académica, de forma que la IA funcione como un apoyo al proceso formativo y a la investigación, sin sustituir las capacidades analíticas propias del estudiante y del investigador.

Según Uribe Rodríguez (2024) "la adopción efectiva de la IA en el entorno empresarial requiere un enfoque integral que combine la gobernanza de TI, la adopción de buenas prácticas, la monitorización continua y el desarrollo del talento humano" (p. 9).

Garantizar que la tecnología se convierta en una herramienta de apoyo que respete los valores de la institución requiere de una base normativa que oriente su aplicación de forma responsable, teniendo en cuenta que, “En esta línea, es esencial desarrollar marcos éticos sólidos que guíen el desarrollo y uso de la IA en la educación superior, y garantizar una toma de decisiones informada, transparente y justa” (Vera, 2023, p. 19). Por esta razón, en el software se establecen políticas y regulaciones claras para garantizar la protección de la privacidad y la ética por parte de la IA integrada, evitando que se exponga información privada y manteniendo un rigor académico a la hora de las sugerencias/ayudas ofrecidas por la IA.

Por su parte, la IA se puede entender como sistemas diseñados para interpretar datos del entorno, aprender de esa información y adaptarse de manera flexible para alcanzar metas concretas (Vera, 2023).

La capacidad de los sistemas inteligentes para transformar la experiencia educativa radica en su facultad de personalización constante, reconociendo que, “la IA puede adaptar el proceso de enseñanza-aprendizaje a las necesidades y preferencias individuales de cada estudiante ofreciendo recursos y actividades de aprendizaje adaptados a su nivel de conocimientos, estilo de aprendizaje y ritmo de progreso” (Vera, 2023, p. 20). por tanto, el diseño de este software que integra IA para la gestión de proyectos de investigación priorizará la usabilidad, la accesibilidad y el respeto a los principios éticos fundamentales.

Aunque la IA tiene el potencial de democratizar el acceso a la educación, ya que puede ofrecer oportunidades de aprendizaje en línea a un gran número de estudiantes, existe el riesgo de

que solo aquellos con acceso a la tecnología y recursos adecuados puedan beneficiarse plenamente de ella (Vera, 2023, p. 19).

Planteamiento del problema

El desarrollo de proyectos de investigación constituye una actividad fundamental en el ámbito educativo y científico, orientada a generar nuevo conocimiento y proponer soluciones a diversas problemáticas. No obstante, este proceso es frecuentemente percibido como complejo y demandante.

A nivel global, para (Representación de la UNESCO en Perú, 2016) "identificar la innovación como una actitud necesaria para el mejoramiento permanente de la calidad educativa en una institución es un pase clave" (p. 7), ya que primeramente se debe superar la fase de reconocimiento de retos para pasar a la toma de decisiones y acciones concretas, donde la planificación y desarrollo de proyectos educativos es un paso fundamental. Este organismo subraya que los proyectos en educación a menudo surgen del "diagnóstico o constatación sistemática y ordenada de la realidad" (Representación de la UNESCO en Perú, 2016, p. 23), lo que implica la necesidad de abordar una situación problemática específica que se busca resolver o mejorar.

En el contexto colombiano, la investigación universitaria enfrenta desafíos particulares que complejizan la gestión de proyectos. Mendivelso Díaz y Parra Guarnizo (2018) señalan que, si bien las universidades son el lugar tradicional para el desarrollo y gestión del conocimiento, existen "retos y dificultades que debe afrontar la universidad actual" (p. 1). entre los que se incluyen la necesidad de transformaciones en infraestructura y recursos humanos.

La productividad científica en el país enfrenta barreras significativas, entre las cuales destaca el limitado dominio del inglés en el profesorado; esta carencia no solo restringe la movilidad académica, sino que también frena la transferencia social del conocimiento. Este escenario, sumado a la escasez de recursos financieros y a la saturación de las agendas docentes,

impide una administración eficiente de las investigaciones en todos los niveles, haciendo imperativa la búsqueda de mecanismos que optimicen la gestión de proyectos (Mendivelso Díaz & Parra Guarnizo, 2018).

Esta situación se particulariza en contextos regionales como el departamento de Sucre, donde los jóvenes investigadores y profesionales se enfrentan a la necesidad de desarrollar competencias científicas para contribuir al mejoramiento de la calidad de vida y la competitividad regional (Cárdenas Díaz & Torregrosa Espinosa, 2023).

Así mismo, Cárdenas Díaz y Torregrosa Espinosa (2023) indican que "en Colombia, la investigación científica enfrenta un gran desafío, ya que en parte no se le ha otorgado la debida importancia" (p. 108), lo que se traduce en una inversión gubernamental en ciencia, tecnología e innovación que puede ser baja en comparación con otros países.

Específicamente para los investigadores en Sucre, la gestión de proyectos implica la necesidad de herramientas y metodologías que faciliten "la elección, el desarrollo y la escritura de la investigación llevada a cabo para alcanzar resultados óptimos" (Cárdenas Díaz & Torregrosa Espinosa, 2023, p. 107).

El problema central que aborda este proyecto es, por tanto, la ausencia de herramientas de software integradas y especializadas que faciliten la gestión eficiente del ciclo de vida de los proyectos de investigación, particularmente aquellas que incorporen las potencialidades de la inteligencia artificial.

La ausencia de estas herramientas se manifiesta en dificultades para la planificación, seguimiento, colaboración, búsqueda de información, redacción estructurada y gestión bibliográfica, tareas que consumen tiempo y esfuerzos significativos de estudiantes e investigadores.

La relevancia de solucionar este problema radica en la necesidad de optimizar los recursos, reducir la carga administrativa y el estrés asociado al proceso investigativo, y fomentar un entorno más productivo y eficiente para la comunidad académica en diversos niveles.

Justificación

La realización de este proyecto de desarrollo de software para la gestión de proyectos de investigación con integración de inteligencia artificial se justifica por su potencial para generar un impacto positivo en múltiples dimensiones.

Desde una perspectiva general, la necesidad de mejorar la calidad educativa a través de la innovación es un paso clave que requiere acciones planificadas, en este sentido, el documento de la Representación de la UNESCO en Perú (2016) destaca que “la planificación y desarrollo de proyectos de innovaciones educativas” se constituye como una “herramienta es fundamental para establecer y coordinar los esfuerzos necesarios para cumplir con las acciones que mantengan la actitud y procesos de innovación vigentes” (p. 6), es por esto que este proyecto se alinea directamente con dicha visión al proponer una herramienta de software que busca transformar y optimizar una faceta crucial del quehacer académico.

A nivel académico y social en el contexto colombiano, la propuesta responde a los desafíos identificados por Mendivelso Díaz y Parra Guarnizo (2018) respecto a la investigación universitaria. Al ofrecer una solución tecnológica que busca optimizar la gestión de proyectos, se contribuye indirectamente a mitigar problemas como la "sobrecarga laboral de los profesores investigadores" (Mendivelso Díaz & Parra Guarnizo, 2018, p. 6). Esto permite fomentar una cultura de investigación más eficiente.

Así mismo la herramienta puede facilitar la producción investigativa válida y mejorar la calidad de los procesos investigativos, aspectos cruciales para el desarrollo del país y la formación de talento humano altamente cualificado.

En el ámbito local, específicamente en regiones como Sucre, este proyecto adquiere una pertinencia particular. Cárdenas Díaz y Torregrosa Espinosa (2023) resaltan la importancia de invertir en ciencia, tecnología e innovación para "obtener beneficios en términos de sostenibilidad y competitividad en la región" (p. 107). Una herramienta como la propuesta puede "desarrollar capacidades y habilidades para la CTI (ciencia, tecnología e innovación)" (Cárdenas Díaz & Torregrosa Espinosa, 2023, p. 107). Esto en jóvenes profesionales e investigadores, proporciona un soporte tecnológico que facilite sus labores y contribuye a "crear una cultura estructurada que favorezca el desarrollo social y económico de la población" (Cárdenas Díaz & Torregrosa Espinosa, 2023, p. 108). Así se busca cerrar brechas y apoyar la generación de conocimiento pertinente a las necesidades locales.

Desde el punto de vista tecnológico e institucional, la integración de un asistente de inteligencia artificial representa una innovación significativa, ofreciendo funcionalidades avanzadas para la automatización de tareas repetitivas, la asistencia en la búsqueda y análisis de información, y la optimización de la toma de decisiones en la gestión de proyectos, esto no solo mejora la eficiencia operativa de los usuarios individuales, sino que también puede fortalecer la capacidad de las instituciones educativas y centros de investigación para gestionar sus actividades científicas de manera más efectiva y con mayor calidad, el proyecto al promover la eficiencia y la calidad en la investigación, se alinea indirectamente con el Objetivo de Desarrollo Sostenible 4 (Educación de Calidad), al fortalecer las capacidades para la generación y difusión de conocimiento.

Objetivos

Objetivo General

Desarrollar un prototipo de software para la gestión del ciclo de vida de proyectos de investigación, mediante la integración de un asistente de inteligencia artificial.

Objetivos Específicos

Analizar los requerimientos funcionales y no funcionales del sistema de gestión de proyectos de investigación para establecer las especificaciones técnicas y de usuario que guiarán el desarrollo del software.

Diseñar la arquitectura del software, la estructura de la base de datos y la interfaz de usuario (UI), a través de diagramas UML y prototipos de bajo nivel, para definir la estructura técnica y visual de la aplicación.

Codificar los módulos del sistema de gestión de proyectos y el componente de integración con la API de Gemini, basándose en las especificaciones y diseños previamente establecidos, para construir el prototipo funcional.

Validar la correcta funcionalidad de los componentes del software mediante la ejecución de pruebas unitarias, con el fin de asegurar que el prototipo cumple con los requerimientos definidos.

Metodología

La presente investigación adoptará un enfoque metodológico mixto, integrando estrategias cualitativas y cuantitativas para alcanzar los objetivos establecidos, dado que, de acuerdo con Hernández-Sampieri & Mendoza (2018), este tipo de diseño secuencial resulta fundamental cuando los datos recolectados en una primera fase sirven para informar, fundamentar y desarrollar la etapa subsecuente; por lo tanto, inicialmente se emplearán técnicas cualitativas para el diseño conceptual y la planificación detallada del software a fin de lograr una comprensión profunda de las necesidades y el contexto de los usuarios, mientras que en las etapas de desarrollo y validación del prototipo predominarán los métodos cuantitativos, orientados a evaluar la funcionalidad y eficacia de la herramienta.

Para la construcción de la herramienta tecnológica, se adoptó como base el marco de trabajo ágil SCRUM. debido a que, según Schwaber & Sutherland (2020), este enfoque facilita la generación de valor a través de la creación de soluciones adaptativas frente a problemas complejos; asimismo, al fundamentarse en el empirismo, optimiza la previsibilidad y controla los riesgos mediante entregas iterativas e incrementales. Reconociendo la naturaleza académica de este proyecto, no se aplicó de manera estricta al cien por ciento, sino que se adaptaron sus principios fundamentales para lograr un desarrollo iterativo e incremental que permitiera alcanzar un nivel de madurez tecnológica TRL 4 (prototipo validado en entorno de laboratorio).

A diferencia de las metodologías tradicionales en cascada, el proceso no se dividió en fases secuenciales aisladas de análisis, diseño, codificación y pruebas. En su lugar, el trabajo se estructuro a través del Product Backlog (ver Anexo 4), en el cual consiste en una lista emergente y ordenada de lo que se necesita para mejorar el producto y funciona como la única fuente del trabajo realizado (Schwaber & Sutherland, 2020, p. 11). Los requerimientos se consolidaron en este artefacto transformado en historias de usuario priorizadas, las cuales se ejecutaron mediante iteraciones cortas (Sprints).

Dentro de cada Sprint, el equipo llevó a cabo de manera concurrente y transversal las siguientes actividades:

1. Diseño conceptual y análisis de requerimientos específicos. Se fundamento en una rigurosa revisión documental de literatura científica y herramientas existentes para identificar mejores prácticas en la gestión de proyectos de investigación y en la aplicación de inteligencia artificial (IA) en contextos académicos. Como explican Hernández-Sampieri y Mendoza (2028) conocer los estudios previos resulta indispensable para actuar con mayor claridad e identificar con precisión los requerimientos funcionales y no funcionales
2. Diseño de arquitectura e interfaz (UI/UX). Con base a los requerimientos del Sprint, se estructuro la lógica del software, seleccionando las tecnologías más apropiadas y elaborando bocetos y prototipos de interfaz buscando la intuitividad y eficiencia. Asimismo, se planifico la integración del módulo de inteligencia artificial (IA), especificando como la API de Gemini optimizara los procesos. El acceso a este modelo mediante una clave API resulta altamente ventajoso porque permite conectarse al modelo ya listo en la nube, sin necesidad de instalarlo o configurarlo localmente (Fuel Pozo. et al., 2025.), asegurando una optima accesibilidad de los recursos
3. Codificación e Integración. Implementación técnica integrando las tecnologías seleccionadas para la solución (FastAPI, Vue.js, PostgreSQL y la API de Gemini). El desarrollo se guio para ofrecer asistencia en la búsqueda de fuentes bibliográficas y en la estructuración de documentos, asegurando una iteración efectiva con el modelo. Para lograr la mejora continua del proceso, al final de cada iteración fue fundamental para evaluar las interacciones, herramientas y procesos (Schwaber & Sutherland, 2020).
4. Pruebas unitarias y de validación**.** Adoptando los principios del *Agile Testing*, el cual constituye una actividad colaborativa que ocurre continuamente, desde el nacimiento de un producto de software hasta su despliegue y operación (Garzón Guerrero, 2022,

> p. 5), se ejecutaron pruebas unitarias. Estas evaluaciones, definidas como porciones de código diseñados para comprobar que una funcionalidad de una aplicación esté funcionando como se espera, sirvieron para verificar individualmente los componentes. También se realizaron pruebas de integración para evaluar la interacción conjunta entre módulos y la conexión con la API de Gemini, la cual destaca por su notable capacidad de comprensión y generación de texto. Adicionalmente, se aplicaron pruebas funcionales y de regresión, indispensables para asegurar que los cambios no hayan alterado ni borrado las funcionalidades existentes (Garzón Guerrero, 2022, p. 14).

A lo largo de estas fases, la recolección de información se apoyará en técnicas de revisión documental empleando instrumentos como fichas de análisis y matrices de comparación en la fase de diseño, debido a que la evaluación de documentos físicos o electrónicos constituye una fuente invaluable de datos cualitativos para comprender el contexto y el fenómeno central del estudio de manera no obstructiva (Hernández-Sampieri & Mendoza, 2018). De esta forma, se garantizará una base teórica y práctica sólida para el posterior desarrollo de las funcionalidades del sistema.

Durante el desarrollo y las pruebas se emplearán como instrumentos principales las actas de reuniones, los registros de casos de prueba y los artefactos propios de Scrum, ya que, respecto a estos últimos, Schwaber y Sutherland (2020) señalan que "están diseñados para maximizar la transparencia de la información clave. Por lo tanto, todas las personas que los inspeccionan tienen la misma base de adaptación" (p. 10).

El análisis de los requerimientos será cualitativo, mientras que los resultados de las pruebas se analizarán tanto cuantitativa como cualitativamente para evaluar la funcionalidad y la eficacia de la interacción con la IA, de manera que todos los hallazgos se documentarán y presentarán en el informe final del proyecto utilizando descripciones narrativas combinadas con representaciones visuales, ya que, según advierten Hernández-Sampieri y Mendoza (2018), "las tablas y figuras tendrán que enriquecer el texto; en lugar de duplicarlo, comunican los hechos esenciales, son fáciles de leer y comprender, a la vez que son coherentes" (p. 582).

Se tienen previstos posibles desafíos, como dificultades técnicas en la integración con la API de Gemini o el riesgo de respuestas imprecisas; consecuentemente, se mantendrá una revisión constante de la documentación oficial, se implementarán estrategias de manejo de errores y se contemplará la configuración de directrices contextuales claras, puesto que complementar a Gemini con bases de datos o instrucciones estructuradas permite superar de manera efectiva problemas como la desactualización del conocimiento y las alucinaciones algorítmicas (Fuel Pozo, et al., 2025).

La subestimación del esfuerzo en los Sprints se abordará mediante la adaptación continua y el refinamiento del Product Backlog, un proceso comprendido como "el acto de dividir y definir aún más los elementos del Product Backlog en elementos más pequeños y precisos" (Schwaber & Sutherland, 2020, p. 11). Por tanto, ante eventuales limitaciones de tiempo derivadas de esta complejidad, se priorizarán las funcionalidades esenciales para garantizar que el prototipo demuestre su viabilidad y cumpla con los objetivos críticos del proyecto

Con esta estructura metodológica busca garantizar un proceso de investigación sistemático y adaptable, conducente a la entrega de un prototipo validado y de calidad.

Resultados

1.4. Análisis de los requerimientos del software

Con el propósito de asegurar el correcto funcionamiento se definieron las especificaciones técnicas y operativas del sistema las cuales se detallan en la Figura 1 a través de un desglose de los requerimientos funcionales identificados que incluye su clasificación por nivel de prioridad junto con los datos de entrada requeridos y las reglas de negocio que rigen el comportamiento esperado del software.

REQUISITOS FUNCIONALES						
Código	**Nombre del Requisito**	**Descripción**	**Prioridad**	**Entradas**	**Procesamiento/Regla de Negocio**	**Salida/resultado esperado**
RF01	Registro de usuario 	El sistema debe permitir que nuevos usuarios se registren mediante un formulario en línea	Alta	Datos personales (nombre, e-mail, contraseña, etc.)	Validar que el e-mail no exista, aplicar reglas de seguridad en la contraseña	El usuario registrado y almacenamiento en la base de datos
RF02	Inicio de sesión 	El sistema debe permitir que los usuarios registrados inicie sesión	Alta	E-mail y contraseña	Verificar credenciales con la base de datos	Acceso al panel de usuario
RF03	Visualización de proyectos 	El sistema debe mostrar los proyectos más recientes y relevantes	Media	Solicitud de usuario	Consulta la base de datos y mostrar proyectos en tarjetas	Lista de proyectos mostrada en pantalla
RF04	Historial de bibliografías 	El sistema debe permitir visualizar el historial de	Media	Usuario logueado	Consultar base de datos del historial asociado al usuario	Mostrar lista de bibliografías con autores, títulos, fecha de publicación, año, etc.
RF05	Consulta de precios (Pricing) 	Visualizar planes y costos de uso de la plataforma	Media	Clic en "Pricing"	Consulta planes en la BD	Lista de precios visible
RF06	Formulario de contacto	Envía consulta o mensajes al soporte	Media	Nombre, email, mensaje, etc.	Confirmación de envío exitoso	Confirmación de envío exitoso
RF07	Registro de nuevos proyectos 	Permite registrar un nuevo proyecto con título, tipo de investigación, nombre de	Alta	Título, tipo, institución, semillero	Validar y almacenar en la BD	Proyecto registrado correctamente

Figura 1. Especificación de los Requerimientos Funcionales y No Funcionales del Sistema

1.5. Limitaciones y Riesgos de los Requerimientos Funcionales (RF) y Requerimientos No Funcionales (RNF)

La identificación proactiva de posibles obstáculos técnicos y operativos es fundamental para garantizar la estabilidad y seguridad del software desarrollado.

Con el fin de anticipar escenarios adversos y establecer estrategias de mitigación, se detallan en la Tabla 1 las limitaciones y riesgos asociados a los requerimientos funcionales y no funcionales, clasificándolos por categorías e identificando el impacto potencial que cada uno representa para la integridad del sistema y la experiencia del usuario.

Tabla 1. Limitaciones y riesgos de los requerimientos

ID Requerimiento(s)	Categoría	Limitación Identificada	Riesgo Asociado	Impacto / Consecuencia
RF01, RF02, RF18, RF26, RF27	Autenticación y Seguridad	Dependencia de protocolos de cifrado y gestión de sesiones robustos. Complejidad en la lógica de recuperación de credenciales.	Implementación deficiente de la seguridad en el registro e inicio de sesión.	Brechas de seguridad, acceso no autorizado, robo de identidad o bloqueo de usuarios legítimos.
RF03, RF04, RF05, RF15, RF28	Gestión de Proyectos	Capacidad de almacenamiento del servidor y límites en el tamaño/formato de archivos cargados.	Inconsistencia en la base de datos al manejar múltiples versiones de un mismo proyecto.	Corrupción de archivos, pérdida de progreso del usuario y degradación del rendimiento del sistema.

RF06, RF10, RF11	Búsqueda y Filtrado	Eficiencia de los algoritmos de búsqueda sobre grandes volúmenes de datos académicos.	Resultados de búsqueda irrelevantes o tiempos de respuesta excesivos.	Frustración del usuario, ineficiencia en la investigación y abandono de la plataforma.
RF07, RF16, RF17	IA y Automatización	Dependencia de APIs externas y límites en el procesamiento de lenguaje natural (NLP).	Sugerencias de IA inexactas o alucinaciones del modelo en contextos académicos.	Desinformación, pérdida de credibilidad de la herramienta y errores en la bibliografía.
RF08, RF12, RF13, RF14	Interfaz y Exportación	Dificultad para mantener la fidelidad del formato en exportaciones (PDF/Citas) entre distintos visores.	Generación de documentos corruptos o con formato incompatible con normas (APA, MLA).	Rechazo de trabajos académicos por errores de forma y fallos en la presentación de resultados.
RNF01, RNF02	Rendimiento y Disponibilidad	Escalabilidad del hardware frente a picos de tráfico simultáneo.	Caídas del sistema durante periodos de alta demanda (entregas académicas).	Interrupción del servicio y pérdida de confianza de la comunidad universitaria.
RNF03, RNF04	Seguridad y Usabilidad	Curva de aprendizaje para usuarios no técnicos y	Interfaz poco intuitiva o vulnerabilidades ante ataques de	Baja adopción de la herramienta y exposición de

		mantenimiento de parches de seguridad.	inyección SQL/XSS.	datos sensibles de investigaciones.
RNF05	Mantenibilidad	Estructura de código acoplada que dificulte actualizaciones futuras.	Dificultad para corregir errores o añadir nuevas funcionalidades sin romper el sistema actual.	Obsolescencia tecnológica temprana y altos costos de mantenimiento preventivo.

Nota. Fuente: Elaboración Propia

1.6. Alcance del Proyecto

El alcance de este proyecto se define bajo un enfoque de desarrollo ágil, centrado en la creación de una plataforma funcional para la gestión y consulta de investigaciones académicas apoyada por Inteligencia Artificial.

Inclusiones (Lo que se entregará)

- Gestión Documental en los Módulos de carga, visualización y organización de documentos académicos por proyectos.
- Interacción con IA con un asistente capaz de generar sugerencias, resúmenes y citas en formatos estándar (APA, MLA).
- Búsqueda y Filtrado usando funciones de búsqueda básica y avanzada para la localización eficiente de recursos que provee la api de Gemini.
- Seguridad con un sistema de autenticación de usuarios y recuperación de credenciales.
- Exportación con la funcionalidad para descargar resúmenes de proyectos en formato PDF.

Exclusiones (Fuera del alcance)

- Edición colaborativa en tiempo real (tipo Google Docs).
- Integración con bases de datos externas de pago (Scopus, Web of Science) en esta fase inicial.
- Desarrollo de una aplicación móvil nativa (se prioriza el entorno Web).

¿Hasta dónde se puede llegar?

El sistema está diseñado para ser un gestor documental académico eficiente con soporte inteligente, limitado por la capacidad de procesamiento de los modelos de lenguaje actuales y el almacenamiento contratado.

¿Qué puede pasar?

El incumplimiento de estos requerimientos, especialmente los no funcionales (Seguridad y Rendimiento), transformaría una herramienta de productividad en un riesgo para la integridad de la información académica de los usuarios.

1.7. Diseño de la plataforma

Arquitectura del software

Definir la arquitectura es el paso que permite que el sistema tenga una base técnica firme y sea fácil de escalar a futuro, así que, con la idea de explicar cómo se organiza la plataforma, en este apartado se incluyen los diagramas UML que muestran tanto la estructura de la información como el modo en que trabajan sus componentes.

Diagrama de clases

El modelado de la estructura estática del sistema es fundamental para comprender las relaciones entre los datos y la lógica de negocio.

En la Figura 2 se ilustra el diagrama de clases del sistema donde se definen las entidades principales como usuarios y proyectos junto con sus atributos y métodos permitiendo visualizar la jerarquía y las relaciones de composición o asociación que estructuran la base del desarrollo.

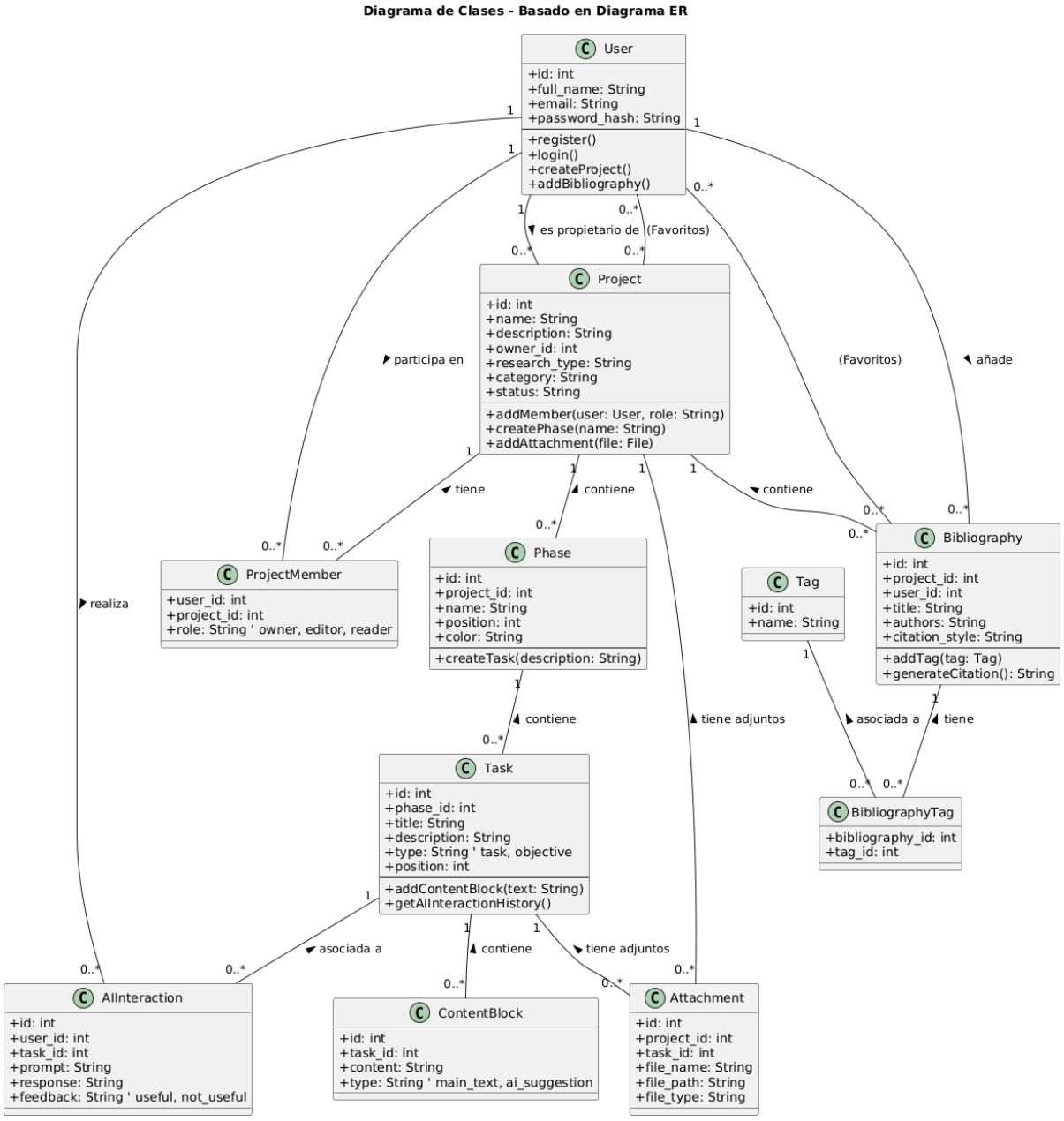


Figura 2. Diagrama de Clases

Diagrama de actividades

La dinámica operativa del sistema requiere una representación clara del flujo de trabajo que siguen los usuarios al interactuar con la plataforma.

El proceso secuencial que sigue un usuario desde el acceso a la plataforma hasta la gestión de sus investigaciones se describe en la Figura 3 la cual esquematiza el flujo de creación de un proyecto detallando las decisiones y acciones que ocurren tanto en el Frontend como en el Backend.

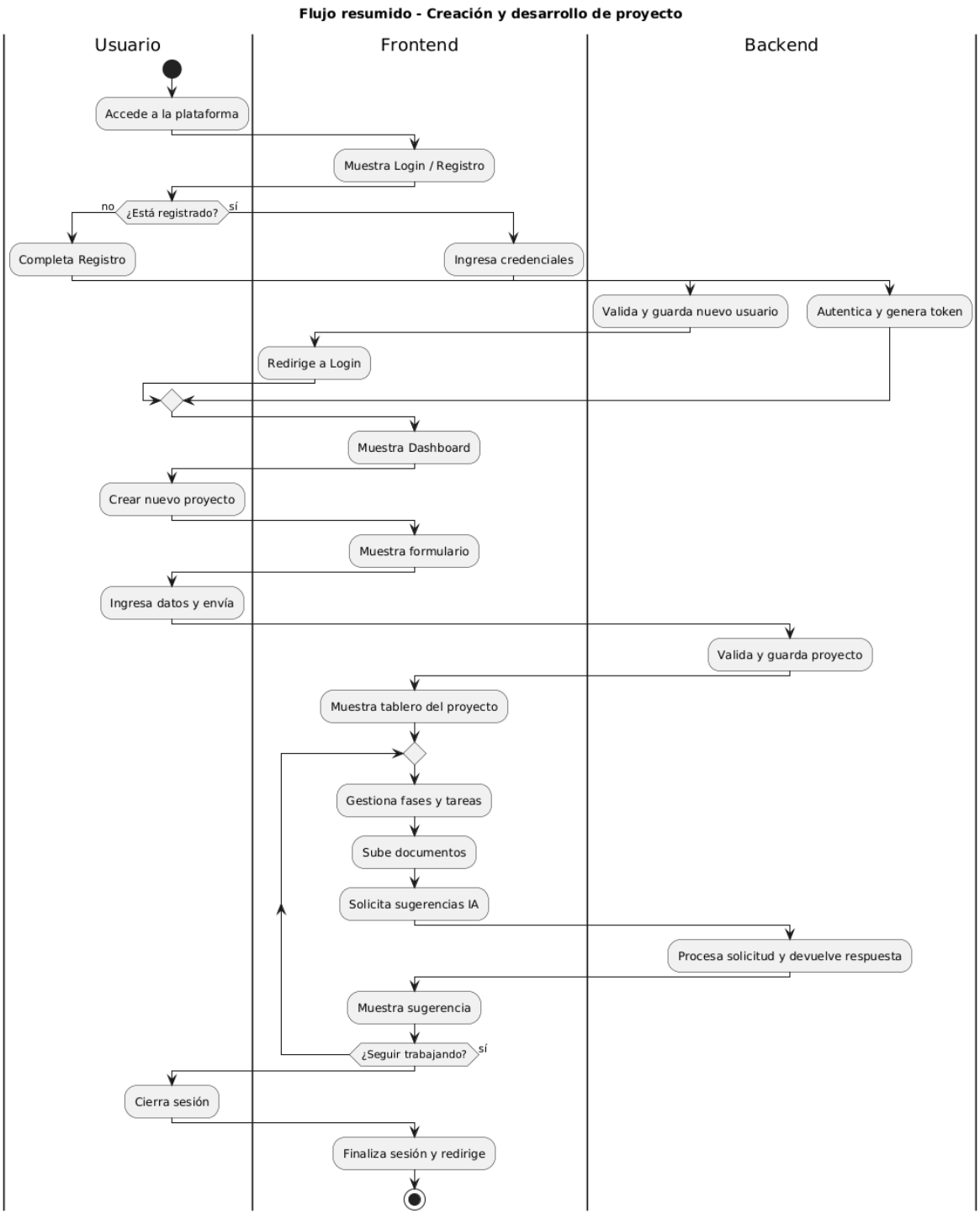


Figura 3. Flujo de Creación de un Proyecto

De igual manera la gestión documental requiere un procedimiento estructurado para garantizar la integridad de la información tal como se observa en la Figura 4 que presenta el flujo para adjuntar documentos validando formatos y metadatos antes de su almacenamiento definitivo.

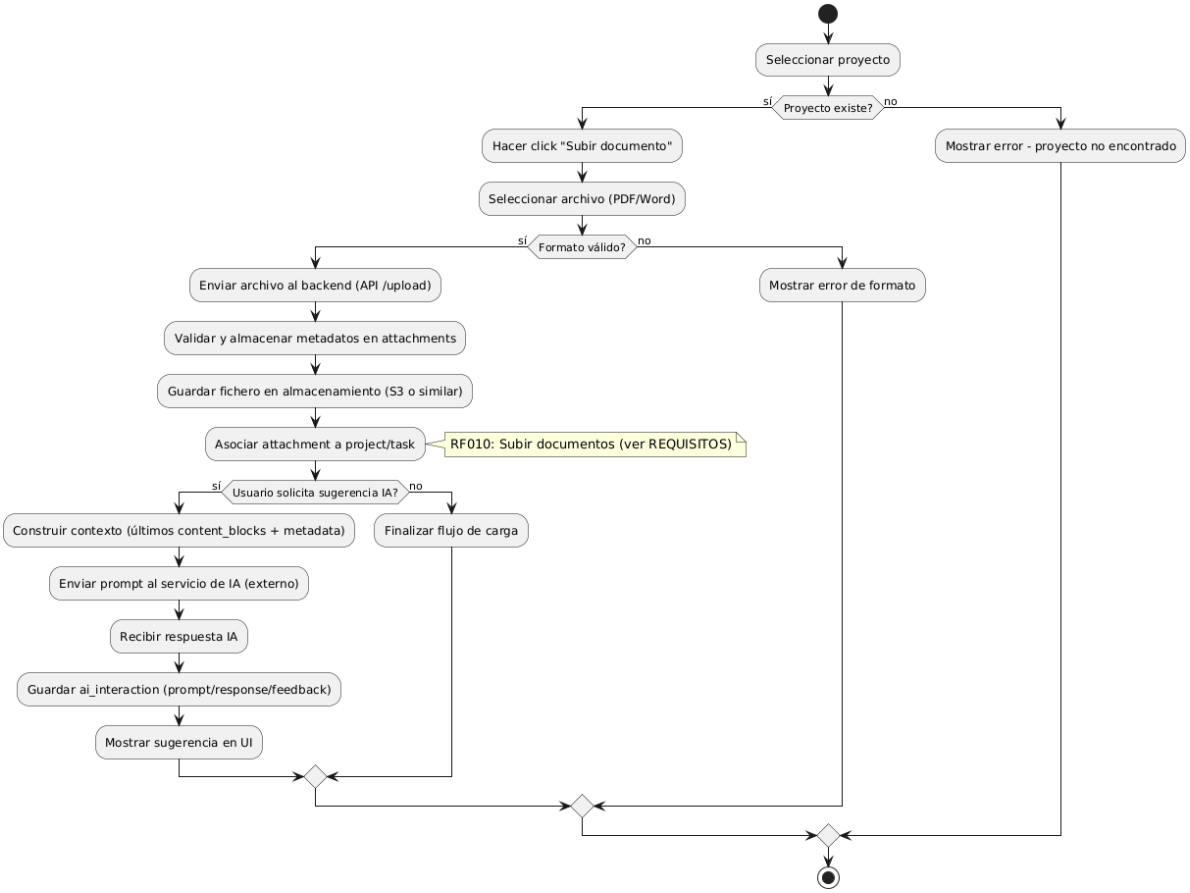


Figura 4. Flujo para Adjuntar un Documento al Proyecto

Diagramas de secuencia

Para detallar la interacción técnica entre los componentes del sistema a lo largo del tiempo se han elaborado diagramas que muestran el intercambio de mensajes.

La comunicación entre la interfaz de usuario y el servidor durante el proceso de autenticación se detalla en la Figura 5 donde se aprecia el envío de credenciales y la respuesta del sistema mediante tokens de seguridad para validar el acceso.

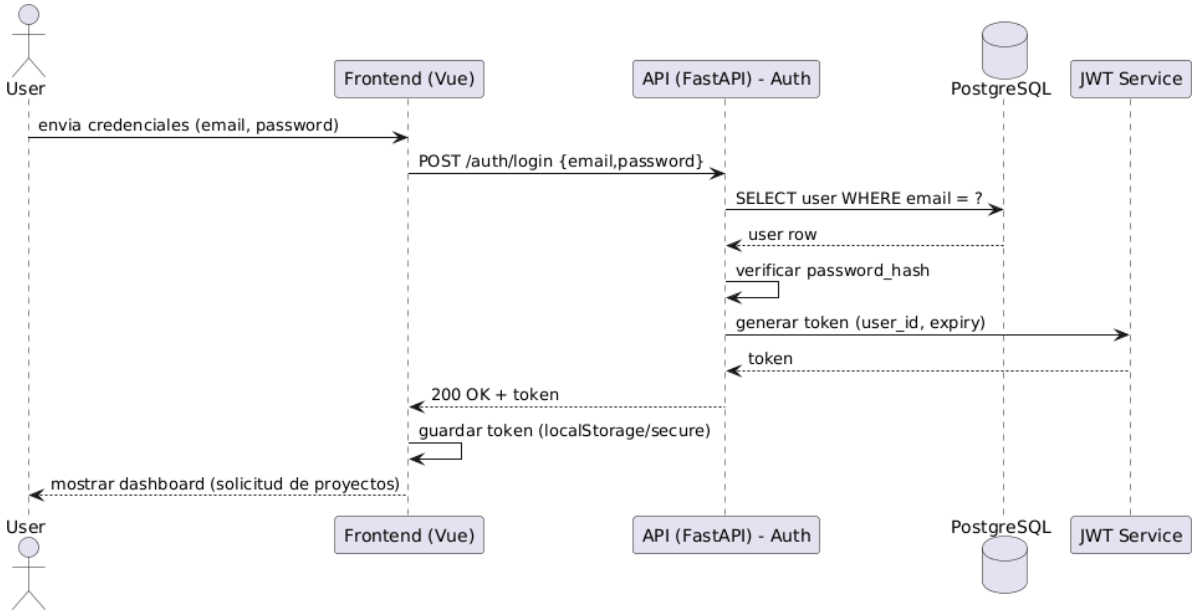


Figura 5. Secuencia para Iniciar Sesión

Una vez autenticado el usuario la interacción para dar de alta una nueva investigación implica múltiples verificaciones que se observan en la Figura 6 evidenciando cómo el Frontend solicita al API la creación del registro y su persistencia en la base de datos.

Figura 6. Secuencia para Crear un Proyecto

La carga de archivos implica una interacción más compleja que involucra almacenamiento externo tal como se representa en la Figura 7 describiendo el flujo de datos desde la selección del

archivo por parte del usuario hasta su confirmación de guardado en el sistema de almacenamiento de objetos.

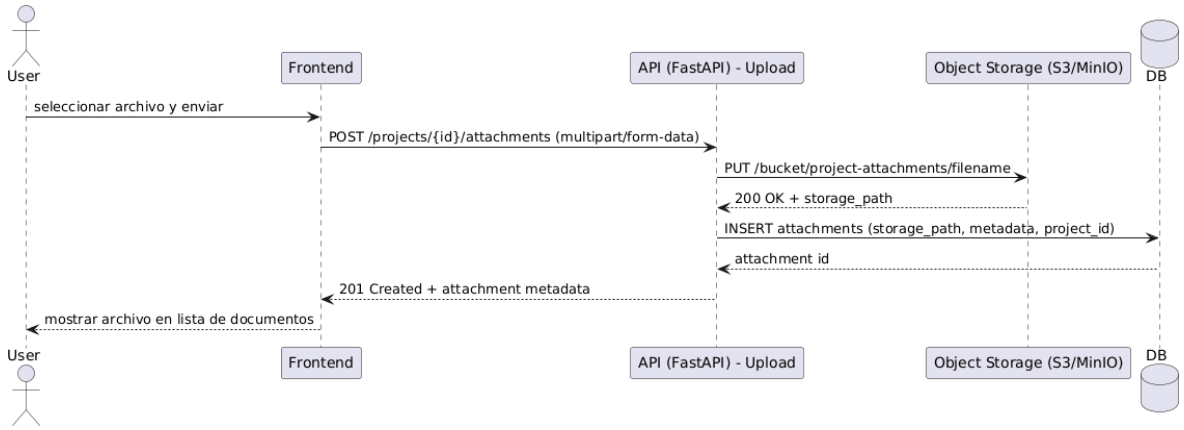


Figura 7. Secuencia para Adjuntar un Documento al Proyecto

La funcionalidad clave de inteligencia artificial requiere una comunicación asíncrona con servicios externos la cual se modela en la Figura 8 mostrando cómo el sistema procesa la solicitud de contexto del usuario y gestiona la respuesta generada por el proveedor de IA.

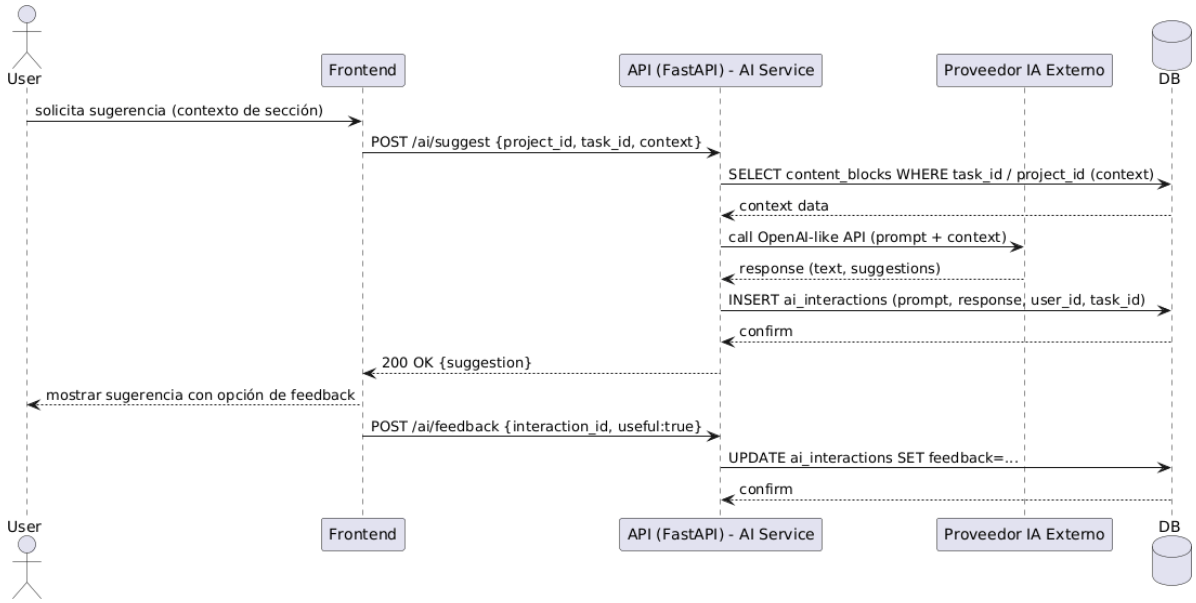


Figura 8. Secuencia de Solicitud al Asistente IA

Diagrama de paquetes

La organización lógica del código fuente es esencial para mantener la modularidad y escalabilidad del proyecto.

La estructura modular del software se organiza mediante una jerarquía de carpetas y dependencias que se visualiza en la Figura 9 separando claramente las capas de presentación lógica de negocio y acceso a datos tanto en el cliente como en el servidor.

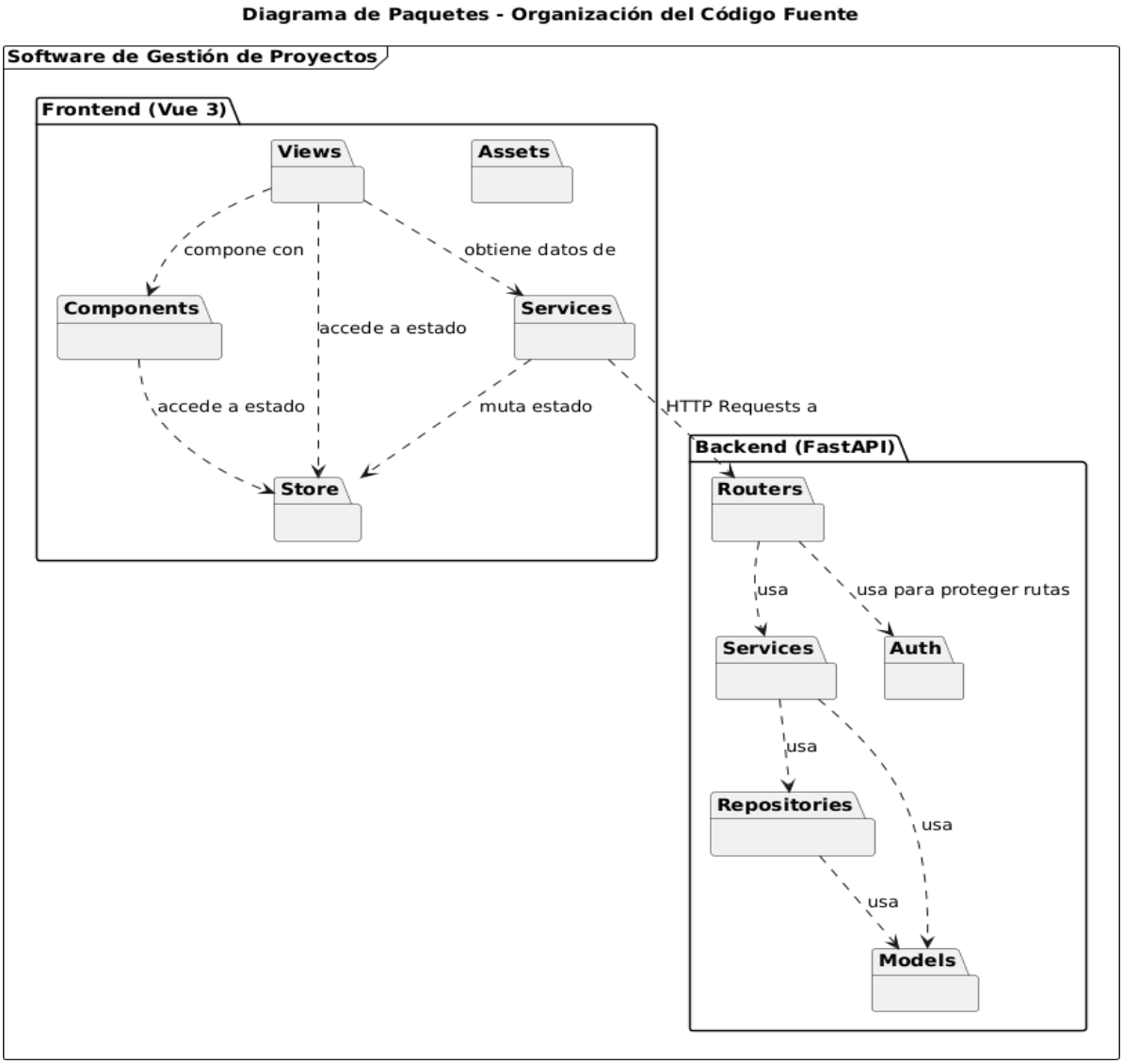


Figura 9. Estructura de Carpetas del Código Fuente

Diagrama de despliegue

La disposición física de los artefactos de software sobre la infraestructura de hardware determina la operatividad del sistema en producción.

La arquitectura física necesaria para la ejecución de la plataforma se representa en la Figura 10 donde se identifican los nodos de procesamiento como el dispositivo del usuario y los servidores en la nube que alojan la aplicación web y la base de datos interconectados a través de protocolos de internet.

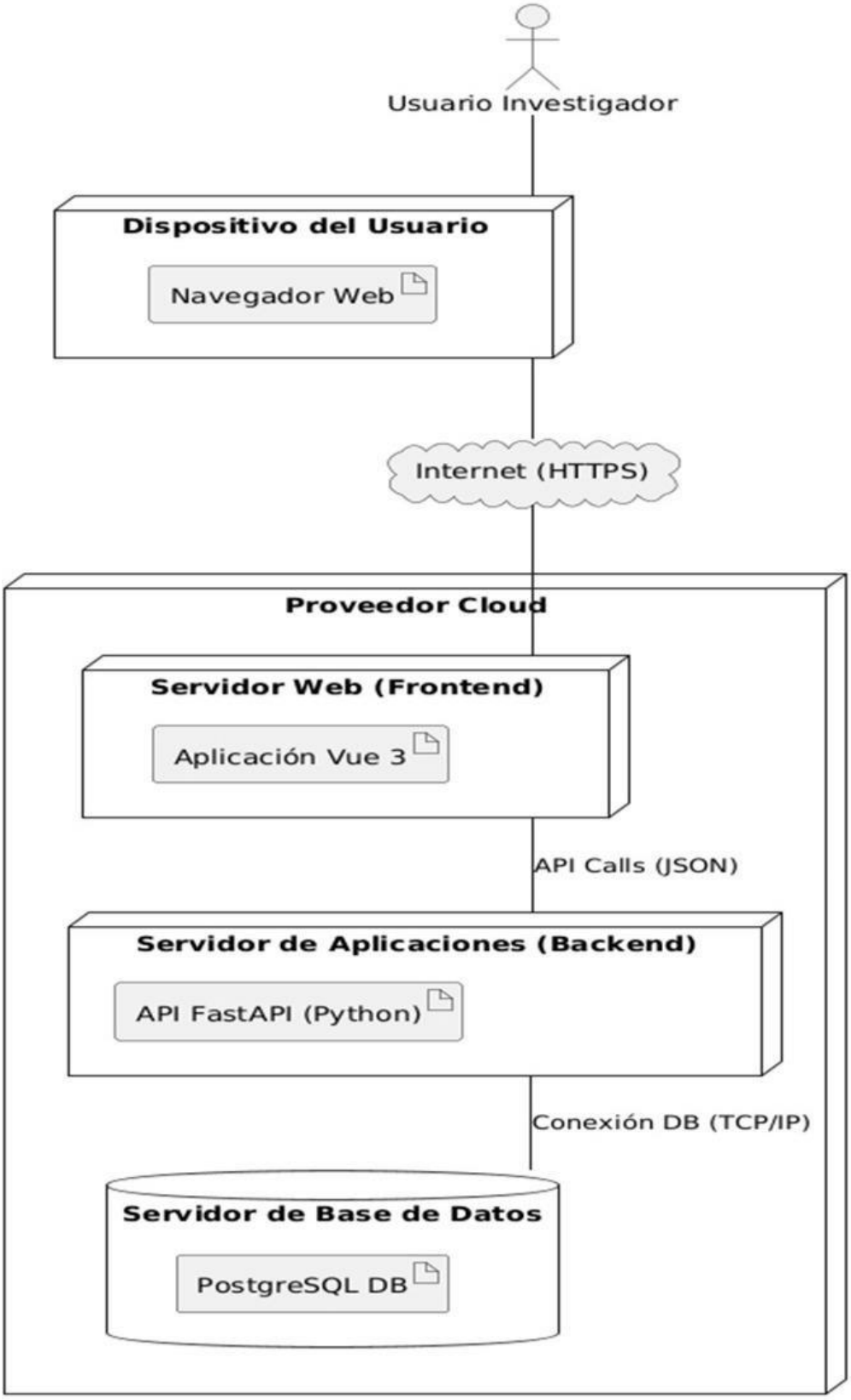


Figura 10. Diagrama de Despliegue

Estructura de la base de datos

El almacenamiento persistente de la información se sustenta en un modelo relacional robusto y normalizado.

El diseño lógico de los datos se presenta en la Figura 11 a través de un diagrama entidad-relación que expone las 14 tablas del sistema mostrando cómo se vinculan los usuarios con sus proyectos fases y tareas para garantizar la integridad referencial.

Figura 11. Diagrama Entidad/Relación de la Base de Datos

Interfaz de usuario

En la parte de abajo se exponen los distintos mockups de la vista principal con sus respectivas secciones, además del laboratorio de investigación donde se gestionan los distintos proyectos creados.

Header

La experiencia de usuario comienza con una navegación intuitiva y accesible desde la entrada a la aplicación.

El componente superior de navegación (header) que recibe al usuario se muestra en la Figura 12 destacando los elementos de bienvenida y los accesos directos a las funcionalidades principales de gestión de proyectos.

Figura 12. Header en la Vista Principal de la Plataforma

Sección IA

Es crucial comunicar al usuario las capacidades asistidas por inteligencia artificial que ofrece la herramienta.

Para informar sobre las funcionalidades inteligentes disponibles se diseñó la sección visualizada en la Figura 13 la cual utiliza tarjetas descriptivas para explicar características como la ideación inteligente y la redacción asistida.

Figura 13. Sección Informativa sobre Características de la IA de la Plataforma

Planes

La sostenibilidad del modelo de negocio se refleja en la presentación clara de las opciones de servicio.

La oferta de servicios se estructura visualmente en la Figura 14 donde se comparan los diferentes niveles de suscripción detallando los beneficios y costos asociados a cada plan para facilitar la decisión del usuario.

Figura 14. Sección de Planes Disponibles en la Plataforma

Footer

El cierre de la interfaz busca redirigir la atención hacia la acción principal de la plataforma. Como elemento de cierre y llamado a la acción se presenta en la Figura 15 el pie de página (footer) que invita al usuario a iniciar su labor investigativa facilitando el acceso inmediato a la creación de proyectos.

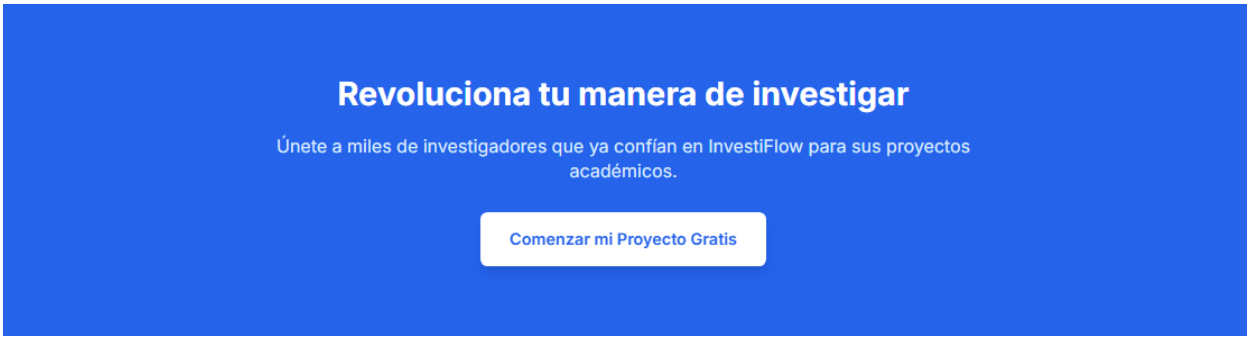


Figura 15. Footer de la Plataforma

Dashboard

El panel de control es el centro neurálgico donde el usuario administra su portafolio de investigaciones.

La vista general que permite al investigador monitorear su progreso se observa en la Figura 16 organizando la información de los proyectos activos mediante tarjetas interactivas que resumen el estado actual de cada investigación.

Figura 16. Dashboard con Todos los Proyectos del Usuario Registrado

Vista tablero Kanban de un proyecto

La gestión operativa de cada investigación se facilita mediante metodologías visuales ágiles.

Para la administración detallada del ciclo de vida del proyecto se implementó la interfaz mostrada en la Figura 17 la cual utiliza un tablero Kanban para visualizar y mover las tareas a través de sus diferentes estados de ejecución.

Figura 17. Vista de Tablero Kanban con las Fases de un Proyecto y sus Respectivas Tareas

Asistente de chat

La interacción conversacional con la IA requiere un espacio dedicado que preserve el contexto del proyecto.

El entorno de comunicación con el asistente inteligente se aprecia en la Figura 18 ofreciendo una interfaz de chat que permite al usuario realizar consultas sobre su investigación manteniendo un historial accesible de las conversaciones previas.

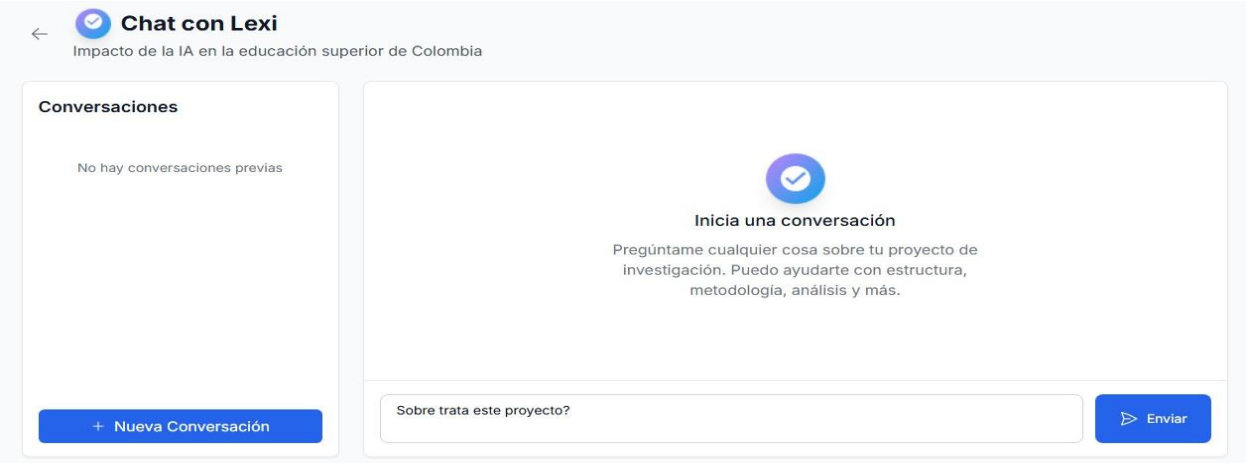


Figura 18. Vista de Asistente de Chat con Historial de Conversaciones

Editor integrado

La redacción de documentos de investigación demanda herramientas que integren la escritura con la asistencia tecnológica en tiempo real.

La herramienta principal de producción textual se presenta en la Figura 19 combinando un procesador de texto robusto con funcionalidades de IA invocables directamente sobre el lienzo de escritura para optimizar la redacción.

Figura 19. Editor Integrado de la Plataforma Potencia con IA

1.8. Codificación de los módulos del software

Módulo de gestión de usuarios

La implementación del Backend inicia con la definición de las estructuras de datos que soportan la identidad de los actores del sistema.

La representación en código de la entidad de usuarios se detalla en la Figura 20 utilizando el ORM para mapear los atributos de perfil y credenciales a la tabla correspondiente en la base de datos.

app > models > user.py > User

```python
from datetime import datetime

from sqlalchemy import Boolean, Column, DateTime, Integer, String
from sqlalchemy.orm import relationship

from app.database import Base


class User(Base):
    __tablename__ = "users"

    id = Column(Integer, primary_key=True, index=True)
    email = Column(String, unique=True, index=True, nullable=False)
    full_name = Column(String, nullable=False)
    hashed_password = Column(String, nullable=False)

    phone_number = Column(String, nullable=True)
    university = Column(String, nullable=True)
    research_group = Column(String, nullable=True)
    career = Column(String, nullable=True)

    is_active = Column(Boolean, default=True)
    is_verified = Column(Boolean, default=False)
    created_at = Column(DateTime, default=datetime.utcnow)
    updated_at = Column(DateTime, default=datetime.utcnow, onupdate=datetime.utcnow)

    projects = relationship(
        "Project", back_populates="owner", cascade="all, delete-orphan"
    )
```

Figura 20. Modelo de la Tabla “users” en Base de Datos

Endpoints

La exposición de servicios web seguros es necesaria para permitir la interacción del cliente con los datos del usuario.

La lógica de control para la verificación de identidad se implementa mediante el Endpoint mostrado en la Figura 21 el cual valida el token de sesión para retornar la información del perfil autenticado.

app > api > api_v1 > endpoints > users.py > ...

```python
from fastapi import APIRouter, Depends, HTTPException, status
from sqlalchemy.orm import Session

from app.core.dependencies import get_current_user
from app.database import get_db
from app.models.user import User
from app.schemas.user import UserResponse, UserUpdate
from app.services.user_service import user_service

router = APIRouter()


@router.get("/me", response_model=UserResponse)
def get_current_user_profile(
    current_user: User = Depends(get_current_user),
) -> UserResponse:
    """
    Obtener el perfil del usuario autenticado.

    Este endpoint requiere autenticación JWT.
    """
    return UserResponse.model_validate(current_user)


@router.get("/{user_id}", response_model=UserResponse)
def get_user(user_id: int, db: Session = Depends(get_db)) -> UserResponse:
    """Obtener información de un usuario por ID"""
    user = user_service.get_user_by_id(db=db, user_id=user_id)
    if not user:
        raise HTTPException(
            status_code=status.HTTP_404_NOT_FOUND, detail="Usuario no encontrado"
        )
    return UserResponse.model_validate(user)

```

Figura 21. Endpoint para Validar el Perfil del Usuario Autenticado

Para facilitar la integración y el mantenimiento se documentaron las rutas disponibles tal como se observa en la Figura 22 listando los métodos HTTP habilitados para la gestión de cuentas de usuario.

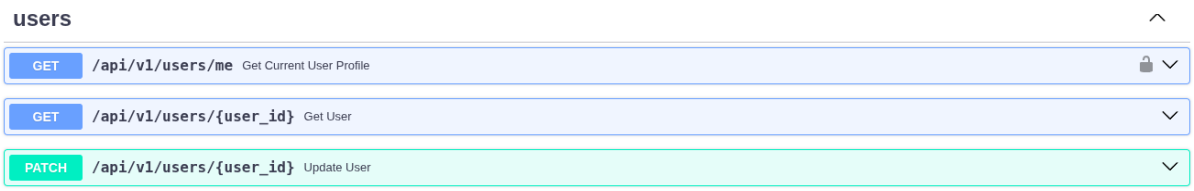


Figura 22. Documentación de Endpoints del Módulo de Usuarios

Frontend

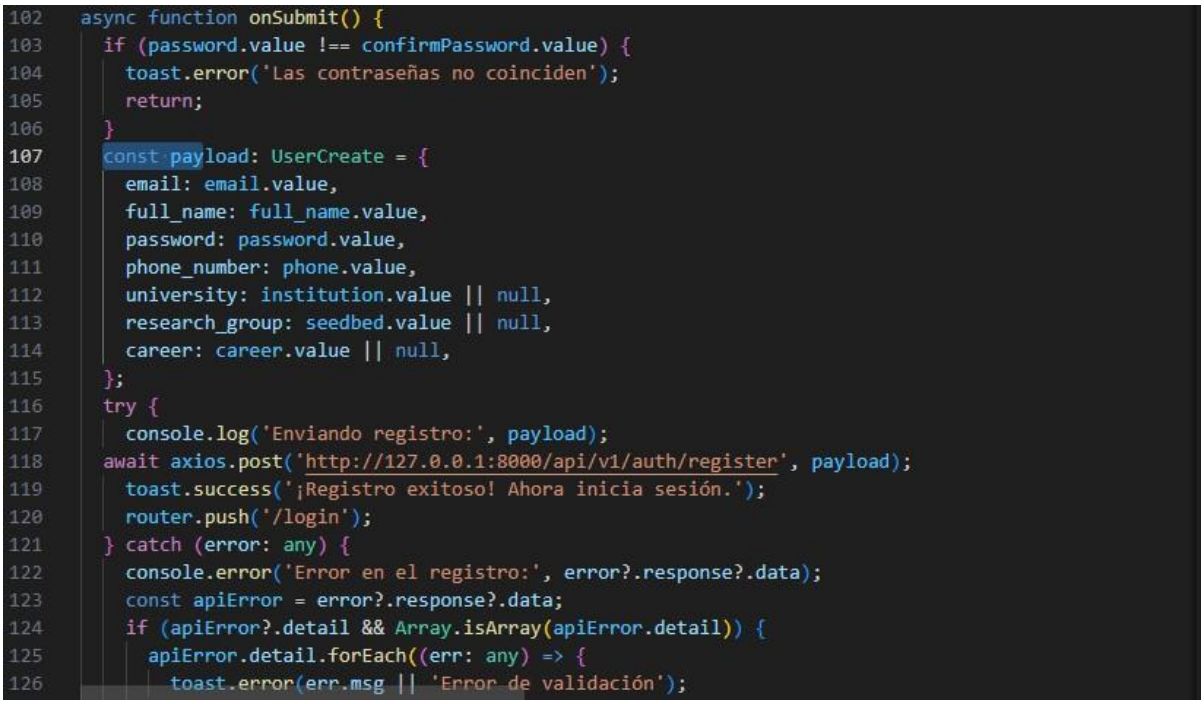
La capa de presentación consume los servicios del Backend para ofrecer una experiencia interactiva.

El manejo de la autenticación desde el lado del cliente se codifica como se muestra en la Figura 23 gestionando el envío de credenciales y el almacenamiento seguro del token de respuesta.

```
74  const onLoginSubmit = async () => {
75    try {
76      const body = new URLSearchParams();
77      body.append('username', email.value);
78      body.append('password', password.value);
79      const response = await axios.post<Token>(
80        'http://127.0.0.1:8000/api/v1/auth/login',
81        body,
82        { headers: { 'Content-Type': 'application/x-www-form-urlencoded' } }
83      );
84      localStorage.setItem('access_token', response.data.access_token);
85      localStorage.setItem('refresh_token', response.data.refresh_token);
86      toast.success('¡Login exitoso!');
87      eyeOpen.value = true; // Abrir el ojo al iniciar sesión
88      setTimeout(() => { router.push('/laboratorio'); }, 700); // Pequeña pausa para mostrar el ojo a
89    } catch (error: any) {
90      const apiError = error?.response?.data;
91      if (apiError?.detail && Array.isArray(apiError.detail)) {
92        apiError.detail.forEach((err: any) => {
93          toast.error(err.msg || 'Error de validación');
94        });
95      } else {
96        toast.error(apiError?.detail || 'Error en el login, verifica los datos ingresados.');
97      }
```

Figura 23. Método para Iniciar Sesión

De manera similar el proceso de registro de nuevos investigadores se implementa según el código de la Figura 24 que captura los datos del formulario y gestiona la comunicación asíncrona con el servicio de creación de cuentas.



```
async function onSubmit() {
  if (password.value !== confirmPassword.value) {
    toast.error('Las contraseñas no coinciden');
    return;
  }
  const payload: UserCreate = {
    email: email.value,
    full_name: full_name.value,
    password: password.value,
    phone_number: phone.value,
    university: institution.value || null,
    research_group: seedbed.value || null,
    career: career.value || null,
  };
  try {
    console.log('Enviando registro:', payload);
  await axios.post('http://127.0.0.1:8000/api/v1/auth/register', payload);
    toast.success('¡Registro exitoso! Ahora inicia sesión.');
    router.push('/login');
  } catch (error: any) {
    console.error('Error en el registro:', error?.response?.data);
    const apiError = error?.response?.data;
    if (apiError?.detail && Array.isArray(apiError.detail)) {
      apiError.detail.forEach((err: any) => {
        toast.error(err.msg || 'Error de validación');
```

Figura 24. Método para Registrar Usuario

Módulo de proyectos

El núcleo de la aplicación reside en la capacidad de modelar digitalmente un proyecto de investigación y sus propiedades.

La estructura de datos que define un proyecto de investigación se observa en la Figura 25 estableciendo los campos para títulos descripciones y relaciones con los usuarios propietarios mediante el uso de SQLAlchemy.

```python
class Project(Base):
    __tablename__ = "projects"

    id = Column(Integer, primary_key=True, index=True)
    name = Column(String(255), nullable=False, index=True)
    description = Column(Text, nullable=True)
    owner_id = Column(Integer, ForeignKey("users.id"), nullable=False, index=True)
    research_type = Column(String(50), nullable=True)
    institution = Column(String(255), nullable=True)
    research_group = Column(String(255), nullable=True)
    category = Column(String(100), nullable=True)
    status = Column(String(50), default=ProjectStatus.PLANNING.value, nullable=False)
    created_at = Column(
        DateTime(timezone=True),
        default=lambda: datetime.now(timezone.utc),
        nullable=False,
    )
    updated_at = Column(
        DateTime(timezone=True),
        default=lambda: datetime.now(timezone.utc),
        onupdate=lambda: datetime.now(timezone.utc),
        nullable=False,
    )

    # Relaciones
    owner = relationship("User", back_populates="projects")
    phases = relationship(
        "Phase", back_populates="project", cascade="all, delete-orphan"
    )
    attachment = relationship(
        "Attachment", back_populates="project", cascade="all, delete-orphan", uselist=False,
    )
```

Figura 25. Modelo para la Tabla “projects” de la Base de Datos

Endpoints

La gestión del ciclo de vida de los proyectos se centraliza en una API RESTful dedicada, lo que permite que la creación de nuevas investigaciones en el servidor se maneje a través del controlador que se muestra en la Figura 26, el cual se encarga de recibir los datos iniciales para generar el registro correspondiente en la base de datos.

```python
@router.post("/", response_model=ProjectResponse, status_code=status.HTTP_201_CREATED)
def create_project(
    *,
    db: Session = Depends(get_db),
    project_in: ProjectCreate,
    current_user: User = Depends(get_current_user),
) -> ProjectResponse:
    """
    Crear un nuevo proyecto.

    - **name**: Nombre del proyecto (requerido)
    - **description**: Descripción del proyecto (opcional)
    - **research_type**: Tipo de investigación (opcional)
    - **institution**: Institución (opcional)
    - **research_group**: Grupo de investigación (opcional)
    - **category**: Categoría del proyecto (opcional)
    - **status**: Estado del proyecto (opcional, por defecto: planning)
    """
    try:
        project = project_service.create_project(
            db=db,
            project_in=project_in,
            owner_id=current_user.id,  # type: ignore
        )
        return project
    except HTTPException:
        raise
    except Exception as e:
        raise HTTPException(
            status_code=status.HTTP_500_INTERNAL_SERVER_ERROR,
            detail=f"Error interno del servidor: {str(e)}",
        )
```

Figura 26. Creación de un Proyecto

Para permitir la asociación de material bibliográfico al proyecto se implementó la función detallada en la Figura 27 que administra la recepción y almacenamiento de archivos adjuntos.

```python
@router.post("/{project_id}/documentos", status_code=status.HTTP_201_CREATED)
async def upload_document(
    *,
    db: Session = Depends(get_db),
    project_id: int,
    file: UploadFile = File(...),
    current_user: User = Depends(get_current_user),
) -> AttachmentResponse:
    """
    Subir un nuevo documento al proyecto.

    Solo el propietario del proyecto puede subir documentos.
    """
    try:
        document = attachment_service.create_attachment(
            db=db,
            file=file,
            parent_type="project",
            parent_id=project_id,
            user_id=current_user.id,  # type: ignore
        )

        return AttachmentResponse.model_validate(document)

    except Exception:
        raise
```

Figura 27. Adjuntar Documento a un Proyecto

La recuperación de los recursos almacenados se realiza a través del servicio mostrado en la Figura 28 garantizando que los documentos adjuntos sean accesibles para su visualización o descarga.

```python
@router.get("/{project_id}/documentos")
async def get_project_document(
    *,
    db: Session = Depends(get_db),
    project_id: int,
    current_user: User = Depends(get_current_user),
) -> Optional[AttachmentResponse]:
    """
    Obtener el documento adjunto del proyecto.

    Solo el propietario del proyecto puede acceder al documento.
    Retorna None si no hay documento adjunto.
    """
    try:
        attachment = attachment_service.get_attachment_by_parent(
            db=db,
            parent_type="project",
            parent_id=project_id,
            user_id=current_user.id,  # type: ignore
        )

        if attachment:
            return AttachmentResponse.model_validate(attachment)
        return None

    except Exception:
        raise
```

Figura 28. Obtener Documento de un Proyecto

El conjunto de operaciones disponibles para la administración de investigaciones se resume en la Figura 29 presentando la lista de rutas API configuradas para el módulo de proyectos.

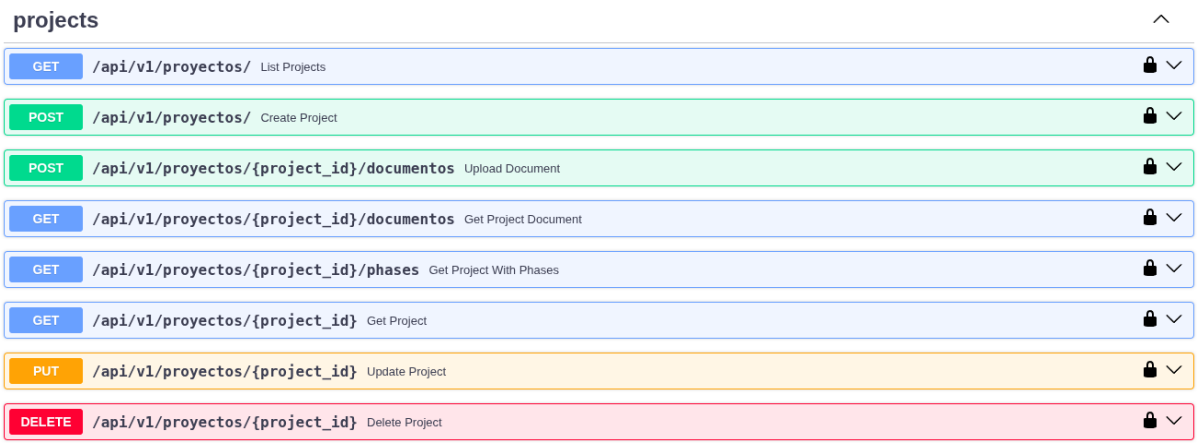
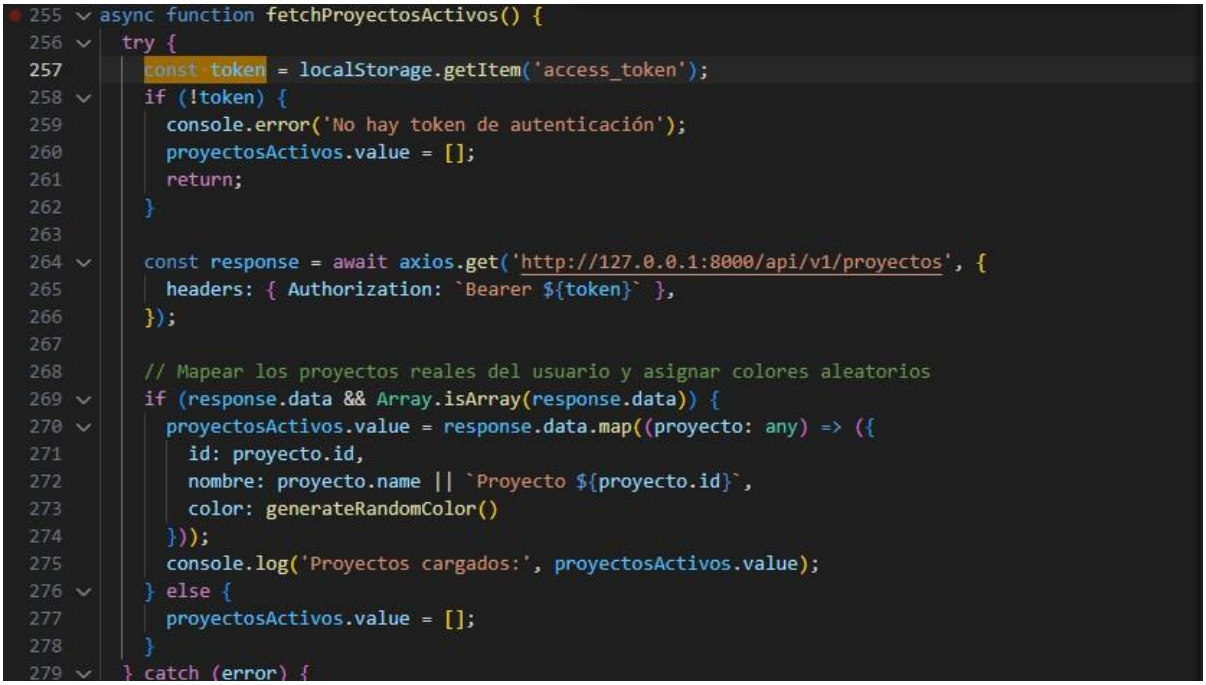


Figura 29. Documentación de los Endpoints del Módulo de Proyectos

Frontend

La interfaz de usuario requiere de una lógica específica para consumir y presentar los datos de los proyectos de forma dinámica, por lo cual la recuperación del listado de investigaciones se realiza mediante la función asíncrona detallada en la Figura 30, que se encarga de consultar el API y actualizar el estado de la aplicación con los proyectos activos.



```
async function fetchProyectosActivos() {
  try {
    const token = localStorage.getItem('access_token');
    if (!token) {
      console.error('No hay token de autenticación');
      proyectosActivos.value = [];
      return;
    }

    const response = await axios.get('http://127.0.0.1:8000/api/v1/proyectos', {
      headers: { Authorization: `Bearer ${token}` },
    });

    // Mapear los proyectos reales del usuario y asignar colores aleatorios
    if (response.data && Array.isArray(response.data)) {
      proyectosActivos.value = response.data.map((proyecto: any) => ({
        id: proyecto.id,
        nombre: proyecto.name || `Proyecto ${proyecto.id}`,
        color: generateRandomColor()
      }));
      console.log('Proyectos cargados:', proyectosActivos.value);
    } else {
      proyectosActivos.value = [];
    }
  } catch (error) {
```

Figura 30. Método para Obtener Todos los Proyectos del Usuario Autenticado

La integración de archivos en la interfaz se maneja a través de la lógica mostrada en la Figura 31 que prepara y envía los documentos seleccionados por el usuario hacia el Backend.

```
async function createProject(projectData: ProjectCreate): Promise<ProjectResponse> {
  loading.value = true
  error.value = null
  try {
    const { data } = await apiClient.post<ProjectResponse>('/proyectos/', projectData)

    // Agregar el nuevo proyecto a la lista (convertir ProjectResponse a ProjectResponse)
    const newProjectListItem: ProjectResponse = {···
    }

    projects.value.unshift(newProjectListItem)
    return data
  } catch (err: any) {
    if (err instanceof ApiValidationError) {
      const firstError = err.getFirstError()
      if (firstError) {
        const message = `${firstError.field}: ${firstError.errorMessage}`
        error.value = message || 'Error al crear proyecto'
      }
    } else {
      error.value = err.response?.data?.detail || 'Error al crear proyecto'
    }

    console.error('Failed to create project:', err)
    throw err
  } finally {
    loading.value = false
  }
}
```

Figura 31. Método para Crear un Proyecto

La integración de archivos en la interfaz se maneja a través de la lógica mostrada en la Figura 32 que prepara y envía los documentos seleccionados por el usuario hacia el Backend.

```
async function uploadProjectDocument(projectId: number, file: File): Promise<AttachmentResponse> {
  loading.value = true
  error.value = null
  try {
    const formData = new FormData()
    formData.append('file', file)

    const { data } = await apiClient.post<AttachmentResponse>(
      `/proyectos/${projectId}/documentos`,
      formData,
      {
        headers: {
          'Content-Type': 'multipart/form-data'
        }
      }
    )
    return data
  } catch (err: any) {
    error.value = err.response?.data?.detail || 'Error al subir documento'
    console.error(`Failed to upload document for project ${projectId}:`, err)
    throw err
  } finally {
    loading.value = false
  }
}
```

Figura 32. Método para Adjuntar un Documento a un Proyecto

Para visualizar los archivos asociados a una investigación la aplicación emplea el método de la Figura 33 recuperando la referencia del documento para su despliegue en el navegador.

```
async function getProjectDocument(projectId: number): Promise<AttachmentResponse | null> {
  loading.value = true
  error.value = null
  try {
    const { data } = await apiClient.get<AttachmentResponse | null>(`/proyectos/${projectId}/documentos`)
    return data
  } catch (err: any) {
    error.value = err.response?.data?.detail || 'Error al obtener documento'
    console.error(`Failed to get document for project ${projectId}:`, err)
    throw err
  } finally {
    loading.value = false
  }
}
```

Figura 33. Método para Obtener Documento Adjunto a un Proyecto

Módulo de fases

En lo que respecta al desarrollo del Backend, es necesario implementar una estructura que no solo agrupe las tareas, sino que también mantenga un registro preciso del estado en el que se

encuentran los documentos dentro de cada etapa específica del proyecto, por consiguiente, se definieron los modelos y servicios que permiten gestionar este flujo de trabajo de manera ordenada.

Modelo para tabla “phases”

La segmentación del trabajo investigativo requiere una entidad que agrupe tareas y entregables lógicamente.

La definición de las etapas que componen un proyecto se codifica en el modelo de la Figura 34 estructurando la relación entre el proyecto padre y sus subdivisiones temporales o temáticas.

```python
from app.database import Base


class Phase(Base):
    __tablename__ = "phases"

    id = Column(Integer, primary_key=True, index=True)
    name = Column(String, nullable=False)
    position = Column(Integer, nullable=False)
    color = Column(String, nullable=True)

    # Relaciones con otras tablas
    project_id = Column(Integer, ForeignKey("projects.id"), nullable=False)
    project = relationship("Project", back_populates="phases")

    tasks = relationship("Task", back_populates="phase", cascade="all, delete-orphan")

    # Relación uno a uno con Attachment para adjuntar un documento a una fase
    attachment = relationship(
        "Attachment",
        back_populates="phase",
        cascade="all, delete-orphan",
        uselist=False,
    )
```

Figura 34. Modelo de la Tabla "phases" de la Base de Datos

Endpoints

Para controlar cómo avanzan las etapas y mantener un registro del estado de los documentos en cada momento, el sistema utiliza servicios web específicos que se encargan de gestionar toda la lógica de las fases desde el Backend.

El mecanismo para añadir nuevas etapas al flujo de trabajo se implementa en el código de la Figura 35 permitiendo la definición dinámica de la estructura del proyecto.

```python
from typing import List, Optional

from fastapi import APIRouter, Depends, File, UploadFile, status
from sqlalchemy.orm import Session

from app.core.dependencies import get_current_user
from app.database import get_db
from app.models.user import User
from app.schemas.attachment import AttachmentResponse
from app.schemas.phase import PhaseCreate, PhaseListResponse, PhaseOrder, PhaseUpdate
from app.services.attachment_service import attachment_service
from app.services.phase_service import phase_service

router = APIRouter()


@router.post("/", response_model=PhaseListResponse, status_code=status.HTTP_201_CREATED)
async def create_phase(
    *,
    db: Session = Depends(get_db),
    phase_in: PhaseCreate,
    current_user: User = Depends(get_current_user),
):
    user_id = current_user.id
    return phase_service.create_phase(
        db=db,
        phase_in=phase_in,
        owner_id=user_id,  # type: ignore
    )
```

Figura 35. Endpoint para Crear Fase

La capacidad de vincular evidencia documental a una etapa específica se habilita con la función de la Figura 36 que asocia los archivos subidos directamente a la fase correspondiente.

```python
@router.post("/{phase_id}/documentos", status_code=status.HTTP_201_CREATED)
async def upload_document(
    *,
    db: Session = Depends(get_db),
    phase_id: int,
    file: UploadFile = File(...),
    current_user: User = Depends(get_current_user),
) -> AttachmentResponse:
    """
    Subir un nuevo documento a la fase.

    Solo el propietario del proyecto al que pertenece la fase puede subir documentos.
    """
    try:
        document = attachment_service.create_attachment(
            db=db,
            file=file,
            parent_type="phase",
            parent_id=phase_id,
            user_id=current_user.id,  # type: ignore
        )

        return AttachmentResponse.model_validate(document)

    except Exception:
        raise
```

Figura 36. Endpoint para Adjuntar Documento a una Fase

Para acceder a la información contenida en una etapa particular se utiliza el servicio expuesto en la Figura 37 recuperando los documentos vinculados a dicha fase.

```python
@router.get("/{phase_id}/documentos")
async def get_phase_document(
    *,
    db: Session = Depends(get_db),
    phase_id: int,
    current_user: User = Depends(get_current_user),
) -> Optional[AttachmentResponse]:
    """
    Obtener el documento adjunto de la fase.

    Solo el propietario del proyecto al que pertenece la fase puede acceder al documento.
    Retorna None si no hay documento adjunto.
    """
    try:
        attachment = attachment_service.get_attachment_by_parent(
            db=db,
            parent_type="phase",
            parent_id=phase_id,
            user_id=current_user.id,  # type: ignore
        )

        if attachment:
            return AttachmentResponse.model_validate(attachment)
        return None

    except Exception:
        raise
```

Figura 37. Endpoint para Obtener Documento Adjunto a una Fase

La especificación técnica de las operaciones soportadas para la gestión de etapas se muestra en la Figura 38 listando los endpoints disponibles para este módulo.

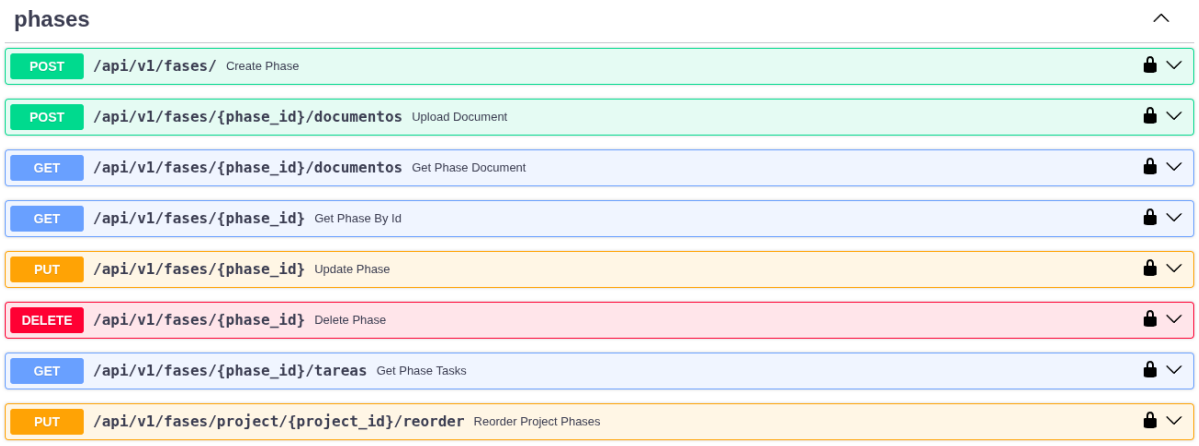


Figura 38. Documentación de Endpoints del Módulo de Fases

Frontend

La lógica de cliente para instanciar una nueva etapa en el tablero se observa en la Figura 39 gestionando la petición de creación y la actualización visual de la interfaz.

```
async function createPhase(phaseData: PhaseCreate): Promise<PhaseResponse> {
  loading.value = true
  error.value = null
  try {
    const { data } = await apiClient.post<PhaseResponse>('/fases/', phaseData)

    // Agregar la nueva fase a la lista
    phases.value.push(data)

    // Invalidar caché del proyecto
    const projectsStore = useProjectsStore()
    projectsStore.invalidateProjectCache(data.project_id)

    return data
  } catch (err: any) {
    if (err instanceof ApiValidationError) {
      const firstError = err.getFirstError()
      if (firstError) {
        const message = `${firstError.field}: ${firstError.errorMessage}`
        error.value = message || 'Error al crear fase'
      }
    } else {
      error.value = err.response?.data?.detail || 'Error al crear fase'
    }

    console.error('Failed to create phase:', err)
    throw err
  } finally {
    loading.value = false
  }
}
```

Figura 39. Método para Crear una Fase

El manejo de archivos dentro de una etapa específica se realiza mediante el código de la Figura 40 que facilita la carga de documentos contextualizados a la fase.

```
async function uploadPhaseDocument(phaseId: number, file: File): Promise<AttachmentResponse> {
  loading.value = true
  error.value = null
  try {
    const formData = new FormData()
    formData.append('file', file)

    const { data } = await apiClient.post<AttachmentResponse>(
      `/fases/${phaseId}/documentos`,
      formData,
      {
        headers: {
          'Content-Type': 'multipart/form-data'
        }
      }
    )
    return data
  } catch (err: any) {
    const apiError: ApiError = err.response?.data
    error.value = apiError.detail || 'Error al subir documento'
    console.error(`Failed to upload document for phase ${phaseId}:`, err)
    throw err
  } finally {
    loading.value = false
  }
}
```

Figura 40. Método para Adjuntar un Documento a una Fase

La recuperación de los recursos asociados a una etapa para su visualización se ejecuta con la función presentada en la Figura 41 completando el flujo de información documental.

```
async function getPhaseDocument(phaseId: number): Promise<AttachmentResponse | null> {
  loading.value = true
  error.value = null
  try {
    const { data } = await apiClient.get<AttachmentResponse | null>(`/fases/${phaseId}/documentos`)
    return data
  } catch (err: any) {
    const apiError: ApiError = err.response?.data
    error.value = apiError.detail || 'Error al obtener documento'
    console.error(`Failed to get document for phase ${phaseId}:`, err)
    throw err
  } finally {
    loading.value = false
  }
}
```

Figura 41. Método para Obtener un Documento Adjunto a una Fase

Módulo de tareas

Lograr un seguimiento preciso de la investigación requiere alcanzar el nivel más granular de gestión posible, por lo cual el sistema permite modelar las actividades específicas que deben realizarse para cumplir con los objetivos de cada etapa.

La representación de las unidades de trabajo individuales se define en el código de la Figura 42 estableciendo las propiedades de las tareas y su vinculación con las fases del proyecto

```python
class Task(Base):
    __tablename__ = "tasks"

    id = Column(Integer, primary_key=True, index=True)
    title = Column(String(255), nullable=False, index=True)
    description = Column(Text, nullable=True)
    position = Column(Integer, nullable=False)
    status = Column(SqlEnum(TaskStatus), default=TaskStatus.PENDING.value, nullable=False)
    start_date = Column(DateTime(timezone=True), nullable=True)
    end_date = Column(DateTime(timezone=True), nullable=True)
    completed = Column(Boolean, default=False, nullable=False)
    created_at = Column(
        DateTime(timezone=True),
        default=lambda: datetime.now(timezone.utc),
        nullable=False,
    )
    updated_at = Column(
        DateTime(timezone=True),
        default=lambda: datetime.now(timezone.utc),
        onupdate=lambda: datetime.now(timezone.utc),
        nullable=False,
    )

    # Foreign Keys
    phase_id = Column(Integer, ForeignKey("phases.id"), nullable=False)

    # Relaciones
    phase = relationship("Phase", back_populates="tasks")
    attachment = relationship(
        "Attachment",
        back_populates="task",
        cascade="all, delete-orphan",
        uselist=False,
    )
```

Figura 42. Modelo de la Tabla "tasks" de la Base de Datos

Endpoints

La administración y el seguimiento de las actividades del sistema requieren de endpoints específicos, por lo cual la lógica del servidor para registrar nuevas tareas se detalla en la Figura 43, donde se facilita la inserción organizada de cada actividad dentro de su fase correspondiente.

```python
@router.post("/", response_model=TaskResponse, status_code=status.HTTP_201_CREATED)
async def create_task(
    *,
    db: Session = Depends(get_db),
    task_in: TaskCreate,
    current_user: User = Depends(get_current_user),
):
    """
    Crea una nueva tarea para el usuario autenticado.

    Args:
        db (Session): Sesión de base de datos proporcionada por la dependencia.
        task_in (TaskCreate): Datos requeridos para crear una nueva tarea.
        current_user (User): El usuario actualmente autenticado.

    Returns:
        TaskResponse: Los datos de la tarea creada.
    """
    user_id = current_user.id
    return task_service.create_task(
        db=db,
        task_in=task_in,
        owner_id=user_id,  # type: ignore
    )
```

Figura 43. Endpoint de Creación de Tarea

Para enriquecer las tareas con material de soporte se dispone del servicio mostrado en la Figura 44 que gestiona la asociación de archivos a una actividad concreta.

```python
@router.post("/{task_id}/documentos", status_code=status.HTTP_201_CREATED)
async def upload_document(
    *,
    db: Session = Depends(get_db),
    task_id: int,
    file: UploadFile = File(...),
    current_user: User = Depends(get_current_user),
) -> AttachmentResponse:
    """
    Subir un nuevo documento a la tarea.

    Solo el propietario del proyecto al que pertenece la tarea puede subir documentos.
    """
    try:
        document = attachment_service.create_attachment(
            db=db,
            file=file,
            parent_type="task",
            parent_id=task_id,
            user_id=current_user.id,  # type: ignore
        )

        return AttachmentResponse.model_validate(document)

    except Exception:
        raise
```

Figura 44. Adjuntar Documento a una Tarea

La consulta de los materiales adjuntos a una actividad se realiza mediante el código de la Figura 45 permitiendo el acceso a la documentación de soporte de la tarea.

```python
@router.get("/{task_id}/documentos")
async def get_task_document(
    *,
    db: Session = Depends(get_db),
    task_id: int,
    current_user: User = Depends(get_current_user),
) -> Optional[AttachmentResponse]:
    """
    Obtener el documento adjunto de la tarea.

    Solo el propietario del proyecto al que pertenece la tarea puede acceder al documento.
    Retorna None si no hay documento adjunto.
    """
    try:
        attachment = attachment_service.get_attachment_by_parent(
            db=db,
            parent_type="task",
            parent_id=task_id,
            user_id=current_user.id,  # type: ignore
        )

        if attachment:
            return AttachmentResponse.model_validate(attachment)
        return None

    except Exception:
        raise
```

Figura 45. Obtener Documento Adjunto a una Tarea

El catálogo de servicios web disponibles para la manipulación de actividades se detalla en la Figura 46 presentando las rutas API del módulo de tareas.

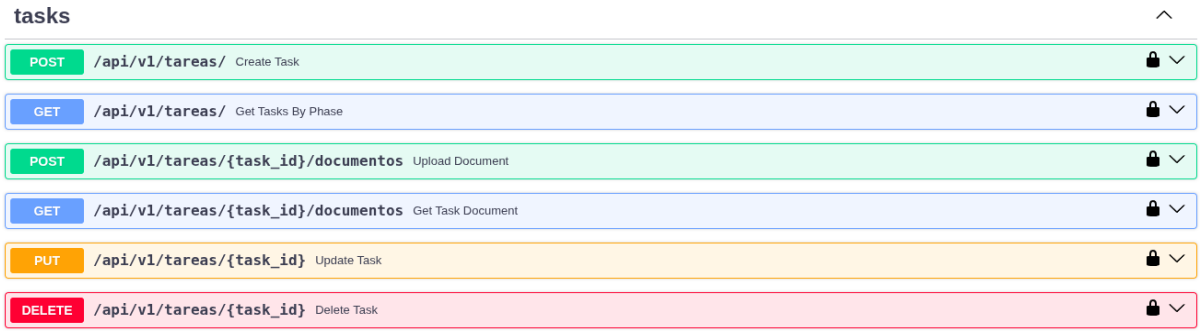


Figura 46. Documentación de los Endpoints del Módulo de Tareas

Frontend

La gestión de tareas en el cliente conecta las acciones del usuario con el servidor para que, al integrarse en las fases, cada actividad funcione como un registro del estado del documento o de las metas a cumplir en ese punto del proceso.

La implementación de la creación de actividades desde la interfaz se observa en la Figura 47 enviando los detalles de la nueva tarea al Backend para su registro.

```
async function createTask(taskData: TaskCreate): Promise<TaskResponse> {
  loading.value = true
  error.value = null
  try {
    const { data } = await apiClient.post<TaskResponse>('/tareas/', taskData)

    // Agregar la nueva tarea a la lista
    tasks.value.push(data)

    return data
  } catch (err: any) {
    if (err instanceof ApiValidationError) {
      const firstError = err.getFirstError()
      if (firstError) {
        const message = `${firstError.field}: ${firstError.errorMessage}`
        error.value = message || 'Error al crear tarea'
      }
    } else {
      error.value = err.response?.data?.detail || 'Error al crear tarea'
    }

    console.error('Failed to create task:', err)
    throw err
  } finally {
    loading.value = false
  }
}
```

Figura 47. Método para Crear una Tarea

Así mismo para permitir al usuario subir evidencias o recursos a una tarea se utiliza el código de la Figura 48 que maneja la transferencia del archivo seleccionado, el cual se adjunta a la tarea seleccionada.

```
async function uploadTaskDocument(taskId: number, file: File): Promise<AttachmentResponse> {
  loading.value = true
  error.value = null
  try {
    const formData = new FormData()
    formData.append('file', file)

    const { data } = await apiClient.post<AttachmentResponse>(
      `/tareas/${taskId}/documentos`,
      formData,
      {
        headers: {
          'Content-Type': 'multipart/form-data'
        }
      }
    )
    return data
  } catch (err: any) {
    error.value = err.response?.data?.detail || 'Error al subir documento'
    console.error(`Failed to upload document for task ${taskId}:`, err)
    throw err
  } finally {
    loading.value = false
  }
}
```

Figura 48. Método para Adjuntar un Documento a una Tarea

La visualización de los archivos vinculados a una actividad se logra mediante la función de la Figura 49 que solicita y procesa la información del documento adjunto.

```
async function getTaskDocument(taskId: number): Promise<AttachmentResponse | null> {
  loading.value = true
  error.value = null
  try {
    const { data } = await apiClient.get<AttachmentResponse | null>(`/tareas/${taskId}/documentos`)
    return data
  } catch (err: any) {
    error.value = err.response?.data?.detail || 'Error al obtener documento'
    console.error(`Failed to get document for task ${taskId}:`, err)
    throw err
  } finally {
    loading.value = false
  }
}
```

Figura 49. Método para Obtener Documento Adjunto a una Tarea

1.9. Validación y Testing

Es importante asegurar que cada funcionalidad del servidor responda de manera correcta y aislada, ya que esto es un paso esencial para la estabilidad del proyecto, por eso se procedió con la ejecución de los tests unitarios aplicados directamente sobre la REST API desarrollada en FastAPI.

Es así como, al utilizar las herramientas de la librería Pytest, se pudo comprobar de forma automática que la lógica de negocio y los puntos de acceso operan sin errores, lo que permitió mejorar la experiencia de desarrollo ya que se escribía código con la tranquilidad de tener pruebas que respalden que todos los módulos funcionen correctamente, además nos permitió obtener los reportes que se presentan a continuación para cada uno de los módulos del sistema.

Módulo de usuarios

La garantía de calidad del software inicia con la verificación automática de las funciones críticas de acceso.

Los resultados de las pruebas automatizadas para la gestión de identidad se presentan en la Figura 50 confirmando el correcto funcionamiento del registro y autenticación de usuarios.

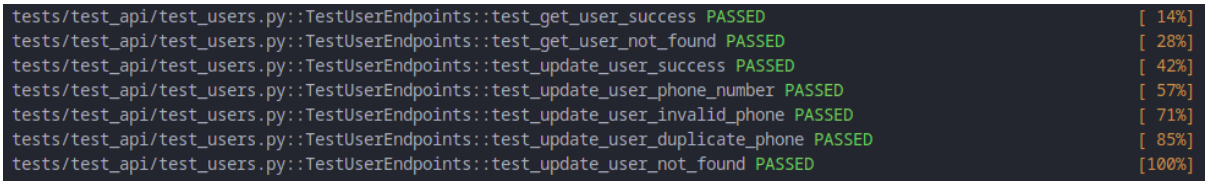


Figura 50. Resultado de Tests Unitarios del Módulo de Usuarios

Módulo de proyectos

Es fundamental asegurar que la lógica de gestión de investigaciones opere sin errores antes del despliegue.

La validación del núcleo de la aplicación se evidencia en la Figura 51 mostrando la ejecución exitosa de los casos de prueba diseñados para la creación y manipulación de proyectos.

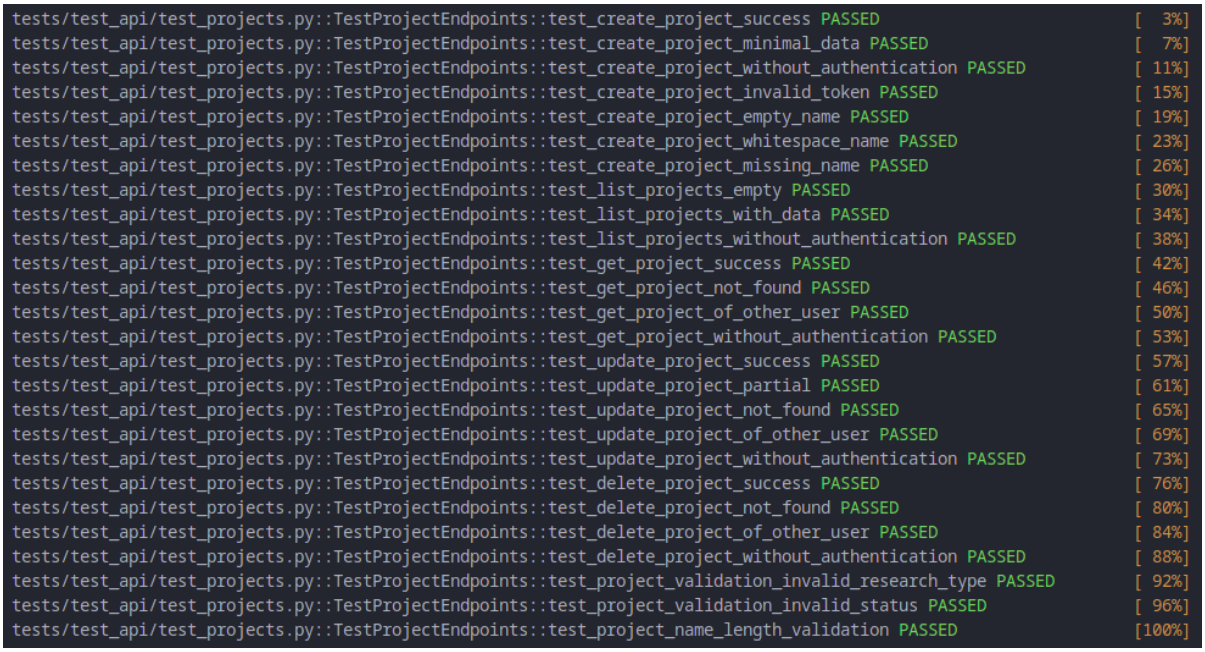


Figura 51. Resultado de Tests Unitarios del Módulo de Proyectos

Módulo de fases

La integridad del flujo de trabajo depende de la correcta validación de las etapas del proyecto.

El reporte de ejecución de pruebas para la segmentación de proyectos se observa en la Figura 52 verificando la estabilidad de las operaciones relacionadas con las fases.

```
tests/test_api/test_phases.py::TestPhaseEndpoints::test_create_phase_success PASSED                    [  3%]
tests/test_api/test_phases.py::TestPhaseEndpoints::test_create_phase_auto_position PASSED              [  7%]
tests/test_api/test_phases.py::TestPhaseEndpoints::test_create_phase_with_existing_position PASSED     [ 10%]
tests/test_api/test_phases.py::TestPhaseEndpoints::test_create_phase_without_authentication PASSED     [ 14%]
tests/test_api/test_phases.py::TestPhaseEndpoints::test_create_phase_invalid_project PASSED            [ 17%]
tests/test_api/test_phases.py::TestPhaseEndpoints::test_create_phase_project_of_other_user PASSED     [ 21%]
tests/test_api/test_phases.py::TestPhaseEndpoints::test_create_phase_invalid_name PASSED              [ 25%]
tests/test_api/test_phases.py::TestPhaseEndpoints::test_create_phase_invalid_color PASSED             [ 28%]
tests/test_api/test_phases.py::TestPhaseEndpoints::test_get_phase_by_id_success PASSED                [ 32%]
tests/test_api/test_phases.py::TestPhaseEndpoints::test_get_phase_not_found PASSED                    [ 35%]
tests/test_api/test_phases.py::TestPhaseEndpoints::test_get_phase_of_other_user PASSED                [ 39%]
tests/test_api/test_phases.py::TestPhaseEndpoints::test_update_phase_success PASSED                   [ 42%]
tests/test_api/test_phases.py::TestPhaseEndpoints::test_update_phase_position PASSED                  [ 46%]
tests/test_api/test_phases.py::TestPhaseEndpoints::test_update_phase_not_found PASSED                 [ 50%]
tests/test_api/test_phases.py::TestPhaseEndpoints::test_update_phase_of_other_user PASSED             [ 53%]
tests/test_api/test_phases.py::TestPhaseEndpoints::test_delete_phase_success PASSED                   [ 57%]
tests/test_api/test_phases.py::TestPhaseEndpoints::test_delete_phase_updates_positions PASSED         [ 60%]
tests/test_api/test_phases.py::TestPhaseEndpoints::test_delete_phase_not_found PASSED                 [ 64%]
tests/test_api/test_phases.py::TestPhaseEndpoints::test_delete_phase_of_other_user PASSED             [ 67%]
tests/test_api/test_phases.py::TestPhaseEndpoints::test_get_phase_tasks PASSED                        [ 71%]
tests/test_api/test_phases.py::TestPhaseEndpoints::test_reorder_phases_success PASSED                 [ 75%]
tests/test_api/test_phases.py::TestPhaseEndpoints::test_reorder_phases_invalid_project PASSED         [ 78%]
tests/test_api/test_phases.py::TestPhaseEndpoints::test_reorder_phases_invalid_phase PASSED           [ 82%]
tests/test_api/test_phases.py::TestPhaseEndpoints::test_reorder_phases_project_of_other_user PASSED   [ 85%]
tests/test_api/test_phases.py::TestPhaseEndpoints::test_phase_validation_name_whitespace PASSED       [ 89%]
tests/test_api/test_phases.py::TestPhaseEndpoints::test_phase_validation_negative_position PASSED     [ 92%]
tests/test_api/test_phases.py::TestPhaseEndpoints::test_phase_validation_missing_required_fields PASSED [ 96%]
tests/test_api/test_phases.py::TestPhaseEndpoints::test_phase_without_authentication_endpoints PASSED [100%]
```

Figura 52. Resultado de Tests Unitarios del Módulo de Fases

Módulo de tareas

La funcionalidad de asignación y seguimiento de actividades se somete a verificación exhaustiva.

La primera parte de los resultados de las pruebas sobre las actividades individuales se muestra en la Figura 53 validando las operaciones básicas de creación y actualización.

```
tests/test_api/test_tasks.py::TestTaskEndpoints::test_create_task_success PASSED                      [  2%]
tests/test_api/test_tasks.py::TestTaskEndpoints::test_create_task_minimal_data PASSED                 [  5%]
tests/test_api/test_tasks.py::TestTaskEndpoints::test_create_task_with_dates PASSED                   [  8%]
tests/test_api/test_tasks.py::TestTaskEndpoints::test_create_task_without_authentication PASSED       [ 10%]
tests/test_api/test_tasks.py::TestTaskEndpoints::test_create_task_invalid_phase PASSED                [ 13%]
tests/test_api/test_tasks.py::TestTaskEndpoints::test_create_task_phase_of_other_user PASSED          [ 16%]
tests/test_api/test_tasks.py::TestTaskEndpoints::test_create_task_invalid_title PASSED                [ 18%]
tests/test_api/test_tasks.py::TestTaskEndpoints::test_create_task_invalid_dates PASSED                [ 21%]
tests/test_api/test_tasks.py::TestTaskEndpoints::test_create_task_negative_position PASSED            [ 24%]
tests/test_api/test_tasks.py::TestTaskEndpoints::test_get_tasks_by_phase_success PASSED               [ 27%]
tests/test_api/test_tasks.py::TestTaskEndpoints::test_get_tasks_by_phase_empty PASSED                 [ 29%]
tests/test_api/test_tasks.py::TestTaskEndpoints::test_get_tasks_by_phase_without_authentication PASSED [ 32%]
tests/test_api/test_tasks.py::TestTaskEndpoints::test_get_tasks_by_phase_invalid_phase PASSED         [ 35%]
tests/test_api/test_tasks.py::TestTaskEndpoints::test_get_tasks_by_phase_of_other_user PASSED         [ 37%]
tests/test_api/test_tasks.py::TestTaskEndpoints::test_update_task_success PASSED                      [ 40%]
tests/test_api/test_tasks.py::TestTaskEndpoints::test_update_task_partial PASSED                      [ 43%]
tests/test_api/test_tasks.py::TestTaskEndpoints::test_update_task_dates PASSED                        [ 45%]
tests/test_api/test_tasks.py::TestTaskEndpoints::test_update_task_not_found PASSED                    [ 48%]
tests/test_api/test_tasks.py::TestTaskEndpoints::test_update_task_of_other_user PASSED                [ 51%]
```

Figura 53. Resultados Parte 1 de Tests Unitarios del Módulo de Tareas

Complementando la validación anterior la Figura 54 presenta el resto de los casos de prueba ejecutados para el módulo de tareas asegurando la robustez de la gestión de actividades.

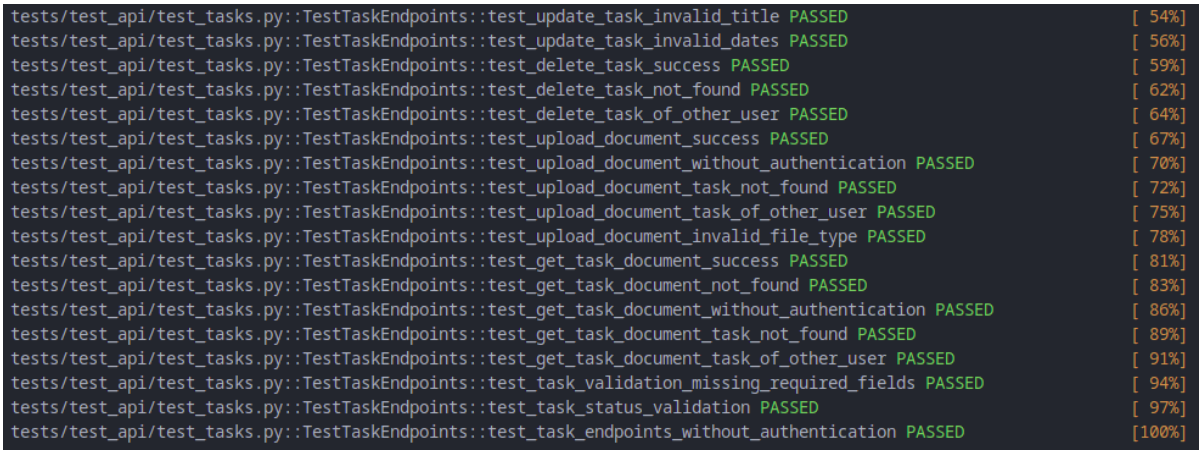


Figura 54. Resultado Parte 2 de Tests Unitarios del Módulo de Tareas

Conclusiones

El desarrollo del prototipo de software para la gestión de proyectos de investigación ha permitido validar la viabilidad técnica y funcional de integrar asistentes de inteligencia artificial en el flujo de trabajo académico, evidenciando que la adaptación de los principios del marco de trabajo ágil SCRUM resultó ser una decisión acertada para guiar el ciclo de vida del desarrollo del prototipo, ya que la estructuración del trabajo en iteraciones y la priorización de requisitos permitieron gestionar la complejidad del proyecto de manera profesional e incremental, facilitando así la adaptación ante desafíos técnicos y asegurando una entrega continua de valor que permitió focalizar los esfuerzos en los módulos críticos sin perder de vista los tiempos establecidos en el cronograma.

Asimismo, la fase de diseño arquitectónico y modelado previo fue determinante para la eficiencia de la etapa de codificación, pues al contar con una arquitectura clara basada en la separación de responsabilidades entre el Frontend y el Backend se logró una implementación modular y organizada que redujo significativamente la ambigüedad al momento de programar, permitiendo a los desarrolladores tener una visión sistémica precisa de la interacción entre la base de datos y la interfaz de usuario antes de escribir las líneas de código finales.

En cuanto a la calidad del software, la implementación rigurosa de pruebas unitarias y de integración aportó un alto grado de fiabilidad y seguridad al desarrollo puesto que la ejecución exitosa de los casos de prueba aseguró que la incorporación de nuevas funcionalidades no afectara la estabilidad de los módulos existentes, convirtiéndose en una práctica crucial para garantizar el correcto funcionamiento de procesos sensibles como la autenticación de usuarios y la persistencia de datos bajo distintas condiciones de operación.

Por otro lado, la integración de la API de Gemini demostró ser una solución potente y eficaz para dotar al sistema de capacidades cognitivas avanzadas, dado que la configuración del asistente

no se limitó a una conexión técnica, sino que incluyó el diseño de instrucciones específicas orientadas a salvaguardar la ética y el rigor académico, logrando con ello que la inteligencia artificial actúe como un soporte contextualizado y seguro en la redacción y sugerencia bibliográfica alineado con las necesidades de los investigadores de optimizar tiempos sin sacrificar la integridad científica.

Finalmente, el enfoque de desarrollo basado en la web cumplió con el objetivo transversal de accesibilidad y democratización tecnológica debido a que la eliminación de instalaciones locales complejas y la posibilidad de acceso desde cualquier dispositivo con navegador permiten abordar directamente la problemática de la falta de herramientas integradas en la región, confirmando así que la plataforma desarrollada constituye un recurso estratégico para fomentar una cultura investigativa más productiva y estructurada en el ámbito universitario.

Referencias

Cárdenas Díaz, Y., & Torregrosa Espinosa, A. (2023). Experiencias vividas por jóvenes investigadores de Sucre: una ventana al mundo de la investigación. En G. P. Anaya Pérez, *Reflexiones de Jóvenes Investigadores. Vol. 2* (Vol. 2, págs. 106-117). Sincelejo: Corporación Universitaria del Caribe – CECAR. Obtenido de https://repositorio.cecar.edu.co/handle/cecar/10025

Chen Cheng, C. C. (2023). La inteligencia artificial y su impacto en la industria de la ingeniería. *REICIT, 3*(1), 26-38. doi:https://doi.org/10.48204/reict.v3n1.3948

Fuel Pozo, M., Saltos, A. G., Llumiquinga, Y. T., Aguirre, K. L., Jara, M. C., & Mejia-Escobar, C. (6 de Octubre de 2025). *arXiv.* Obtenido de arXiv: https://arxiv.org/abs/2510.02653

Garzon Guerrero, E. A. (2022). *Creación de los procesos de pruebas de tipo unitarias y pruebas funcionales de tipo regresión automatizados dentro del ciclo de desarrollo de vida del software bajo metodología Scrum con el enfoque de Devops.* Bogotá D.C.: Universidad Distrital Francisco José de Caldas. Obtenido de http://hdl.handle.net/11349/31798

Hernández-Sampieri, R., & Mendoza, C. P. (2018). *Metodología de la investigación: las rutas: cuantitativa ,cualitativa y mixta.* Ciudad de México: Mc Graw Hill educación. Obtenido de https://hdl.handle.net/20.500.14624/1206

Mendivelso Díaz, M. E., & Parra Guarnizo, J. A. (2018). *La investigación en la universidad colombiana, retos para el futuro.* Universidad Cooperativa de Colombia, Facultad de Ciencias Sociales, Psicología, Villavicencio. Villavicencio: Universidad Cooperativa de Colombia. Obtenido de https://repository.ucc.edu.co/entities/publication/8a508fef-f2ec-4e12-90a7-8b6a142acf5a

Representación de la UNESCO en Perú. (2016). *Formulación de proyectos.* Lima: Representación de la UNESCO en Perú. Obtenido de https://unesdoc.unesco.org/ark:/48223/pf0000247006

Schwaber, K., & Sutherland, J. (2020). *La Guía Definitiva de Scrum: Las Reglas del Juego.* Scrum.org. Obtenido de https://repositorio.uvm.edu.ve/handle/123456789/59

Uribe Rodríguez, L. (2024). Integración de Inteligencia Artificial en la gestión de tecnologías de la información: un enfoque aplicado en el desarrollo empresarial. *Encuentro Internacional de Educación en Ingeniería ACOFI 2024* (págs. 1-10). Bogota, Colombia: Asociación Colombiana de Facultades de Ingeniería (ACOFI). doi:https://doi.org/10.26507/paper.3761

Vera, F. (2023). Integración de la Inteligencia Artificial en la Educación superior: Desafíos y oportunidades. *Transformar*, *4*(1), 17-31. Obtenido de https://www.revistatransformar.cl/index.php/transformar/article/view/84

Anexos

Anexo 1: Codificación de los módulos

Módulo de usuarios

Endpoint para actualizar un usuario

```python

@router.patch("/{user_id}", response_model=UserResponse)
def update_user(
    user_id: int, user_update: UserUpdate, db: Session = Depends(get_db)
) -> UserResponse:
    """Actualizar información de un usuario"""
    try:
        updated_user = user_service.update_user(
            db=db, user_id=user_id, user_update=user_update
        )
        return updated_user
    except HTTPException:
        raise
    except Exception as e:
        raise HTTPException(
            status_code=status.HTTP_500_INTERNAL_SERVER_ERROR,
            detail=f"Error interno del servidor: {str(e)}",
        )

```

Módulo de proyectos

Enums para campos de la tabla proyectos

```python
from app.database import Base


class ProjectStatus(str, Enum):
    """Estados posibles de un proyecto"""

    PLANNING = "planning"
    IN_PROGRESS = "in_progress"
    ON_HOLD = "on_hold"
    COMPLETED = "completed"
    CANCELLED = "cancelled"


class ResearchType(str, Enum):
    """Tipos de investigación"""

    BASIC = "basic"
    APPLIED = "applied"
    EXPERIMENTAL = "experimental"
    THEORETICAL = "theoretical"
    QUALITATIVE = "qualitative"
    QUANTITATIVE = "quantitative"
    MIXED = "mixed"
```

Importaciones necesarias para los endpoints del módulo de proyectos en el Backend

```python
from typing import Any, List, Optional

from fastapi import APIRouter, Depends, File, HTTPException, UploadFile, status
from sqlalchemy.orm import Session

from app.core.dependencies import get_current_user
from app.database import get_db
from app.models.user import User
from app.schemas.attachment import AttachmentResponse
from app.schemas.project import (
    ProjectCreate,
    ProjectListResponse,
    ProjectResponse,
    ProjectUpdate,
)
from app.services.attachment_service import attachment_service
from app.services.project_service import project_service

router = APIRouter()

```

Endpoints en el Backend

Obtener todos los proyectos

```python
@router.get("/", response_model=List[ProjectListResponse])
def list_projects(
    *,
    db: Session = Depends(get_db),
    current_user: User = Depends(get_current_user),
) -> Any:
    """
    Listar todos los proyectos del usuario autenticado.

    Retorna una lista con todos los proyectos creados por el usuario actual,
    ordenados por fecha de creación (más recientes primero).
    """
    try:
        projects = project_service.get_user_projects(db=db, owner_id=current_user.id)
        return projects
    except HTTPException:
        raise
    except Exception as e:
        raise HTTPException(
            status_code=status.HTTP_500_INTERNAL_SERVER_ERROR,
            detail=f"Error interno del servidor: {str(e)}",
        )
```

Obtener un proyecto por id

```python
@router.get("/{project_id}", response_model=ProjectResponse)
def get_project(
    *,
    db: Session = Depends(get_db),
    project_id: int,
    current_user: User = Depends(get_current_user),
) -> ProjectResponse:
    """
    Obtener los detalles de un proyecto específico.

    Solo el propietario del proyecto puede acceder a sus detalles.
    """
    try:
        project = project_service.get_user_project_by_id(
            db=db,
            project_id=project_id,
            owner_id=current_user.id,  # type: ignore
        )
        return project
    except HTTPException:
        raise
    except Exception as e:
        raise HTTPException(
            status_code=status.HTTP_500_INTERNAL_SERVER_ERROR,
            detail=f"Error interno del servidor: {str(e)}",
        )
```

Obtener un proyecto con sus fases

```python
@router.get("/{project_id}/phases")
async def get_project_with_phases(
    *,
    db: Session = Depends(get_db),
    project_id: int,
    current_user: User = Depends(get_current_user),
):
    try:
        return project_service.get_project_with_phases(
            db=db,
            project_id=project_id,
            owner_id=current_user.id,  # type: ignore
        )

    except HTTPException:
        raise

    except Exception:
        raise
```

Actualizar un proyecto

```python
@router.put("/{project_id}", response_model=ProjectResponse)
def update_project(
    *,
    db: Session = Depends(get_db),
    project_id: int,
    project_in: ProjectUpdate,
    current_user: User = Depends(get_current_user),
) -> ProjectResponse:
    """
    Actualizar un proyecto existente.

    Solo el propietario del proyecto puede actualizarlo.
    Todos los campos son opcionales, solo se actualizarán los campos proporcionados.
    """
    try:
        project = project_service.update_user_project(
            db=db,
            project_id=project_id,
            project_in=project_in,
            owner_id=current_user.id,  # type: ignore
        )
        return project
    except HTTPException:
        raise
    except Exception as e:
        raise HTTPException(
            status_code=status.HTTP_500_INTERNAL_SERVER_ERROR,
            detail=f"Error interno del servidor: {str(e)}",
        )
```

Enpoints en el Frontend

Obtener proyecto y sus fases con GET /api/v1/proyectos/{proyecto_id}/phases

```
async function fetchProjectWithPhases(
  id: number,
  options: { forceRefresh?: boolean } = {}
): Promise<ProjectWithPhases> {
  const { forceRefresh = false } = options

  // Check cache first (if not forcing refresh)
  if (!forceRefresh && projectCache.value.has(id)) {
    const cached = projectCache.value.get(id)!
    const cacheAge = Date.now() - cached.timestamp
    const CACHE_MAX_AGE = 5 * 60 * 1000 // 5 minutes

    // If cache is still fresh, use it
    if (cacheAge < CACHE_MAX_AGE) {
      currentProject.value = cached.data
      currentProjectId.value = id
      return cached.data
    }
  }

  loading.value = true
  error.value = null

  // Get cached etag if available (declare outside try for catch block access)
  const cached = projectCache.value.get(id)

  try {
    const headers: Record<string, string> = {}

    if (cached?.etag && !forceRefresh) {
      headers['If-None-Match'] = cached.etag
    }
```

```
    const url = `/proyectos/${id}/phases` + (forceRefresh ? `?t=${Date.now()}` : '')

    const response = await apiClient.get<ProjectWithPhases>(
      url,
      { headers }
    )

    // If we get 304 Not Modified, use cached data
    if (response.status === 304 && cached) {
      // Update timestamp but keep data
      projectCache.value.set(id, {
        ...cached,
        timestamp: Date.now()
      })
      currentProject.value = cached.data
      currentProjectId.value = id
      return cached.data
    }

    // New data received
    const data = response.data
    const etag = response.headers['etag'] || null

    // Update cache
    projectCache.value.set(id, {
      data,
      etag,
      timestamp: Date.now()
    })
```

```
    currentProject.value = data
    currentProjectId.value = id
    return data
  } catch (err: any) {
    // If we have cached data and error is not critical, use cache
    if (cached && err.response?.status !== 404) {
      console.warn('Using cached data due to error:', err)
      currentProject.value = cached.data
      currentProjectId.value = id
      return cached.data
    }

    error.value = err.response?.data?.detail || 'Error al cargar proyecto con fases'
    console.error(`Failed to fetch project with phases ${id}:`, err)
    throw err
  } finally {
    loading.value = false
  }
}
```

Obtener un proyecto por id

```
async function fetchProjectById(id: number): Promise<ProjectResponse> {
  loading.value = true
  error.value = null
  try {
    const { data } = await apiClient.get<ProjectResponse>(`/proyectos/${id}`)
    return data
  } catch (err: any) {
    error.value = err.response?.data?.detail || 'Error al cargar proyecto'
    console.error(`Failed to fetch project ${id}:`, err)
    throw err
  } finally {
    loading.value = false
  }
}
```

Actualizar un proyecto con PUT /api/v1/proyectos/{proyecto_id}

```
async function updateProject(id: number, projectData: ProjectUpdate): Promise<ProjectResponse> {
  loading.value = true
  error.value = null
  try {
    const { data } = await apiClient.put<ProjectResponse>(`/proyectos/${id}`, projectData)

    // Actualizar en la lista de proyectos
    const index = projects.value.findIndex(p => p.id === id)
    if (index !== -1) {
      const updatedProjectListItem: ProjectResponse = {···
      }
      projects.value[index] = updatedProjectListItem
    }

    // Actualizar proyecto actual si es el mismo
    if (currentProject.value && currentProject.value.id === id) {
      currentProject.value = { ...currentProject.value, ...data }
    }

    return data
  } catch (err: any) {
    error.value = err.response?.data?.detail || 'Error al actualizar proyecto'
    console.error(`Failed to update project ${id}:`, err)
    throw err
  } finally {
    loading.value = false
  }
}
```

Eliminar proyecto con /api/v1/proyectos/{proyecto_id}

```
async function deleteProject(id: number): Promise<void> {
  loading.value = true
  error.value = null
  try {
    await apiClient.delete(`/proyectos/${id}`)

    // Remover de la lista
    projects.value = projects.value.filter(p => p.id !== id)

    // Limpiar proyecto actual si es el mismo
    if (currentProjectId.value === id) {
      currentProject.value = null
      currentProjectId.value = null
    }
  } catch (err: any) {
    error.value = err.response?.data?.detail || 'Error al eliminar proyecto'
    console.error(`Failed to delete project ${id}:`, err)
    throw err
  } finally {
    loading.value = false
  }
}
```

Módulo de fases

Endpoints en el Backend

Obtener una fase por id

```python
@router.get("/{phase_id}", response_model=PhaseListResponse)
async def get_phase_by_id(
    *,
    db: Session = Depends(get_db),
    phase_id: int,
    current_user: User = Depends(get_current_user),
):
    user_id = current_user.id
    return phase_service.get_phase_by_id(
        db=db,
        phase_id=phase_id,
        owner_id=user_id,  # type: ignore
    )


@router.get("/{phase_id}/tareas")
async def get_phase_tasks(
    *,
    db: Session = Depends(get_db),
    phase_id: int,
    current_user: User = Depends(get_current_user),
):
    return phase_service.get_phase_tasks(
        db=db,
        phase_id=phase_id,
    )
```

Reordenar fases en el tablero

```python
@router.put("/project/{project_id}/reorder", response_model=List[PhaseListResponse])
async def reorder_project_phases(
    *,
    db: Session = Depends(get_db),
    project_id: int,
    phase_orders: List[PhaseOrder],
    current_user: User = Depends(get_current_user),
):
    """
    Reordena las fases de un proyecto.
    """
    user_id = current_user.id
    phase_orders_dict = [order.model_dump() for order in phase_orders]
    return phase_service.reorder_phases(
        db=db,
        project_id=project_id,
        phase_orders=phase_orders_dict,
        owner_id=user_id,  # type: ignore
    )
```

Actualizar una fase por id

```python
@router.put("/{phase_id}", response_model=PhaseListResponse)
async def update_phase(
    *,
    db: Session = Depends(get_db),
    phase_id: int,
    phase_in: PhaseUpdate,
    current_user: User = Depends(get_current_user),
):
    user_id = current_user.id
    return phase_service.update_phase(
        db=db,
        phase_id=phase_id,
        phase_in=phase_in,
        owner_id=user_id,  # type: ignore
    )


@router.delete("/{phase_id}", status_code=status.HTTP_204_NO_CONTENT)
async def delete_phase(
    *,
    db: Session = Depends(get_db),
    phase_id: int,
    current_user: User = Depends(get_current_user),
):
    user_id = current_user.id
    phase_service.delete_phase(
        db=db,
        phase_id=phase_id,
        owner_id=user_id,  # type: ignore
    )
```

Endpoints en el Frontend

Obtener una fase por id

```
async function getPhaseById(phaseId: number): Promise<PhaseResponse> {
  loading.value = true
  error.value = null
  try {
    const { data } = await apiClient.get<PhaseResponse>(`/fases/${phaseId}`)
    currentPhase.value = data
    return data
  } catch (err: any) {
    const apiError: ApiError = err.response?.data
    error.value = apiError.detail || 'Error al obtener fase'
    console.error(`Failed to get phase ${phaseId}:`, err)
    throw err
  } finally {
    loading.value = false
  }
}
```

Actualizar fase con PUT /api/v1/fases/{fase_id}

```
async function updatePhase(phaseId: number, phaseData: PhaseUpdate): Promise<PhaseResponse> {
  loading.value = true
  error.value = null
  try {
    const { data } = await apiClient.put<PhaseResponse>(`/fases/${phaseId}`, phaseData)

    // Actualizar en la lista de fases
    const index = phases.value.findIndex(p => p.id === phaseId)
    if (index !== -1) {
      phases.value[index] = data
    }

    // Actualizar fase actual si es la misma
    if (currentPhase.value && currentPhase.value.id === phaseId) {
      currentPhase.value = data
    }

    // Invalidar caché del proyecto
    const projectsStore = useProjectsStore()
    projectsStore.invalidateProjectCache(data.project_id)

    return data
  } catch (err: any) {
    const apiError: ApiError = err.response?.data
    error.value = apiError.detail || 'Error al actualizar fase'
    console.error(`Failed to update phase ${phaseId}:`, err)
    throw err
  } finally {
    loading.value = false
  }
}
```

Eliminar fase con DELETE /api/v1/fases/{fase_id}

```
async function deletePhase(phaseId: number): Promise<void> {
  loading.value = true
  error.value = null
  try {
    await apiClient.delete(`/fases/${phaseId}`)

    // Obtener project_id antes de eliminar para invalidar caché
    const phase = phases.value.find(p => p.id === phaseId)
    const projectId = phase?.project_id

    // Remover de la lista
    phases.value = phases.value.filter(p => p.id !== phaseId)

    // Limpiar fase actual si es la misma
    if (currentPhase.value && currentPhase.value.id === phaseId) {
      currentPhase.value = null
    }

    // Invalidar caché del proyecto
    if (projectId) {
      const projectsStore = useProjectsStore()
      projectsStore.invalidateProjectCache(projectId)
    }
  } catch (err: any) {
    const apiError: ApiError = err.response?.data
    error.value = apiError.detail || 'Error al eliminar fase'
    console.error(`Failed to delete phase ${phaseId}:`, err)
    throw err
  } finally {
    loading.value = false
  }
}
```

Actualizar la posición de una fase con PUT /fases/project/{project_id}/reorder

```
async function reorderProjectPhases(projectId: number, phaseOrders: PhaseOrder[]): Promise<PhaseResponse[]> {
  loading.value = true
  error.value = null
  try {
    const { data } = await apiClient.put<PhaseResponse[]>(
      `/fases/project/${projectId}/reorder`,
      phaseOrders
    )

    // Actualizar las fases en la lista con las nuevas posiciones
    data.forEach(updatedPhase => {
      const index = phases.value.findIndex(p => p.id === updatedPhase.id)
      if (index !== -1) {
        phases.value[index] = updatedPhase
      }
    })

    // Invalidar caché del proyecto
    const projectsStore = useProjectsStore()
    projectsStore.invalidateProjectCache(projectId)

    return data
  } catch (err: any) {
    const apiError: ApiError = err.response?.data
    error.value = apiError.detail || 'Error al reordenar fases'
    console.error(`Failed to reorder phases for project ${projectId}:`, err)
    throw err
  } finally {
    loading.value = false
  }
}
```

Módulo de tareas

Endpoints en el Backend

Enums para campo TaskStatus del modelo de tareas

```python
from app.database import Base


class TaskStatus(str, Enum):
    """Estados posibles de una tarea"""

    PENDING = "pending"
    IN_PROGRESS = "in_progress"
    COMPLETED = "completed"
    ON_HOLD = "on_hold"

```

Importaciones necesarias para los endpoints del módulo de tareas

```python
from typing import Optional

from fastapi import APIRouter, Depends, File, UploadFile, status
from sqlalchemy.orm import Session

from app.core.dependencies import get_current_user
from app.database import get_db
from app.models.user import User
from app.schemas import TaskCreate, TaskResponse, TaskUpdate
from app.schemas.attachment import AttachmentResponse
from app.services import task_service
from app.services.attachment_service import attachment_service

router = APIRouter()
```

Obtener una tarea de una fase

```python
@router.get("/", response_model=list[TaskResponse])
async def get_tasks_by_phase(
    *,
    db: Session = Depends(get_db),
    phase_id: int,
    current_user: User = Depends(get_current_user),
):
    """
    Obtiene todas las tareas asociadas a una fase específica para el usuario autenticado.

    Args:
        db (Session): Sesión de base de datos proporcionada por la dependencia.
        phase_id (int): ID de la fase para la cual se desean obtener las tareas.
        current_user (User): El usuario actualmente autenticado.

    Returns:
        list[TaskResponse]: Una lista de tareas asociadas a la fase especificada.
    """
    user_id = current_user.id
    return task_service.get_phase_tasks(
        db=db,
        phase_id=phase_id,
        owner_id=user_id,  # type: ignore
    )
```

Actualizar tarea por id

```python
@router.put("/{task_id}", response_model=TaskResponse)
async def update_task(
    *,
    db: Session = Depends(get_db),
    task_id: int,
    task_in: TaskUpdate,
    current_user: User = Depends(get_current_user),
):
    """
    Actualiza una tarea existente para el usuario autenticado.

    Args:
        db (Session): Sesión de base de datos proporcionada por la dependencia.
        task_id (int): ID de la tarea que se desea actualizar.
        task_in (TaskUpdate): Datos para actualizar la tarea.
        current_user (User): El usuario actualmente autenticado.
    Returns:
        TaskResponse: Los datos de la tarea actualizada.
    """
    user_id = current_user.id
    return task_service.update_task(
        db=db,
        task_id=task_id,
        task_in=task_in,
        owner_id=user_id,  # type: ignore
    )
```

Eliminar tarea por id

```python
@router.delete("/{task_id}", status_code=status.HTTP_204_NO_CONTENT)
async def delete_task(
    *,
    db: Session = Depends(get_db),
    task_id: int,
    current_user: User = Depends(get_current_user),
):
    """
    Elimina una tarea existente para el usuario autenticado.

    Args:
        db (Session): Sesión de base de datos proporcionada por la dependencia.
        task_id (int): ID de la tarea que se desea eliminar.
        current_user (User): El usuario actualmente autenticado.

    Returns:
        None
    """
    user_id = current_user.id
    return task_service.delete_task(
        db=db,
        task_id=task_id,
        owner_id=user_id,  # type: ignore
    )
```

Endpoints en el Frontend

Actualizar una tarea con PUT /api/v1/tareas/{tarea_id}

```
async function updateTask(taskId: number, taskData: TaskUpdate): Promise<TaskResponse> {
  loading.value = true
  error.value = null
  try {
    const { data } = await apiClient.put<TaskResponse>(`/tareas/${taskId}`, taskData)

    // Actualizar en la lista de tareas
    const index = tasks.value.findIndex(t => t.id === taskId)
    if (index !== -1) {
      tasks.value[index] = data
    }

    // Actualizar tarea actual si es la misma
    if (currentTask.value && currentTask.value.id === taskId) {
      currentTask.value = data
    }

    return data
  } catch (err: any) {
    error.value = err.response?.data?.detail || 'Error al actualizar tarea'
    console.error(`Failed to update task ${taskId}:`, err)
    throw err
  } finally {
    loading.value = false
  }
}
```

Eliminar una tarea con DELETE /api/v1/tareas/{tarea_id}

```
async function deleteTask(taskId: number): Promise<void> {
  loading.value = true
  error.value = null
  try {
    await apiClient.delete(`/tareas/${taskId}`)

    // Remover de la lista
    tasks.value = tasks.value.filter(t => t.id !== taskId)

    // Limpiar tarea actual si es la misma
    if (currentTask.value && currentTask.value.id === taskId) {
      currentTask.value = null
    }
  } catch (err: any) {
    const apiError: ApiError = err.response?.data
    error.value = apiError.detail || 'Error al eliminar tarea'
    console.error(`Failed to delete task ${taskId}:`, err)
    throw err
  } finally {
    loading.value = false
  }
}
```

Mover una tarea entre fases con PUT /api/v1/tareas/{tarea_id}/mover

```
async function moveTaskToPhase(taskId: number, newPhaseId: number): Promise<TaskResponse | undefined> {
  const newDataToUpdateInTask: { new_phase_id: number; new_position?: number } = { new_phase_id: newPhaseId }

  // Primero obtenemos las tareas de la fase destino para calcular la nueva posición
  const targetPhaseTasks = tasksByPhase.value(newPhaseId)
  const newPosition = targetPhaseTasks.length

  newDataToUpdateInTask.new_position = newPosition

  loading.value = true
  error.value = null
  try {
    const { data } = await apiClient.put<TaskResponse>(`/tareas/${taskId}/mover`, newDataToUpdateInTask)

    // Actualizar en la lista de tareas
    const index = tasks.value.findIndex(t => t.id === taskId)
    if (index !== -1) {
      tasks.value[index] = data
    }

    // Actualizar tarea actual si es la misma
    if (currentTask.value && currentTask.value.id === taskId) {
      currentTask.value = data
    }

    return data
  } catch (err: any) {
    const apiError: ApiError = err.response?.data
    error.value = apiError.detail || 'Error al mover tarea'
  } finally {
    loading.value = false
  }
}
```

Endpoints en el Frontend

Obtener las tareas de una fase con GET /api/v1/fases/{fase_id}/tareas

```
async function getTasksByPhase(phaseId: number): Promise<TaskResponse[]> {
  loading.value = true
  error.value = null
  try {
    const { data } = await apiClient.get<TaskResponse[]>(`/fases/${phaseId}/tareas`)

    // Actualizar las tareas de esta fase en el store
    const otherTasks = tasks.value.filter(t => t.phase_id !== phaseId)
    tasks.value = [...otherTasks, ...data]

    return data
  } catch (err: any) {
    error.value = err.response?.data?.detail || 'Error al obtener tareas'
    console.error(`Failed to get tasks for phase ${phaseId}:`, err)
    throw err
  } finally {
    loading.value = false
  }
}
```

Anexo 2: Cronograma

Para visualizar la distribución temporal de las actividades planificadas durante las 11 semanas del proyecto se presenta en la Figura 55 y la Figura 56 el cronograma detallado que abarca desde la conceptualización hasta el despliegue final estableciendo los tiempos estimados para cada fase del desarrollo.

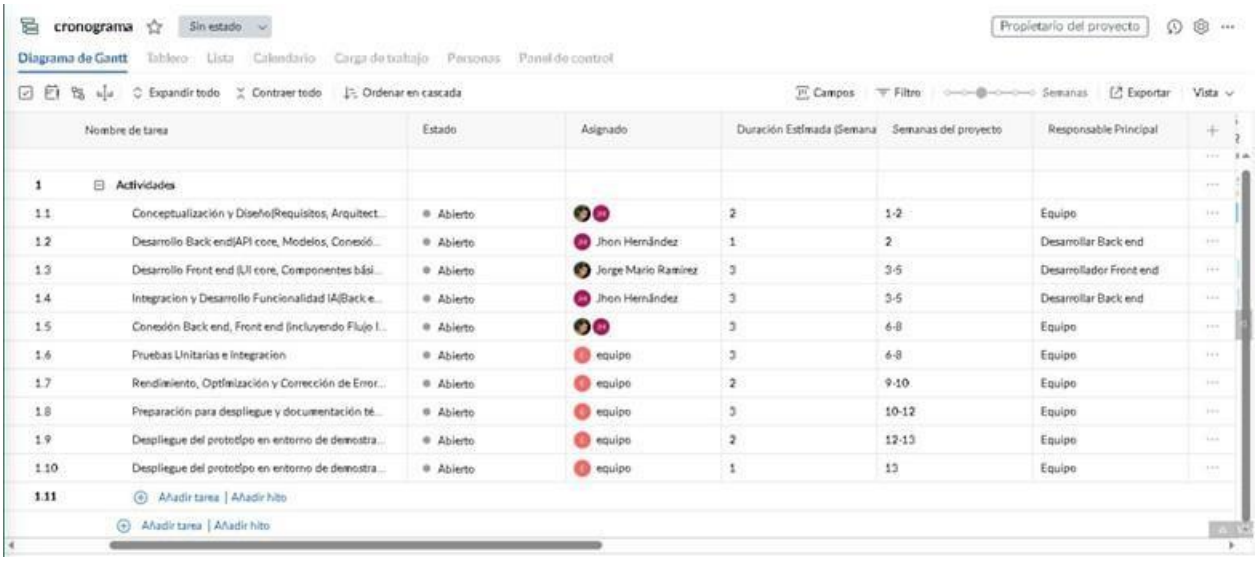


Figura 55. Cronograma de Actividades Parte 1

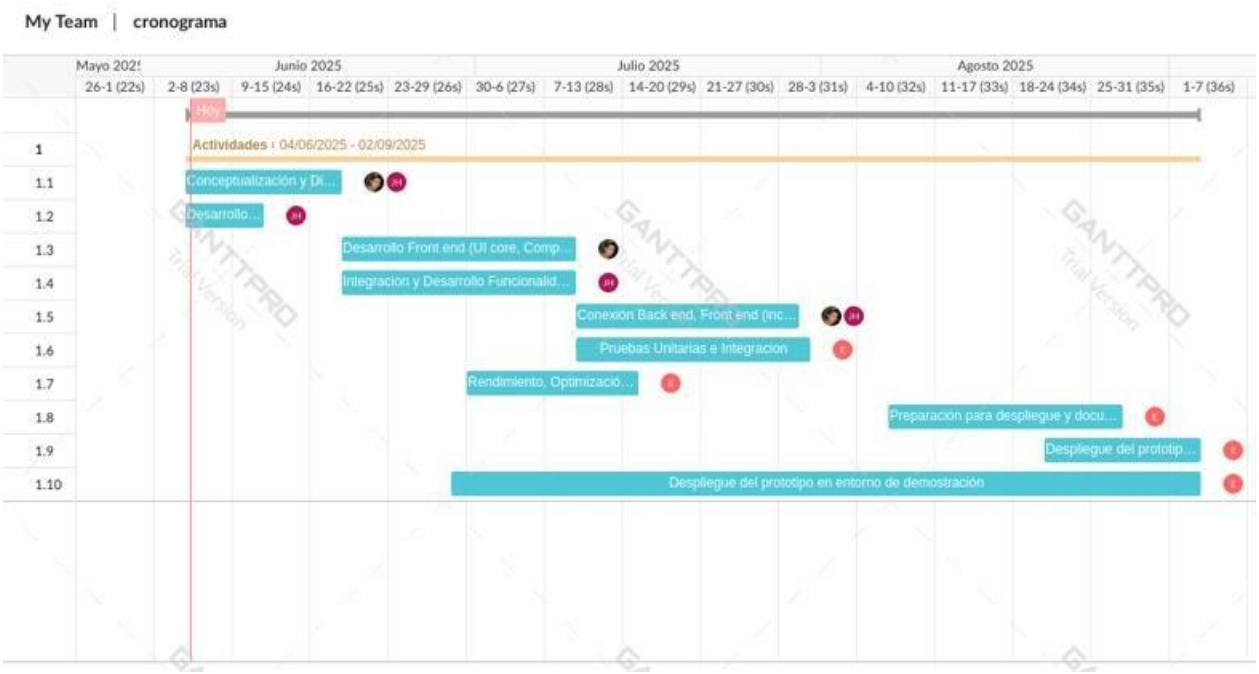


Figura 56. Cronograma de Actividades Parte 2

Anexo 3: Presupuesto

Para garantizar la viabilidad económica del proyecto y la correcta asignación de recursos, se presenta en la Tabla 2 el presupuesto detallado que contempla los costos asociados al talento humano y los servicios de infraestructura tecnológica necesarios para la implementación del sistema.

Tabla 2. Presupuesto del Proyecto

Rubro/ Categoría	Descripción del gasto	Unidad	Cantidad	Valor
		1. Recursos Humanos		
1.1	Desarrollador Backend & IA (Jhon H.))	Mes	2.(6 días)	1.6000.000
1.2	Desarrollador Frontend (Jorge R.)	Mes	2.(6 días)	1.6000.000
Subtotal				7.040.000
		2. Software y servicios Cloud		
2.1	Base de Datos: Neon PostgreSQL	Mes	2.(6 días)	20 usd (73.544,36$)
2.2	API de IA: Google Gemini	Mes	2.(6 días)	5 usd (18.386,09)
2.3	Hosting	Global	1	Gratis
Subtotal				65,00 usd (239019.17)
Total general				7.279.019.17

Nota. El presupuesto detalla los costos operativos necesarios para la implementación del proyecto, incluyendo el talento humano y servicios en la nube. Los valores en dólares (USD) fueron calculados según la tasa de cambio vigente al momento de la elaboración del documento.

Anexo 4: Scrum

Product Backlog priorizado

A continuación, se presenta el *Product Backlog* priorizado del proyecto en la Tabla 3, en la cual se detallan las historias de usuario identificadas, su prioridad, estimación y los requisitos funcionales asociados.

Tabla 3. Product Backlog de SCRUM

ID	NOMBRE DE LA HISTORIA	PRIORIDAD	ESTIMACIÓN	REQUISITOS ASOCIADOS
HU-01	Registro de nuevos usuarios	Alta	8	RF01, RF26
HU-02	Inicio y cierre de sesión	Alta	5	RF02, RF18, RF26
HU-03	Creación de nuevos proyectos	Alta	5	RF07
HU-04	Visualización de proyectos del usuario	Alta	5	RF08, RF32
HU-05	Carga de documentos en un proyecto	Alta	8	RF10
HU-06	Búsqueda básica de investigaciones	Alta	8	RF12
HU-07	Asistente IA para sugerencias académicas	Alta	13	RF13, RF31
HU-08	Navegación principal autenticada	Alta	3	RF32
HU-09	Visualización y edición de documentos	Alta	5	RF11

HU-10	Filtros de investigación	Alta	5	RF17
HU-11	Generación de citas y bibliografías	Alta	8	RF14
HU-12	Descarga de proyecto en PDF	Alta	5	RF15
HU-13	Visualización landing page	Media	3	RF28
HU-14	Gestión de favoritos	Media	3	RF16

Nota. Fuente: Elaboración propia

Sprint 1

Objetivo del Sprint

Al finalizar el Sprint, un nuevo usuario podrá registrarse en la plataforma, iniciar sesión, crear un proyecto y cargarle un documento asociado, también existirá una navegación básica para el usuario autenticado.

La duración del sprint es de 3 semanas hábiles, que va desde el jueves, 7 de agosto de 2025 hasta el lunes, 1 de septiembre de 2025, la Capacidad Planificada fue de 21 Story Points.

Sprint Backlog (Historias de Usuario Seleccionadas)

HU-01: Registro de nuevos usuarios (8 SP)

HU-02: Inicio y cierre de sesión (5 SP)

HU-03: Creación de nuevos proyectos (5 SP)

HU-08: Navegación principal autenticada (3 SP)

Incremento del Sprint 1 (Entregable Funcional)

El usuario podrá ver una página de inicio de sesión.

Hacer clic en un enlace "Registrarse" y crear una cuenta.

Iniciar sesión con sus credenciales.

Al iniciar sesión, ver un panel vacío con un botón para "Crear Nuevo Proyecto".

Crear un proyecto, que aparecerá en una lista.

Cerrar su sesión de forma segura.

Sprint 2

Objetivo del Sprint

Sobre la base existente, el usuario podrá entrar a un proyecto, cargar documentos PDF/Word y realizar búsquedas básicas dentro de sus investigaciones. Se introducirá una primera versión del asistente IA.

La duración del sprint es de 2 semanas desde el martes, 2 de septiembre de 2025 hasta el viernes, 16 de septiembre de 2025, la capacidad planificada es aproximadamente de 21 Story Points (se ajusta según la velocidad final del Sprint 1).

Sprint Backlog (Historias de Usuario Seleccionadas)

HU-05: Carga de documentos en un proyecto (8 SP)

HU-06: Búsqueda básica de investigaciones (8 SP)

HU-09: Visualización y edición de documentos (5 SP) (Priorizar visualización sobre edición)

Incremento del Sprint 2 (Entregable Funcional)

El usuario podrá hacer clic en un proyecto de su lista para ver sus detalles.

Usar un botón para "Cargar Documento" y subir un archivo.

Ver la lista de documentos cargados en un proyecto.

Utilizar una barra de búsqueda para encontrar proyectos o documentos por título.

Sprint 3

Objetivo del Sprint

Mejorar la utilidad de la plataforma con filtros avanzados, la capacidad de generar citas y bibliografías con el asistente IA, y la opción de descargar un resumen del proyecto en PDF.

La Duración del sprint es de 2 semanas, desde el miércoles, 17 de septiembre de 2025 hasta el martes, 30 de septiembre de 2025.

Sprint Backlog (Historias de Usuario Seleccionadas)

HU-07: Asistente IA para sugerencias académicas (13 SP) (Esta HU es grande, podría necesitar descomponerse más).

HU-10: Filtros de investigación (5 SP)

HU-12: Descarga de proyecto en PDF (5 SP)

Incremento del Sprint 3 (Entregable Funcional)

El usuario podrá interactuar con un asistente de IA para obtener sugerencias.

Generar citas en formatos estándar (APA, MLA).

Filtrar la vista de proyectos por categorías.

Descargar una vista simplificada del proyecto como un archivo PDF.