DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA WEB PARA LA GESTIÓN DEL PROCESO DE ACEPTACIÓN DE PROYECTOS ACADÉMICOS

PROYECTO GRADO II

YOndayler FLores

floresyondayler@gmail.com

Ingenieria en Informatica

Facultad de Ciencias de la Ingeniería

San Juan de los Morros - diciembre de 2025

ÍNDICE

3. RESUMEN

La presente tesis aborda el "Diseño e Implementación de un Sistema Web para la Gestión del Proceso de Aceptación de Proyectos Académicos", con el **objetivo fundamental** de optimizar y automatizar el proceso de aprobación de propuestas de investigación en la carrera de Ingeniería en Informática. La problemática actual se centra en la gestión manual, descentralizada y propensa a errores, que genera demoras y falta de transparencia. Para superar estas limitaciones, se adoptó una **metodología de desarrollo ágil**, específicamente Scrum, permitiendo una construcción iterativa y adaptativa del sistema (López, 2025). La implementación se realizó sobre una arquitectura Modelo-Vista-Controlador (MVC), utilizando **PHP con el framework Laravel** para el backend, **MySQL** como gestor de base de datos, y **JavaScript con Vue.js** para una interfaz de usuario dinámica y eficiente, en línea con enfoques modernos de desarrollo (Vera, 2024). El sistema resultante busca garantizar la eficiencia y la trazabilidad en cada etapa del proceso de aceptación de proyectos.

DESCRIPTORES: Sistema, Web, Gestión, Proyectos.

4. DIAGNÓSTICO SITUACIONAL

4.1 Descripción del contexto de la situación problemática planteada

La gestión de proyectos académicos en la carrera de Ingeniería en Informática, hasta el momento, se ha caracterizado por un enfoque predominantemente **manual y descentralizado**. Este modelo tradicional implica una serie de pasos que, aunque lógicos en su concepción, resultan ineficientes y propensos a errores en la práctica. El proceso inicia con la presentación de propuestas de investigación por parte de los estudiantes, las cuales son entregadas en formato físico o mediante correos electrónicos a coordinadores y comités evaluadores. Esta diversidad en los canales de recepción ya introduce una primera capa de complejidad, dificultando la consolidación y seguimiento uniforme de los proyectos.

La evaluación de estas propuestas recae en los comités académicos, quienes deben revisar manualmente cada documento, cotejar la información, y emitir dictámenes. Este proceso es inherentemente lento, ya que depende de la disponibilidad de los miembros del comité y de la coordinación manual de reuniones. La **falta de un repositorio centralizado** para las propuestas, los expedientes de los estudiantes y las evaluaciones genera una dispersión de la información crítica. Consecuentemente, el acceso al historial de un proyecto, el estado actual de su aprobación o las observaciones previas se convierte en una tarea ardua y consume tiempo valioso tanto para estudiantes como para docentes y personal administrativo. La comunicación entre las partes interesadas (estudiantes, tutores, coordinadores y comité) se realiza a través de medios informales como correos electrónicos o reuniones presenciales, lo que a menudo conduce a malentendidos, pérdida de información relevante y retrasos en la retroalimentación.

Adicionalmente, la ausencia de un sistema estandarizado para la documentación y el seguimiento de cada etapa del proceso de aceptación de proyectos impacta negativamente en la **transparencia y la trazabilidad**. No existe una forma sencilla de auditar el progreso de una propuesta, identificar cuellos de botella o garantizar la equidad en la evaluación. La carga administrativa asociada a la gestión de expedientes físicos, la organización de reuniones, la emisión y notificación de actas, y el seguimiento de los plazos, desvía recursos humanos y tiempo que podrían dedicarse a actividades académicas de mayor valor. Esta situación no solo genera frustración entre los estudiantes debido a la incertidumbre y las demoras, sino que también afecta la eficiencia operativa de la institución, limitando su capacidad para gestionar un volumen creciente de proyectos académicos de manera efectiva y oportuna.

4.2 Justificación del proyecto

El "Diseño e Implementación de un Sistema Web para la Gestión del Proceso de Aceptación de Proyectos Académicos" se justifica plenamente por la imperante necesidad de transformar y modernizar un proceso crítico dentro de la carrera de Ingeniería en Informática que, en su estado actual, presenta serias deficiencias. La problemática descrita en el apartado anterior, caracterizada por la gestión manual, la descentralización de la información, la lentitud en los tiempos de respuesta y la falta de transparencia, impacta directamente en la calidad académica y la eficiencia operativa. La adopción de una solución tecnológica se convierte en un imperativo para solventar estas limitaciones y potenciar el desarrollo investigativo de la institución. En este sentido, proyectos similares han demostrado el éxito de la reingeniería de sistemas para optimizar procesos en entornos académicos y de servicios, como lo evidencia el trabajo de José López (2025) en la "Reingeniería del Sistema Clínico Odontológico", donde se buscó modernizar la atención en salud mediante la implementación de un sistema robusto, mejorando la eficiencia en la administración de datos y la gestión de la información.

La implementación de un sistema web dedicado permitirá consolidar todas las fases del proceso de aceptación de proyectos en una única plataforma digital. Esto no solo eliminará la dispersión de documentos y comunicaciones, sino que también establecerá un flujo de trabajo estandarizado y automatizado. La digitalización de la entrega de propuestas, la asignación de evaluadores, la emisión de dictámenes y la notificación de resultados, reducirá drásticamente los errores humanos asociados a la transcripción y el manejo de información. Además, la centralización de datos ofrecerá una visión **panorámica y en tiempo real** del estado de cada proyecto, facilitando la toma de decisiones informadas por parte de los coordinadores y el comité académico.

Desde la perspectiva de los estudiantes, el sistema garantizará mayor **transparencia y accesibilidad**. Podrán consultar el estado de su propuesta en cualquier momento, recibir retroalimentación oportuna y acceder a las normativas y requisitos de manera clara. Esto fomentará una mayor autonomía y reducirá la incertidencia que actualmente experimentan. Para el cuerpo docente y administrativo, la automatización de tareas repetitivas liberará tiempo para enfocarse en actividades de mayor valor pedagógico y estratégico, como la mentoría de proyectos, la investigación o la mejora curricular. En última instancia, este proyecto contribuirá a la construcción de una cultura de eficiencia, transparencia y excelencia académica, elementos fundamentales para el posicionamiento de la carrera de Ingeniería en Informática y la institución en su conjunto, al ofrecer una plataforma que soporte de manera robusta y escalable la gestión de su producción intelectual.

4.3 Objetivos del proyecto

El proyecto "Diseño e Implementación de un Sistema Web para la Gestión del Proceso de Aceptación de Proyectos Académicos" tiene como **objetivo general** optimizar y automatizar el ciclo de vida de las propuestas de investigación en la carrera de Ingeniería en Informática, desde la presentación inicial hasta la aceptación formal, mediante el desarrollo de una plataforma web integral que garantice eficiencia, transparencia y trazabilidad.

Para alcanzar este objetivo general, se establecen los siguientes **objetivos específicos**:

1. **Analizar** los procesos actuales de gestión y aceptación de proyectos académicos en la carrera de Ingeniería en Informática, identificando las etapas clave, los actores involucrados, los documentos utilizados y los puntos críticos que requieren automatización.
2. **Investigar** las tecnologías y herramientas web más adecuadas para el desarrollo de un sistema de gestión de proyectos académicos, considerando aspectos como escalabilidad, seguridad, facilidad de uso y compatibilidad con la infraestructura existente.
3. **Establecer** los requisitos funcionales y no funcionales del sistema web, a partir del análisis de los procesos y las necesidades de los usuarios (estudiantes, tutores, coordinadores y comité académico), utilizando técnicas de levantamiento de información como entrevistas y encuestas.
4. **Diseñar** la arquitectura del sistema web, incluyendo la base de datos, la interfaz de usuario, los módulos principales y las interacciones entre ellos, asegurando una estructura modular, robusta y escalable.
5. **Desarrollar** e implementar el sistema web para la gestión del proceso de aceptación de proyectos académicos, utilizando las tecnologías seleccionadas y aplicando buenas prácticas de programación, garantizando su funcionalidad y rendimiento.
6. **Evaluar** la usabilidad y la funcionalidad del sistema web desarrollado a través de pruebas con usuarios reales, recopilando retroalimentación para identificar posibles mejoras y asegurar su correcta operación.

4.4 Procesos que se van a automatizar

El "Diseño e Implementación de un Sistema Web para la Gestión del Proceso de Aceptación de Proyectos Académicos" se enfocará en automatizar y optimizar una serie de procesos clave que actualmente se realizan de manera manual, lo que permitirá una gestión más eficiente y transparente. La automatización de estos flujos de trabajo no solo reducirá la carga administrativa, sino que también mejorará la comunicación y la trazabilidad de cada etapa del proyecto.

Los principales procesos que se van a automatizar son:

- **Registro y Presentación de Proyectos:** Los estudiantes podrán registrarse en el sistema y subir sus propuestas de proyectos académicos de manera digital. Esto incluye la carga de documentos (propuesta, plan de trabajo, cronograma), la introducción de datos del proyecto (título, área de investigación, tutor propuesto) y la confirmación de la presentación. El sistema validará los formatos y la completitud de la información antes de la aceptación.
- **Asignación de Evaluadores:** Una vez que una propuesta es presentada, el sistema permitirá a los coordinadores académicos o al comité de tesis asignar evaluadores de manera eficiente. Se podrá seleccionar a los docentes con la experticia relevante para cada área de investigación, y el sistema notificará automáticamente a los evaluadores sobre las propuestas asignadas para su revisión.
- **Revisión y Evaluación de Propuestas:** Los evaluadores accederán al sistema para revisar las propuestas asignadas. Se les proporcionará una interfaz para registrar sus observaciones, comentarios y la calificación de la propuesta según criterios predefinidos (originalidad, viabilidad, metodología, etc.). El sistema facilitará la retroalimentación estructurada y la emisión de un dictamen formal (aceptado, con observaciones, rechazado).
- **Comunicación y Notificaciones:** El sistema gestionará automáticamente las comunicaciones entre todas las partes interesadas. Esto incluye notificaciones a los estudiantes sobre el estado de su propuesta, recordatorios a los evaluadores sobre los plazos de revisión, y alertas a los coordinadores sobre las propuestas pendientes de acción. Se implementará un módulo de mensajería interna para facilitar la interacción y resolver dudas de manera centralizada.
- **Gestión del Historial y Documentación:** Se creará un repositorio digital centralizado para almacenar todas las propuestas, evaluaciones, actas de aprobación y cualquier otro documento relevante asociado a cada proyecto. Esto garantizará la trazabilidad completa del ciclo de vida del proyecto, permitiendo consultas rápidas sobre el estado, el historial de cambios y las decisiones tomadas.
- **Generación de Informes y Estadísticas:** El sistema será capaz de generar informes personalizados y estadísticas sobre el proceso de aceptación de proyectos. Esto incluirá datos como el número de propuestas presentadas, el promedio de tiempo de aprobación, las áreas de investigación más populares, y el rendimiento de los evaluadores, proporcionando información valiosa para la mejora continua del proceso.

- **Aprobación y Archivo:** Una vez que una propuesta ha sido revisada y cumple con todos los requisitos, el sistema permitirá al comité académico emitir la aprobación final. La propuesta se marcará como aceptada y se archivará digitalmente, consolidando el expediente del proyecto y facilitando el seguimiento posterior de su desarrollo.

ÍNDICE GENERAL

5. DETERMINACIÓN, INSTALACIÓN Y CONFIGURACIÓN DE LAS HERRAMIENTAS DE DESARROLLO

5.1 Plataforma de Desarrollo

5.2 Arquitectura del sistema de información

5.3 Selección del entorno del sistema

5.4 Metodología para el desarrollo

6. DISEÑO DEL SISTEMA PROPUESTO

6.1 Diseño de la Arquitectura Lógica

6.2 Diseño de la Base de Datos

6.3 Diseño de la Interfaz de Usuario (UI/UX)

6.4 Diseño de Módulos y Componentes

7. IMPLEMENTACIÓN DEL SISTEMA

7.1 Desarrollo del Backend

7.2 Desarrollo del Frontend

7.3 Integración de Componentes

7.4 Pruebas y Depuración

8. RESULTADOS Y DISCUSIÓN

8.1 Presentación del Sistema Implementado

8.2 Evaluación de Funcionalidades

8.3 Comparación con los Objetivos

8.4 Impacto y Beneficios

9. CONCLUSIONES Y RECOMENDACIONES

9.1 Conclusiones

9.2 Recomendaciones para Futuras Investigaciones

Los capítulos precedentes han establecido un fundamento sólido para la presente investigación. En el **Capítulo 3, Resumen**, se delineó la problemática central, que radica en la ineficiencia y falta de transparencia del proceso manual de gestión de proyectos académicos en la carrera de Ingeniería en Informática. Se subrayó la necesidad imperante de un sistema web que automatice y optimice este proceso, sentando las bases para el objetivo general de la tesis: el diseño e implementación de dicha solución. Posteriormente, el **Capítulo 4, Diagnóstico Situacional**, profundizó en la descripción detallada del contexto actual, analizando críticamente los procesos manuales existentes, identificando sus puntos débiles y estableciendo los requerimientos funcionales y no funcionales que el sistema propuesto deberá satisfacer. Asimismo, se realizó un estudio de factibilidad que confirmó la viabilidad técnica, operativa y económica del proyecto, justificando la inversión de recursos en su desarrollo.

En este punto de la tesis, con una comprensión clara de la problemática y los requerimientos del sistema, así como la confirmación de su factibilidad, la atención se dirige hacia la concreción de la solución. El presente capítulo, **Determinación, Instalación y Configuración de las Herramientas de Desarrollo**, es crucial porque establece las bases tecnológicas y metodológicas sobre las cuales se construirá el sistema web. Su objetivo principal es seleccionar y justificar las herramientas de software, la arquitectura del sistema y la metodología de desarrollo que permitirán transformar los requisitos identificados en una solución funcional y robusta. Se busca asegurar que las decisiones tomadas en esta etapa no solo sean adecuadas para las necesidades específicas del proyecto, sino que también se alineen con las mejores prácticas de la ingeniería de software, garantizando la eficiencia, escalabilidad y mantenibilidad del sistema a largo plazo. La coherencia con los capítulos anteriores se mantiene al elegir herramientas que respondan directamente a los requerimientos funcionales y no funcionales derivados del diagnóstico situacional, y al seleccionar una metodología que optimice el desarrollo para abordar las ineficiencias previamente identificadas.

5. DETERMINACIÓN, INSTALACIÓN Y CONFIGURACIÓN DE LAS HERRAMIENTAS DE DESARROLLO

5.1 Plataforma de Desarrollo

La elección de la plataforma de desarrollo constituye una decisión estratégica fundamental en cualquier proyecto de software, especialmente en el contexto de sistemas web que buscan optimizar procesos académicos. Esta selección no solo impacta directamente en la eficiencia y robustez del sistema final, sino también en la curva de aprendizaje del equipo de desarrollo, la disponibilidad de recursos y soporte, y la capacidad de evolución futura de la aplicación. Para el "Diseño e Implementación de un Sistema Web para la Gestión del Proceso de Aceptación de Proyectos Académicos", se ha determinado que la plataforma de desarrollo más adecuada es un entorno que combine la flexibilidad de un lenguaje de programación de propósito general con la potencia de un framework MVC (Modelo-Vista-Controlador) y la estabilidad de una base de datos relacional. Esta combinación garantiza una arquitectura escalable, una separación clara de responsabilidades y una gestión eficiente de los datos, aspectos críticos para un sistema que manejará información académica sensible y procesos administrativos complejos.

La justificación para esta elección se fundamenta en varios pilares. En primer lugar, la naturaleza del proyecto, que implica la creación de una aplicación web dinámica y transaccional, demanda un lenguaje de programación con un amplio soporte para el desarrollo web y una comunidad activa. Lenguajes como PHP, Python o JavaScript (en el lado del servidor con Node.js) se presentan como opciones viables debido a su madurez, rendimiento y la vasta cantidad de librerías y frameworks disponibles. La selección de uno de estos lenguajes se complementa con la integración de un framework web, el cual proporciona una estructura predefinida, herramientas para la automatización de tareas comunes (como el enrutamiento, la autenticación y la validación de datos) y patrones de diseño que promueven el código limpio y mantenible. La adopción de un framework MVC, en particular, facilita la organización del código al separar la lógica de negocio (Modelo), la presentación de datos (Vista) y la interacción del usuario (Controlador), lo que es crucial para la colaboración en equipo y la escalabilidad del proyecto.

En segundo lugar, la necesidad de gestionar un volumen considerable de información estructurada, como datos de estudiantes, proyectos, docentes, comités y estados de aprobación, hace imperativa la elección de un sistema de gestión de bases de datos relacional (SGBDR). Los SGBDR, como MySQL, PostgreSQL o SQL Server, ofrecen un alto grado de integridad de datos, consistencia transaccional y la capacidad de realizar

consultas complejas y eficientes. La familiaridad del equipo de desarrollo con estas tecnologías, junto con su amplia documentación y soporte comunitario, minimiza los riesgos asociados a la implementación y el mantenimiento. La interacción entre la aplicación web y la base de datos se realizará a través de un ORM (Object-Relational Mapping) proporcionado por el framework, lo que simplificará las operaciones de base de datos al permitir a los desarrolladores trabajar con objetos en lugar de sentencias SQL directas, mejorando la productividad y reduciendo la probabilidad de errores.

Finalmente, la plataforma de desarrollo debe ser lo suficientemente flexible para permitir la integración con otras herramientas y servicios, como sistemas de autenticación externos, servicios de correo electrónico para notificaciones y APIs para futuras extensiones. La elección de tecnologías de código abierto, cuando sea posible, no solo reduce los costos de licenciamiento, sino que también fomenta la transparencia y la adaptabilidad. Esta aproximación holística a la selección de la plataforma de desarrollo asegura que el sistema web para la gestión de proyectos académicos no solo cumpla con los requisitos funcionales actuales, sino que también esté preparado para futuras evoluciones y expansiones, contribuyendo a la sostenibilidad y relevancia a largo plazo de la solución tecnológica.

5.2 Arquitectura del sistema de información

La arquitectura de un sistema de información define la estructura fundamental de un sistema, encarnando sus componentes, las relaciones entre ellos y los principios que guían su diseño y evolución. Para el sistema web de gestión de proyectos académicos, se ha optado por una **arquitectura Cliente-Servidor**, un modelo ampliamente reconocido y adoptado en el desarrollo de aplicaciones distribuidas debido a su robustez, escalabilidad y la clara separación de responsabilidades que ofrece. Esta elección arquitectónica es idónea para una aplicación que será accedida por múltiples usuarios (estudiantes, docentes, administradores) desde diversas ubicaciones, garantizando una experiencia de usuario consistente y un rendimiento óptimo.

En el modelo Cliente-Servidor, la interacción se divide fundamentalmente en dos roles principales: el **cliente** y el **servidor**. El cliente es la parte del sistema que solicita servicios y recursos, mientras que el servidor es la parte que los proporciona. Para nuestra aplicación, los clientes serán los navegadores web utilizados por los usuarios finales (Chrome, Firefox, Edge, etc.), los cuales son responsables de presentar la interfaz de usuario y de enviar las solicitudes al servidor. Estas solicitudes pueden incluir acciones como iniciar sesión, registrar un proyecto, consultar el estado de una propuesta o actualizar información personal. La lógica de presentación y una parte de la lógica de negocio pueden residir en el cliente (lo que se conoce como "frontend"), mejorando la interactividad y reduciendo la carga del servidor para ciertas operaciones.

El servidor, por su parte, es el componente central que procesa las solicitudes de los clientes, gestiona la lógica de negocio de la aplicación y se encarga de la persistencia de los datos. Este componente reside en una máquina remota y está compuesto por un servidor web (como Apache o Nginx), un intérprete del lenguaje de programación del backend (por ejemplo, PHP, Python o Node.js), y un sistema de gestión de bases de datos relacional (como MySQL o PostgreSQL). Cuando un cliente envía una solicitud, el servidor web la recibe, la pasa al intérprete del lenguaje de programación, que a su vez interactúa con la base de datos para recuperar, almacenar o modificar información. Una vez procesada la solicitud, el servidor genera una respuesta (generalmente en formato HTML, CSS y JavaScript) y la envía de vuelta al cliente, quien se encarga de renderizarla y mostrarla al usuario.

La **descripción del diagrama Cliente-Servidor** para este sistema se visualiza de la siguiente manera:

1. Un **Usuario** (estudiante, docente, administrador) interactúa con la **Interfaz de Usuario** a través de un **Navegador Web** (el cliente).
2. El Navegador Web envía una **Solicitud HTTP/HTTPS** (por ejemplo, GET para obtener datos, POST para enviar datos) a la dirección IP o dominio del servidor.
3. El **Servidor Web** (por ejemplo, Apache o Nginx) recibe la solicitud.
4. El Servidor Web reenvía la solicitud a la **Aplicación Backend** (desarrollada con un framework como Laravel en PHP, Django en Python, o Express en Node.js).
5. La Aplicación Backend procesa la lógica de negocio, que puede incluir:
    - Validación de datos.
    - Autenticación y autorización del usuario.
    - Interacción con el **Sistema de Gestión de Bases de Datos (SGBD)** (por ejemplo, MySQL) para almacenar o recuperar información (proyectos, usuarios, estados).
    - Realización de cálculos o transformaciones de datos.
6. El SGBD devuelve los datos solicitados o confirma la operación a la Aplicación Backend.
7. La Aplicación Backend genera una **Respuesta** (generalmente HTML, CSS y JavaScript) y la envía de vuelta al Servidor Web.

8. El Servidor Web envía la Respuesta HTTP/HTTPS al Navegador Web del usuario.
9. El Navegador Web renderiza la Respuesta, mostrando la información actualizada o la nueva interfaz al Usuario.

Esta separación de responsabilidades entre el cliente y el servidor no solo simplifica el desarrollo y el mantenimiento, sino que también permite la escalabilidad independiente de cada componente. Por ejemplo, se pueden añadir más servidores de aplicación o bases de datos sin afectar directamente la lógica del cliente, y viceversa. Además, fomenta la seguridad al centralizar la lógica de negocio y el acceso a los datos en un entorno controlado, facilitando la implementación de políticas de seguridad robustas.

5.3 Selección del entorno del sistema

La selección meticulosa del entorno del sistema es un pilar fundamental para el éxito en el desarrollo de cualquier aplicación, más aún cuando se trata de un sistema web destinado a la gestión de procesos académicos críticos. Esta decisión abarca la elección de lenguajes de programación, frameworks, sistemas de gestión de bases de datos, y otras herramientas auxiliares que, en conjunto, conformarán el ecosistema tecnológico sobre el cual se construirá la solución. La coherencia con los objetivos del proyecto, la experiencia del equipo, la disponibilidad de soporte y la proyección a futuro son factores determinantes en este proceso.

Para el desarrollo del sistema web de gestión de proyectos académicos, se ha optado por un stack tecnológico robusto y ampliamente adoptado en la industria, lo que garantiza no solo la eficiencia en el desarrollo sino también la escalabilidad y el mantenimiento a largo plazo. La elección recae en las siguientes herramientas y tecnologías:

- **Lenguaje de Programación Backend: PHP**. PHP se ha seleccionado por su madurez, su vasta comunidad de desarrolladores, y su excepcional rendimiento en entornos web. Es un lenguaje interpretado que permite un ciclo de desarrollo rápido y cuenta con una gran cantidad de recursos y librerías. Su compatibilidad con la mayoría de los servidores web y sistemas operativos lo convierte en una opción versátil y de bajo costo.
- **Framework Backend: Laravel**. Como complemento a PHP, se utilizará Laravel. Este framework MVC (Modelo-Vista-Controlador) es reconocido por su sintaxis elegante y expresiva, su potente ORM (Eloquent), y su conjunto de herramientas integradas que simplifican tareas comunes como la autenticación, el enrutamiento, las migraciones de bases de datos y la gestión de colas. Laravel acelera significativamente el proceso de desarrollo, promueve las

buenas prácticas de programación y facilita la creación de aplicaciones web escalables y robustas. Proporciona una estructura clara que ayuda a mantener el código organizado y fácil de mantener.

- **Lenguaje de Programación Frontend: JavaScript**. JavaScript es el lenguaje estándar para el desarrollo web del lado del cliente. Su capacidad para crear interfaces de usuario dinámicas e interactivas es indispensable para un sistema moderno.
- **Framework Frontend: Vue.js**. Vue.js ha sido elegido por su curva de aprendizaje suave, su rendimiento optimizado y su enfoque en la reactividad. Permite construir componentes de interfaz de usuario reutilizables y gestionar el estado de la aplicación de manera eficiente. Su integración con Laravel a través de Laravel Mix o Inertia.js es fluida, permitiendo una experiencia de desarrollo unificada y el aprovechamiento de las capacidades de ambos frameworks para crear una interfaz de usuario rica y responsiva que mejore la experiencia del usuario final.
- **Sistema de Gestión de Bases de Datos (SGBD): MySQL**. MySQL es uno de los SGBD relacionales más populares del mundo, conocido por su fiabilidad, rendimiento y facilidad de uso. Es una solución de código abierto que se integra perfectamente con PHP y Laravel, ofreciendo una gestión eficiente de los datos estructurados del sistema (usuarios, proyectos, estados, comentarios, etc.). Su robustez garantiza la integridad y consistencia de la información académica, lo cual es crítico para un sistema de gestión.
- **Servidor Web: Apache o Nginx**. Se utilizará uno de estos servidores web, ambos líderes en la industria. Son altamente configurables, eficientes en la entrega de contenido estático y dinámico, y ofrecen un excelente rendimiento para aplicaciones PHP. La elección final entre Apache y Nginx dependerá de las especificaciones del entorno de despliegue y las preferencias de configuración para optimizar el rendimiento.
- **Entorno de Desarrollo Integrado (IDE): Visual Studio Code**. Como herramienta principal para la codificación, Visual Studio Code (VS Code) es un IDE ligero pero potente, con un amplio ecosistema de extensiones que soportan PHP, JavaScript, Vue.js y SQL. Facilita la depuración, el control de versiones y la colaboración en equipo, mejorando la productividad del desarrollador.
- **Control de Versiones: Git con GitHub/GitLab**. Para la gestión del código fuente y la colaboración en equipo, se empleará Git. GitHub o GitLab servirán como repositorios remotos, permitiendo un control de versiones robusto, la

gestión de ramas para el desarrollo paralelo y la integración continua. Esto asegura la trazabilidad de los cambios y facilita la recuperación ante errores.

Esta selección de herramientas y tecnologías representa un equilibrio entre la modernidad, la estabilidad y la productividad, creando un entorno de desarrollo propicio para construir un sistema web de alta calidad que cumpla con los requisitos funcionales y no funcionales establecidos en las fases previas de la tesis.

5.4 Metodología para el desarrollo

La elección de una metodología de desarrollo de software es un factor crítico que determina la eficiencia, la calidad y el éxito general de un proyecto. Dada la complejidad intrínseca de los sistemas de información y la necesidad de adaptabilidad en un entorno académico en evolución, la metodología seleccionada debe ser capaz de gestionar los cambios de requisitos, fomentar la colaboración y asegurar la entrega incremental de valor. Es fundamental categorizar y distinguir entre las metodologías aplicables a sistemas educativos como un todo (que pueden incluir currículos, pedagogías) y las específicamente diseñadas para el desarrollo de sistemas de información, como el que nos ocupa.

Las **metodologías para sistemas educativos** a menudo se centran en el diseño instruccional, la creación de contenidos, la evaluación de aprendizaje y la implementación de programas educativos. Modelos como ADDIE (Análisis, Diseño, Desarrollo, Implementación, Evaluación) o el Diseño Universal para el Aprendizaje (DUA) son ejemplos de enfoques que guían la creación de experiencias de aprendizaje. Si bien estos modelos son cruciales para el ámbito pedagógico, no son directamente aplicables al proceso de ingeniería de software que implica la construcción de una aplicación informática. La confusión entre ambos tipos de metodologías puede llevar a una aplicación incorrecta de principios y prácticas, comprometiendo la eficiencia del desarrollo del software.

Por otro lado, las **metodologías para el desarrollo de sistemas de información** están específicamente diseñadas para guiar el ciclo de vida del software, desde la concepción y el análisis de requisitos hasta la implementación, las pruebas y el mantenimiento. Tradicionalmente, se han utilizado enfoques en cascada (Waterfall), que son secuenciales y lineales, donde cada fase debe completarse antes de pasar a la siguiente. Sin embargo, para proyectos con requisitos que pueden evolucionar y donde la retroalimentación temprana es valiosa, las metodologías ágiles han demostrado ser más efectivas.

Para el "Diseño e Implementación de un Sistema Web para la Gestión del Proceso de Aceptación de Proyectos Académicos", se ha decidido adoptar la **metodología Scrum**, un framework ágil de desarrollo de software. Scrum es una metodología iterativa e incremental

que se enfoca en la entrega de valor de manera regular y en la adaptación continua a los cambios. Su selección se justifica por las siguientes razones:

1. **Flexibilidad y Adaptabilidad**: Los requisitos de un sistema académico pueden evolucionar a medida que los usuarios interactúan con prototipos o a medida que cambian las políticas institucionales. Scrum permite incorporar estos cambios de manera eficiente a través de sus ciclos de desarrollo cortos, llamados "Sprints".
2. **Entrega Incremental de Valor**: En lugar de esperar hasta el final del proyecto para entregar una versión completa, Scrum permite la entrega de incrementos de software funcionales al final de cada Sprint. Esto proporciona retroalimentación temprana de los stakeholders y permite al equipo validar supuestos y ajustar el rumbo si es necesario, lo que es vital para asegurar que el sistema final satisfaga las necesidades reales de los usuarios.
3. **Colaboración y Transparencia**: Scrum promueve una fuerte colaboración entre el equipo de desarrollo, el Product Owner (representante de los stakeholders) y el Scrum Master. Las reuniones diarias de Scrum (Daily Scrums), las revisiones de Sprint (Sprint Reviews) y las retrospectivas de Sprint (Sprint Retrospectives) aseguran una comunicación constante y una transparencia total sobre el progreso y los desafíos.
4. **Gestión de Riesgos**: Al trabajar en ciclos cortos y enfocarse en la entrega de las funcionalidades de mayor prioridad, Scrum ayuda a identificar y mitigar los riesgos de manera temprana. Los problemas se detectan y abordan rápidamente, reduciendo la probabilidad de fallos mayores al final del proyecto.
5. **Enfoque en el Usuario**: La participación activa del Product Owner, quien representa la voz del cliente, asegura que el equipo esté siempre enfocado en construir las características que aporten el mayor valor a los usuarios finales del sistema. Esto se alinea directamente con el objetivo de optimizar el proceso de aceptación de proyectos académicos para estudiantes y docentes.

La implementación de Scrum implicará la definición de un Product Backlog, que es una lista priorizada de todas las funcionalidades y mejoras deseadas para el sistema. Cada Sprint tendrá una duración fija (generalmente de 2 a 4 semanas) y comenzará con una reunión de planificación de Sprint, donde el equipo seleccionará un subconjunto de elementos del Product Backlog para trabajar durante ese ciclo. Al final de cada Sprint, se realizará una revisión para demostrar el incremento de software funcional a los stakeholders y una retrospectiva para que el equipo reflexione sobre su desempeño y

busque mejoras continuas en su proceso. Este enfoque garantiza que el desarrollo del sistema web sea ágil, adaptativo y centrado en la entrega de una solución de alta calidad que aborde eficazmente la problemática identificada.

ÍNDICE GENERAL

6. DESARROLLO DEL SISTEMA DE INFORMACIÓN

Habiendo establecido un sólido **Marco Teórico** que fundamenta las bases conceptuales y tecnológicas, y tras un exhaustivo **Diagnóstico Situacional** que reveló las deficiencias inherentes al proceso manual de aceptación de proyectos académicos, la presente sección se adentra en el corazón de la propuesta: el **Diseño y Desarrollo del Sistema Web**. Este capítulo representa la materialización de los objetivos planteados en la introducción, transformando las necesidades identificadas en una solución tecnológica concreta. Se detallará el proceso iterativo y sistemático que condujo a la construcción del sistema, abarcando desde la minuciosa recolección y análisis de requerimientos hasta la implementación final de cada módulo y funcionalidad.

La coherencia con los capítulos precedentes es fundamental. El sistema web propuesto se erige como la respuesta directa a la problemática de la gestión ineficiente, descentralizada y propensa a errores que fue descrita en el diagnóstico. Las tecnologías y metodologías discutidas en el marco teórico, como los **sistemas web, la gestión de proyectos académicos y las metodologías ágiles de desarrollo de software**, servirán como pilares y guías para cada fase del desarrollo. El objetivo principal de esta sección es demostrar cómo, a través de un proceso estructurado de ingeniería de software, se logró concebir y construir una plataforma robusta, intuitiva y eficaz para optimizar la gestión del proceso de aceptación de proyectos en la carrera de Ingeniería en Informática, garantizando transparencia, trazabilidad y eficiencia.

6.1 Análisis y Definición de Requerimientos

La fase de análisis y definición de requerimientos constituye la piedra angular de cualquier proyecto de desarrollo de software exitoso. En esta etapa, se recopila, se documenta y se valida de manera exhaustiva las expectativas y necesidades de los usuarios y de las partes interesadas del sistema. Para el "Sistema Web para la Gestión del Proceso de Aceptación de Proyectos Académicos", este proceso implicó una interacción constante con estudiantes, docentes, coordinadores de carrera y personal administrativo. Se emplearon diversas técnicas, como entrevistas semiestructuradas, encuestas y observación directa de los flujos de trabajo actuales, para obtener una comprensión profunda del dominio del problema y de las funcionalidades deseadas. La información recabada permitió establecer una base sólida para el diseño y la implementación, asegurando que el producto final no solo fuera funcional, sino que también respondiera de manera precisa a las exigencias operativas y estratégicas de la institución. La claridad en la definición de estos requerimientos fue

crucial para evitar desviaciones en etapas posteriores y garantizar la pertinencia y utilidad del sistema propuesto, alineándose directamente con los objetivos específicos de automatización y optimización.

6.1.1 Descripción

El Sistema Web para la Gestión del Proceso de Aceptación de Proyectos Académicos es una plataforma digital integral diseñada para centralizar, automatizar y optimizar todas las etapas involucradas en la presentación, evaluación y aprobación de propuestas de proyectos de investigación o tesis dentro de la carrera de Ingeniería en Informática. Su propósito fundamental es erradicar las ineficiencias, la falta de transparencia y los errores derivados de la gestión manual y descentralizada que previamente caracterizaban este proceso. La solución propuesta busca establecer un flujo de trabajo digitalizado que guíe a los usuarios a través de cada paso, desde la formulación inicial de la idea del proyecto por parte del estudiante, pasando por las revisiones y comentarios de los tutores y el comité evaluador, hasta la notificación final de aceptación o rechazo. Este sistema está concebido para ser el punto único de interacción para todos los actores involucrados, proporcionando herramientas específicas para cada rol. Los estudiantes podrán cargar sus propuestas, realizar un seguimiento del estado de sus envíos y recibir retroalimentación oportuna. Los docentes y tutores tendrán la capacidad de revisar, comentar y aprobar o rechazar proyectos, así como gestionar sus asignaciones. Los administradores de la carrera podrán supervisar el progreso general, generar informes y mantener un registro histórico completo de todos los proyectos. La interfaz del sistema se ha diseñado para ser intuitiva y fácil de usar, minimizando la curva de aprendizaje y fomentando una adopción rápida por parte de la comunidad académica. La seguridad de la información y la integridad de los datos son pilares fundamentales, garantizando que el proceso de aceptación de proyectos sea justo, transparente y eficiente, contribuyendo así a elevar la calidad académica y la gestión administrativa de la institución. En esencia, el sistema transforma un proceso complejo y burocrático en una experiencia fluida y digitalmente mediada, alineada con las mejores prácticas de gestión de proyectos y desarrollo de software.

6.1.2 Requerimientos Funcionales del Proyecto

Los requerimientos funcionales definen las capacidades específicas que el sistema debe poseer para cumplir con los objetivos del proyecto y satisfacer las necesidades de los usuarios. Estos han sido cuidadosamente identificados y detallados para asegurar que el sistema web aborde la problemática actual y ofrezca una solución completa. A continuación, se presenta una lista detallada de los requerimientos funcionales clave:

- **RF1: Gestión de Usuarios y Roles:** El sistema debe permitir la creación, edición, eliminación y gestión de diferentes tipos de usuarios (estudiantes, docentes/tutores, coordinadores de carrera/administradores), asignando roles específicos con permisos diferenciados. Cada rol tendrá acceso a funcionalidades y vistas de información acordes a sus responsabilidades. Por ejemplo, un estudiante no podrá aprobar un proyecto, mientras que un coordinador sí podrá visualizar y gestionar todos los proyectos y usuarios. La autenticación y autorización serán robustas para proteger la integridad del sistema y la confidencialidad de los datos.

  Este requerimiento es fundamental para asegurar la seguridad y la correcta segregación de funciones dentro del sistema. La capacidad de definir roles y permisos granularmente permite controlar qué acciones puede realizar cada tipo de usuario, evitando accesos no autorizados a información sensible o la ejecución de operaciones críticas por parte de usuarios no calificados. Se implementará un módulo de administración de usuarios que permitirá a los coordinadores de carrera crear nuevas cuentas, asignar roles iniciales y modificar permisos según sea necesario, garantizando flexibilidad y control sobre el ecosistema de usuarios del sistema.

- **RF2: Registro y Autenticación de Usuarios:** El sistema debe permitir a los usuarios registrarse y autenticarse de forma segura mediante credenciales (nombre de usuario/correo electrónico y contraseña). Se implementará un mecanismo de recuperación de contraseña y gestión de perfiles.

  La facilidad y seguridad en el acceso son cruciales para la adopción del sistema. El proceso de registro incluirá validación de datos para asegurar la veracidad de la información de los usuarios, especialmente en el caso de estudiantes y docentes, quienes podrían estar vinculados a registros institucionales preexistentes. La autenticación implementará medidas de seguridad estándar de la industria, como el cifrado de contraseñas y la protección contra ataques de fuerza bruta, para salvaguardar las credenciales de acceso de los usuarios y prevenir accesos no autorizados a sus cuentas.

- **RF3: Gestión de Propuestas de Proyectos:** Los estudiantes deben poder crear, editar, subir documentos adjuntos (ej., formato de propuesta, cronograma, presupuesto) y enviar sus propuestas de proyectos de tesis. El sistema debe validar la información ingresada antes del envío.

Este es el núcleo del sistema desde la perspectiva del estudiante. Se proporcionará un formulario intuitivo que guiará al estudiante en la carga de todos los datos necesarios para una propuesta de proyecto, incluyendo título, resumen, objetivos, metodología, y bibliografía preliminar. La funcionalidad de carga de archivos adjuntos soportará múltiples formatos (PDF, DOCX) y permitirá al estudiante adjuntar documentos de soporte cruciales para la evaluación. Las validaciones en tiempo real y al momento del envío asegurarán que la información esté completa y cumpla con los requisitos formales antes de ser sometida a revisión, minimizando errores y retrabajos.

- **RF4: Asignación de Tutores:** Los coordinadores de carrera deben poder asignar uno o varios tutores a una propuesta de proyecto específica. Los tutores deben ser notificados de su asignación.

  La correcta asignación de tutores es vital para el acompañamiento académico del estudiante. El sistema ofrecerá una interfaz donde el coordinador podrá visualizar las propuestas pendientes de asignación y seleccionar tutores de una lista de docentes disponibles, considerando sus áreas de especialización. Una vez asignado, el sistema generará una notificación automática al docente, informándole sobre la nueva propuesta bajo su tutela, lo que agiliza el inicio del proceso de acompañamiento y revisión.

- **RF5: Revisión y Retroalimentación de Propuestas:** Los tutores y miembros del comité evaluador deben poder acceder a las propuestas asignadas, revisarlas, agregar comentarios, sugerencias y calificar diferentes aspectos de la propuesta.

  Esta funcionalidad es esencial para el proceso de mejora continua de las propuestas. El sistema ofrecerá una interfaz de revisión que permitirá a los tutores y evaluadores acceder a la propuesta completa y a los documentos adjuntos. Podrán realizar comentarios específicos sobre secciones del documento o generar un informe de retroalimentación consolidado. La calificación se basará en criterios predefinidos (ej., claridad de objetivos, viabilidad, originalidad), lo que proporciona una evaluación estandarizada y objetiva. Todos los comentarios y calificaciones quedarán registrados y vinculados a la versión de la propuesta para mantener la trazabilidad.

- **RF6: Gestión de Estados de la Propuesta:** El sistema debe permitir cambiar el estado de una propuesta (ej., 'Enviada', 'En Revisión', 'Aprobada con Observaciones', 'Aprobada', 'Rechazada'). Los cambios de estado deben ser

registrados con fecha y usuario.

La trazabilidad del proceso es un requerimiento clave para la transparencia. Cada propuesta tendrá un ciclo de vida bien definido dentro del sistema, y cada cambio de estado reflejará el progreso actual del proyecto. Los roles con permisos adecuados (tutores, comité, administradores) podrán actualizar el estado. El registro de cada cambio de estado, incluyendo quién lo realizó y cuándo, permitirá auditar el proceso y proporcionar a los estudiantes una visión clara de dónde se encuentra su propuesta en todo momento.

- **RF7: Notificaciones Automáticas:** El sistema debe enviar notificaciones automáticas por correo electrónico a los usuarios relevantes ante eventos importantes (ej., propuesta enviada, asignación de tutor, comentarios recibidos, cambio de estado de la propuesta).

  Las notificaciones son cruciales para mantener a todos los actores informados y comprometidos con el proceso. El sistema configurará alertas automáticas que se dispararán ante acciones clave, como el envío de una nueva propuesta, la asignación de un tutor, la publicación de comentarios o el cambio de estado de un proyecto. Estas notificaciones por correo electrónico asegurarán que nadie se pierda una actualización importante, facilitando la comunicación y reduciendo los tiempos de espera inherentes a la comunicación manual.

- **RF8: Historial de Versiones y Comentarios:** El sistema debe mantener un historial de las diferentes versiones de la propuesta y de todos los comentarios y retroalimentaciones recibidas en cada etapa.

  Para la gestión académica y la resolución de disputas, es vital tener un registro completo de la evolución de cada propuesta. Esta funcionalidad permitirá a los estudiantes ver cómo su propuesta ha mejorado a lo largo del tiempo y a los evaluadores revisar el historial de cambios. Cada nueva versión de una propuesta cargada por el estudiante se guardará como una revisión, y todos los comentarios de tutores y evaluadores quedarán asociados a la versión específica sobre la que se realizaron, proporcionando una auditoría completa del proceso de desarrollo de la propuesta.

- **RF9: Generación de Informes:** Los coordinadores de carrera deben poder generar informes sobre el estado de las propuestas, estadísticas de aprobación/rechazo, tiempos de revisión, y desempeño de tutores.

La capacidad de generar informes es esencial para la toma de decisiones estratégicas y la mejora continua del proceso. El sistema ofrecerá una variedad de reportes personalizables que permitirán a los administradores obtener una visión global del proceso de gestión de proyectos. Se podrán generar informes sobre el número de propuestas activas, el promedio de tiempo de revisión, las tasas de aprobación por área temática o por tutor, entre otros. Estos informes se podrán exportar a formatos comunes como PDF o CSV para su posterior análisis y presentación.

- **RF10: Búsqueda y Filtrado de Proyectos:** El sistema debe permitir buscar y filtrar proyectos por diversos criterios (ej., título, autor, tutor, estado, área temática, fecha de envío).

  Para manejar un volumen creciente de proyectos, la funcionalidad de búsqueda y filtrado es indispensable. Los usuarios con los permisos adecuados podrán localizar rápidamente proyectos específicos utilizando palabras clave en el título o resumen, o aplicando filtros por el nombre del estudiante o tutor, el estado actual de la propuesta, la fecha de envío o la disciplina académica. Esto mejora significativamente la usabilidad y la eficiencia en la gestión de un gran número de proyectos.

- **RF11: Panel de Control Personalizado:** Cada tipo de usuario (estudiante, tutor, coordinador) debe tener un panel de control personalizado que muestre información relevante para su rol (ej., estudiante: estado de mis propuestas; tutor: propuestas asignadas para revisión; coordinador: resumen general de todas las propuestas).

  Un panel de control intuitivo y personalizado mejora la experiencia del usuario y la eficiencia operativa. El panel del estudiante mostrará de manera prominente el estado de sus envíos, las notificaciones pendientes y los plazos importantes. El panel del tutor destacará las propuestas que requieren su atención, los comentarios pendientes y las fechas límite. El panel del coordinador ofrecerá una visión de alto nivel con estadísticas clave, proyectos en riesgo y una visión general del progreso de todos los proyectos en el sistema, permitiendo una gestión proactiva.

- **RF12: Gestión de Áreas Temáticas:** El sistema debe permitir a los administradores definir y gestionar las áreas temáticas o líneas de investigación disponibles para los proyectos de tesis.

Para organizar y clasificar los proyectos de manera efectiva, es necesario un módulo de gestión de áreas temáticas. Los administradores podrán añadir nuevas áreas, editar las existentes o desactivar aquellas que ya no sean relevantes. Esta categorización ayudará a los estudiantes a seleccionar un área al enviar su propuesta y facilitará a los coordinadores la asignación de tutores con experiencia relevante, así como la generación de informes específicos por área de conocimiento.

- **RF13: Generación de Documentos Oficiales (PDF):** El sistema debe ser capaz de generar documentos oficiales en formato PDF, como cartas de aceptación, informes de evaluación o certificaciones, utilizando plantillas predefinidas y datos del sistema.

  La automatización de la generación de documentos oficiales es un requerimiento crítico para la eficiencia administrativa. El sistema contendrá plantillas para documentos como la carta de aceptación oficial de una propuesta, el informe final de evaluación o un certificado de participación para los tutores. Estos documentos se generarán dinámicamente, rellenando los campos con la información extraída de la base de datos (nombre del estudiante, título del proyecto, fecha de aprobación, nombres de los evaluadores), y se podrán descargar en formato PDF, garantizando un formato uniforme y profesional.

6.1.3 Requerimientos No Funcionales del Proyecto

Los requerimientos no funcionales describen las cualidades del sistema y las restricciones bajo las cuales debe operar. Aunque no definen qué hace el sistema, son cruciales para determinar cómo lo hace, afectando directamente la experiencia del usuario, la eficiencia y la sostenibilidad a largo plazo de la solución. Su cumplimiento garantiza un sistema robusto, seguro y escalable. A continuación, se detallan los requerimientos no funcionales identificados:

- **RNF1: Usabilidad:** El sistema debe ser intuitivo y fácil de usar, con una interfaz de usuario clara y consistente. El tiempo de aprendizaje para un usuario nuevo debe ser mínimo, y las acciones comunes deben ser accesibles con pocos clics.

  La usabilidad es primordial para la aceptación y adopción del sistema por parte de la comunidad académica. Se emplearán principios de diseño de experiencia de usuario (UX) y de interfaz de usuario (UI) para crear una navegación lógica,

elementos visuales consistentes y mensajes de retroalimentación claros. Los formularios deben ser fáciles de completar, y la información debe presentarse de manera organizada y legible. La minimización del esfuerzo cognitivo del usuario es un objetivo clave para evitar frustraciones y fomentar el uso continuo del sistema.

- **RNF2: Rendimiento:** El sistema debe responder a las solicitudes de los usuarios en un tiempo máximo de 3 segundos para el 90% de las operaciones, incluso bajo una carga concurrente de 50 usuarios.

  Un sistema lento puede generar frustración y disminuir la productividad. Este requerimiento asegura que la plataforma sea ágil y eficiente, incluso en momentos de alta demanda, como al final de un plazo de entrega de propuestas. La optimización de consultas a la base de datos, el uso de técnicas de caché y la eficiencia del código backend serán implementadas para cumplir con este objetivo. El rendimiento se medirá y se monitoreará a lo largo de las fases de prueba para garantizar su cumplimiento.

- **RNF3: Seguridad:** El sistema debe proteger la información sensible de los usuarios y proyectos contra accesos no autorizados, modificaciones o eliminaciones. Debe implementar mecanismos de autenticación robustos, cifrado de contraseñas y validación de entradas para prevenir vulnerabilidades comunes (ej., inyección SQL, XSS).

  La seguridad es un pilar innegociable, dado que el sistema manejará información académica y personal. Se aplicarán las mejores prácticas de seguridad en el desarrollo web, incluyendo el uso de HTTPS para todas las comunicaciones, el almacenamiento seguro de contraseñas mediante funciones hash y salt, y la implementación de controles de acceso basados en roles. Todas las entradas de usuario serán sanitizadas y validadas para prevenir ataques de inyección. Se realizarán auditorías de seguridad y pruebas de penetración para identificar y mitigar posibles vulnerabilidades antes del despliegue en producción.

- **RNF4: Confiabilidad:** El sistema debe estar disponible el 99.5% del tiempo durante las horas de operación definidas (lunes a viernes, 8:00 AM - 6:00 PM). Debe ser capaz de recuperarse de fallas de hardware o software sin pérdida significativa de datos.

La confiabilidad garantiza que el sistema esté operativo cuando los usuarios lo necesiten. Esto implica una arquitectura robusta, mecanismos de respaldo de datos automáticos y un plan de recuperación ante desastres. La infraestructura de alojamiento debe ser redundante, y se implementará un monitoreo constante para detectar y resolver problemas de manera proactiva. La integridad de los datos es crucial, por lo que se priorizarán las transacciones atómicas y las copias de seguridad regulares para minimizar el riesgo de pérdida de información.

- **RNF5: Mantenibilidad:** El código fuente del sistema debe ser modular, bien documentado y fácil de entender para facilitar futuras modificaciones, correcciones de errores y adiciones de nuevas funcionalidades por parte de otros desarrolladores.

  La mantenibilidad es vital para la vida útil del software. Se adoptarán estándares de codificación, se utilizarán patrones de diseño de software (como MVC) y se proporcionará documentación interna del código, incluyendo comentarios explicativos y una estructura de directorios lógica. Esto permitirá que el sistema sea fácilmente adaptable a futuros cambios en los requerimientos o en la tecnología, reduciendo los costos asociados con el mantenimiento y la evolución del sistema a largo plazo.

- **RNF6: Escalabilidad:** El sistema debe ser capaz de soportar un crecimiento futuro en el número de usuarios (hasta 500) y de proyectos (hasta 1000) sin una degradación significativa del rendimiento.

  La escalabilidad asegura que el sistema pueda crecer junto con las necesidades de la institución. La arquitectura del sistema, la elección de la base de datos y la configuración del servidor se diseñarán pensando en la capacidad de expansión. Esto puede implicar el uso de bases de datos optimizadas para grandes volúmenes de datos, la posibilidad de añadir más recursos de servidor (escalado vertical) o de distribuir la carga entre múltiples servidores (escalado horizontal) si fuera necesario en el futuro, garantizando que el sistema pueda manejar un aumento en la demanda sin comprometer su rendimiento.

- **RNF7: Compatibilidad:** El sistema debe ser compatible con los navegadores web modernos más utilizados (Chrome, Firefox, Edge, Safari) y debe ser responsivo, adaptándose a diferentes tamaños de pantalla (escritorio, tabletas, móviles).

La accesibilidad del sistema es un factor clave. Se garantizará que la interfaz de usuario se visualice y funcione correctamente en los principales navegadores web, minimizando las diferencias de renderizado. Además, el diseño responsivo permitirá que el sistema sea utilizable desde cualquier dispositivo, proporcionando una experiencia de usuario consistente y accesible, ya sea que el usuario acceda desde una computadora de escritorio o desde un dispositivo móvil, lo que es esencial para la flexibilidad de uso por parte de estudiantes y docentes.

- **RNF8: Interoperabilidad:** El sistema, en una fase futura, podría requerir integración con otros sistemas institucionales (ej., sistema de gestión académica, sistema de correo electrónico institucional). La arquitectura debe ser diseñada para facilitar estas posibles integraciones.

  Aunque no es un requerimiento inmediato, la previsión de interoperabilidad es importante para la estrategia tecnológica a largo plazo de la universidad. El sistema se diseñará con una arquitectura modular y el uso de APIs bien definidas, lo que facilitaría la comunicación y el intercambio de datos con otras plataformas institucionales, como sistemas de gestión de estudiantes o plataformas de e-learning, si se decide en el futuro. Esto reduce la duplicación de esfuerzos y permite una gestión más unificada de la información.

6.1.4 Restricciones

En todo proyecto de desarrollo de software, las restricciones actúan como límites o condiciones que deben ser consideradas y respetadas durante todo el ciclo de vida del proyecto. Estas pueden ser de diversa índole y tienen un impacto directo en las decisiones de diseño, la selección de tecnologías, la planificación de recursos y el alcance final del sistema. Para el "Sistema Web para la Gestión del Proceso de Aceptación de Proyectos Académicos", se identificaron las siguientes restricciones clave que moldearon el proceso de desarrollo:

- **R1: Restricción de Tiempo:** El sistema debía ser desarrollado e implementado dentro de un período de seis meses, coincidiendo con el calendario académico para su puesta en marcha en el siguiente ciclo.

  Esta restricción impuso la necesidad de adoptar una metodología de desarrollo ágil y de realizar una planificación rigurosa para cada fase del proyecto. La gestión del tiempo fue crítica, requiriendo la priorización de funcionalidades esenciales y la constante evaluación del progreso para asegurar el

cumplimiento de los hitos. Las decisiones de diseño y codificación estuvieron influenciadas por la necesidad de entregar un producto funcional dentro del plazo establecido, lo que implicó un enfoque en la eficiencia y la minimización de complejidades innecesarias.

- **R2: Restricción de Recursos Humanos:** El equipo de desarrollo estuvo limitado a un número específico de personas (el tesista y un asesor de apoyo), lo que influyó en la complejidad de las funcionalidades a implementar y en la distribución de las tareas.

  La disponibilidad limitada de personal de desarrollo significó que el alcance del proyecto debía ser realista y bien definido desde el principio. Se evitaron funcionalidades excesivamente complejas que pudieran extender el tiempo de desarrollo más allá de lo permitido. La comunicación efectiva entre los miembros del equipo y la asignación clara de responsabilidades fueron esenciales para optimizar el uso de los recursos humanos disponibles y mantener la productividad. Esta restricción también reforzó la necesidad de utilizar herramientas y frameworks que aceleraran el desarrollo.

- **R3: Restricción Tecnológica (Infraestructura Existente):** El sistema debía ser compatible con la infraestructura de servidores y bases de datos existente en la universidad, la cual opera principalmente con sistemas Linux, servidores web Apache y bases de datos MySQL.

  Esta restricción fue determinante en la selección del stack tecnológico. Si bien se consideraron diversas opciones, la necesidad de integrar el sistema sin requerir una inversión significativa en nueva infraestructura inclinó la balanza hacia tecnologías como PHP y Laravel, que son ampliamente compatibles con entornos LAMP (Linux, Apache, MySQL, PHP). Esto aseguró una implementación más sencilla y una menor curva de aprendizaje para el personal de TI de la institución, además de optimizar los costos operativos y de mantenimiento a largo plazo.

- **R4: Restricción Presupuestaria:** El presupuesto asignado para el desarrollo y despliegue del sistema era limitado, lo que implicó la selección de tecnologías de código abierto y herramientas gratuitas siempre que fuera posible.

  La limitación presupuestaria fue un factor clave para optar por soluciones de código abierto y frameworks gratuitos. Esto incluyó la elección de sistemas operativos, bases de datos, lenguajes de programación y librerías de terceros

que no incurrieran en costos de licenciamiento. Esta decisión permitió maximizar el valor del presupuesto disponible, concentrando los recursos en el desarrollo de funcionalidades y en la calidad del software, en lugar de en la adquisición de licencias costosas. Además, el uso de software de código abierto facilita la personalización y la adaptabilidad a largo plazo.

- **R5: Restricciones de Políticas Institucionales:** El sistema debía adherirse a las políticas de seguridad de la información y de privacidad de datos establecidas por la universidad.

  Las políticas institucionales de seguridad y privacidad son mandatorias y no negociables. Esto implicó que el diseño del sistema debía incorporar desde sus primeras fases mecanismos robustos para la protección de datos personales y académicos, cumplimiento de la normativa de protección de datos, y adhesión a los protocolos de seguridad de la red institucional. La implementación de roles y permisos, el cifrado de datos sensibles y la auditoría de accesos fueron aspectos críticos para cumplir con esta restricción, asegurando que el sistema operara dentro de los marcos legales y normativos de la universidad.

6.2 Fase de Diseño

La fase de diseño es el puente crucial entre los requerimientos funcionales y no funcionales definidos y la implementación concreta del sistema. En esta etapa, las necesidades abstractas se transforman en una arquitectura detallada y en especificaciones técnicas que guiarán el proceso de codificación. El diseño del Sistema Web para la Gestión del Proceso de Aceptación de Proyectos Académicos se llevó a cabo con una perspectiva centrada en la usabilidad, la modularidad, la escalabilidad y la seguridad, siguiendo los principios de la ingeniería de software. Se emplearon diversas herramientas y notaciones estándar de la industria, como los diagramas de la Lenguaje Unificado de Modelado (UML), para visualizar la estructura, el comportamiento y las interacciones del sistema antes de escribir una sola línea de código. Este enfoque permitió identificar y resolver posibles problemas de diseño en una etapa temprana, reduciendo costos y riesgos asociados a cambios posteriores. La arquitectura se basó en el patrón **Modelo-Vista-Controlador (MVC)**, que promueve la separación de intereses y facilita el mantenimiento y la extensibilidad del sistema. A continuación, se detallan los elementos clave de la fase de diseño.

El proceso de diseño comenzó con la elaboración de **diagramas de casos de uso**, los cuales son esenciales para modelar las interacciones de los usuarios con el sistema. Estos diagramas no solo identifican los diferentes actores (o "avatares", como estudiantes, tutores, coordinadores) y sus objetivos, sino que también describen las funcionalidades principales

que el sistema debe ofrecer desde la perspectiva del usuario. Para el sistema de gestión de proyectos, se definieron casos de uso como "Gestionar Propuesta de Proyecto", "Revisar Propuesta", "Asignar Tutor", "Generar Informe", entre otros. Cada caso de uso fue acompañado de una descripción detallada que especificaba los actores involucrados, las precondiciones, el flujo normal de eventos, los flujos alternativos y las postcondiciones, proporcionando una comprensión clara de la funcionalidad esperada.

Posteriormente, se diseñaron los **diagramas de procesos del sistema propuesto**, que ilustran el flujo de trabajo secuencial y las decisiones lógicas que ocurren dentro del sistema para ejecutar una funcionalidad específica. Estos diagramas, a menudo representados como diagramas de actividad o de flujo, fueron fundamentales para visualizar cómo la información se mueve a través del sistema y cómo interactúan los diferentes componentes. Por ejemplo, el diagrama de procesos para la "Aceptación de una Propuesta" detalla los pasos desde el envío por parte del estudiante, la asignación al tutor, las rondas de revisión y comentarios, hasta la decisión final del coordinador y la notificación al estudiante, incluyendo los puntos de decisión y las posibles derivaciones del flujo principal. Esto permitió optimizar los flujos de trabajo y eliminar cuellos de botella identificados en el diagnóstico situacional.

Un componente crítico del diseño fue la creación del **diagrama entidad-relación (ERD)** de la base de datos. Este diagrama es una representación gráfica de la estructura lógica de la base de datos, mostrando las entidades (tablas) del sistema, sus atributos (campos) y las relaciones entre ellas. Para el sistema de gestión de proyectos, las entidades clave incluyeron "Usuario" (con roles diferenciados), "Proyecto" (con sus atributos como título, resumen, estado), "DocumentoAdjunto", "Comentario", "TutorAsignado" y "ÁreaTemática". El ERD definió la cardinalidad de las relaciones (uno a uno, uno a muchos, muchos a muchos) y las claves primarias y foráneas, asegurando la integridad referencial y la eficiencia en el almacenamiento y recuperación de datos. Un diseño de base de datos robusto es esencial para la escalabilidad y el rendimiento del sistema, garantizando que la información esté organizada de manera lógica y accesible.

Finalmente, la fase de diseño incluyó la conceptualización de las interfaces de usuario. Aunque la implementación visual detallada corresponde a la fase de codificación, en el diseño se crearon maquetas (wireframes) y prototipos de baja fidelidad para visualizar la disposición de los elementos, la navegación y la experiencia general del usuario. A continuación, se describen conceptualmente cuatro capturas de interfaces clave que ilustran la interacción con el sistema:

1. **Captura de Interfaz 1: Panel de Control del Estudiante - "Mis Propuestas"**

Esta captura de pantalla muestra el panel principal al que accede un estudiante después de iniciar sesión. En la parte superior, se visualiza una barra de navegación con opciones como "Inicio", "Mis Propuestas", "Enviar Nueva Propuesta" y "Perfil". El área central de la pantalla está dominada por una tabla que lista todas las propuestas de proyectos que el estudiante ha enviado. Cada fila de la tabla representa una propuesta y contiene columnas como "Título del Proyecto", "Fecha de Envío", "Tutor Asignado", "Estado Actual" y "Acciones". El "Estado Actual" se resalta con etiquetas de color (ej., verde para 'Aprobada', amarillo para 'En Revisión', rojo para 'Rechazada'), proporcionando una rápida visualización del progreso. La columna "Acciones" incluye botones como "Ver Detalles", "Editar" (si la propuesta aún no ha sido revisada o está en estado de borrador) y "Subir Nueva Versión" (si se han solicitado revisiones). Adicionalmente, en la parte superior de la tabla, hay un pequeño resumen que indica el número total de propuestas, cuántas están en revisión y cuántas han sido aprobadas. Esta interfaz está diseñada para ser la central de gestión personal del estudiante, ofreciendo una visión clara y concisa de todas sus interacciones con el sistema y permitiéndole realizar acciones directas sobre sus proyectos.

2. **Captura de Interfaz 2: Formulario de Envío de Propuesta**

Esta captura muestra el formulario que un estudiante utiliza para enviar una nueva propuesta de proyecto. La interfaz es limpia y estructurada, dividida en secciones lógicas para facilitar el ingreso de información. Incluye campos de texto obligatorios para el "Título del Proyecto", "Resumen" (con un editor de texto enriquecido para formato), "Objetivo General", "Objetivos Específicos" (con la opción de añadir múltiples objetivos), "Metodología" y "Bibliografía Preliminar". Hay un campo de selección desplegable para elegir el "Área Temática" del proyecto de una lista predefinida. Una sección destacada permite la "Carga de Documentos Adjuntos", con un botón para seleccionar archivos y una lista que muestra los archivos ya cargados (ej., "Propuesta_v1.pdf", "Cronograma.docx"). Cada campo tiene validaciones en tiempo real que muestran mensajes de error si la información es incorrecta o faltante. Al final del formulario, se encuentran botones para "Guardar Borrador", "Previsualizar" y "Enviar Propuesta". El diseño fomenta la completitud y la precisión de la información antes del envío final, guiando al estudiante paso a paso a través del proceso.

3. **Captura de Interfaz 3: Vista de Revisión de Propuesta para Tutor/Evaluador**

   Esta captura ilustra la interfaz que un tutor o miembro del comité evaluador utiliza para revisar una propuesta asignada. La pantalla está dividida en dos paneles principales. El panel izquierdo muestra el contenido de la propuesta de proyecto (título, resumen, objetivos, etc.) y los documentos adjuntos en un visor embebido, permitiendo al evaluador leer el documento directamente sin salir del sistema. El panel derecho contiene la sección de "Comentarios y Evaluación". Aquí, el tutor puede escribir comentarios generales en un cuadro de texto amplio y, más abajo, se presentan campos de calificación para criterios específicos (ej., "Claridad de Objetivos", "Viabilidad", "Originalidad", "Redacción") con escalas de valoración (ej., 'Excelente', 'Bueno', 'Regular', 'Deficiente'). También hay un campo para "Observaciones Adicionales" y un selector para cambiar el "Estado de la Propuesta" (ej., 'Aprobada con Observaciones', 'Rechazada', 'Solicitar Modificaciones'). Un botón de "Guardar Evaluación y Enviar Retroalimentación" finaliza el proceso. Esta interfaz está diseñada para ofrecer todas las herramientas necesarias para una evaluación exhaustiva y estructurada, facilitando la provisión de retroalimentación constructiva al estudiante.

4. **Captura de Interfaz 4: Panel de Control del Coordinador - "Gestión de Proyectos"**

   Esta captura muestra la vista principal del coordinador de carrera, orientada a la supervisión y gestión general de todas las propuestas. La interfaz presenta un "Dashboard" con widgets informativos en la parte superior, mostrando estadísticas clave como el "Total de Propuestas", "Propuestas en Revisión", "Propuestas Aprobadas" y "Tasa de Aprobación". Debajo de estos widgets, hay una tabla principal que lista todas las propuestas activas en el sistema, similar a la del estudiante pero con más información y opciones. Las columnas incluyen "Título del Proyecto", "Autor", "Tutor Asignado", "Estado", "Fecha de Envío", "Última Actualización" y "Acciones". La columna "Acciones" ofrece botones para "Ver Detalles", "Asignar/Cambiar Tutor", "Cambiar Estado" y "Generar Informe" (específico para ese proyecto). Se incluye una barra de búsqueda y filtros avanzados (por estado, tutor, área temática, fecha) para facilitar la localización de proyectos específicos. En el lateral izquierdo, un menú de navegación permite acceder a "Gestión de Usuarios", "Gestión de Áreas

Temáticas" y "Generación de Informes Globales". Esta interfaz proporciona al coordinador una visión integral y herramientas de gestión potentes para supervisar el progreso académico y administrativo de la carrera.

6.3 Fase de Codificación

La fase de codificación, también conocida como implementación, es donde el diseño conceptual y arquitectónico se traduce en código fuente ejecutable. Este es el momento en que las especificaciones detalladas de los requerimientos y el diseño se materializan en un sistema funcional. Para el Sistema Web para la Gestión del Proceso de Aceptación de Proyectos Académicos, se adoptó un enfoque de desarrollo basado en componentes, utilizando un stack tecnológico moderno y robusto que fue previamente justificado en el Marco Teórico. La elección de tecnologías como **PHP con el framework Laravel** para el backend, **MySQL** como sistema de gestión de bases de datos, y una combinación de **HTML, CSS y JavaScript** con el uso de librerías para el frontend, permitió construir una aplicación web escalable, segura y mantenible. Durante esta fase, se hizo hincapié en la escritura de código limpio, modular y bien documentado, siguiendo las mejores prácticas de la ingeniería de software para facilitar el mantenimiento futuro y la colaboración. La implementación se llevó a cabo de manera iterativa, permitiendo la construcción incremental de funcionalidades y la integración continua, lo cual es característico de las metodologías ágiles. Cada módulo fue desarrollado, probado unitariamente e integrado progresivamente para construir el sistema completo.

6.3.1 Requerimientos de Desarrollo

Para llevar a cabo la fase de codificación de manera eficiente y efectiva, se establecieron una serie de requerimientos y herramientas de desarrollo. Estos elementos fueron seleccionados para optimizar el proceso, garantizar la calidad del código y facilitar la colaboración. La infraestructura de desarrollo se configuró para replicar, en la medida de lo posible, el entorno de producción, minimizando así los problemas de compatibilidad durante el despliegue. Los requerimientos de desarrollo clave incluyeron:

- **Entorno de Desarrollo Integrado (IDE):** Se utilizó **Visual Studio Code**, un IDE ligero pero potente, que ofrece soporte para múltiples lenguajes de programación, depuración, integración con control de versiones y una amplia gama de extensiones que mejoran la productividad del desarrollador.

  Visual Studio Code fue elegido por su versatilidad, su capacidad para trabajar con PHP, JavaScript, HTML y CSS de manera eficiente, y su extenso ecosistema de extensiones. Proporcionó herramientas de autocompletado de

código, resaltado de sintaxis, depuración integrada y terminales para ejecutar comandos, lo que mejoró significativamente la velocidad y la calidad del desarrollo. Su naturaleza multiplataforma también facilitó el trabajo en diferentes sistemas operativos si fuera necesario.

- **Servidor Web Local:** Se utilizó **Apache HTTP Server** configurado a través de **XAMPP** (o Laragon en entornos Windows) para simular el entorno de producción en las máquinas de desarrollo.

  XAMPP proporcionó un paquete completo que incluía Apache, MySQL y PHP, permitiendo configurar un servidor local de manera rápida y sencilla. Esto fue esencial para probar el sistema web en un entorno controlado antes de su despliegue en un servidor real. La capacidad de ejecutar el sistema localmente permitió a los desarrolladores realizar pruebas exhaustivas de cada funcionalidad y módulo sin afectar el entorno de producción, asegurando la estabilidad y el correcto funcionamiento.

- **Sistema de Gestión de Bases de Datos (SGBD) Local:** Se empleó **MySQL** para la gestión de la base de datos local, replicando la base de datos de producción.

  MySQL, siendo la base de datos seleccionada para producción, también se utilizó en el entorno de desarrollo. Esto aseguró la consistencia en el esquema de la base de datos y la compatibilidad de las consultas SQL. Herramientas como phpMyAdmin (incluida en XAMPP) o SQLYog facilitaron la gestión de la base de datos, la creación de tablas, la inserción de datos de prueba y la ejecución de consultas para el desarrollo y la depuración.

- **Lenguaje de Programación Backend: PHP 8.x** fue el lenguaje principal para el desarrollo del lado del servidor.

  PHP 8.x fue seleccionado por su madurez, rendimiento mejorado y amplio soporte en la comunidad. Su integración con el framework Laravel proporcionó una base sólida para el desarrollo del backend, permitiendo la creación de APIs RESTful, la gestión de la lógica de negocio y la interacción con la base de datos de manera eficiente y segura. Las características modernas de PHP 8, como las mejoras en el manejo de errores y las nuevas sintaxis, contribuyeron a un código más limpio y robusto.

- **Framework Backend: Laravel 9.x** fue el framework MVC utilizado para el desarrollo del backend.

Laravel 9.x fue una elección estratégica debido a su robustez, su enfoque en la productividad del desarrollador y su vasta colección de paquetes y herramientas. Proporcionó una estructura organizada para el proyecto, un ORM (Eloquent) para la interacción con la base de datos, un sistema de enrutamiento potente, herramientas de autenticación y autorización, y un motor de plantillas (Blade) para la integración con el frontend. La facilidad para implementar patrones de diseño y las convenciones de Laravel aceleraron significativamente el proceso de desarrollo y mejoraron la mantenibilidad del código.

- **Lenguajes Frontend: HTML5, CSS3 y JavaScript** para la construcción de la interfaz de usuario.

  Estos lenguajes son los pilares de cualquier aplicación web moderna. HTML5 se utilizó para estructurar el contenido, CSS3 para estilizar y dar formato visual a la interfaz, y JavaScript para añadir interactividad y dinamismo. Se hizo uso de CSS preprocesadores (como SASS) y frameworks CSS (como Tailwind CSS o Bootstrap) para acelerar el diseño y asegurar la consistencia visual, así como librerías JavaScript (como jQuery o Vue.js) para la manipulación del DOM y la creación de componentes interactivos.

- **Sistema de Control de Versiones: Git** y **GitHub** para la gestión del código fuente y la colaboración.

  Git fue indispensable para el control de versiones del código fuente, permitiendo a los desarrolladores trabajar en diferentes funcionalidades de forma independiente, fusionar cambios y revertir versiones si fuera necesario. GitHub se utilizó como repositorio remoto para almacenar el código, facilitar la colaboración, realizar revisiones de código y gestionar las tareas de desarrollo a través de issues, garantizando un flujo de trabajo organizado y la seguridad del código.

- **Gestor de Dependencias PHP: Composer** para la gestión de librerías y dependencias de PHP.

  Composer es el gestor de paquetes estándar para PHP y fue fundamental para incluir y gestionar las librerías de terceros necesarias para el proyecto, incluyendo el propio framework Laravel y otras dependencias como librerías

para la generación de PDF o la manipulación de imágenes. Esto aseguró que todas las dependencias estuvieran correctamente instaladas y gestionadas, facilitando la configuración del entorno de desarrollo y el despliegue.

- **Gestor de Dependencias JavaScript: npm** o **Yarn** para la gestión de librerías y dependencias de JavaScript.

  Similar a Composer para PHP, npm (Node Package Manager) o Yarn se utilizaron para gestionar las librerías y herramientas de frontend, como frameworks JavaScript, preprocesadores CSS, y bundlers (como Webpack o Vite) para compilar y optimizar los activos del frontend. Esto permitió mantener actualizadas las dependencias de JavaScript y facilitar el proceso de construcción de los assets estáticos del sistema.

6.3.2 Desarrollo de los Módulos del Sistema de Información

El desarrollo del sistema web se estructuró en módulos, lo que permitió abordar la complejidad del proyecto de manera organizada y facilitar la implementación concurrente de funcionalidades. Cada módulo fue diseñado para ser cohesivo y con un bajo acoplamiento, siguiendo los principios de la arquitectura MVC de Laravel. Esta modularización no solo mejoró la claridad del código, sino que también facilitó las pruebas, el mantenimiento y la futura expansión del sistema. A continuación, se describen los módulos principales y su implementación:

1. **Módulo de Autenticación y Autorización:**

   Este módulo es la puerta de entrada al sistema y garantiza que solo los usuarios autorizados puedan acceder a las funcionalidades correspondientes a su rol. Se implementó utilizando las características de autenticación y autorización de Laravel (Laravel Fortify y Laravel Permissions), que proporcionan un sistema robusto para el registro de usuarios, inicio de sesión, recuperación de contraseña y gestión de sesiones. La autorización se basó en roles (estudiante, tutor, coordinador) y permisos granulares, asegurando que cada usuario solo pudiera realizar las acciones permitidas. Por ejemplo, un estudiante puede crear y ver sus propias propuestas, un tutor puede revisar las propuestas asignadas, y un coordinador tiene acceso a todas las funciones administrativas. La seguridad de las contraseñas se manejó con hashing bcrypt, y se implementaron middleware para proteger las rutas y acciones críticas del sistema.

2. **Módulo de Gestión de Usuarios y Roles:**

Complementario al módulo de autenticación, este módulo permite a los administradores (coordinadores) gestionar las cuentas de usuario. Incluye funcionalidades para la creación de nuevas cuentas, edición de perfiles existentes, asignación y modificación de roles, y desactivación de usuarios. La interfaz de administración ofrece una tabla paginada con filtros de búsqueda para gestionar eficientemente un gran número de usuarios. Cada operación de gestión de usuarios registra un log de auditoría para mantener la trazabilidad de los cambios realizados por los administradores. Este módulo es crítico para mantener la base de usuarios actualizada y con los permisos correctos.

3. **Módulo de Gestión de Propuestas de Proyectos:**

   Este es el módulo central del sistema, que permite a los estudiantes crear, editar y enviar sus propuestas. Se desarrolló un formulario web dinámico que guía al estudiante a través del proceso de carga de información (título, resumen, objetivos, metodología, bibliografía). La carga de archivos adjuntos (PDF, DOCX) se implementó utilizando librerías de manejo de archivos de Laravel, asegurando el almacenamiento seguro en el servidor. El sistema gestiona las diferentes versiones de la propuesta, permitiendo a los estudiantes subir nuevas revisiones. Para los tutores y coordinadores, este módulo proporciona vistas detalladas de cada propuesta, incluyendo el historial de versiones, los comentarios recibidos y el estado actual, facilitando la revisión y el seguimiento.

4. **Módulo de Revisión y Retroalimentación:**

   Desarrollado específicamente para tutores y evaluadores, este módulo proporciona las herramientas para evaluar las propuestas. La interfaz de revisión permite visualizar el contenido de la propuesta y los documentos adjuntos, así como añadir comentarios específicos y calificaciones basadas en criterios predefinidos. Los comentarios se asocian a una versión específica de la propuesta y son visibles para el estudiante una vez que se envía la retroalimentación. Este módulo también permite al tutor o evaluador cambiar el estado de la propuesta (ej., 'En Revisión', 'Aprobada con Observaciones', 'Rechazada'), lo que desencadena notificaciones automáticas a los estudiantes y otros actores relevantes. La implementación se centró en una experiencia de usuario fluida para facilitar la tarea de evaluación.

5. **Módulo de Notificaciones:**

Este módulo es responsable de enviar notificaciones automáticas por correo electrónico a los usuarios ante eventos importantes del sistema. Se integró el sistema de notificaciones de Laravel, que permite definir diferentes tipos de notificaciones (ej., propuesta enviada, tutor asignado, comentarios recibidos, cambio de estado). Las notificaciones se envían de forma asíncrona para no afectar el rendimiento del sistema y contienen información relevante sobre el evento. La configuración de las plantillas de correo electrónico se realizó utilizando Blade, el motor de plantillas de Laravel, permitiendo personalizar el contenido y el formato de los mensajes.

6. **Módulo de Informes y Estadísticas:**

   Diseñado para los coordinadores, este módulo permite generar informes detallados sobre el estado de los proyectos. Se implementaron consultas complejas a la base de datos para extraer métricas como el número de propuestas por estado, tiempo promedio de revisión, distribución por área temática y desempeño de los tutores. Los informes se pueden filtrar por diversos criterios y se pueden exportar a formatos como CSV o PDF. La visualización de datos se mejoró con gráficos y tablas interactivas en el panel de control del coordinador, proporcionando una visión clara y concisa del progreso académico.

   Una funcionalidad clave de este módulo es la **generación de documentos PDF**, la cual es crucial para la emisión de cartas de aceptación, informes de evaluación finales y certificaciones. Para esta funcionalidad, se utilizó una librería como **Dompdf** o **Snappy PDF** (integrada en Laravel a través de paquetes como `barryvdh/laravel-dompdf`). Esta librería permite renderizar vistas HTML en formato PDF, lo que facilita la creación de documentos dinámicos y con formato profesional utilizando las plantillas Blade existentes del sistema. El código subyacente a esta funcionalidad implica la carga de una vista Blade con los datos del proyecto (título, autor, fecha, decisión, etc.), la transformación de esa vista HTML a PDF, y la descarga del archivo resultante. Por ejemplo, para generar una carta de aceptación, se recuperan los datos del proyecto y del estudiante de la base de datos, se rellenan en una plantilla HTML prediseñada, y luego esta se convierte a PDF.

   A continuación, se presenta una descripción conceptual de una captura de código ilustrativa para la generación de PDFs:

   Captura de Código 1: Generador de PDF (Controlador Laravel)

Esta captura muestra un fragmento de código PHP dentro de un controlador de Laravel (`App\Http\Controllers\ProjectController`) que es responsable de generar y descargar un documento PDF, específicamente una carta de aceptación de un proyecto. El código comienza con la definición de un método público, por ejemplo, `generateAcceptanceLetterPdf($projectId)`. Dentro de este método, primero se recupera la información del proyecto y del estudiante asociado desde la base de datos utilizando el ORM Eloquent de Laravel, por ejemplo, `$project = Project::findOrFail($projectId);`. Luego, se prepara un array de datos que se pasarán a la vista Blade que servirá como plantilla para el PDF, como `$data = ['project' => $project, 'student' => $project->student];`. La parte crucial es la invocación de la librería de PDF. Se instancia el generador de PDF (ej., `PDF::loadView('pdfs.acceptance_letter', $data)`), indicando la vista Blade (`pdfs.acceptance_letter`) que contiene el diseño HTML de la carta de aceptación y los datos que debe rellenar. Finalmente, el código invoca un método para descargar el PDF, por ejemplo, `->download('carta_aceptacion_' . $project->id . '.pdf');`, que envía el archivo al navegador del usuario. Este fragmento demuestra la integración fluida entre el framework Laravel, la base de datos y la librería de generación de PDF para crear documentos dinámicos y personalizados.

7. **Captura de Código 2: Fragmento de Validación de Formulario (JavaScript/Frontend)**

   Esta captura de código ilustra un fragmento de JavaScript (posiblemente utilizando Vue.js o JavaScript puro) encargado de la validación de un formulario de envío de propuestas en el lado del cliente. El código muestra una función o un método que se ejecuta cuando el usuario intenta enviar el formulario. Dentro de esta función, se accede a los valores de los campos del formulario (ej., `this.form.title`, `this.form.abstract`). Se aplican una serie de condiciones para verificar la validez de los datos: por ejemplo, `if (!this.form.title || this.form.title.length < 10)` para verificar que el título no esté vacío y tenga una longitud mínima. Se utilizan expresiones regulares para validar formatos específicos, como direcciones de correo electrónico o URLs. Si alguna validación falla, el código establece un estado de error para el campo correspondiente (ej., `this.errors.title = 'El título es obligatorio y debe tener al menos 10 caracteres';`) y, opcionalmente, impide el envío del formulario (`return false;`). Si todos los campos son válidos, el formulario se envía al

servidor. Este fragmento es crucial para proporcionar retroalimentación instantánea al usuario, mejorando la experiencia y reduciendo la carga del servidor al evitar envíos de datos inválidos.

ÍNDICE GENERAL

6. Implementación del Sistema

6.1 Entorno de Desarrollo y Herramientas

6.2 Desarrollo del Backend

6.3 Desarrollo del Frontend

6.4 Integración de Módulos

7. Fase de Pruebas

7.1 Elaboración y Ejecución del Plan de Pruebas

7.2 Análisis de Resultados

8. Conclusiones y Recomendaciones

8.1 Conclusiones

8.2 Recomendaciones

8.3 Líneas Futuras de Investigación

RESUMEN DE CAPÍTULOS ANTERIORES

La presente tesis se ha centrado en el "Diseño e Implementación de un Sistema Web para la Gestión del Proceso de Aceptación de Proyectos Académicos" en la carrera de Ingeniería en Informática. El problema central identificado fue la ineficiencia y falta de transparencia del proceso manual actual, lo que motivó la búsqueda de una solución automatizada. El objetivo general establecido es optimizar y automatizar este proceso mediante el desarrollo de un sistema web integral, utilizando una metodología de desarrollo ágil, específicamente Scrum, para garantizar la adaptabilidad y la entrega incremental de valor.

En los capítulos previos, se ha establecido un **Marco Teórico** robusto, cubriendo fundamentos de sistemas web, gestión de proyectos académicos, metodologías de desarrollo de software y las tecnologías específicas a emplear, como PHP con Laravel para el backend, MySQL para la base de datos y Vue.js para el frontend. Posteriormente, se detalló el **Diagnóstico Situacional**, describiendo la problemática actual de la gestión manual y sus consecuencias negativas. La fase de **Diseño del Sistema** abarcó la definición de la arquitectura (MVC), el modelo de base de datos, el diseño de la interfaz de usuario con un enfoque UI/UX centrado en el usuario, y la modularización de los componentes del sistema. Finalmente, la **Implementación del Sistema** describió el entorno de desarrollo, el proceso de construcción del backend y el frontend, y la integración de todos los módulos, culminando en un prototipo funcional listo para ser validado.

Este capítulo, la **Fase de Pruebas**, representa la etapa crítica donde se verifica la funcionalidad, fiabilidad y eficiencia del sistema desarrollado. Se busca asegurar que la solución no solo cumpla con los requisitos técnicos y funcionales definidos, sino que también resuelva eficazmente la problemática original, proporcionando una herramienta robusta y usable para la gestión de proyectos académicos. La conexión con el contenido previo es directa: la calidad del diseño y la implementación se valida y mide en esta fase, asegurando que el esfuerzo invertido en las etapas anteriores se traduzca en un producto de software de alto valor y rendimiento.

7. FASE DE PRUEBAS

7.1 Elaboración y Ejecución del Plan de Pruebas

La fase de pruebas constituye un pilar fundamental en el ciclo de vida del desarrollo de software, especialmente en proyectos de la envergadura y complejidad como el "Diseño e Implementación de un Sistema Web para la Gestión del Proceso de Aceptación de Proyectos Académicos". Esta etapa no solo busca identificar y corregir errores, sino que también tiene como propósito primordial validar que el sistema cumple con los requisitos funcionales y no funcionales establecidos durante las fases de análisis y diseño, y que, en última instancia, satisface las necesidades de los usuarios finales y resuelve la problemática inicial. La **calidad del software** es directamente proporcional a la rigurosidad y exhaustividad de su proceso de pruebas, siendo este un componente crítico para garantizar la fiabilidad, la usabilidad, la eficiencia y la seguridad del sistema.

La elaboración de un **Plan de Pruebas** detallado es el primer paso estratégico en esta fase. Este plan actúa como una hoja de ruta, definiendo el alcance, los objetivos, los recursos, el cronograma y las metodologías a emplear durante todo el proceso de verificación. Para nuestro sistema de gestión de proyectos académicos, el plan de pruebas se estructuró para abarcar diversas dimensiones de evaluación. Los **objetivos principales** de las pruebas fueron: 1) Verificar la correcta implementación de todas las funcionalidades descritas en la fase de diseño, tales como la gestión de usuarios, la creación y seguimiento de proyectos, la asignación de revisores y la emisión de notificaciones; 2) Asegurar la integridad y consistencia de los datos almacenados en la base de datos MySQL; 3) Evaluar la usabilidad y la experiencia del usuario (UX) de la interfaz desarrollada con Vue.js, garantizando que sea intuitiva y eficiente; 4) Medir el rendimiento del sistema bajo diferentes cargas de trabajo, especialmente en escenarios de múltiples usuarios concurrentes; y 5) Identificar y mitigar posibles vulnerabilidades de seguridad, dado que el sistema manejará información sensible de proyectos académicos y usuarios.

El **alcance de las pruebas** fue definido para cubrir la totalidad del sistema, desde los componentes individuales del backend implementados con Laravel hasta la interacción completa del usuario con el frontend. Se adoptó una estrategia de pruebas por niveles, comenzando con pruebas unitarias, seguidas de pruebas de integración, pruebas de sistema y, finalmente, pruebas de aceptación de usuario (UAT). Esta aproximación gradual permite detectar defectos en etapas tempranas, donde su corrección es menos costosa y compleja.

La **metodología ágil Scrum** utilizada en el desarrollo facilitó una integración continua de las pruebas a lo largo de cada sprint, permitiendo ciclos de retroalimentación rápidos y una adaptación constante del plan de pruebas a medida que el sistema evolucionaba.

Para la **ejecución de las pruebas**, se preparó un entorno de pruebas que replicaba fielmente el entorno de producción, incluyendo el servidor web (Apache/Nginx), la versión de PHP, la base de datos MySQL y todas las dependencias del framework Laravel y Vue.js. Se generaron **datos de prueba representativos**, abarcando casos de uso normales, casos extremos y escenarios de error, para simular situaciones reales y poner a prueba la robustez del sistema. La elaboración de **casos de prueba** se basó directamente en los requisitos funcionales y los casos de uso definidos en la fase de diseño. Cada caso de prueba especificaba una precondición, los pasos a seguir, los datos de entrada, el resultado esperado y el criterio de éxito o fallo.

La **ejecución de las pruebas unitarias** se realizó principalmente sobre los componentes individuales del backend, utilizando herramientas como PHPUnit, integrada con el framework Laravel. Estas pruebas se enfocaron en validar la lógica de negocio de los controladores, modelos y servicios, asegurando que cada función y método operara correctamente de forma aislada. Por ejemplo, se probaron las funciones de autenticación de usuarios, la lógica de validación de proyectos, las operaciones CRUD (Crear, Leer, Actualizar, Eliminar) sobre las entidades de la base de datos, y los algoritmos de asignación de revisores. La automatización de estas pruebas permitió una ejecución rápida y repetible, crucial en un entorno de desarrollo ágil.

Las **pruebas de integración** se centraron en verificar la correcta interacción entre los diferentes módulos y componentes del sistema. Esto incluyó la comunicación entre el frontend (Vue.js) y el backend (Laravel) a través de las APIs RESTful, la interacción entre el backend y la base de datos MySQL, y la integración de servicios externos si los hubiera (como el envío de correos electrónicos para notificaciones). Por ejemplo, se probó que al enviar un formulario de creación de proyecto desde la interfaz, los datos se validaran correctamente en el backend y se persistieran adecuadamente en la base de datos, y que la respuesta al frontend fuera la esperada. Se utilizaron herramientas de testing de integración que simulaban peticiones HTTP y verificaban las respuestas del servidor.

Las **pruebas de sistema** tuvieron como objetivo evaluar el sistema en su conjunto, verificando que cumpliera con todos los requisitos funcionales y no funcionales desde la perspectiva del usuario final. Esto incluyó pruebas de funcionalidad end-to-end, pruebas de rendimiento, pruebas de seguridad y pruebas de usabilidad. Para las pruebas de funcionalidad, se crearon escenarios de uso completos, simulando el flujo de trabajo de un estudiante al presentar un proyecto, de un revisor al evaluar una propuesta y de un

administrador al gestionar el sistema. Se verificó que todas las características operaran según lo especificado y que el sistema manejara adecuadamente los flujos de trabajo complejos.

Las **pruebas de rendimiento** fueron esenciales para asegurar que el sistema pudiera manejar la carga esperada de usuarios y operaciones. Se realizaron pruebas de carga para simular un número creciente de usuarios concurrentes accediendo al sistema y realizando diversas operaciones (inicio de sesión, envío de proyectos, consulta de estado). Se monitorearon métricas clave como el tiempo de respuesta, el uso de CPU y memoria del servidor, y el rendimiento de la base de datos. El objetivo fue identificar cuellos de botella y asegurar que el sistema mantuviera un rendimiento aceptable incluso bajo picos de demanda. Para estas pruebas, se consideraron herramientas de estrés y carga que permitieran simular estos escenarios de manera controlada.

La **seguridad del sistema** fue una preocupación primordial, dada la naturaleza académica y la información sensible que manejaría. Se realizaron pruebas de seguridad para identificar vulnerabilidades comunes, como inyección SQL, cross-site scripting (XSS), cross-site request forgery (CSRF), fallos de autenticación y autorización, y exposición de datos sensibles. Se emplearon técnicas de caja negra y caja blanca, revisando tanto el código fuente como el comportamiento externo del sistema ante ataques simulados. Se verificó la robustez de los mecanismos de autenticación y autorización implementados, asegurando que solo los usuarios con los roles adecuados pudieran acceder a las funcionalidades y datos correspondientes.

Finalmente, las **Pruebas de Aceptación de Usuario (UAT)** representaron la etapa culminante, donde usuarios finales reales (estudiantes, profesores, administradores de la carrera de Ingeniería en Informática) interactuaron con el sistema para validar que este cumpliera con sus expectativas y necesidades operativas. Estas pruebas son cruciales para asegurar la **validación de la solución** desde la perspectiva del negocio. Se proporcionaron escenarios de uso realistas a los usuarios, quienes ejecutaron las funcionalidades principales del sistema y proporcionaron retroalimentación directa sobre la usabilidad, la funcionalidad y la adecuación general de la aplicación. Cualquier desviación o sugerencia de mejora se documentó meticulosamente para su posterior análisis y posible implementación. La ejecución de este plan de pruebas exhaustivo y sistemático fue fundamental para garantizar la calidad y la fiabilidad del sistema web desarrollado.

7.2 Análisis de Resultados

El análisis de los resultados obtenidos durante la fase de pruebas es un proceso crítico que permite evaluar la calidad del software, identificar áreas de mejora y determinar la preparación del sistema para su despliegue en un entorno de producción. Tras la rigurosa ejecución del plan de pruebas, se procedió a la recopilación, clasificación y evaluación de todos los datos generados, incluyendo la cantidad de casos de prueba ejecutados, los casos exitosos, los fallidos y, lo más importante, la naturaleza y severidad de los defectos detectados. Este análisis no solo se enfoca en la cantidad de errores, sino en su impacto potencial en la funcionalidad, rendimiento y seguridad del sistema de gestión de proyectos académicos.

La **gestión de defectos** fue un componente central del análisis de resultados. Cada defecto identificado durante las pruebas unitarias, de integración, de sistema y UAT fue documentado en un sistema de seguimiento de errores. Para cada defecto, se registraron detalles como la descripción, los pasos para reproducirlo, la severidad (crítico, mayor, moderado, menor), la prioridad (alta, media, baja), el módulo afectado, la fecha de detección y el responsable de su corrección. La **severidad de los defectos** se clasificó de la siguiente manera: **Crítico**, si el defecto impedía la funcionalidad principal del sistema o causaba pérdida de datos; **Mayor**, si afectaba una funcionalidad importante pero no bloqueaba completamente el sistema; **Moderado**, si causaba inconvenientes menores o comportamientos inesperados; y **Menor**, si se trataba de problemas estéticos o de usabilidad de bajo impacto. Esta clasificación permitió priorizar las correcciones y asignar recursos de manera eficiente, asegurando que los problemas más graves fueran abordados primero.

A partir de los datos recopilados, se generaron **métricas y KPIs (Key Performance Indicators)** que ofrecieron una visión cuantitativa del estado del software. Entre las métricas clave se incluyeron: el **índice de aprobación de pruebas**, que representó el porcentaje de casos de prueba ejecutados exitosamente; la **densidad de defectos**, calculada como el número de defectos por cada mil líneas de código o por cada funcionalidad; el **tiempo promedio de resolución de defectos**, que midió la eficiencia del equipo de desarrollo en corregir los problemas; y la **trazabilidad de defectos**, que vinculó cada defecto con el requisito original al que afectaba. Por ejemplo, si se detectaba un error en la funcionalidad de "subir propuesta de proyecto", se trazaba hasta el requisito funcional específico que describía dicha acción. Esta trazabilidad fue fundamental para asegurar que todos los requisitos fueran validados y que los defectos no pasaran desapercibidos.

Los resultados de las **pruebas unitarias** mostraron un alto índice de aprobación, lo que indicó una sólida implementación de la lógica de negocio a nivel de componentes individuales en el backend de Laravel. Los pocos defectos encontrados en esta etapa fueron

principalmente de lógica menor o de manejo de excepciones, los cuales fueron corregidos rápidamente. Esto validó la eficacia de la programación modular y el uso de un framework robusto que promueve buenas prácticas de desarrollo.

En las **pruebas de integración**, se identificaron algunos defectos relacionados con la comunicación entre el frontend de Vue.js y el backend de Laravel, especialmente en el manejo de estados de la aplicación y la sincronización de datos en tiempo real. Estos problemas fueron cruciales de detectar en esta fase, ya que afectaban la experiencia del usuario y la coherencia de la información. La mayoría de ellos fueron debidos a errores en la definición de los contratos de las APIs RESTful o en el manejo asíncrono de las peticiones. Su resolución implicó ajustes en los controladores del backend y en los servicios de datos del frontend, asegurando una interacción fluida y sin errores.

Las **pruebas de sistema** revelaron la mayoría de los defectos de mayor impacto, especialmente en escenarios de uso complejos que involucraban múltiples roles de usuario y flujos de trabajo interconectados. Por ejemplo, se identificaron problemas en la correcta propagación de notificaciones a todos los usuarios relevantes tras un cambio de estado de un proyecto, o en la validación de permisos al intentar acceder a funcionalidades restringidas. También se detectaron algunas vulnerabilidades menores de seguridad que, aunque no críticas, requerían atención para fortalecer la protección del sistema. Estos hallazgos llevaron a revisiones significativas en la lógica de autorización y en el sistema de eventos y notificaciones del backend.

Los resultados de las **pruebas de rendimiento** indicaron que el sistema, en su estado inicial, podía manejar eficientemente un número moderado de usuarios concurrentes. Sin embargo, en escenarios de carga máxima simulada, se observaron picos en el tiempo de respuesta y un aumento en el consumo de recursos de la base de datos. Este análisis llevó a la identificación de consultas SQL ineficientes y a la necesidad de optimizar ciertos índices en la base de datos MySQL. Se implementaron mejoras en el caching de datos y en la optimización de consultas, lo que resultó en una mejora sustancial del rendimiento del sistema bajo estrés. Estos ajustes fueron vitales para garantizar la escalabilidad del sistema a medida que la cantidad de usuarios y proyectos aumente.

Finalmente, los resultados de las **Pruebas de Aceptación de Usuario (UAT)** fueron sumamente valiosos. La retroalimentación de los usuarios finales (estudiantes, profesores y administradores) proporcionó una perspectiva real sobre la usabilidad y la adecuación del sistema a sus necesidades diarias. Si bien la funcionalidad básica fue bien recibida, se sugirieron mejoras en la claridad de ciertos mensajes de error, la disposición de algunos elementos en la interfaz de usuario y la adición de pequeñas características de conveniencia. Por ejemplo, se propuso una mejora en el filtro de búsqueda de proyectos o

una reorganización de los campos en el formulario de creación de proyectos. Estas sugerencias, aunque no implicaron defectos críticos, fueron cruciales para pulir la experiencia del usuario y aumentar la satisfacción general. Se priorizaron aquellas que ofrecían el mayor impacto en la usabilidad y se planificó su implementación en futuras iteraciones o como parte de un mantenimiento evolutivo.

En conclusión, el análisis de los resultados de las pruebas demostró que el sistema web para la gestión de proyectos académicos, tras las correcciones pertinentes, cumplía con la mayoría de los requisitos funcionales y no funcionales. Los defectos críticos fueron identificados y resueltos, la estabilidad y seguridad del sistema fueron reforzadas, y el rendimiento fue optimizado. La retroalimentación de los usuarios en la UAT fue fundamental para validar la solución desde una perspectiva práctica y para identificar oportunidades de mejora continua. Este proceso de pruebas riguroso y sistemático permitió asegurar que el sistema no solo fuera funcional, sino también robusto, eficiente, seguro y, lo más importante, útil para los usuarios finales, cumpliendo así con el objetivo principal de la tesis de optimizar y automatizar el proceso de aceptación de proyectos académicos.

ÍNDICE GENERAL

8.5 Líneas de Investigación Futuras y Recomendaciones

9. RECOMENDACIONES

10. REFERENCIAS

11. ANEXOS

RESUMEN DE CAPÍTULOS ANTERIORES:

La presente tesis se ha centrado en el **Diseño e Implementación de un Sistema Web para la Gestión del Proceso de Aceptación de Proyectos Académicos** en la carrera de Ingeniería en Informática. Iniciamos con una **Introducción** que justificó la necesidad del proyecto ante la problemática de una gestión manual, ineficiente y descentralizada de las propuestas de tesis. Se establecieron objetivos claros: diseñar, desarrollar e implementar una solución web que automatice y optimice este proceso, mejorando la transparencia y reduciendo los tiempos de aprobación.

El **Marco Teórico** proporcionó los fundamentos conceptuales sobre sistemas web, gestión de proyectos académicos y metodologías de desarrollo de software, sentando las bases para la comprensión técnica del proyecto. Posteriormente, la **Metodología** detalló el enfoque de investigación (aplicada), el tipo (descriptiva-proyectiva) y las fases del desarrollo, destacando el uso de metodologías ágiles. El **Diagnóstico Situacional** profundizó en la situación actual, identificando los puntos críticos del proceso manual y levantando los requerimientos funcionales y no funcionales del sistema propuesto.

Con base en el diagnóstico, el capítulo de **Diseño del Sistema** delineó la arquitectura MVC, el diseño de la base de datos, la interfaz de usuario y la estructura modular. La fase de **Implementación** describió el desarrollo del backend con Laravel y MySQL, y el frontend con Vue.js, detallando la construcción de cada módulo funcional. Finalmente, el capítulo de **Pruebas y Validación** expuso el plan de pruebas, los tipos de pruebas realizadas (unitarias, integración, sistema, aceptación) y los resultados obtenidos, confirmando la operatividad y eficacia del sistema en un entorno controlado y con la participación de usuarios clave.

OBJETIVOS DE LA SECCIÓN "CONCLUSIONES":

La presente sección de **Conclusiones** tiene como objetivo principal consolidar los hallazgos y resultados de la investigación, evaluando el cumplimiento de los objetivos planteados al inicio del proyecto. Se buscará sintetizar los aportes más significativos del sistema desarrollado, reflexionar sobre su impacto potencial en la gestión académica y señalar las

limitaciones inherentes al proyecto. Finalmente, se establecerán recomendaciones que abran nuevas vías para futuras investigaciones y mejoras en el ámbito de la gestión de proyectos académicos mediante soluciones tecnológicas.

8. CONCLUSIONES

La culminación de este proyecto de tesis, centrado en el **Diseño e Implementación de un Sistema Web para la Gestión del Proceso de Aceptación de Proyectos Académicos**, permite extraer una serie de conclusiones fundamentales que no solo validan el esfuerzo realizado, sino que también reafirman la viabilidad y el impacto positivo de la automatización en entornos académicos. A lo largo de las diversas fases de investigación, análisis, diseño, desarrollo y pruebas, se ha logrado construir una solución tecnológica robusta y funcional que atiende de manera directa las necesidades identificadas en el proceso de gestión de proyectos de tesis en la carrera de Ingeniería en Informática. La transición de un modelo manual y propenso a errores a un sistema digitalizado representa un avance significativo en la eficiencia administrativa y la transparencia académica, sentando un precedente para futuras innovaciones en la gestión universitaria.

8.1 Recapitulación de los Objetivos y su Cumplimiento

Desde el inicio de esta investigación, se establecieron objetivos claros que guiaron cada etapa del desarrollo. El **Objetivo General** fue "Diseñar, desarrollar e implementar un sistema web para la gestión del proceso de aceptación de proyectos académicos que optimice la eficiencia y transparencia en la carrera de Ingeniería en Informática". Tras la conclusión de las fases de diseño, implementación y validación, se puede afirmar con contundencia que este objetivo general ha sido **plenamente alcanzado**. El sistema desarrollado cumple con las funcionalidades requeridas para gestionar integralmente el ciclo de vida de una propuesta de proyecto, desde su registro hasta su aprobación final, proporcionando una plataforma centralizada y accesible.

En cuanto a los **Objetivos Específicos**, se han verificado los siguientes logros:

1. **Diagnosticar la situación actual del proceso de aceptación de proyectos académicos:** Este objetivo se cumplió a través de un exhaustivo análisis del proceso manual existente, la identificación de sus deficiencias (demoras, falta de trazabilidad, errores administrativos) y la recolección de requerimientos funcionales y no funcionales por parte de los usuarios clave (estudiantes, docentes, coordinadores). El diagnóstico permitió comprender a fondo el contexto problemático y establecer una base sólida para el diseño de la solución, tal como se detalló en el Capítulo 4.

2. **Diseñar la arquitectura, base de datos e interfaz de usuario del sistema web:** Este objetivo se concretó en el Capítulo 5, donde se propuso una **arquitectura Modelo-Vista-Controlador (MVC)**, un diseño relacional de la base de datos que garantiza la integridad y consistencia de la información, y un diseño de interfaz de usuario (UI/UX) enfocado en la usabilidad y la experiencia del usuario. Los diagramas de clases, diagramas de casos de uso, prototipos de interfaz y el modelo entidad-relación demuestran el cumplimiento de este objetivo, proporcionando un blueprint detallado para la construcción del sistema.
3. **Implementar el sistema web utilizando tecnologías modernas y frameworks adecuados:** El Capítulo 6 evidenció el éxito en la implementación. Se utilizó PHP con el framework **Laravel** para el backend, garantizando una lógica de negocio robusta y segura. Para el frontend, se empleó JavaScript con el framework **Vue.js**, lo que permitió desarrollar una interfaz dinámica, reactiva e intuitiva. La base de datos **MySQL** fue fundamental para el almacenamiento eficiente de la información. La integración de estas tecnologías resultó en un sistema funcional, modular y escalable, confirmando la consecución de este objetivo.
4. **Realizar pruebas exhaustivas del sistema para asegurar su correcto funcionamiento y validar su eficacia:** Este objetivo fue abordado en el Capítulo 7. Se ejecutaron diferentes tipos de pruebas, incluyendo pruebas unitarias para componentes individuales, pruebas de integración para verificar la comunicación entre módulos, pruebas de sistema para evaluar el comportamiento global y pruebas de aceptación con usuarios finales para validar que el sistema satisficiera sus necesidades. Los resultados de estas pruebas demostraron la operatividad, fiabilidad y eficacia del sistema, minimizando errores y garantizando la calidad del software.
5. **Evaluar el impacto del sistema en la eficiencia y transparencia del proceso de aceptación de proyectos:** Aunque la evaluación del impacto a gran escala requeriría un período de uso más prolongado, las pruebas de aceptación y la retroalimentación inicial de los usuarios clave sugieren un impacto altamente positivo. El sistema ha **reducido drásticamente los tiempos de procesamiento**, ha eliminado la pérdida de documentos, ha proporcionado una **trazabilidad completa** de cada etapa del proyecto y ha centralizado la información, lo que se traduce directamente en una mejora tangible de la eficiencia y una mayor transparencia para todos los actores involucrados.

En síntesis, todos los objetivos planteados al inicio de la investigación han sido satisfactoriamente alcanzados, lo que consolida la tesis como un proyecto exitoso y una contribución valiosa para la mejora de los procesos académicos.

8.2 Principales Hallazgos y Aportes del Sistema

La investigación y el desarrollo del sistema web han generado una serie de **hallazgos significativos** y han producido aportes concretos que merecen ser destacados. Uno de los principales hallazgos es la confirmación de que la **automatización de procesos administrativos** en el ámbito académico no solo es deseable, sino indispensable en la era digital. La gestión manual no solo consume recursos valiosos, sino que también introduce un margen de error humano considerable y dificulta la supervisión y el seguimiento de los avances, aspectos que el sistema desarrollado ha logrado mitigar de forma efectiva. La resistencia inicial a la digitalización, si bien presente en algunos casos, se disipa rápidamente al evidenciarse los beneficios tangibles en términos de rapidez y fiabilidad.

Los **aportes específicos** del sistema son múltiples y se pueden categorizar de la siguiente manera:

- **Centralización y Unificación de la Información:** El sistema proporciona un repositorio único para todas las propuestas de proyectos, eliminando la dispersión de documentos físicos y digitales. Esto facilita el acceso a la información en tiempo real para estudiantes, docentes asesores y miembros del comité de aprobación, promoviendo una visión unificada y actualizada del estado de cada proyecto.
- **Optimización de Tiempos de Procesamiento:** La automatización de flujos de trabajo, notificaciones y estados de aprobación ha reducido significativamente los tiempos de espera y las demoras inherentes al proceso manual. Las solicitudes se tramitan de manera más ágil, permitiendo a los estudiantes recibir retroalimentación y aprobaciones en un lapso de tiempo considerablemente menor.
- **Mejora de la Trazabilidad y Transparencia:** Cada acción realizada dentro del sistema, desde la presentación de una propuesta hasta su aprobación o rechazo, queda registrada con fecha y hora, y el usuario responsable. Esto confiere una **trazabilidad completa** del proceso, aumentando la transparencia y la rendición de cuentas. Los estudiantes pueden monitorear el progreso de sus proyectos en cualquier momento, y los coordinadores tienen una visión global del estado de todas las propuestas.

- **Reducción de Errores Administrativos:** La validación de datos en tiempo real, la estandarización de formularios y la eliminación de la manipulación manual de documentos minimizan la ocurrencia de errores humanos, como la pérdida de información o el registro incorrecto de datos, que eran comunes en el proceso previo.
- **Interfaz de Usuario Intuitiva y Amigable:** El diseño centrado en el usuario, implementado con Vue.js, ha resultado en una interfaz que es fácil de navegar y comprender, incluso para usuarios con conocimientos tecnológicos limitados. Esto facilita la adopción del sistema y reduce la curva de aprendizaje, maximizando su utilidad.
- **Soporte para la Toma de Decisiones:** Al centralizar la información y proporcionar un seguimiento detallado, el sistema genera datos valiosos que pueden ser utilizados por las autoridades académicas para analizar tendencias, identificar cuellos de botella y tomar decisiones informadas para la mejora continua del proceso de gestión de proyectos.
- **Replicabilidad y Escalabilidad:** La arquitectura modular y el uso de tecnologías estándar (Laravel, Vue.js, MySQL) confieren al sistema un alto grado de replicabilidad, permitiendo su adaptación a otras carreras o instituciones con procesos similares. Además, su diseño escalable facilita la incorporación de nuevas funcionalidades en el futuro sin comprometer su rendimiento actual.

Estos aportes no solo benefician directamente a la carrera de Ingeniería en Informática, sino que también establecen un modelo de referencia para la modernización de la gestión académica en otros contextos universitarios, demostrando el potencial transformador de las tecnologías de la información.

8.3 Impacto y Beneficios de la Implementación

La implementación del sistema web para la gestión de proyectos académicos ha generado un impacto multifacético y ha traído consigo una serie de beneficios tangibles e intangibles para la comunidad universitaria. El **impacto más inmediato** se observa en la optimización del tiempo. Lo que antes podía tomar semanas en trámites y revisiones manuales, ahora se reduce a días, o incluso horas, gracias a la automatización de notificaciones, la estandarización de formatos y la centralización de las comunicaciones. Este ahorro de tiempo no solo beneficia a los estudiantes, quienes pueden dedicar más esfuerzo a la investigación en sí, sino también a los docentes y coordinadores, liberándolos de tareas administrativas repetitivas para que puedan enfocarse en actividades de mayor valor académico.

Entre los **beneficios clave** derivados de la implementación, se destacan:

- **Incremento de la Eficiencia Operativa:** La eliminación de la burocracia en papel y la automatización de los flujos de trabajo han mejorado sustancialmente la eficiencia operativa. Los procesos son más rápidos, con menos interrupciones y una asignación más efectiva de recursos, lo que se traduce en una gestión académica más fluida y productiva.
- **Mayor Transparencia y Rendición de Cuentas:** El sistema proporciona una visibilidad completa del estado de cada proyecto en tiempo real. Los estudiantes pueden seguir el progreso de su propuesta, los asesores pueden ver el estado de las revisiones y los coordinadores tienen un panorama general de todas las propuestas. Esta transparencia fomenta la confianza en el proceso y asegura una mayor rendición de cuentas por parte de todos los involucrados.
- **Estandarización y Consistencia:** Al digitalizar los formularios y los flujos de aprobación, el sistema impone una estandarización de los procedimientos. Esto asegura que todas las propuestas sigan el mismo camino, minimizando la subjetividad y garantizando una aplicación consistente de las normativas de la universidad. La consistencia en el proceso reduce confusiones y posibles disputas.
- **Mejora en la Comunicación Interna:** Las notificaciones automáticas y el sistema de mensajería integrado facilitan una comunicación efectiva y oportuna entre estudiantes, asesores y el comité. Esto reduce la necesidad de reuniones presenciales y correos electrónicos dispersos, centralizando la información y asegurando que todos estén al tanto de los desarrollos importantes.
- **Facilitación del Acceso a la Información:** La disponibilidad de la plataforma web 24/7 permite a los usuarios acceder a la información de sus proyectos desde cualquier lugar y en cualquier momento, siempre que dispongan de conexión a internet. Esto es particularmente beneficioso para estudiantes y docentes con horarios complejos o que se encuentran fuera del campus.
- **Reducción del Uso de Papel:** La digitalización del proceso contribuye a una gestión más sostenible y ecológica al disminuir drásticamente la necesidad de imprimir documentos. Esto no solo genera ahorros en costos de insumos, sino que también alinea la institución con prácticas ambientales responsables.
- **Generación de Datos para Análisis:** El sistema acumula una vasta cantidad de datos sobre las propuestas de proyectos, los tiempos de aprobación, las áreas de investigación más populares, entre otros. Esta información es

invaluable para futuras investigaciones, para la planificación estratégica de la carrera y para la identificación de áreas de mejora en el currículo o en los procesos académicos.

En definitiva, el sistema implementado no es solo una herramienta tecnológica, sino un catalizador para la transformación digital de la gestión académica, impulsando una cultura de eficiencia, transparencia y modernización en la universidad.

8.4 Limitaciones Identificadas

A pesar del éxito general en el diseño e implementación del sistema web, es fundamental reconocer y exponer las **limitaciones inherentes** al alcance de este proyecto de tesis. La honestidad académica exige la identificación de aquellos aspectos que no pudieron ser abordados en su totalidad o que representan áreas de mejora para futuras iteraciones. Una de las principales limitaciones se relaciona con el **alcance temporal y los recursos disponibles**. Al tratarse de un proyecto académico realizado en un tiempo determinado, no fue posible abarcar todas las funcionalidades que un sistema de gestión de proyectos de tesis podría potencialmente incluir en un entorno de producción a gran escala.

Otras limitaciones importantes son:

- **Integración con Otros Sistemas Universitarios:** El sistema fue diseñado como una solución autónoma para la gestión de proyectos académicos. No se contempló ni se implementó una integración directa con otros sistemas de información existentes en la universidad, como sistemas de gestión estudiantil (SGA), plataformas de e-learning o sistemas de gestión documental. Esta falta de integración puede requerir la duplicación de algunos datos o procesos en el futuro, aunque no afecta la funcionalidad central del sistema.
- **Escalabilidad a Nivel Institucional Completo:** Si bien la arquitectura del sistema es modular y permite cierta escalabilidad, su diseño y pruebas se realizaron específicamente para la carrera de Ingeniería en Informática. La implementación a nivel de toda la universidad, abarcando múltiples carreras con reglamentos y procesos distintos, requeriría una fase de análisis y adaptación considerablemente mayor, que escapa al alcance de esta tesis.
- **Gestión de Usuarios y Roles Complejos:** El sistema implementa un esquema de roles básico (estudiante, docente asesor, coordinador de carrera, miembro de comité). Sin embargo, una gestión de usuarios y permisos más granular y compleja, que atienda a estructuras organizacionales muy específicas o a un gran número de usuarios con roles híbridos, no fue un enfoque prioritario.

- **Módulos de Reportes y Analíticas Avanzadas:** Aunque el sistema almacena datos valiosos, la generación de reportes y analíticas avanzadas para la toma de decisiones estratégicas (ej. tendencias de áreas de investigación, desempeño de asesores, análisis de tiempos promedio de aprobación) se encuentra en un nivel básico. Un módulo de inteligencia de negocios más sofisticado sería una mejora deseable, pero quedó fuera del alcance inicial.
- **Pruebas de Carga y Rendimiento a Gran Escala:** Las pruebas realizadas se enfocaron en la funcionalidad y la usabilidad, con un número limitado de usuarios concurrentes. No se llevaron a cabo pruebas de carga y rendimiento exhaustivas bajo condiciones de alto tráfico o con un volumen masivo de datos, lo que podría ser relevante para una implementación a nivel institucional con miles de usuarios.
- **Aspectos de Seguridad Avanzada:** Se implementaron medidas de seguridad estándar (autenticación, autorización, validación de entradas), pero no se realizaron auditorías de seguridad exhaustivas ni pruebas de penetración por parte de expertos externos. La seguridad es un campo en constante evolución, y la protección contra ataques sofisticados siempre es un desafío.
- **Documentación para Usuarios Finales:** Si bien se generó documentación técnica para el desarrollo, la creación de manuales de usuario detallados y material de capacitación extensivo para todos los roles no fue un entregable principal de la tesis, aunque es crucial para la adopción y el uso efectivo del sistema a largo plazo.

Estas limitaciones no desmerecen los logros del proyecto, sino que ofrecen una visión realista de su alcance y sirven como punto de partida para futuras mejoras y expansiones.

8.5 Líneas de Investigación Futuras y Recomendaciones

La finalización de este proyecto no solo marca la consecución de sus objetivos, sino que también abre un abanico de **oportunidades para futuras investigaciones** y mejoras. El sistema web implementado constituye una base sólida, pero existen diversas avenidas para su expansión y perfeccionamiento, lo que puede traducirse en nuevos proyectos académicos o iniciativas de desarrollo institucional. La naturaleza dinámica de la tecnología y las necesidades cambiantes de los usuarios garantizan que siempre habrá espacio para la innovación.

Se proponen las siguientes **líneas de investigación futuras y recomendaciones**:

1. **Integración con Sistemas Académicos Existentes:**

- **Recomendación:** Explorar la viabilidad e implementar la integración del sistema con el Sistema de Gestión Académica (SGA) de la universidad, para la sincronización automática de datos de estudiantes, docentes y asignaturas. Esto evitaría la redundancia de información y mejoraría la coherencia de los datos.
- **Línea de Investigación:** Estudiar las mejores prácticas y tecnologías para la integración de sistemas heterogéneos en entornos universitarios, evaluando APIs, web services y protocolos de comunicación estándar.

2. **Desarrollo de Módulos de Analíticas y Business Intelligence:**

    - **Recomendación:** Implementar un módulo avanzado de reportes y paneles de control (dashboards) que permitan a los coordinadores y autoridades académicas visualizar métricas clave, como el número de proyectos por área, tiempos promedio de aprobación, tasas de éxito y desempeño de los asesores.
    - **Línea de Investigación:** Investigar la aplicación de técnicas de minería de datos y aprendizaje automático para predecir tendencias en la investigación académica, identificar posibles cuellos de botella en el proceso o sugerir áreas de enfoque para la investigación institucional.

3. **Expansión a Otras Carreras o Facultades:**

    - **Recomendación:** Adaptar el sistema para que pueda ser utilizado por otras carreras o facultades dentro de la universidad, considerando las particularidades de sus reglamentos y procesos de aprobación de proyectos. Esto implicaría un análisis de requerimientos a nivel institucional y la parametrización de flujos de trabajo.
    - **Línea de Investigación:** Diseñar un modelo de sistema de gestión de proyectos académicos altamente configurable y modular que pueda ser implementado en diversas instituciones educativas con mínima adaptación de código.

4. **Implementación de Notificaciones Avanzadas y Recordatorios:**

- **Recomendación:** Mejorar el sistema de notificaciones para incluir alertas personalizables por correo electrónico, SMS o incluso notificaciones push en una futura aplicación móvil, para recordar plazos, cambios de estado o solicitudes de revisión.
- **Línea de Investigación:** Estudiar el impacto de diferentes estrategias de notificación en la adherencia a los plazos y la eficiencia de la comunicación en entornos académicos.

5. **Desarrollo de una Aplicación Móvil Complementaria:**
    - **Recomendación:** Crear una aplicación móvil nativa o híbrida que permita a estudiantes y docentes realizar consultas rápidas, recibir notificaciones y gestionar tareas básicas desde sus dispositivos móviles.
    - **Línea de Investigación:** Investigar la usabilidad y la experiencia del usuario en aplicaciones móviles para la gestión académica, comparando diferentes enfoques de desarrollo (nativo vs. híbrido).
6. **Mejora Continua de la Interfaz de Usuario (UI/UX):**
    - **Recomendación:** Realizar evaluaciones periódicas de usabilidad con usuarios reales y aplicar principios de diseño centrado en el usuario para mejorar continuamente la interfaz gráfica y la experiencia de usuario.
    - **Línea de Investigación:** Explorar la implementación de nuevas tendencias de diseño de interfaz, como interfaces conversacionales o el uso de inteligencia artificial para personalizar la experiencia del usuario.
7. **Fortalecimiento de la Seguridad y Resiliencia del Sistema:**
    - **Recomendación:** Realizar auditorías de seguridad periódicas, pruebas de penetración y aplicar las últimas recomendaciones en ciberseguridad para proteger la información sensible y garantizar la resiliencia del sistema frente a posibles ataques.
    - **Línea de Investigación:** Investigar la aplicación de tecnologías blockchain para mejorar la inmutabilidad y la transparencia en el registro de aprobaciones de proyectos académicos.

Estas líneas de investigación y recomendaciones no solo prolongan la vida útil del sistema, sino que también aseguran su relevancia y su capacidad para adaptarse a las demandas futuras del entorno académico, consolidando su papel como una herramienta fundamental para la gestión eficiente y transparente de los proyectos de tesis.

Hasta este punto de la tesis, se ha presentado un análisis exhaustivo del problema de la gestión manual de proyectos académicos en la carrera de Ingeniería en Informática, delineando las ineficiencias y desafíos inherentes a este enfoque tradicional. Se ha detallado la propuesta de solución a través del diseño y la implementación de un **Sistema Web para la Gestión del Proceso de Aceptación de Proyectos Académicos**, explicando la arquitectura, las tecnologías empleadas (Laravel, MySQL, Vue.js) y la metodología de desarrollo ágil (Scrum) que guiaron su construcción. Los capítulos previos han cubierto desde el diagnóstico situacional, pasando por el marco teórico que sustenta la solución, hasta el desarrollo práctico del sistema y su fase de pruebas, culminando con la presentación de los resultados y las conclusiones derivadas de la investigación.

Los resultados obtenidos demuestran que el sistema desarrollado no solo cumple con los objetivos planteados, sino que también introduce mejoras significativas en la eficiencia, transparencia y trazabilidad del proceso de aceptación de proyectos. La automatización de tareas, la centralización de la información y la provisión de herramientas para la colaboración han transformado un proceso anteriormente engorroso en uno más dinámico y controlable. Sin embargo, la implementación de cualquier sistema informático en un entorno académico dinámico es un punto de partida, no un fin en sí mismo. La mejora continua y la adaptación a nuevas necesidades son aspectos cruciales para asegurar la sostenibilidad y la relevancia a largo plazo de la solución.

En este contexto, la presente sección, "9. Recomendaciones", tiene como objetivo principal proponer directrices y acciones concretas que permitan optimizar aún más el sistema implementado, asegurar su correcta adopción y evolución dentro de la institución, y abrir nuevas avenidas para futuras investigaciones. Estas recomendaciones se fundamentan en las lecciones aprendidas durante todo el ciclo de vida del proyecto, en las conclusiones obtenidas de la fase de pruebas y en la visión a futuro de la gestión académica de proyectos. Se buscará ofrecer una perspectiva integral que abarque desde mejoras técnicas en el software hasta consideraciones estratégicas para su integración institucional y potencial expansión académica.

ÍNDICE GENERAL

9. RECOMENDACIONES

Las recomendaciones que se presentan a continuación son el corolario lógico de la investigación y desarrollo de este sistema web. Se formulan con el propósito de maximizar el valor del sistema implementado, asegurar su sostenibilidad y adaptabilidad a largo plazo, y fomentar una cultura de mejora continua en la gestión de proyectos académicos. Estas directrices se estructuran en tres ejes principales: la evolución técnica del sistema, las consideraciones institucionales para su adopción y aprovechamiento, y las futuras líneas de investigación que pueden surgir a partir de este trabajo.

9.1 Recomendaciones para la Evolución y Mejora del Sistema Desarrollado

El sistema web para la gestión del proceso de aceptación de proyectos académicos ha sido diseñado con una arquitectura modular y escalable, lo que facilita su futura expansión y adaptación. Sin embargo, el entorno tecnológico y las necesidades de los usuarios evolucionan constantemente, por lo que es imperativo considerar una serie de mejoras y adiciones para mantener su relevancia y eficiencia.

9.1.1 Integración de Nuevas Funcionalidades

Si bien el sistema actual cubre las funcionalidades esenciales para la gestión del proceso de aceptación, existen áreas que podrían enriquecer significativamente la experiencia de los usuarios y la robustez del flujo de trabajo. Una de las primeras adiciones recomendadas es la implementación de un **módulo de gestión de tribunales de tesis avanzado**. Actualmente, la asignación de revisores se realiza de manera eficiente, pero un módulo más sofisticado podría incluir funcionalidades como la gestión de disponibilidad de docentes, la consideración de especialidades temáticas para una asignación más precisa, y la automatización de la generación de documentos de asignación y notificaciones. Esto reduciría la carga administrativa del comité académico y garantizaría una selección óptima de los evaluadores, impactando directamente en la calidad de la retroalimentación a los estudiantes.

Otra funcionalidad clave a considerar es un **módulo de seguimiento de tesis en curso**. Una vez que un proyecto es aceptado, el sistema podría extender su alcance para monitorear el progreso de la tesis hasta su defensa final. Esto implicaría la integración de hitos de seguimiento, la posibilidad de cargar avances periódicos, la comunicación bidireccional entre el estudiante y su tutor, y la generación de reportes de progreso. Dicho módulo permitiría a la coordinación de carrera tener una visión global y en tiempo real del estado de todas las tesis, facilitando la identificación temprana de posibles retrasos o problemas y

permitiendo una intervención oportuna. La implementación de un **calendario interactivo** que muestre fechas límite, reuniones con tutores y defensas de tesis sería un complemento valioso para esta funcionalidad, mejorando la planificación y la organización tanto para estudiantes como para docentes.

Finalmente, la adición de un **repositorio de documentos y plantillas** integrado sería de gran utilidad. Esto permitiría a los estudiantes acceder fácilmente a todas las plantillas oficiales para la elaboración de sus propuestas, informes de avance y el documento final de tesis. Asimismo, el sistema podría almacenar versiones de documentos revisados, facilitando el control de versiones y la trazabilidad de los cambios. La capacidad de generar automáticamente ciertos documentos administrativos (cartas de aceptación, actas de asignación de tribunal) a partir de los datos del sistema simplificaría aún más las operaciones, eliminando la necesidad de procesos manuales de edición y formateo, y asegurando la uniformidad en la documentación oficial de la institución.

9.1.2 Optimización del Rendimiento y Escalabilidad

Aunque el sistema ha sido probado y funciona correctamente con el volumen actual de usuarios y proyectos, es fundamental prepararlo para un crecimiento futuro. La **optimización continua de la base de datos** es crucial. Se recomienda realizar auditorías periódicas de las consultas SQL más frecuentes para identificar cuellos de botella y aplicar índices adecuados. La normalización de la base de datos ya implementada es un buen punto de partida, pero la revisión de la eficiencia de las relaciones y la posibilidad de desnormalización estratégica para ciertas consultas de lectura intensiva podría mejorar significativamente los tiempos de respuesta. La implementación de una estrategia de **caching** a nivel de aplicación (Laravel Cache) o a nivel de base de datos (Redis, Memcached) para datos frecuentemente accedidos pero que no cambian a menudo, como listas de usuarios o configuraciones generales, reduciría la carga sobre el servidor de base de datos y aceleraría la entrega de contenido a los usuarios.

En cuanto a la **escalabilidad de la infraestructura**, se sugiere explorar opciones de despliegue en la nube (AWS, Google Cloud, Azure) que permitan escalar horizontalmente los recursos computacionales y de almacenamiento según la demanda. Un enfoque de microservicios, aunque más complejo de implementar inicialmente, podría ser una evolución a largo plazo para desacoplar las funcionalidades y permitir que cada una escale de forma independiente. Para el corto y mediano plazo, la configuración de un balanceador de carga y la replicación de instancias del servidor de aplicaciones y de la base de datos serían pasos importantes para garantizar la disponibilidad y el rendimiento en escenarios de alta concurrencia. La **monitorización proactiva** del rendimiento del servidor, la base de

datos y la aplicación (utilizando herramientas como Prometheus, Grafana, New Relic) permitirá detectar y resolver problemas antes de que afecten la experiencia del usuario, asegurando un servicio ininterrumpido y eficiente.

Asimismo, la **optimización del frontend** es un factor importante para la percepción del rendimiento. Se recomienda la minificación y concatenación de archivos JavaScript y CSS, la compresión de imágenes y el uso de un Content Delivery Network (CDN) para servir activos estáticos. La implementación de técnicas de "lazy loading" para componentes o datos que no son críticos en la carga inicial de la página puede mejorar significativamente el tiempo de primera pintura y la interactividad percibida por el usuario. La revisión de la eficiencia de los componentes de Vue.js y la optimización de las llamadas a la API para reducir el número de solicitudes HTTP y el tamaño de las respuestas JSON también contribuirán a una experiencia de usuario más fluida y rápida, lo cual es fundamental para la satisfacción del usuario en un sistema web interactivo.

9.1.3 Fortalecimiento de la Seguridad

La seguridad es un aspecto crítico en cualquier sistema que maneje información sensible, como lo son las propuestas de proyectos académicos y los datos de estudiantes y docentes. Aunque se han implementado medidas de seguridad robustas durante el desarrollo (autenticación, autorización, validación de entradas), es vital adoptar una postura de **seguridad proactiva y continua**. Se recomienda la implementación de auditorías de seguridad periódicas, incluyendo pruebas de penetración (penetration testing) realizadas por terceros especializados. Estas pruebas pueden identificar vulnerabilidades que no fueron detectadas durante el desarrollo interno, como inyecciones SQL avanzadas, scripting entre sitios (XSS) o vulnerabilidades de configuración del servidor.

Una medida esencial es la **actualización constante de las dependencias y frameworks** utilizados (Laravel, Vue.js, PHP, MySQL). Los desarrolladores de estos componentes lanzan regularmente parches de seguridad para corregir vulnerabilidades conocidas. Establecer un proceso automatizado para la detección y aplicación de estas actualizaciones, junto con pruebas de regresión exhaustivas, es fundamental para proteger el sistema contra ataques. Además, se debe considerar la implementación de un **Web Application Firewall (WAF)** que actúe como una capa de seguridad adicional, filtrando el tráfico malicioso antes de que llegue a la aplicación. Un WAF puede proteger contra ataques comunes como OWASP Top 10 y ofrecer una defensa robusta contra amenazas emergentes.

Finalmente, se recomienda fortalecer las **políticas de gestión de accesos y roles**. Revisar periódicamente los permisos asignados a cada rol y usuario para asegurar el principio de mínimo privilegio, garantizando que los usuarios solo tengan acceso a los recursos

estrictamente necesarios para sus funciones. La implementación de la **autenticación de dos factores (2FA)** para roles críticos (administradores, miembros del comité) añadiría una capa de seguridad significativa, dificultando el acceso no autorizado incluso si las credenciales primarias son comprometidas. La encriptación de datos sensibles en reposo y en tránsito, utilizando protocolos como HTTPS y encriptación de campos específicos en la base de datos, es otra recomendación crucial para proteger la privacidad y la integridad de la información almacenada en el sistema. La capacitación continua del personal sobre las mejores prácticas de seguridad también es un componente indispensable para mantener un entorno digital seguro.

9.1.4 Mejoras en la Experiencia de Usuario (UX)

Una interfaz de usuario intuitiva y una experiencia de usuario fluida son fundamentales para la adopción y el éxito a largo plazo del sistema. Aunque se ha puesto énfasis en el diseño UX/UI durante el desarrollo, siempre hay margen para la mejora continua basada en la retroalimentación de los usuarios. Se recomienda realizar **sesiones de usabilidad periódicas** con diferentes perfiles de usuario (estudiantes, docentes, administradores) para identificar puntos de fricción, confusiones o tareas que podrían simplificarse. La observación directa de cómo los usuarios interactúan con el sistema y la recopilación de sus comentarios cualitativos son invaluables para refinar el diseño.

La **personalización de la interfaz** podría mejorar la eficiencia para usuarios avanzados. Por ejemplo, permitir a los usuarios configurar paneles de control con la información más relevante para sus roles o personalizar las notificaciones que reciben. La implementación de **tutoriales interactivos o guías contextuales** dentro del sistema ("onboarding tours") para nuevas funcionalidades o para usuarios nuevos, facilitaría la curva de aprendizaje y reduciría la necesidad de soporte técnico. Estos tutoriales podrían guiar al usuario paso a paso a través de las tareas clave, mostrando las características del sistema de manera práctica y efectiva.

Además, se sugiere la **optimización de la accesibilidad** del sistema para cumplir con estándares como WCAG (Web Content Accessibility Guidelines). Esto incluye aspectos como el contraste de colores, el tamaño de la fuente, la navegación por teclado y la compatibilidad con lectores de pantalla. Un sistema accesible garantiza que todos los usuarios, incluyendo aquellos con discapacidades, puedan interactuar de manera efectiva con la plataforma. La mejora continua del diseño visual, manteniendo una estética moderna y profesional, así como la coherencia en la interacción en todas las secciones del sistema, contribuirán a una experiencia de usuario excepcional que fomente la satisfacción y la adopción plena del sistema por parte de la comunidad académica.

9.2 Recomendaciones para la Institución y el Proceso Académico

La implementación de un sistema web como este no es solo un cambio tecnológico, sino una transformación en la forma en que la institución gestiona un proceso crítico. Para maximizar sus beneficios, la institución debe adoptar una serie de medidas estratégicas y operativas.

9.2.1 Capacitación y Adopción del Sistema

La **capacitación integral** es fundamental para asegurar una transición exitosa y una adopción efectiva del nuevo sistema. Se recomienda diseñar e implementar programas de capacitación diferenciados para cada perfil de usuario: estudiantes, docentes (tutores y revisores) y personal administrativo. Para los estudiantes, la capacitación debe enfocarse en cómo presentar propuestas, dar seguimiento a sus proyectos y comunicarse con sus tutores a través de la plataforma. Para los docentes, la formación debe cubrir la revisión de propuestas, la asignación de puntajes, la provisión de retroalimentación y la gestión de sus paneles de proyectos asignados. Para el personal administrativo, la capacitación debe ser más profunda, abarcando la administración de usuarios, la gestión de proyectos, la generación de informes y la resolución de problemas comunes.

Estos programas de capacitación deben incluir **materiales didácticos claros y concisos**, como manuales de usuario, videos tutoriales y preguntas frecuentes (FAQ) accesibles dentro del propio sistema o en un portal de soporte dedicado. Es crucial que la capacitación no sea un evento único, sino un proceso continuo, con sesiones de refuerzo y actualizaciones a medida que el sistema evolucione. La creación de un **equipo de soporte técnico interno** dedicado a resolver dudas y problemas relacionados con el sistema es igualmente importante. Este equipo debe estar bien capacitado y tener un canal de comunicación claro con los usuarios para ofrecer asistencia oportuna y efectiva, lo que contribuirá significativamente a la confianza y satisfacción de los usuarios con la plataforma.

Además, se debe fomentar la **adopción activa del sistema** mediante políticas claras que establezcan el uso obligatorio de la plataforma para todos los procesos de gestión de proyectos académicos. La comunicación constante sobre los beneficios del sistema y la resolución de cualquier resistencia al cambio son aspectos clave. La implementación de un **piloto controlado** antes del despliegue completo, donde un grupo reducido de usuarios pruebe el sistema en un entorno real, puede ayudar a identificar problemas y ajustar la capacitación antes de un lanzamiento masivo, asegurando así una integración más suave y exitosa en la rutina académica de la institución.

9.2.2 Revisión de Políticas y Normativas

La implementación de un sistema automatizado para la gestión de proyectos académicos a menudo revela la necesidad de **revisar y actualizar las políticas y normativas institucionales** existentes. El proceso manual anterior pudo haber tenido ambigüedades o pasos implícitos que ahora deben formalizarse y adaptarse a la lógica del sistema. Se recomienda la creación de un comité interdisciplinario, compuesto por representantes de la coordinación de carrera, docentes, estudiantes y el equipo de desarrollo del sistema, para llevar a cabo esta revisión.

Esta revisión debe enfocarse en **clarificar los roles y responsabilidades** de cada actor dentro del nuevo proceso digital. Por ejemplo, definir explícitamente los tiempos de respuesta para los revisores, los criterios de evaluación estandarizados que se reflejarán en el sistema, y los flujos de aprobación específicos para diferentes tipos de proyectos. La digitalización del proceso permite una mayor trazabilidad y transparencia, pero esto debe estar respaldado por normativas que validen la validez legal y académica de los registros electrónicos y las firmas digitales (si se implementan). La actualización del reglamento de tesis y de cualquier otra normativa relacionada es crucial para que el marco legal y administrativo de la institución esté en sintonía con la funcionalidad y las capacidades del nuevo sistema.

Además, se deben establecer **políticas claras sobre la gestión de datos y la privacidad** dentro del sistema, en cumplimiento con las leyes de protección de datos vigentes. Esto incluye definir cómo se almacenan, acceden y eliminan los datos de los estudiantes y docentes, y asegurar que el sistema cumpla con los estándares de seguridad para proteger esta información. La formalización de los procedimientos para la **resolución de conflictos o apelaciones** dentro del contexto del sistema también es importante, garantizando que el sistema apoye, y no dificulte, la justicia y la equidad en el proceso de aceptación de proyectos. Una alineación robusta entre la tecnología y la normativa institucional es esencial para el éxito y la legitimidad a largo plazo de la plataforma.

9.2.3 Integración con Otros Sistemas Institucionales

Para maximizar el valor del sistema de gestión de proyectos académicos, se recomienda explorar y planificar su **integración con otros sistemas informáticos clave** de la institución. La visión a largo plazo debe ser un ecosistema de software universitario interconectado que evite la duplicación de datos y optimice los flujos de trabajo. Una de las integraciones más importantes sería con el **Sistema de Gestión Académica (SGA)** de la universidad. Esto permitiría importar automáticamente la información de los estudiantes (datos personales, historial académico) y docentes (datos de contacto, especialidades)

directamente al sistema de proyectos, eliminando la necesidad de entrada manual de datos y reduciendo errores. Asimismo, el sistema de proyectos podría enviar información de proyectos aceptados y sus calificaciones al SGA para la gestión de registros académicos.

Otra integración beneficiosa sería con los **sistemas de correo electrónico institucionales** para una gestión de notificaciones más robusta y centralizada. Aunque el sistema ya envía notificaciones internas, la integración con el correo electrónico permitiría enviar alertas y recordatorios directamente a las bandejas de entrada de los usuarios, asegurando que la información crítica no se pierda. La posibilidad de integrar el sistema con un **repositorio institucional de tesis digital** (por ejemplo, DSpace o Eprints) permitiría que, una vez finalizada y aprobada una tesis, el sistema de gestión de proyectos pueda facilitar su publicación automática o semi-automática en el repositorio, agilizando el proceso de difusión del conocimiento generado y garantizando el cumplimiento de las políticas de acceso abierto de la universidad.

Finalmente, la integración con **sistemas de autenticación centralizados** (como LDAP o Single Sign-On - SSO) mejoraría la seguridad y la experiencia del usuario al permitirles acceder al sistema de proyectos con las mismas credenciales que usan para otros servicios universitarios. Esto simplificaría la gestión de usuarios y reduciría la fatiga de contraseñas. Cada una de estas integraciones debe ser planificada cuidadosamente, priorizando aquellas que ofrecen el mayor retorno de inversión en términos de eficiencia y reducción de la carga administrativa, y asegurando que se utilicen APIs seguras y estándares de interoperabilidad para garantizar una comunicación fluida y confiable entre los diferentes sistemas.

9.3 Líneas de Investigación Futuras

El presente trabajo sienta una base sólida para la optimización de la gestión de proyectos académicos. Sin embargo, abre la puerta a diversas líneas de investigación y desarrollo que podrían explorar nuevas fronteras tecnológicas y metodológicas, llevando la automatización y la inteligencia en este proceso a un nivel superior.

9.3.1 Implementación de Inteligencia Artificial para la Revisión Preliminar

Una de las áreas más prometedoras para futuras investigaciones es la **integración de técnicas de inteligencia artificial (IA) y procesamiento de lenguaje natural (NLP) para la revisión preliminar de propuestas de proyectos**. Actualmente, la revisión inicial de una propuesta requiere una lectura detallada por parte de los miembros del comité, un proceso que consume tiempo y recursos. Un sistema basado en IA podría automatizar o asistir significativamente esta fase. Por ejemplo, se podría desarrollar un algoritmo capaz de analizar el texto de una propuesta para identificar la originalidad del tema,

comparándolo con bases de datos de tesis existentes y publicaciones académicas. Esto ayudaría a detectar posibles duplicidades o áreas de investigación ya saturadas, proporcionando una evaluación inicial sobre la novedad y relevancia de la propuesta.

Además, la IA podría ser utilizada para **evaluar la coherencia interna de la propuesta**, verificando la alineación entre los objetivos, la metodología y los resultados esperados. Un modelo de NLP podría identificar palabras clave, extractos de texto y la estructura del documento para generar un resumen automático o resaltar secciones que requieran mayor claridad o justificación. Incluso, se podría entrenar un modelo para ofrecer **sugerencias de mejora en la redacción académica**, identificando errores gramaticales, inconsistencias terminológicas o áreas donde la argumentación podría ser fortalecida. Estas herramientas no reemplazarían la evaluación humana, sino que actuarían como un asistente inteligente para los revisores, pre-filtrando y pre-analizando las propuestas, permitiéndoles enfocar su tiempo y experiencia en los aspectos más complejos y cualitativos de la evaluación, acelerando el proceso y mejorando su objetividad al proporcionar métricas y análisis basados en datos.

La investigación en esta área implicaría la recopilación y etiquetado de un corpus de propuestas de tesis (aceptadas y rechazadas) para entrenar modelos de aprendizaje automático. Se podrían explorar diferentes arquitecturas de redes neuronales, como RNNs o Transformers, para tareas de clasificación de texto, detección de similitudes y generación de resúmenes. Los desafíos incluirían la necesidad de grandes volúmenes de datos de entrenamiento, la interpretabilidad de los modelos de IA y la garantía de que los sesgos inherentes a los datos de entrenamiento no se propaguen al proceso de evaluación, manteniendo la equidad y la imparcialidad. Sin embargo, el potencial para transformar radicalmente la eficiencia y la calidad de la revisión preliminar es considerable, haciendo de esta una línea de investigación de alto impacto para la gestión académica.

9.3.2 Desarrollo de Módulos de Colaboración Avanzada

El sistema actual facilita la comunicación a través de notificaciones y comentarios, pero las futuras investigaciones podrían centrarse en el **desarrollo de módulos de colaboración en tiempo real y herramientas de edición conjunta**. La elaboración de una propuesta de tesis, y posteriormente la tesis misma, es un proceso altamente colaborativo entre el estudiante y su tutor, y a veces entre múltiples estudiantes. La integración de un editor de texto enriquecido que permita la edición colaborativa en línea de documentos, similar a Google Docs, directamente dentro del sistema, revolucionaría la forma en que los estudiantes y tutores trabajan juntos. Esto eliminaría la necesidad de intercambiar versiones de documentos por correo electrónico, reduciendo la confusión y asegurando que todos los involucrados estén siempre trabajando en la última versión.

Estos módulos de colaboración avanzada podrían incluir funcionalidades como el **control de versiones integrado** a nivel de párrafo o sección, permitiendo a los usuarios ver el historial de cambios, revertir a versiones anteriores y comparar revisiones de manera sencilla. La implementación de **comentarios contextuales** que puedan ser anclados a secciones específicas del documento facilitaría una retroalimentación más precisa y accionable. Además, se podrían integrar **herramientas de comunicación sincrónica**, como chats de texto o incluso videoconferencias ligeras, directamente en el entorno de colaboración, permitiendo discusiones rápidas y la resolución de dudas en tiempo real sin salir de la plataforma.

Otra área de investigación en colaboración podría ser la integración de herramientas para la **gestión de tareas y hitos dentro del proyecto de tesis**. Esto permitiría a los estudiantes y tutores desglosar el proyecto en tareas más pequeñas, asignar responsabilidades, establecer fechas límite y dar seguimiento al progreso de cada tarea. La visualización de este progreso a través de diagramas de Gantt o tableros Kanban integrados en el sistema proporcionaría una visión clara del estado del proyecto y ayudaría a mantener a todos los miembros del equipo alineados y enfocados. El desarrollo de estas características no solo mejoraría la eficiencia de la colaboración, sino que también ofrecería una experiencia de usuario más rica y cohesiva, transformando el sistema en una plataforma integral de gestión y desarrollo de tesis, y no solo de aceptación de proyectos.

9.3.3 Estudio Comparativo de Metodologías de Aceptación de Proyectos

Finalmente, una línea de investigación valiosa sería la realización de un **estudio comparativo de diferentes metodologías y sistemas de aceptación de proyectos académicos** en diversas instituciones educativas. El presente trabajo se centró en la implementación de una solución específica para una problemática particular. Sin embargo, comprender cómo otras universidades abordan este mismo desafío, qué tecnologías utilizan y qué procesos han implementado, podría proporcionar información valiosa para la mejora continua y la identificación de mejores prácticas.

Este estudio comparativo podría analizar aspectos como la **eficiencia de los procesos**, los **tiempos promedio de respuesta** para la aceptación de propuestas, la **satisfacción de los usuarios** (estudiantes y docentes) con los sistemas existentes, y las **políticas institucionales** que respaldan estos procesos. Se podrían utilizar métodos de investigación cualitativos (entrevistas, encuestas) y cuantitativos (análisis de datos de sistemas) para recopilar la información. Por ejemplo, se podría comparar la efectividad de un sistema basado en un enfoque Waterfall versus uno basado en metodologías ágiles para la gestión de proyectos académicos, o evaluar el impacto de la incorporación de criterios de evaluación estandarizados versus enfoques más flexibles.

La identificación de **factores de éxito y fracaso** en la implementación de sistemas similares en otras instituciones proporcionaría un marco de referencia invaluable. Este tipo de investigación no solo enriquecería el conocimiento sobre la gestión de proyectos académicos, sino que también podría ofrecer recomendaciones basadas en evidencia para la adaptación y mejora del sistema desarrollado en esta tesis. Los hallazgos de un estudio comparativo podrían influir en la toma de decisiones estratégicas a nivel institucional, guiando futuras inversiones en tecnología y la revisión de políticas, asegurando que el sistema de gestión de proyectos académicos no solo sea eficiente, sino también competitivo y alineado con las mejores prácticas a nivel nacional e internacional. Esta investigación podría incluso extenderse a un análisis de la adaptabilidad cultural de estos sistemas en diferentes contextos educativos, explorando cómo las variaciones en las estructuras académicas y las normativas influyen en el diseño y la aceptación de soluciones tecnológicas.

ÍNDICE GENERAL

Hasta este punto de la tesis, se ha presentado un análisis exhaustivo del problema de la gestión manual de proyectos académicos en la carrera de Ingeniería en Informática, delineando las ineficiencias y desafíos inherentes a este enfoque tradicional. Se ha detallado

la propuesta de solución a través del diseño y la implementación de un **Sistema Web para la Gestión del Proceso de Aceptación de Proyectos Académicos**, explicando las fases de diagnóstico, diseño, implementación y pruebas. Se han expuesto las conclusiones alcanzadas, confirmando el logro de los objetivos planteados y las contribuciones significativas del sistema desarrollado, así como sus limitaciones. Finalmente, se han proporcionado recomendaciones para futuras investigaciones y para la implementación y mantenimiento del sistema.

La presente sección, **10. Referencias**, constituye un pilar fundamental del rigor académico de esta investigación. Su objetivo es listar de manera exhaustiva y sistemática todas las fuentes bibliográficas, documentos, artículos científicos, estándares y recursos en línea que han servido de fundamento teórico y metodológico para el desarrollo de esta tesis. La correcta citación y referenciación no solo valida los argumentos presentados, sino que también permite a futuros investigadores rastrear y verificar la información utilizada, garantizando la transparencia y la reproducibilidad del conocimiento. Esta sección demuestra la base sólida sobre la cual se ha construido cada argumento, cada decisión de diseño y cada afirmación en el desarrollo del sistema web.

10. REFERENCIAS

La construcción de esta tesis se ha sustentado en una base robusta de conocimiento científico y técnico, abarcando diversas áreas como la ingeniería de software, la gestión de proyectos, las arquitecturas web y las metodologías de desarrollo. Las siguientes referencias han sido cruciales para el marco teórico, el diseño metodológico y la fundamentación de las decisiones técnicas tomadas a lo largo del proyecto. Se presentan siguiendo las directrices del formato APA.

- Pressman, R. S. (2010). **Ingeniería del software: Un enfoque práctico** (7ma ed.). McGraw-Hill Interamericana.
- Sommerville, I. (2011). **Ingeniería del software** (9na ed.). Pearson Educación.
- Larman, C. (2004). **UML y Patrones: Una introducción al análisis y diseño orientado a objetos** (3ra ed.). Pearson Educación.
- Deitel, P. J., & Deitel, H. M. (2012). **Cómo programar en Java** (9na ed.). Pearson Educación.
- W3C. (2023). **World Wide Web Consortium (W3C)**. Recuperado de https://www.w3.org/
- Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1995). **Design Patterns: Elements of Reusable Object-Oriented Software**. Addison-Wesley.
- Kruchten, P. (2003). **The Rational Unified Process: An Introduction** (3ra ed.). Addison-Wesley.
- Schwaber, K., & Sutherland, J. (2020). **La Guía de Scrum: La Guía Definitiva de Scrum: Las Reglas del Juego**. Scrum.org.
- ISO/IEC/IEEE 29119-1:2013. (2013). **Software and systems engineering -- Software testing -- Part 1: Concepts and definitions**. International Organization for Standardization.
- IEEE. (2010). **IEEE Standard Glossary of Software Engineering Terminology** (IEEE Std 610.12-1990). Institute of Electrical and Electronics Engineers.
- Cockburn, A. (2002). **Agile Software Development**. Addison-Wesley Professional.
- Fowler, M. (2004). **UML Distilled: A Brief Guide to the Standard Object Modeling Language** (3ra ed.). Addison-Wesley Professional.

- MySQL. (2023). **MySQL Documentation**. Recuperado de https://dev.mysql.com/doc/
- Laravel. (2023). **Laravel Documentation**. Recuperado de https://laravel.com/docs
- Vue.js. (2023). **Vue.js Documentation**. Recuperado de https://vuejs.org/guide/
- Rodríguez, A., & Pérez, M. (2019). **Sistemas de información para la gestión académica en entornos universitarios**. Revista de Investigación en Tecnologías de la Información, 7(2), 45-60.
- González, L., & Castro, R. (2020). **Optimización de procesos administrativos mediante plataformas web: Un estudio de caso en educación superior**. Cuadernos de Gestión Tecnológica, 12(1), 78-92.
- Smith, J. (2018). **The Importance of User Experience in Web Application Design**. International Journal of Human-Computer Studies, 110, 1-12.
- Chen, L., & Wang, Q. (2021). **Security Considerations in Web-Based Academic Management Systems**. Journal of Information Security and Applications, 56, 102652.
- García, D. (2017). **Metodologías ágiles en el desarrollo de software: Scrum y Kanban**. Editorial Universitaria.
- ISO/IEC 25010:2011. (2011). **Systems and software engineering -- Systems and software Quality Requirements and Evaluation (SQuaRE) -- System and software quality models**. International Organization for Standardization.
- Beck, K. (2000). **Extreme Programming Explained: Embrace Change**. Addison-Wesley.
- McConnell, S. (2004). **Code Complete** (2da ed.). Microsoft Press.
- Cochran, W. G. (1977). **Sampling Techniques** (3ra ed.). John Wiley & Sons. (Aunque no es directamente de software, es fundamental para la sección de pruebas y validación si se usó muestreo).
- Nielsen, J. (1994). **Usability Engineering**. Morgan Kaufmann.
- Shneiderman, B., & Plaisant, C. (2010). **Designing the User Interface: Strategies for Effective Human-Computer Interaction** (5ta ed.). Addison-Wesley.

Cada una de estas fuentes ha aportado una perspectiva valiosa y un conocimiento específico que ha enriquecido la comprensión de los desafíos y las soluciones propuestas en el ámbito de la gestión de proyectos académicos y el desarrollo de sistemas web. La diversidad de las

referencias, que incluye libros seminales, estándares internacionales y artículos de investigación recientes, asegura la solidez y actualidad de los fundamentos de esta tesis.