DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA WEB PARA LA GESTIÓN Y ACEPTACIÓN DE PROYECTOS DE TESIS

PROYECTO GRADO II

Yondayler Flores

floresyondayler@gmail.com

Ingeniería en Informática

Facultad de Ciencias de la Ingeniería

San Juan de los Morros - diciembre de 2025

ÍNDICE

3. RESUMEN

La presente tesis aborda la problemática de la gestión manual y descentralizada de los proyectos de tesis de grado en la carrera de Ingeniería en Informática, un proceso caracterizado por su lentitud, falta de transparencia y propensión a errores administrativos. El **objetivo principal** de esta investigación fue diseñar, desarrollar e implementar un sistema web integral que automatice y optimice el ciclo de vida de las propuestas de tesis, desde su presentación inicial por parte del estudiante hasta su aceptación formal por parte del comité académico, mejorando significativamente la eficiencia y la trazabilidad del proceso.

Para la consecución de este objetivo, se empleó una **metodología de desarrollo ágil**, específicamente Scrum, que facilitó la construcción incremental y adaptativa del software, permitiendo ajustes continuos basados en la retroalimentación de los usuarios clave. La arquitectura del sistema se fundamentó en el patrón **Modelo-Vista-Controlador (MVC)**, utilizando un conjunto de tecnologías modernas y robustas: PHP con el framework Laravel para el desarrollo del backend, MySQL como sistema de gestión de bases de datos relacional, y JavaScript con el framework Vue.js para la creación de una interfaz de usuario dinámica, reactiva e intuitiva en el frontend.

Los **resultados** obtenidos demuestran la viabilidad y eficacia del sistema propuesto. Se logró implementar una plataforma que centraliza la información, automatiza las notificaciones, facilita la revisión y evaluación por parte de los docentes, y proporciona un seguimiento en tiempo real del estado de cada proyecto de tesis. Esto se traduce en una reducción sustancial de los tiempos de procesamiento, una mayor transparencia en la toma de decisiones y una disminución de la carga administrativa para estudiantes y personal académico. La interfaz de usuario intuitiva y la robustez del backend garantizan una experiencia de usuario óptima y la integridad de los datos.

En **conclusión**, el sistema web desarrollado representa una solución tecnológica innovadora y pertinente para la gestión de proyectos de tesis, alineándose con las necesidades de digitalización de los procesos académicos. Su implementación no solo optimiza la eficiencia operativa, sino que también contribuye a una experiencia académica más fluida y transparente para todos los actores involucrados, sentando las bases para futuras mejoras y expansiones en la gestión académica digital.

4. DIAGNÓSTICO SITUACIONAL

4.1 Descripción del contexto de la situación problemática planteada

La gestión de los proyectos de tesis de grado en las instituciones de educación superior representa un pilar fundamental para la culminación exitosa de los estudios universitarios y la formación de profesionales competentes. Sin embargo, en el contexto específico de la carrera de Ingeniería en Informática, la administración de este proceso ha estado históricamente marcada por una serie de desafíos inherentes a su naturaleza manual y descentralizada. La Universidad, como entidad formadora, se enfrenta anualmente a un volumen considerable de estudiantes que inician la fase de elaboración de sus proyectos de tesis, lo que implica la necesidad de un sistema robusto y eficiente para gestionar las propuestas, asignaciones, revisiones y aprobaciones. Actualmente, el proceso se caracteriza por una dependencia excesiva de documentos físicos, comunicaciones fragmentadas y una carencia de herramientas tecnológicas integradas que soporten el ciclo de vida completo de una propuesta de tesis.

El entorno actual de la gestión de proyectos de tesis se describe como un sistema predominantemente analógico, donde los estudiantes deben presentar sus propuestas en formatos impresos, a menudo en múltiples copias, para su posterior distribución entre los miembros del comité académico o los potenciales tutores. Esta fase inicial ya introduce una serie de ineficiencias, como la posibilidad de pérdida de documentos, la dificultad para realizar un seguimiento del estado de cada propuesta y la generación de una considerable carga administrativa para el personal de la secretaría académica. La revisión de estas propuestas, un paso crítico para asegurar la calidad y pertinencia de los temas de investigación, se realiza de manera secuencial y, en muchos casos, sin un mecanismo centralizado para el registro de observaciones o la consolidación de dictámenes. Los docentes, en su rol de revisores, reciben las propuestas de forma individual, emiten sus juicios y los devuelven a la secretaría, donde se intenta compilar la información para tomar una decisión final.

Esta metodología manual conlleva a una serie de problemáticas significativas. En primer lugar, la **lentitud del proceso** es una de las quejas más recurrentes. Los tiempos de espera para que una propuesta sea revisada, aprobada o rechazada pueden extenderse por semanas o incluso meses, lo que retrasa el inicio de la fase de desarrollo de la tesis y, consecuentemente, la graduación de los estudiantes. Esta dilación no solo genera frustración en el alumnado, sino que también impacta negativamente en la planificación académica y administrativa de la carrera. En segundo lugar, la **falta de transparencia** es un

problema inherente a la gestión descentralizada. Los estudiantes carecen de una visibilidad clara sobre el estado de su propuesta, quién la está revisando, cuáles son los criterios de evaluación aplicados o cuándo pueden esperar una respuesta. Esta opacidad genera incertidumbre y dificulta la comunicación efectiva entre los estudiantes, los docentes y la administración.

Adicionalmente, la propensión a **errores administrativos** es elevada. La manipulación manual de grandes volúmenes de documentos aumenta el riesgo de extravío, la duplicidad de información o la inconsistencia en los datos registrados. La asignación de tutores, por ejemplo, se realiza a menudo de forma empírica, sin considerar la carga de trabajo de los docentes o sus áreas de especialización de manera sistemática, lo que puede llevar a un desequilibrio en la distribución de responsabilidades y a una subutilización de los recursos humanos. La ausencia de un repositorio centralizado y digitalizado de las propuestas y sus dictámenes dificulta la trazabilidad del proceso, impidiendo auditorías eficientes o la generación de estadísticas fiables sobre la gestión de tesis a lo largo del tiempo. La comunicación entre los diferentes actores (estudiantes, tutores, comité académico) es a menudo ineficiente, basándose en correos electrónicos dispersos o reuniones presenciales que consumen tiempo y recursos. Esta situación no solo afecta la eficiencia operativa, sino que también puede mermar la calidad de las propuestas de tesis al no facilitar un ciclo de retroalimentación estructurado y oportuno.

En resumen, el contexto actual de la gestión de proyectos de tesis en la carrera de Ingeniería en Informática se caracteriza por ser un sistema obsoleto, ineficiente y propenso a errores, que no se alinea con las capacidades tecnológicas y las expectativas de una carrera orientada a la innovación y el desarrollo de software. La necesidad de una solución que aborde estas deficiencias es imperativa para modernizar el proceso, mejorar la experiencia de los estudiantes y optimizar la labor del personal académico y administrativo.

4.2 Justificación del proyecto

La justificación de este proyecto de tesis se fundamenta en la imperiosa necesidad de transformar y modernizar la gestión de proyectos de tesis en la carrera de Ingeniería en Informática, abordando directamente las deficiencias y problemáticas identificadas en el diagnóstico situacional. La persistencia de un sistema manual y descentralizado no solo genera ineficiencias operativas, sino que también impacta negativamente en la calidad académica, la experiencia estudiantil y la optimización de los recursos institucionales. La implementación de un sistema web integral no es meramente una mejora incremental, sino una **transformación estratégica** que busca alinear los procesos administrativos con los principios de eficiencia, transparencia y accesibilidad que caracterizan a la era digital.

En primer lugar, la **eficiencia operativa** es un pilar fundamental de esta justificación. La automatización de los procesos de recepción, revisión, aprobación y seguimiento de las propuestas de tesis reducirá drásticamente los tiempos de espera, eliminando cuellos de botella y agilizando el flujo de trabajo. Esto se traduce en una menor carga administrativa para el personal de la secretaría académica, quienes podrán dedicar su tiempo a tareas de mayor valor añadido, y en una optimización del tiempo de los docentes, quienes podrán concentrarse en la revisión académica en lugar de la gestión documental. La digitalización de los formularios y la estandarización de los flujos de trabajo minimizarán los errores humanos, garantizando la integridad y consistencia de la información.

En segundo lugar, la **transparencia y la trazabilidad** son aspectos críticos que el sistema web busca potenciar. Al proporcionar una plataforma centralizada, los estudiantes tendrán acceso en tiempo real al estado de sus propuestas, a los comentarios de los revisores y a las decisiones tomadas por el comité. Esta visibilidad no solo reduce la incertidumbre y la ansiedad de los estudiantes, sino que también fomenta un ambiente de confianza y equidad en el proceso. Para la administración, el sistema ofrecerá una trazabilidad completa de cada propuesta, desde su presentación hasta su aprobación final, facilitando auditorías, la generación de informes detallados y la toma de decisiones informadas basadas en datos concretos sobre el rendimiento del proceso.

La **mejora en la comunicación** es otro beneficio sustancial. El sistema web actuará como un canal de comunicación unificado y estructurado entre estudiantes, tutores, revisores y el comité académico. Las notificaciones automáticas sobre cambios de estado, solicitudes de información adicional o fechas límite asegurarán que todos los actores estén siempre informados, eliminando la dependencia de correos electrónicos dispersos o comunicaciones verbales que pueden llevar a malentendidos. Esta comunicación fluida y eficiente es esencial para un proceso tan colaborativo como la elaboración de una tesis.

Desde una perspectiva académica, este proyecto contribuye directamente a la **modernización de la gestión educativa** y a la **mejora de la calidad de las tesis**. Al estandarizar los criterios de evaluación y facilitar un feedback estructurado, el sistema puede elevar el nivel de las propuestas presentadas. Además, al ser un proyecto de desarrollo de software dentro de la carrera de Ingeniería en Informática, representa una oportunidad invaluable para aplicar los conocimientos teóricos y prácticos adquiridos, demostrando la capacidad de los estudiantes para diseñar e implementar soluciones tecnológicas complejas que resuelvan problemas reales. Esto no solo valida la formación recibida, sino que también sirve como un ejemplo práctico de cómo la tecnología puede ser un motor de cambio y mejora en el ámbito universitario.

Finalmente, la implementación de este sistema web se alinea con las tendencias actuales de digitalización y automatización en la educación superior, posicionando a la carrera de Ingeniería en Informática como un referente en la adopción de tecnologías para optimizar sus propios procesos. La inversión en este tipo de soluciones no solo beneficia a la comunidad universitaria actual, sino que sienta las bases para una gestión más eficiente y adaptable a futuros desafíos, contribuyendo a la reputación y el prestigio de la institución.

4.3 Objetivos del proyecto

La formulación de objetivos claros y medibles es fundamental para guiar el desarrollo de cualquier proyecto de investigación y desarrollo, asegurando que los esfuerzos se dirijan hacia la consecución de resultados concretos y relevantes. En el contexto de esta tesis, los objetivos se han estructurado para abordar de manera integral la problemática identificada en el diagnóstico situacional, buscando no solo resolver las deficiencias actuales, sino también establecer una plataforma robusta y escalable para la gestión futura de proyectos de tesis.

El **Objetivo General** de este proyecto es:

- **Diseñar, desarrollar e implementar un sistema web integral para la gestión y aceptación de proyectos de tesis en la carrera de Ingeniería en Informática, con el fin de optimizar la eficiencia, transparencia y trazabilidad del proceso, y mejorar la comunicación entre estudiantes, tutores y el comité académico.**

Este objetivo general encapsula la visión global del proyecto, abarcando desde la concepción hasta la puesta en marcha de la solución tecnológica. Para lograr este objetivo ambicioso, se han definido una serie de **Objetivos Específicos** que desglosan las tareas y metas en componentes más manejables y evaluables:

1. **Analizar los procesos actuales de gestión de proyectos de tesis:** Este objetivo inicial implica una investigación exhaustiva de los flujos de trabajo existentes, la identificación de los actores involucrados, los documentos utilizados y los puntos críticos que generan ineficiencias. Se buscará comprender a fondo las necesidades y requisitos de los usuarios finales (estudiantes, docentes, personal administrativo) para definir las funcionalidades esenciales del sistema. Esto incluirá la recopilación de información a través de entrevistas, encuestas y observación directa, con el fin de establecer una base sólida para el diseño del sistema.

2. **Diseñar la arquitectura del sistema web, la base de datos y la interfaz de usuario:** Una vez comprendidos los requisitos, se procederá al diseño detallado de la solución. Esto implica la definición de la arquitectura de software (por ejemplo, Modelo-Vista-Controlador), el diseño de la base de datos relacional para almacenar de manera eficiente toda la información relevante (propuestas, usuarios, estados, comentarios), y la creación de una interfaz de usuario (UI) intuitiva y amigable. El diseño de la UI se centrará en la usabilidad y la experiencia del usuario (UX), asegurando que el sistema sea fácil de aprender y utilizar para todos los perfiles de usuario. Se considerarán principios de seguridad desde la fase de diseño para proteger la información sensible.

3. **Desarrollar e implementar el sistema web utilizando tecnologías modernas:** Este objetivo se centra en la fase de construcción del software. Se utilizarán tecnologías de desarrollo web robustas y ampliamente adoptadas en la industria, como PHP con el framework Laravel para el desarrollo del backend, MySQL como sistema de gestión de bases de datos relacional para la persistencia de datos, y JavaScript con el framework Vue.js para la creación de una interfaz de usuario dinámica, reactiva e intuitiva en el frontend. La implementación seguirá las mejores prácticas de codificación y se realizará de manera modular para facilitar el mantenimiento y la escalabilidad futura.

4. **Realizar pruebas exhaustivas del sistema:** La calidad del software es primordial. Este objetivo implica la ejecución de diversas fases de pruebas para asegurar que el sistema funcione correctamente, cumpla con los requisitos definidos y sea robusto ante diferentes escenarios de uso. Se llevarán a cabo pruebas unitarias para verificar el funcionamiento de componentes individuales, pruebas de integración para asegurar la correcta interacción entre módulos, pruebas de sistema para evaluar el comportamiento global del sistema, y pruebas de aceptación por parte de los usuarios finales para validar que la solución satisface sus necesidades y expectativas en un entorno real.

5. **Desplegar el sistema web en un entorno de producción y capacitar a los usuarios finales:** Una vez que el sistema haya superado las pruebas y se considere estable, se procederá a su despliegue en un servidor de producción, haciéndolo accesible a los usuarios de la carrera de Ingeniería en Informática. Paralelamente, se diseñarán y ejecutarán programas de capacitación para los

diferentes perfiles de usuario (estudiantes, docentes, personal administrativo), asegurando que adquieran las habilidades necesarias para utilizar el sistema de manera efectiva y aprovechar al máximo sus funcionalidades.

6. **Evaluar la efectividad del sistema en la mejora de la eficiencia y transparencia del proceso de gestión de proyectos de tesis:** Finalmente, se realizará una evaluación post-implementación para medir el impacto real del sistema. Esto implicará la recopilación de métricas cuantitativas (tiempos de procesamiento, reducción de errores) y cualitativas (satisfacción del usuario, percepción de transparencia) para determinar en qué medida el sistema ha logrado los objetivos planteados y ha mejorado el proceso de gestión de tesis. Esta evaluación proporcionará retroalimentación valiosa para futuras mejoras y para validar la contribución del proyecto.

Estos objetivos específicos, al ser alcanzados, garantizarán la consecución del objetivo general, resultando en un sistema web funcional, eficiente y que aporte un valor significativo a la gestión académica de la carrera de Ingeniería en Informática.

4.4 Procesos que se van a automatizar

La automatización de procesos es el núcleo central de este proyecto, buscando transformar un sistema manual y propenso a errores en una plataforma digital eficiente y transparente. Los procesos identificados para ser automatizados abarcan el ciclo de vida completo de una propuesta de tesis, desde su concepción inicial por parte del estudiante hasta su aceptación formal y el seguimiento posterior. La digitalización de estos flujos de trabajo no solo optimizará la operatividad, sino que también proporcionará una base de datos estructurada para la toma de decisiones y la mejora continua.

Los principales procesos que serán objeto de automatización son los siguientes:

1. **Gestión de Propuestas de Tesis:**
    - **Registro y Envío de Propuestas:** Los estudiantes podrán registrar sus datos personales y académicos, y subir sus propuestas de tesis directamente a través del sistema web. Esto incluirá la carga de documentos en formatos estandarizados (PDF, DOCX), eliminando la necesidad de impresiones físicas. El sistema validará automáticamente los campos obligatorios y los formatos de archivo, reduciendo errores en la presentación inicial.

- **Validación Preliminar:** El sistema permitirá al personal administrativo realizar una validación inicial de las propuestas, verificando que cumplan con los requisitos formales mínimos antes de ser enviadas al comité académico o a los revisores. Esto puede incluir la verificación de la completitud de la información y la correcta adjunción de anexos.

- **Asignación de Revisores/Comité:** El sistema facilitará la asignación de revisores o miembros del comité a cada propuesta. Se podrá implementar un mecanismo que considere la especialidad de los docentes y su carga de trabajo actual para una distribución equitativa y pertinente de las revisiones.

2. **Revisión y Evaluación de Propuestas:**

    - **Acceso y Revisión en Línea:** Los revisores y miembros del comité tendrán acceso seguro a las propuestas asignadas a través de la plataforma. Podrán visualizar los documentos, añadir comentarios directamente en el sistema y registrar sus evaluaciones utilizando formularios estandarizados.

    - **Registro de Comentarios y Calificaciones:** El sistema permitirá a los revisores ingresar sus observaciones, sugerencias y calificaciones de manera estructurada. Esto facilitará la consolidación de feedback y la generación de un dictamen final.

    - **Generación de Dictámenes:** Basado en las evaluaciones de los revisores, el sistema podrá generar dictámenes automáticos (aprobado, rechazado, con observaciones, requiere ajustes mayores), agilizando la toma de decisiones y la comunicación de resultados.

3. **Comunicación y Notificaciones Automatizadas:**

    - **Notificaciones a Estudiantes:** El sistema enviará notificaciones automáticas a los estudiantes sobre el estado de su propuesta (recibida, en revisión, aprobada, rechazada, con observaciones), eliminando la incertidumbre y la necesidad de consultas manuales.

- **Notificaciones a Revisores y Tutores:** Los docentes recibirán alertas sobre nuevas asignaciones de revisión o tutoría, así como recordatorios de plazos.

- **Comunicación Interna:** Se implementarán funcionalidades para facilitar la comunicación interna entre los miembros del comité, tutores y personal administrativo, a través de mensajes internos o foros de discusión asociados a cada proyecto.

4. **Asignación y Gestión de Tutores:**

    - **Registro de Disponibilidad de Tutores:** Los docentes podrán registrar sus áreas de especialización y su disponibilidad para tutorías, lo que permitirá una asignación más informada y eficiente.

    - **Asignación de Tutores a Proyectos Aprobados:** Una vez que una propuesta es aprobada, el sistema facilitará la asignación de un tutor, considerando las preferencias del estudiante y la disponibilidad y especialización del docente.

    - **Seguimiento de la Carga de Tutores:** El sistema podrá monitorear la carga de trabajo de cada tutor, ayudando a distribuir equitativamente las asignaciones y evitando la sobrecarga de algunos docentes.

5. **Seguimiento y Monitoreo de Proyectos:**

    - **Visualización del Progreso:** Tanto estudiantes como administradores podrán visualizar el progreso de los proyectos de tesis a través de un panel de control intuitivo, que mostrará hitos, fechas límite y el estado actual.

    - **Registro de Hitos y Entregables:** Los tutores y estudiantes podrán registrar el cumplimiento de hitos importantes y la entrega de avances del proyecto, facilitando el seguimiento académico.

6. **Gestión de Usuarios y Roles:**

    - **Administración de Perfiles:** El sistema permitirá la creación y gestión de perfiles para diferentes tipos de usuarios (estudiantes, docentes, administradores, miembros del comité), cada uno con

sus permisos y funcionalidades específicas.

- **Control de Acceso Basado en Roles (RBAC):** Se implementará un robusto sistema de control de acceso que garantice que cada usuario solo pueda acceder a la información y funcionalidades para las que tiene autorización, asegurando la seguridad y privacidad de los datos.

7. **Generación de Informes y Estadísticas:**

   - **Informes Personalizados:** El sistema será capaz de generar informes detallados sobre el estado de las propuestas, tiempos de revisión, tasas de aprobación, distribución de tutores, entre otros.
   - **Estadísticas y Métricas:** Se proporcionarán paneles de control con estadísticas clave que permitan a la administración evaluar el rendimiento del proceso de gestión de tesis, identificar tendencias y tomar decisiones estratégicas para su mejora continua.

8. **Almacenamiento y Gestión Documental:**

   - **Repositorio Digital:** El sistema funcionará como un repositorio centralizado para todas las propuestas de tesis, dictámenes, anexos y documentos relacionados, eliminando la dependencia de archivos físicos y facilitando la búsqueda y recuperación de información.
   - **Control de Versiones:** Se podrá implementar un control de versiones para los documentos, permitiendo rastrear los cambios y acceder a versiones anteriores de las propuestas.

La automatización de estos procesos no solo busca la eficiencia operativa, sino que también sienta las bases para una gestión más estratégica de los proyectos de tesis, permitiendo a la carrera de Ingeniería en Informática optimizar sus recursos, mejorar la calidad de la investigación y ofrecer una experiencia académica superior a sus estudiantes.

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

La fase de determinación, instalación y configuración de las herramientas de desarrollo constituye un pilar fundamental en la construcción de cualquier sistema de información robusto y eficiente, especialmente en el contexto de un proyecto de tesis que busca resolver una problemática compleja como la gestión de proyectos de tesis. La selección adecuada de estas herramientas no solo impacta directamente en la productividad del equipo de desarrollo, sino que también define la escalabilidad, mantenibilidad y seguridad del producto final. En este capítulo, se detallan las decisiones técnicas cruciales que sustentaron el diseño y la implementación del sistema web para la gestión y aceptación de proyectos de tesis, abarcando desde la plataforma de desarrollo y la arquitectura del sistema hasta la selección del entorno específico y la metodología de trabajo adoptada. Cada elección fue cuidadosamente ponderada, buscando un equilibrio entre la eficiencia en el desarrollo, la robustez tecnológica y la capacidad de adaptación a futuros requerimientos, siempre con el objetivo de entregar una solución que no solo cumpla con los requisitos funcionales y no funcionales identificados en el diagnóstico situacional, sino que también se alinee con las mejores prácticas de la ingeniería de software.

5.1 Plataforma de Desarrollo

La elección de la plataforma de desarrollo es un paso crítico que sienta las bases tecnológicas sobre las cuales se construirá el sistema. Para el sistema de gestión y aceptación de proyectos de tesis, se optó por una combinación de tecnologías de código abierto ampliamente reconocidas por su robustez, flexibilidad y el vasto soporte de la comunidad, lo que garantiza una curva de aprendizaje eficiente y una gran cantidad de recursos disponibles para la resolución de problemas. En el **backend**, se seleccionó **PHP** como lenguaje de programación principal, complementado por el framework **Laravel**. PHP, en su versión 8.x, ofrece mejoras significativas en rendimiento y sintaxis, lo que lo convierte en una opción potente para el desarrollo de aplicaciones web empresariales. Laravel, por su parte, es un framework MVC (Modelo-Vista-Controlador) que facilita la creación de aplicaciones web complejas de manera rápida y estructurada, proporcionando una gran cantidad de funcionalidades preconstruidas como ORM (Eloquent), sistema de rutas, autenticación, y una API RESTful sencilla de implementar. La elección de Laravel se justificó por su capacidad para acelerar el desarrollo, su enfoque en la seguridad, su ecosistema de paquetes (Composer) y su sintaxis expresiva que promueve un código limpio y mantenible. Esta combinación permite construir una lógica de negocio sólida y una API eficiente para la comunicación con el frontend.

Para el **frontend**, la decisión recayó en **JavaScript** como lenguaje base, utilizando el framework **Vue.js**. JavaScript es el lenguaje estándar para el desarrollo web interactivo, y Vue.js se ha consolidado como una opción popular debido a su facilidad de aprendizaje, su rendimiento optimizado y su enfoque en la reactividad. Vue.js permite construir interfaces de usuario dinámicas y componentes reutilizables, lo que mejora la experiencia del usuario y la modularidad del código. Su integración con Laravel es fluida, permitiendo una arquitectura de aplicación de una sola página (SPA) o una aplicación híbrida donde Vue.js maneja la interactividad en el lado del cliente, comunicándose con el backend de Laravel a través de peticiones HTTP asíncronas (AJAX). Esta separación de responsabilidades entre el backend y el frontend no solo mejora la escalabilidad del sistema, sino que también facilita el trabajo en equipos y la especialización de los desarrolladores. La base de datos seleccionada para el almacenamiento persistente de la información fue **MySQL**, un sistema de gestión de bases de datos relacional (RDBMS) de código abierto, conocido por su fiabilidad, rendimiento y amplia adopción en la industria. MySQL es compatible con Laravel a través de su ORM Eloquent, lo que simplifica las operaciones de base de datos y permite una gestión eficiente de la información de usuarios, proyectos, estados y roles dentro del sistema.

5.2 Arquitectura del sistema de información

La arquitectura del sistema de información es el esqueleto conceptual que define la estructura, el comportamiento y la interacción de los componentes de software. Para el sistema de gestión y aceptación de proyectos de tesis, se adoptó una arquitectura basada en el patrón **Modelo-Vista-Controlador (MVC)**, complementada con un enfoque de **API RESTful** para la comunicación entre el frontend y el backend. El patrón MVC es fundamental para el desarrollo de aplicaciones web modernas, ya que promueve una clara separación de responsabilidades, lo que resulta en un código más organizado, mantenible y escalable. En esta arquitectura, el **Modelo** es responsable de la lógica de negocio y la gestión de datos. En el contexto de nuestro sistema, el Modelo interactúa directamente con la base de datos MySQL, encapsulando la lógica para la creación, lectura, actualización y eliminación (CRUD) de entidades como usuarios, proyectos de tesis, comités evaluadores, estados de aprobación y comentarios. Utilizando el ORM Eloquent de Laravel, los modelos representan las tablas de la base de datos y proporcionan una interfaz orientada a objetos para interactuar con ellas, abstrayendo la complejidad de las consultas SQL.

El **Controlador** actúa como un intermediario entre el Modelo y la Vista, procesando las solicitudes del usuario, interactuando con el Modelo para obtener o manipular datos, y seleccionando la Vista apropiada para mostrar la respuesta. En el sistema de tesis, los controladores de Laravel reciben las peticiones HTTP (GET, POST, PUT, DELETE) desde

el frontend, validan los datos de entrada, invocan los métodos correspondientes en los modelos para realizar las operaciones de negocio (ej. guardar un nuevo proyecto, actualizar el estado de una propuesta, obtener la lista de proyectos), y luego devuelven una respuesta en formato JSON al frontend. Esta respuesta JSON es consumida por la Vista para renderizar la interfaz de usuario. Finalmente, la **Vista** es la encargada de la presentación de los datos al usuario y de capturar sus interacciones. En nuestra arquitectura, la Vista se implementa principalmente con **Vue.js** en el lado del cliente. Vue.js consume los datos proporcionados por el Controlador (a través de la API RESTful) y los renderiza en componentes de interfaz de usuario interactivos. Esto incluye formularios para la presentación de proyectos, tablas para la visualización de estados, paneles de control para administradores y evaluadores, y elementos de navegación. La separación entre el backend (Laravel) que provee la API y el frontend (Vue.js) que consume esa API, permite que ambos puedan desarrollarse de forma independiente, facilitando la escalabilidad horizontal y la posibilidad de que el backend sirva a múltiples clientes (web, móvil, etc.) en el futuro.

5.3 Selección del entorno del sistema

La selección y configuración del entorno del sistema abarca tanto las herramientas utilizadas durante el ciclo de desarrollo como el ambiente donde la aplicación será desplegada y ejecutada en producción. Un entorno de desarrollo bien configurado es crucial para la eficiencia y la colaboración del equipo. Para el desarrollo local, se utilizó **Visual Studio Code (VS Code)** como Entorno de Desarrollo Integrado (IDE). VS Code es un editor de código ligero pero extremadamente potente, con una vasta colección de extensiones que mejoran la productividad para PHP, JavaScript, Vue.js, HTML, CSS y Git. Sus características incluyen resaltado de sintaxis, autocompletado inteligente, depuración integrada, integración con control de versiones y un terminal incorporado, lo que lo convierte en una herramienta indispensable para el desarrollo full-stack. Para gestionar las dependencias de PHP, se empleó **Composer**, el gestor de paquetes estándar para PHP. Composer permite declarar las librerías de las que depende el proyecto y las instala automáticamente, asegurando que todas las dependencias de Laravel y otras librerías de PHP estén correctamente configuradas y actualizadas. De manera similar, para las dependencias de JavaScript y Vue.js, se utilizó **npm (Node Package Manager)**, que facilita la instalación y gestión de paquetes frontend, así como la ejecución de scripts de construcción.

El entorno de desarrollo local se configuró utilizando **Laragon**, una solución de servidor web local que integra Apache, Nginx, MySQL, PHP y Node.js en un paquete fácil de usar para Windows. Laragon simplifica la creación de entornos de desarrollo aislados para múltiples proyectos, permitiendo a los desarrolladores trabajar en diferentes versiones de

PHP o bases de datos sin conflictos. Alternativamente, para entornos de desarrollo más complejos o para asegurar la paridad con el entorno de producción, se podría haber optado por **Docker**, una plataforma de contenedores que permite empaquetar la aplicación y todas sus dependencias en un contenedor aislado, garantizando que el software se ejecute de la misma manera en cualquier entorno. Para el control de versiones del código fuente, se implementó **Git**, un sistema de control de versiones distribuido. Git es esencial para la colaboración en equipo, permitiendo a múltiples desarrolladores trabajar en el mismo proyecto de forma concurrente, fusionar cambios, revertir a versiones anteriores y gestionar ramas de desarrollo. Se utilizó una plataforma de alojamiento de repositorios como **GitHub** o **GitLab** para almacenar el código de forma remota, facilitando la integración continua y el despliegue continuo (CI/CD) en etapas posteriores. Finalmente, para el entorno de producción, se prevé el despliegue en un servidor **Linux** (preferiblemente Ubuntu o CentOS) con **Nginx** como servidor web, **PHP-FPM **para procesar las solicitudes PHP, y** MySQL** como servidor de base de datos. Esta configuración es estándar en la industria para aplicaciones Laravel y Vue.js, ofreciendo un alto rendimiento, seguridad y escalabilidad.

5.4 Metodología para el desarrollo

La elección de una metodología de desarrollo adecuada es tan crucial como la selección de las herramientas tecnológicas, ya que define el marco de trabajo para la planificación, ejecución y gestión del proyecto. Para el diseño e implementación del sistema web de gestión y aceptación de proyectos de tesis, se adoptó la metodología **Scrum**, un marco de trabajo ágil e iterativo que se alinea perfectamente con la naturaleza evolutiva y los requisitos cambiantes de los proyectos de desarrollo de software. Scrum se caracteriza por su enfoque empírico, su capacidad de adaptación y su énfasis en la entrega incremental de valor, lo cual es fundamental para un proyecto que busca optimizar un proceso académico complejo y que requiere la retroalimentación constante de los usuarios finales (estudiantes, docentes, administradores). La implementación de Scrum se estructuró en torno a los siguientes pilares:

1. **Roles de Scrum:**
    - **Product Owner:** Responsable de maximizar el valor del producto resultante del trabajo del Equipo de Desarrollo. En este proyecto, el Product Owner representaría a los stakeholders de la universidad (coordinadores de tesis, jefes de carrera) y sería el encargado de definir y priorizar los elementos del Product Backlog, asegurando que el equipo trabaje en las funcionalidades más importantes para el sistema.

- **Scrum Master:** Responsable de asegurar que Scrum sea entendido y adoptado. Actúa como un facilitador, eliminando impedimentos, protegiendo al equipo de interrupciones externas y guiando al equipo en la auto-organización y la mejora continua.

- **Equipo de Desarrollo:** Compuesto por los desarrolladores encargados de construir el incremento del producto en cada Sprint. Este equipo es auto-organizado y multifuncional, con todas las habilidades necesarias para entregar un incremento de software funcional al final de cada iteración.

2. **Eventos de Scrum:**

    - **Sprint Planning:** Al inicio de cada Sprint (iteración de tiempo fijo, típicamente de 2 a 4 semanas), el Equipo de Desarrollo, junto con el Product Owner, planifica el trabajo a realizar. Se seleccionan elementos del Product Backlog para formar el Sprint Backlog y se define el objetivo del Sprint.

    - **Daily Scrum (Stand-up Meeting):** Una reunión diaria de 15 minutos donde el Equipo de Desarrollo sincroniza sus actividades y planifica el trabajo para las próximas 24 horas, identificando impedimentos y asegurando la alineación con el objetivo del Sprint.

    - **Sprint Review:** Al final de cada Sprint, se realiza una reunión para inspeccionar el incremento y adaptar el Product Backlog si es necesario. El Equipo de Desarrollo presenta el trabajo completado a los stakeholders, quienes proporcionan retroalimentación valiosa.

    - **Sprint Retrospective:** Después del Sprint Review y antes del siguiente Sprint Planning, el Equipo de Desarrollo y el Scrum Master se reúnen para inspeccionar cómo fue el Sprint en términos de personas, relaciones, procesos y herramientas, e identificar mejoras para el próximo Sprint.

3. **Artefactos de Scrum:**

- **Product Backlog:** Una lista priorizada de todas las funcionalidades, mejoras y correcciones que se desean para el producto. Es dinámico y evoluciona a lo largo del proyecto.
- **Sprint Backlog:** Un subconjunto de elementos del Product Backlog seleccionados para un Sprint específico, junto con el plan para entregar el incremento y lograr el objetivo del Sprint.
- **Incremento:** El conjunto de todos los elementos del Product Backlog completados durante un Sprint y los Sprints anteriores, que es potencialmente entregable y funcional.

La adopción de Scrum permitió una **flexibilidad** considerable para adaptarse a los requisitos cambiantes y a la retroalimentación temprana de los usuarios, lo cual es vital en un proyecto de desarrollo de software donde la comprensión inicial de las necesidades puede evolucionar. La entrega incremental de funcionalidades en cada Sprint aseguró que los stakeholders pudieran ver y probar partes del sistema en funcionamiento de manera regular, lo que facilitó la identificación de mejoras y la validación de las soluciones propuestas. Además, la transparencia inherente a Scrum, a través de los Daily Scrums y los Sprint Reviews, promovió una comunicación efectiva y una colaboración estrecha entre el equipo de desarrollo y los usuarios finales, garantizando que el sistema final no solo fuera técnicamente robusto, sino que también satisficiera plenamente las necesidades operativas de la gestión de proyectos de tesis en la carrera de Ingeniería en Informática.

ÍNDICE GENERAL

RESUMEN DE CAPÍTULOS ANTERIORES

Hasta el momento, la presente tesis ha establecido una base sólida para el diseño e implementación de un sistema web destinado a la gestión y aceptación de proyectos de tesis. Inicialmente, el **Capítulo 3: Resumen**, delineó la problemática central: la ineficiencia y falta de transparencia en la gestión manual de proyectos de tesis en la carrera de Ingeniería en Informática, proponiendo como solución un sistema web integral que automatice y optimice este proceso. Se destacó el uso de una metodología ágil (Scrum) y una arquitectura Modelo-Vista-Controlador (MVC), con tecnologías clave como PHP con Laravel, MySQL y JavaScript con Vue.js.

Posteriormente, el **Capítulo 4: Diagnóstico Situacional**, profundizó en la descripción del contexto problemático, analizando las deficiencias del sistema actual, tales como la lentitud en los trámites, la dispersión de la información y la falta de un seguimiento claro para estudiantes y docentes. Este diagnóstico permitió identificar las necesidades críticas y las oportunidades de mejora que un sistema automatizado podría ofrecer, sentando las bases para la definición de los requerimientos del nuevo sistema.

Finalmente, el **Capítulo 5: Determinación, Instalación y Configuración de las Herramientas de Desarrollo**, detalló la selección estratégica de la metodología de desarrollo ágil Scrum, la arquitectura MVC y las tecnologías específicas que conformarían el stack tecnológico del proyecto. Se justificó la elección de PHP con el framework Laravel para el backend, MySQL como sistema de gestión de bases de datos relacional, y JavaScript con el framework Vue.js para el frontend, explicando cómo estas herramientas se alinearían con los objetivos del proyecto y las necesidades identificadas. Este capítulo concluyó con la descripción del proceso de instalación y configuración del entorno de desarrollo, preparando el terreno para la fase de construcción del sistema.

6. DESARROLLO DEL SISTEMA DE INFORMACIÓN

La fase de desarrollo del sistema de información representa el núcleo de la presente investigación, donde los cimientos teóricos y las decisiones tecnológicas previamente establecidas se materializan en una solución funcional y robusta. Este capítulo detalla el proceso iterativo y progresivo de construcción del sistema web para la gestión y aceptación de proyectos de tesis, desde la conceptualización de sus funcionalidades hasta la implementación de sus módulos. Se conecta directamente con el diagnóstico situacional al traducir las necesidades identificadas en requerimientos concretos, y con la selección de herramientas al aplicar las tecnologías y la metodología definidas para construir cada componente del sistema. El objetivo primordial de esta fase es transformar los planos de diseño en un producto de software tangible que resuelva la problemática de la gestión manual y descentralizada, ofreciendo una plataforma eficiente, transparente y accesible para todos los actores involucrados en el proceso de titulación.

6.1. Análisis de Requerimientos

El análisis de requerimientos es una etapa crítica en el ciclo de vida del desarrollo de software, sirviendo como puente entre la comprensión de la problemática y la definición de la solución tecnológica. En el contexto de este proyecto, esta fase implicó una inmersión profunda en las operaciones actuales de gestión de proyectos de tesis, la interacción con usuarios clave (estudiantes, docentes asesores, miembros del comité de tesis) y la documentación exhaustiva de sus expectativas y necesidades. La información recopilada durante el diagnóstico situacional fue fundamental para este proceso, permitiendo identificar con precisión qué funcionalidades debía ofrecer el sistema y bajo qué condiciones debía operar. La claridad en la definición de los requerimientos es esencial para evitar desviaciones en el desarrollo, asegurar la satisfacción del usuario final y garantizar que el sistema cumpla con su propósito de optimizar la gestión de proyectos de tesis.

6.1.1 Descripción

La descripción del análisis de requerimientos para el sistema web de gestión y aceptación de proyectos de tesis se centra en la comprensión exhaustiva de las necesidades y expectativas de los usuarios finales, así como de los procesos de negocio que el sistema debe automatizar y optimizar. Este proceso se inició con la revisión detallada de la problemática expuesta en el diagnóstico situacional, donde se evidenciaron las ineficiencias, la falta de transparencia y la dispersión de la información inherentes a la

gestión manual. A partir de esta base, se realizaron entrevistas estructuradas y no estructuradas con estudiantes de Ingeniería en Informática que se encontraban en la fase de tesis, docentes asesores y miembros del comité de titulación. Estas interacciones permitieron obtener una visión integral de los flujos de trabajo actuales, los puntos de dolor, las expectativas de mejora y las funcionalidades deseadas.

El objetivo principal de esta descripción es establecer un entendimiento común y detallado de lo que el sistema debe hacer, cómo debe comportarse y qué restricciones debe considerar. Se documentaron los diferentes roles de usuario (estudiante, asesor, miembro del comité, administrador), sus interacciones con el sistema y las tareas específicas que cada uno realizaría. Por ejemplo, para los estudiantes, se identificó la necesidad de un portal para la presentación de propuestas, el seguimiento del estado de su proyecto y la comunicación con sus asesores. Para los docentes, la plataforma debía permitir la revisión de propuestas, la asignación de comentarios, la aprobación o rechazo de proyectos y la gestión de su carga de asesorías. Los miembros del comité requerían herramientas para la evaluación global de proyectos, la asignación de tribunales y la generación de informes consolidados. La administración, por su parte, necesitaba funcionalidades para la gestión de usuarios, la configuración de parámetros del sistema y la supervisión general del proceso. Esta fase de descripción no solo se limitó a enumerar funcionalidades, sino que también buscó comprender el contexto operativo, las normativas universitarias aplicables y las expectativas de rendimiento y seguridad, sentando las bases para la definición formal de los requerimientos funcionales y no funcionales.

La documentación generada en esta etapa incluyó diagramas de casos de uso, que ilustran las interacciones entre los actores y el sistema, y descripciones detalladas de cada caso de uso, especificando los flujos de eventos principales y alternativos. Estos artefactos fueron cruciales para visualizar el comportamiento esperado del sistema desde la perspectiva del usuario y para facilitar la comunicación entre el equipo de desarrollo y los stakeholders. La claridad y precisión en esta descripción son fundamentales para asegurar que el sistema final no solo cumpla con las expectativas, sino que también aborde de manera efectiva la problemática central de la gestión de proyectos de tesis, transformando un proceso manual y propenso a errores en uno automatizado, eficiente y transparente. La adopción de un enfoque iterativo, característico de la metodología Scrum, permitió refinar continuamente esta descripción a medida que se obtenía nueva información y se validaban los entendimientos iniciales con los usuarios.

6.1.2 Requerimientos Funcionales del Proyecto

Los requerimientos funcionales definen las funciones específicas que el sistema debe realizar para satisfacer las necesidades de los usuarios y cumplir con los objetivos del proyecto. Estos requerimientos son la columna vertebral del sistema, detallando qué acciones puede ejecutar el usuario y qué procesos internos debe llevar a cabo la aplicación. Para el sistema web de gestión y aceptación de proyectos de tesis, se identificaron y documentaron una serie de requerimientos funcionales esenciales, derivados directamente del análisis de la situación problemática y las expectativas de los stakeholders.

Entre los requerimientos funcionales clave se encuentran:

- **Gestión de Usuarios y Roles:** El sistema debe permitir la creación, edición y eliminación de usuarios, asignando roles específicos como estudiante, docente asesor, miembro del comité de tesis y administrador. Cada rol tendrá permisos diferenciados para acceder a las funcionalidades del sistema. Por ejemplo, los estudiantes podrán presentar proyectos, los asesores revisarlos y los administradores gestionar la configuración general.

- **Autenticación y Autorización Segura:** Los usuarios deben poder iniciar sesión de forma segura en el sistema. El sistema debe implementar mecanismos de autenticación robustos y de autorización basados en roles para garantizar que cada usuario solo acceda a las funcionalidades y datos para los que tiene permiso. Esto incluye la recuperación de contraseña y la gestión de sesiones.

- **Presentación de Propuestas de Tesis:** Los estudiantes deben tener la capacidad de crear y enviar nuevas propuestas de tesis a través de un formulario intuitivo. Este formulario debe permitir adjuntar documentos relevantes (ej. anteproyecto, cronograma, bibliografía preliminar) y proporcionar información detallada sobre el tema, objetivos, metodología y justificación del proyecto.

- **Seguimiento del Estado del Proyecto:** El sistema debe proporcionar a los estudiantes una interfaz para visualizar el estado actual de su propuesta de tesis (ej. "Pendiente de Revisión", "En Revisión", "Aprobado", "Rechazado", "Con Observaciones"). Además, debe mostrar el historial de revisiones y los comentarios recibidos.

- **Asignación de Asesores:** El sistema debe permitir a los administradores o al comité de tesis asignar docentes asesores a las propuestas de tesis. Esta asignación puede basarse en la disponibilidad del docente, su área de

especialización o la carga de trabajo actual.

- **Revisión y Evaluación de Propuestas:** Los docentes asesores y los miembros del comité deben poder acceder a las propuestas asignadas, revisarlas, añadir comentarios detallados, realizar observaciones y emitir un veredicto (aprobar, rechazar o solicitar modificaciones). El sistema debe registrar la fecha y hora de cada revisión.

- **Notificaciones Automáticas:** El sistema debe enviar notificaciones automáticas por correo electrónico a los usuarios sobre eventos importantes, como el cambio de estado de una propuesta, la asignación de un asesor, la recepción de comentarios o la fecha límite para la entrega de documentos.

- **Gestión de Observaciones y Modificaciones:** Si una propuesta es devuelta con observaciones, el sistema debe permitir al estudiante realizar las modificaciones solicitadas y volver a enviar la propuesta para una nueva revisión. Debe mantener un historial de versiones de los documentos.

- **Generación de Informes:** El sistema debe ser capaz de generar diversos informes para los administradores y el comité de tesis, tales como: listado de proyectos aprobados/rechazados, proyectos en curso, carga de trabajo de asesores, estadísticas de tiempo de revisión, entre otros. Estos informes deben ser exportables a formatos comunes (ej. PDF, Excel).

- **Gestión de Plazos y Fechas Límite:** El sistema debe permitir la configuración de plazos y fechas límite para la presentación de propuestas, revisiones y otras etapas del proceso. Debe alertar a los usuarios sobre los plazos próximos.

- **Comunicación Interna:** Integrar un módulo de mensajería interna o comentarios para facilitar la comunicación directa entre estudiantes, asesores y miembros del comité dentro de la plataforma, centralizando las interacciones relacionadas con el proyecto.

- **Gestión de Documentos Adjuntos:** Permitir la carga, descarga y visualización de documentos en diferentes formatos (PDF, DOCX, etc.) asociados a cada propuesta de tesis, garantizando la seguridad y accesibilidad de los archivos.

Cada uno de estos requerimientos funcionales fue detallado con sus respectivos casos de uso, precondiciones, postcondiciones y flujos de eventos, asegurando una comprensión clara de su implementación. La priorización de estos requerimientos se realizó en

colaboración con los stakeholders, utilizando técnicas de la metodología Scrum para enfocar los esfuerzos de desarrollo en las funcionalidades de mayor valor para el usuario final y para la resolución de la problemática central.

6.1.3 Requerimientos No Funcionales del Proyecto

Mientras que los requerimientos funcionales describen "qué" debe hacer el sistema, los requerimientos no funcionales especifican "cómo" debe comportarse el sistema, es decir, las cualidades o atributos que el sistema debe poseer. Estos requerimientos son cruciales para la calidad, la usabilidad, la eficiencia y la sostenibilidad del sistema a largo plazo. Para el sistema web de gestión y aceptación de proyectos de tesis, se identificaron los siguientes requerimientos no funcionales:

- **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 de 50 usuarios concurrentes.
    - Las operaciones de carga de documentos no deben exceder los 10 segundos para archivos de hasta 10 MB.
    - La generación de informes complejos debe completarse en un máximo de 15 segundos.
- **Seguridad:**
    - El sistema debe proteger la información sensible de los usuarios (credenciales, datos personales, contenido de las propuestas) mediante cifrado de datos en tránsito (HTTPS) y en reposo.
    - Debe implementar un control de acceso basado en roles estricto, asegurando que los usuarios solo puedan acceder a las funcionalidades y datos para los que tienen autorización.
    - Debe ser resistente a ataques comunes de seguridad web, como inyección SQL, Cross-Site Scripting (XSS) y Cross-Site Request Forgery (CSRF), utilizando las características de seguridad provistas por el framework Laravel.
    - Debe contar con un sistema de registro de auditoría para rastrear las acciones críticas realizadas por los usuarios.

- **Usabilidad:**
    - La interfaz de usuario debe ser intuitiva, fácil de aprender y de usar para todos los roles de usuario, minimizando la necesidad de capacitación extensa.
    - Debe proporcionar retroalimentación clara y oportuna al usuario sobre el estado de sus acciones (ej. mensajes de éxito, error, carga).
    - El diseño debe ser consistente en todas las páginas y módulos del sistema, siguiendo principios de diseño de interfaz de usuario modernos y accesibles.
    - Debe ser accesible para usuarios con diferentes niveles de habilidad tecnológica.
- **Fiabilidad:**
    - El sistema debe estar disponible el 99.5% del tiempo durante las horas de operación (lunes a viernes, 8:00 AM - 6:00 PM).
    - Debe manejar errores de manera elegante, mostrando mensajes informativos al usuario y registrando los errores para su posterior análisis.
    - Debe garantizar la integridad de los datos, evitando la pérdida o corrupción de información debido a fallos del sistema.
- **Escalabilidad:**
    - El sistema debe ser capaz de soportar un aumento en el número de usuarios y proyectos sin una degradación significativa del rendimiento.
    - La arquitectura debe permitir la adición de nuevas funcionalidades o módulos en el futuro con un esfuerzo de desarrollo razonable.
- **Mantenibilidad:**
    - El código fuente debe ser modular, bien documentado y seguir estándares de codificación para facilitar su comprensión, modificación y depuración por parte de futuros desarrolladores.

- La arquitectura del sistema debe permitir la fácil actualización de componentes tecnológicos (frameworks, librerías) sin requerir una reescritura completa.

- **Compatibilidad:**

  - El sistema debe ser compatible con los navegadores web modernos más utilizados (Chrome, Firefox, Edge, Safari) en sus últimas versiones estables.

  - Debe ser responsivo, adaptándose correctamente a diferentes tamaños de pantalla (escritorio, tabletas, dispositivos móviles) para una experiencia de usuario consistente.

Estos requerimientos no funcionales fueron considerados desde las primeras etapas de diseño y desarrollo, influyendo en la elección de la arquitectura, las tecnologías y las prácticas de codificación. Su cumplimiento es fundamental para asegurar que el sistema no solo funcione correctamente, sino que también sea una herramienta de alta calidad, segura y sostenible para la institución.

6.1.4 Restricciones

Las restricciones representan los límites o factores que pueden influir en el diseño, desarrollo e implementación del sistema. Identificar y comprender estas restricciones desde el inicio es crucial para gestionar las expectativas, planificar adecuadamente los recursos y evitar posibles obstáculos durante el ciclo de vida del proyecto. Para el sistema web de gestión y aceptación de proyectos de tesis, se identificaron las siguientes restricciones:

- **Restricciones Tecnológicas:**

  - El sistema debe ser desarrollado utilizando el stack tecnológico previamente definido en el Capítulo 5: PHP con el framework Laravel para el backend, MySQL como sistema de gestión de bases de datos y JavaScript con el framework Vue.js para el frontend. Esta restricción se estableció para aprovechar la experiencia del equipo de desarrollo, la disponibilidad de recursos y la compatibilidad con la infraestructura existente.

  - La infraestructura de servidor disponible para el despliegue del sistema puede tener limitaciones en cuanto a capacidad de procesamiento, memoria y almacenamiento, lo que debe ser

considerado en el diseño de la arquitectura y la optimización del rendimiento.

- **Restricciones de Tiempo:**
    - El proyecto está sujeto a un cronograma de tesis universitario, lo que impone plazos estrictos para la finalización de cada fase (análisis, diseño, desarrollo, pruebas e implementación). Cualquier retraso puede impactar la culminación exitosa de la tesis.
    - La disponibilidad de los stakeholders (estudiantes, docentes, comité) para la recopilación de requerimientos y la validación del sistema puede ser limitada debido a sus propias cargas académicas y laborales, lo que requiere una planificación cuidadosa de las sesiones de trabajo.
- **Restricciones de Recursos Humanos:**
    - El equipo de desarrollo es limitado, compuesto principalmente por el tesista, lo que implica una gestión eficiente del tiempo y las tareas para cubrir todas las fases del proyecto.
    - La experiencia en ciertas tecnologías o dominios específicos puede requerir un tiempo adicional para el aprendizaje y la investigación.
- **Restricciones Presupuestarias:**
    - Aunque el proyecto es de tesis, puede haber limitaciones en la adquisición de licencias de software, herramientas de desarrollo premium o servicios de hosting avanzados, lo que impulsa la elección de tecnologías de código abierto y soluciones costo-efectivas.
- **Restricciones Organizacionales y Normativas:**
    - El sistema debe adherirse a las políticas y normativas internas de la universidad con respecto a la gestión de datos académicos, la privacidad de la información y los procedimientos de titulación.

    - Cualquier cambio significativo en los procesos de gestión de tesis debe ser aprobado por las autoridades académicas correspondientes, lo que puede introducir demoras o requerir ajustes en el diseño del sistema.

- **Restricciones de Datos:**
    - La migración de datos históricos (si aplica) puede ser una restricción si los datos existentes no están en un formato estructurado o son inconsistentes, lo que podría requerir un esfuerzo adicional de limpieza y transformación.

La consideración de estas restricciones permitió al equipo de desarrollo tomar decisiones informadas, mitigar riesgos potenciales y establecer un alcance realista para el proyecto. La gestión proactiva de estas limitaciones fue clave para mantener el proyecto en curso y asegurar la entrega de un sistema que cumpla con los objetivos dentro de los parámetros establecidos.

6.2 Fase de diseño

La fase de diseño es el puente entre el "qué" (definido en el análisis de requerimientos) y el "cómo" (implementación en la codificación). En esta etapa, se traduce la comprensión de las necesidades del usuario en una arquitectura y un conjunto de especificaciones detalladas que guiarán la construcción del sistema. El diseño no solo se enfoca en la funcionalidad, sino también en la calidad, la eficiencia, la seguridad y la mantenibilidad del software. Para el sistema web de gestión y aceptación de proyectos de tesis, la fase de diseño se abordó de manera integral, abarcando la arquitectura del sistema, el diseño de la base de datos y el diseño de la interfaz de usuario, siempre bajo la premisa de la metodología ágil Scrum, que permite la iteración y la adaptación continua.

El proceso de diseño comenzó con la conceptualización de la arquitectura general del sistema, basándose en el patrón Modelo-Vista-Controlador (MVC) previamente seleccionado. Esta elección arquitectónica es fundamental para la separación de responsabilidades, facilitando la modularidad, la escalabilidad y la mantenibilidad del código. El diseño arquitectónico detalló cómo interactuarían los diferentes componentes del sistema: el frontend (desarrollado con Vue.js) se comunicaría con el backend (desarrollado con Laravel) a través de APIs RESTful, y el backend, a su vez, interactuaría con la base de datos (MySQL). Se definieron las capas de presentación, lógica de negocio y acceso a datos, asegurando una estructura clara y organizada.

Posteriormente, se procedió con el diseño de la base de datos, una etapa crítica para garantizar la integridad, consistencia y eficiencia en el almacenamiento y recuperación de la información. Se elaboró un Modelo Entidad-Relación (MER) que representó las entidades clave del sistema (usuarios, proyectos, revisiones, documentos, etc.) y las relaciones entre ellas. A partir del MER, se generó el esquema de la base de datos relacional, especificando las tablas, los campos, los tipos de datos, las claves primarias y foráneas, y los índices necesarios para optimizar las consultas. Se aplicaron principios de normalización para minimizar la redundancia y mejorar la integridad de los datos, asegurando que la estructura de la base de datos fuera robusta y escalable para soportar el volumen de información esperado.

Finalmente, el diseño de la interfaz de usuario (UI) y la experiencia de usuario (UX) fue una prioridad para garantizar que el sistema fuera intuitivo, accesible y agradable de usar. Se crearon wireframes y mockups para visualizar la disposición de los elementos en pantalla, los flujos de navegación y la interacción del usuario con el sistema. Se prestó especial atención a la usabilidad para cada rol de usuario, diseñando interfaces que simplificaran la presentación de proyectos para estudiantes, la revisión para asesores y la gestión para administradores. Se definieron paletas de colores, tipografías y componentes de interfaz que siguieran principios de diseño modernos y que fueran consistentes en toda la aplicación. El objetivo era crear una experiencia de usuario fluida y eficiente, que redujera la curva de aprendizaje y fomentara la adopción del sistema por parte de todos los usuarios. Esta fase de diseño fue iterativa, con revisiones y retroalimentación constante de los stakeholders para asegurar que las soluciones de diseño se alinearan con sus expectativas y necesidades.

6.3 Fase de Codificación

La fase de codificación, también conocida como implementación, es donde los diseños y especificaciones detalladas se transforman en código fuente ejecutable. Esta etapa es la materialización de todo el trabajo previo de análisis y diseño, y requiere una aplicación rigurosa de las mejores prácticas de programación, el uso eficiente de las herramientas seleccionadas y una atención meticulosa a la calidad del código. Para el sistema web de gestión y aceptación de proyectos de tesis, la fase de codificación se llevó a cabo siguiendo los principios de la metodología Scrum, con ciclos de desarrollo iterativos (sprints) que permitieron la construcción incremental de funcionalidades y la integración continua.

El desarrollo se dividió en componentes de frontend y backend, aprovechando la separación de responsabilidades inherente a la arquitectura MVC y al uso de APIs RESTful. En el backend, se utilizó PHP con el framework Laravel, que proporcionó una base sólida para la gestión de la lógica de negocio, la interacción con la base de datos y la exposición de los

servicios web. Laravel facilitó la implementación de funcionalidades como la autenticación de usuarios, la gestión de rutas, la validación de datos, la manipulación de la base de datos a través de Eloquent ORM y la gestión de archivos. Se siguieron los estándares de codificación de Laravel y las convenciones de PHP para asegurar la legibilidad y mantenibilidad del código.

En el frontend, se empleó JavaScript con el framework Vue.js para construir una interfaz de usuario dinámica y reactiva. Vue.js permitió la creación de componentes reutilizables, la gestión eficiente del estado de la aplicación y la interacción asíncrona con el backend. Se utilizaron herramientas como Vue Router para la gestión de rutas del lado del cliente y Vuex para la gestión centralizada del estado, lo que contribuyó a una experiencia de usuario fluida y a una arquitectura de frontend organizada. La comunicación entre el frontend y el backend se realizó mediante solicitudes HTTP a las APIs RESTful implementadas en Laravel, asegurando una interacción eficiente y desacoplada.

Durante toda la fase de codificación, se hizo hincapié en la escritura de código limpio, modular y bien documentado. Se implementaron pruebas unitarias y de integración para asegurar la correcta funcionalidad de cada componente y la interacción entre ellos. La gestión de versiones se realizó utilizando Git, lo que permitió un control efectivo de los cambios en el código fuente y facilitó la colaboración. Esta fase no solo se centró en la funcionalidad, sino también en la optimización del rendimiento, la seguridad y la escalabilidad, asegurando que el sistema final fuera robusto y eficiente.

6.3.1 Requerimientos de desarrollo

Los requerimientos de desarrollo, en este contexto, se refieren a las especificaciones técnicas y el entorno necesario para llevar a cabo la fase de codificación de manera efectiva, así como a las directrices y estándares que guiaron la implementación del software. Estos requerimientos aseguran que el equipo de desarrollo (en este caso, el tesista) cuente con las herramientas adecuadas y siga un conjunto de principios para construir un sistema de alta calidad. Se basan en las decisiones tomadas en la fase de determinación de herramientas y se detallan para garantizar la coherencia y la eficiencia en la implementación.

- **Entorno de Desarrollo:**
    - **Sistema Operativo:** Se utilizó un sistema operativo basado en Linux (ej. Ubuntu) o macOS, que proporciona un entorno robusto y compatible con las herramientas de desarrollo web.

- **Servidor Web Local:** Se configuró un servidor web local (ej. Apache o Nginx) para ejecutar la aplicación PHP/Laravel.
- **Base de Datos Local:** Se instaló y configuró MySQL Server para el desarrollo y las pruebas de la base de datos.
- **Intérprete de PHP:** Se utilizó una versión estable de PHP (ej. PHP 8.x) compatible con Laravel.
- **Node.js y npm/Yarn:** Necesario para la gestión de dependencias de frontend (Vue.js) y la compilación de activos.
- **Composer:** Gestor de dependencias de PHP para Laravel.
- **Editor de Código/IDE:** Se empleó un entorno de desarrollo integrado (IDE) como Visual Studio Code o PhpStorm, con extensiones para PHP, JavaScript, Vue.js y control de versiones (Git).

- **Estándares de Codificación:**
  - Se siguieron los estándares de codificación PSR (PHP Standard Recommendations) para PHP, especialmente PSR-1 (Basic Coding Standard) y PSR-12 (Extended Coding Style Guide), para asegurar la consistencia y legibilidad del código.
  - Para JavaScript y Vue.js, se adoptaron las guías de estilo recomendadas por la comunidad Vue y se utilizó un linter (ej. ESLint) para hacer cumplir las reglas de estilo y detectar posibles errores.
  - Se implementaron comentarios de código claros y concisos para explicar la lógica compleja, las decisiones de diseño y la funcionalidad de las clases, métodos y componentes.

- **Control de Versiones:**
  - Se utilizó Git como sistema de control de versiones distribuido, con un repositorio remoto (ej. GitHub o GitLab) para almacenar el código fuente, gestionar las ramas de desarrollo y facilitar el seguimiento de cambios.

    - Se siguió una estrategia de ramificación (ej. Git Flow o GitHub Flow) para organizar el desarrollo de nuevas funcionalidades, corrección de errores y lanzamientos.

- **Pruebas Unitarias y de Integración:**
    - Se escribieron pruebas unitarias para las clases y métodos críticos del backend utilizando PHPUnit, el framework de pruebas de PHP.
    - Se implementaron pruebas de integración para verificar la interacción entre diferentes componentes del sistema y las APIs RESTful.
    - Para el frontend, se consideraron pruebas de componentes utilizando herramientas como Vue Test Utils.
- **Documentación Técnica:**
    - Además de los comentarios en el código, se generó documentación técnica para describir la arquitectura del sistema, el diseño de la base de datos, las APIs expuestas y los procedimientos de despliegue.

El cumplimiento de estos requerimientos de desarrollo fue fundamental para asegurar que el proceso de codificación fuera eficiente, que el código resultante fuera de alta calidad y que el sistema fuera mantenible y escalable a largo plazo. La adherencia a estos estándares y el uso de un entorno de desarrollo bien configurado contribuyeron significativamente al éxito de la fase de implementación.

6.3.2 Desarrollo de los módulos del sistema de información

El desarrollo de los módulos del sistema de información constituyó la implementación práctica de los requerimientos funcionales y no funcionales, estructurando el sistema en componentes lógicos y cohesivos. Cada módulo fue diseñado para abordar un conjunto específico de funcionalidades, facilitando la gestión, el mantenimiento y la escalabilidad del sistema. La construcción de estos módulos se realizó de manera iterativa, integrando las tecnologías seleccionadas (Laravel para el backend, Vue.js para el frontend y MySQL para la base de datos) para crear una solución completa y funcional.

6.3.2.1. Módulo de Autenticación y Gestión de Usuarios

Este módulo es la puerta de entrada al sistema y es fundamental para garantizar la seguridad y el control de acceso. Su desarrollo incluyó:

- **Registro de Usuarios:** Implementación de un formulario de registro para nuevos usuarios (estudiantes, docentes), con validación de datos y cifrado de contraseñas.
- **Inicio de Sesión (Login):** Desarrollo de la funcionalidad de inicio de sesión seguro, utilizando el sistema de autenticación de Laravel (Laravel Breeze o Jetstream, o implementado manualmente con Laravel Sanctum para APIs). Se gestionaron las sesiones de usuario y se implementaron mecanismos para prevenir ataques de fuerza bruta.
- **Recuperación de Contraseña:** Funcionalidad para que los usuarios puedan restablecer sus contraseñas a través de un proceso seguro, generalmente enviando un enlace de restablecimiento por correo electrónico.
- **Gestión de Roles y Permisos:** Implementación de un sistema de control de acceso basado en roles (RBAC). Se definieron roles como "Estudiante", "Docente Asesor", "Miembro del Comité" y "Administrador", y se asignaron permisos específicos a cada rol para acceder a diferentes funcionalidades y recursos del sistema. Esto se logró utilizando paquetes de Laravel como Spatie/Laravel-Permission o implementando una lógica personalizada.
- **CRUD de Usuarios (Administrador):** El rol de administrador tiene la capacidad de crear, leer, actualizar y eliminar cuentas de usuario, así como de asignar y modificar roles.
- **Perfil de Usuario:** Cada usuario puede ver y actualizar su información personal (nombre, correo electrónico, etc.) y cambiar su contraseña.

El backend de este módulo se desarrolló en Laravel, utilizando sus características de autenticación y autorización. El frontend en Vue.js proporcionó las interfaces de usuario para el registro, inicio de sesión y gestión de perfiles, comunicándose con el backend a través de APIs RESTful.

6.3.2.2. Módulo de Presentación y Gestión de Proyectos

Este módulo es el corazón del sistema para los estudiantes y el punto de inicio del ciclo de vida de un proyecto de tesis. Su desarrollo abarcó:

- **Formulario de Presentación de Propuestas:** Creación de un formulario web detallado para que los estudiantes envíen sus propuestas de tesis. Este formulario incluyó campos para el título, resumen, objetivos (general y específicos), justificación, metodología, cronograma preliminar, bibliografía y palabras clave.

- **Carga de Documentos Adjuntos:** Implementación de la funcionalidad para adjuntar archivos (ej. anteproyecto en PDF, DOCX) a la propuesta. Se gestionó el almacenamiento seguro de estos archivos en el servidor y se implementaron validaciones para el tipo y tamaño de archivo.

- **Visualización y Edición de Propuestas:** Los estudiantes pueden ver el detalle de sus propuestas enviadas y, si el estado lo permite (ej. "Con Observaciones"), editarlas y volver a enviarlas.

- **Seguimiento del Estado:** Interfaz para que los estudiantes puedan consultar el estado actual de su propuesta (ej. "Pendiente de Revisión", "Aprobado", "Rechazado") y ver el historial de cambios de estado.

- **Listado de Proyectos (Administrador/Comité):** Los administradores y miembros del comité pueden visualizar un listado completo de todas las propuestas, con opciones de filtrado y búsqueda por estado, estudiante, asesor, etc.

- **Asignación de Asesores:** Funcionalidad para que los administradores o el comité asignen uno o más docentes asesores a una propuesta de tesis específica.

El backend en Laravel manejó la lógica de negocio para la creación, almacenamiento y actualización de proyectos, así como la gestión de archivos. El frontend en Vue.js proporcionó las interfaces dinámicas para la presentación, visualización y edición de propuestas, con componentes reactivos que mejoraron la experiencia del usuario.

6.3.2.3. Módulo de Revisión y Evaluación

Este módulo está diseñado para los docentes asesores y miembros del comité, permitiéndoles evaluar las propuestas de tesis. Las funcionalidades desarrolladas incluyen:

- **Listado de Proyectos Asignados:** Los asesores y miembros del comité pueden ver un listado de las propuestas de tesis que les han sido asignadas para revisión.

- **Detalle de la Propuesta para Revisión:** Interfaz que muestra toda la información de la propuesta, incluyendo los documentos adjuntos, para que el revisor pueda evaluarla.
- **Formulario de Evaluación:** Un formulario donde los revisores pueden ingresar sus comentarios detallados, observaciones, sugerencias y emitir un veredicto (aprobar, rechazar, solicitar modificaciones). Se pueden incluir criterios de evaluación predefinidos.
- **Historial de Revisiones:** El sistema registra todas las revisiones realizadas a una propuesta, incluyendo el revisor, la fecha, los comentarios y el veredicto.
- **Cambio de Estado del Proyecto:** Basado en el veredicto del revisor, el sistema actualiza automáticamente el estado de la propuesta de tesis.
- **Asignación de Tribunales (Comité):** Funcionalidad para que el comité de tesis asigne los miembros del tribunal examinador una vez que la propuesta ha sido aprobada.

La lógica de evaluación y el almacenamiento de comentarios y veredictos se implementaron en el backend con Laravel, asegurando la integridad de los datos. El frontend en Vue.js proporcionó las interfaces intuitivas para la revisión y evaluación, facilitando el proceso para los docentes.

6.3.2.4. Módulo de Notificaciones

Este módulo es crucial para mantener informados a todos los usuarios sobre el progreso y los eventos importantes relacionados con sus proyectos de tesis. Su desarrollo incluyó:

- **Envío de Notificaciones por Correo Electrónico:** Implementación de un sistema para enviar correos electrónicos automáticos a los usuarios en respuesta a eventos específicos, como:
    - Propuesta enviada exitosamente.
    - Cambio de estado de la propuesta (ej. "Aprobado", "Rechazado", "Con Observaciones").
    - Asignación de asesor.
    - Recepción de nuevos comentarios o feedback.
    - Recordatorios de plazos próximos.

- **Plantillas de Correo Electrónico:** Diseño de plantillas de correo electrónico personalizables para diferentes tipos de notificaciones, asegurando un formato consistente y profesional.

- **Gestión de Notificaciones Internas (Opcional):** Consideración de un sistema de notificaciones dentro de la propia aplicación, visible en un panel de control o centro de notificaciones para cada usuario.

Laravel, con su sistema de Mail y Queues, fue fundamental para la implementación robusta y escalable de las notificaciones por correo electrónico, permitiendo el envío asíncrono para no afectar el rendimiento de la aplicación principal.

6.3.2.5. Módulo de Generación de Informes

Este módulo proporciona a los administradores y al comité de tesis herramientas para monitorear el progreso general y obtener estadísticas relevantes. Las funcionalidades desarrolladas fueron:

- **Informes de Proyectos:** Generación de informes sobre el estado de todas las propuestas de tesis (ej. número de proyectos en cada estado, proyectos por asesor, proyectos por área temática).

- **Informes de Usuarios:** Listados de estudiantes, docentes y miembros del comité, con sus respectivas cargas de trabajo o proyectos asignados.

- **Estadísticas de Proceso:** Informes sobre el tiempo promedio de revisión, la tasa de aprobación/rechazo, etc.

- **Exportación de Datos:** Capacidad de exportar los informes generados a formatos comunes como PDF y Excel, para su análisis externo o presentación.

- **Filtros y Búsquedas Avanzadas:** Interfaces para aplicar filtros y criterios de búsqueda a los datos antes de generar los informes, permitiendo una personalización de la información.

El backend en Laravel se encargó de la lógica de consulta y procesamiento de datos para la generación de informes, utilizando librerías para la exportación a PDF (ej. Dompdf o Snappy) y Excel (ej. Maatwebsite/Laravel-Excel). El frontend en Vue.js proporcionó las interfaces para la selección de criterios de informe y la visualización de los resultados.

El desarrollo de cada uno de estos módulos se realizó con un enfoque en la modularidad, la reutilización de código y la adherencia a los estándares de calidad, asegurando que el sistema final fuera una solución integral y eficiente para la gestión y aceptación de proyectos de tesis.

ÍNDICE GENERAL

7. FASE DE PRUEBAS

La fase de pruebas representa un pilar fundamental en el ciclo de vida del desarrollo de software, especialmente en el contexto de un sistema web diseñado para la gestión y aceptación de proyectos de tesis. Su propósito primordial es asegurar que el sistema desarrollado no solo cumpla con los requisitos funcionales y no funcionales previamente definidos, sino que también opere de manera estable, eficiente y segura en un entorno real. Tras las etapas de diseño, desarrollo e implementación de los módulos del sistema, tal como se detalló en el capítulo anterior, se hace imperativo someter la aplicación a un escrutinio riguroso para identificar y corregir cualquier anomalía, error o comportamiento inesperado. Este proceso sistemático de verificación y validación es crucial para garantizar la calidad del producto final y la satisfacción de los usuarios, quienes dependerán de la fiabilidad del sistema para gestionar un proceso tan crítico como la aprobación de sus proyectos de tesis.

En las secciones previas, se ha establecido el contexto problemático de la gestión manual de tesis, se han determinado las herramientas tecnológicas (PHP con Laravel, MySQL, Vue.js) y se ha descrito el proceso de desarrollo bajo una metodología ágil. La fase de pruebas, por tanto, no es un evento aislado, sino una continuación lógica y esencial que valida la efectividad de las decisiones tomadas en cada etapa anterior. Se busca confirmar que la arquitectura MVC implementada, la base de datos diseñada y la lógica de negocio codificada en el backend y el frontend, interactúen de manera armónica y produzcan los resultados esperados. La coherencia con los objetivos generales y específicos planteados en la introducción es el hilo conductor que guía esta fase, asegurando que el sistema web contribuya efectivamente a la automatización y optimización del proceso de gestión de proyectos de tesis en la carrera de Ingeniería en Informática.

El objetivo de este capítulo es detallar la metodología empleada para la verificación del sistema, desde la concepción del plan de pruebas hasta la ejecución de los casos de prueba y el análisis exhaustivo de los resultados obtenidos. Se abordarán los diferentes tipos de pruebas aplicadas, las herramientas utilizadas para su gestión y ejecución, y el proceso iterativo de detección, reporte y corrección de defectos. La finalidad última es presentar una evaluación objetiva del rendimiento y la funcionalidad del sistema, proporcionando una base sólida para las conclusiones y recomendaciones finales de esta investigación.

7.1 Elaboración y Ejecución del Plan de Pruebas

La elaboración de un plan de pruebas exhaustivo y su posterior ejecución metódica son pasos ineludibles para garantizar la calidad y fiabilidad de cualquier sistema de software, y el sistema web para la gestión y aceptación de proyectos de tesis no fue la excepción. Este plan se concibió como un documento estratégico que delineó el alcance, los objetivos, los recursos, el cronograma y las responsabilidades asociadas a la fase de verificación del sistema. Su diseño se basó en los requisitos funcionales y no funcionales identificados durante la etapa de diagnóstico situacional, así como en las especificaciones de diseño detalladas en el capítulo de desarrollo, asegurando una cobertura integral de todas las funcionalidades implementadas.

El **objetivo principal** del plan de pruebas fue validar que el sistema web cumpliera con todas las expectativas de los usuarios y los requisitos técnicos, operando sin errores significativos y proporcionando una experiencia de usuario intuitiva y eficiente. Para lograrlo, se definieron varios tipos de pruebas, cada uno enfocado en un aspecto particular del sistema, permitiendo una evaluación multifacética de su comportamiento. La metodología ágil (Scrum) adoptada para el desarrollo facilitó un enfoque iterativo en las pruebas, donde cada incremento del software era sometido a pruebas continuas, permitiendo la detección temprana de defectos y su corrección oportuna, lo que redujo significativamente el riesgo de encontrar problemas mayores en las etapas finales.

La estructura del plan de pruebas incluyó los siguientes componentes clave:

- **Objetivos de las Pruebas:** Definir claramente qué se esperaba lograr con la fase de pruebas, como la verificación de la funcionalidad, la usabilidad, el rendimiento y la seguridad del sistema.
- **Alcance de las Pruebas:** Especificar qué módulos y funcionalidades del sistema serían probados y cuáles, por limitaciones de tiempo o recursos, quedarían fuera del alcance inmediato (aunque en este proyecto se buscó una cobertura lo más amplia posible).
- **Estrategia de Pruebas:** Detallar los tipos de pruebas a realizar (unitarias, de integración, de sistema, de aceptación), las técnicas a emplear y el orden en que se ejecutarían.
- **Criterios de Inicio y Fin:** Establecer las condiciones bajo las cuales se iniciaría la fase de pruebas (ej. módulos desarrollados y desplegados en entorno de pruebas) y cuándo se consideraría finalizada (ej. un porcentaje mínimo de casos de prueba aprobados, número de defectos críticos resueltos).

- **Entorno de Pruebas:** Describir la configuración de hardware y software necesaria para ejecutar las pruebas, incluyendo servidores, bases de datos, navegadores web y sistemas operativos.

- **Roles y Responsabilidades:** Asignar roles específicos al equipo de desarrollo y a los usuarios clave involucrados en las pruebas.

- **Herramientas de Pruebas:** Identificar las herramientas de software que se utilizarían para la gestión de casos de prueba, la ejecución de pruebas automatizadas y el seguimiento de defectos.

La ejecución del plan de pruebas se llevó a cabo de manera sistemática, comenzando con las pruebas de menor nivel y progresando hacia las de mayor complejidad. A continuación, se detallan los tipos de pruebas aplicados:

1. **Pruebas Unitarias:** Estas pruebas se enfocaron en verificar la funcionalidad de componentes individuales del código fuente, como funciones, métodos y clases. En el contexto del backend desarrollado con PHP y el framework Laravel, se utilizaron las capacidades de prueba unitaria provistas por **PHPUnit**. Cada controlador, modelo y servicio crítico fue sometido a pruebas para asegurar que su lógica interna operara correctamente, validando entradas, salidas y el manejo de excepciones. Por ejemplo, se probaron métodos para la creación de usuarios, la validación de formularios de proyectos, la asignación de roles y la interacción con la base de datos. En el frontend, para los componentes de Vue.js, se emplearon herramientas como **Vue Test Utils** y **Jest** para verificar que los componentes renderizaran correctamente, respondieran a eventos de usuario y gestionaran su estado de manera adecuada. Estas pruebas tempranas fueron fundamentales para detectar errores en la lógica de negocio antes de la integración de los módulos.

2. **Pruebas de Integración:** Una vez que los componentes individuales fueron probados, las pruebas de integración se centraron en verificar la interacción entre diferentes módulos del sistema. Esto incluyó la comunicación entre el frontend (Vue.js) y el backend (Laravel API), la persistencia de datos en la base de datos MySQL, y la interacción entre distintos servicios del backend. Por ejemplo, se probó el flujo completo de un estudiante subiendo un proyecto de tesis, desde la interfaz de usuario, pasando por la validación en el servidor, hasta el almacenamiento exitoso en la base de datos. También se verificó la correcta recuperación y visualización de estos datos por parte de un asesor o

miembro del comité. Estas pruebas fueron cruciales para identificar problemas de compatibilidad, errores en la transmisión de datos entre capas y fallos en la lógica de negocio que involucraban múltiples componentes.

3. **Pruebas de Sistema:** Las pruebas de sistema evaluaron el sistema web en su totalidad, verificando que todas las funcionalidades operaran de acuerdo con los requisitos especificados y que el sistema se comportara como una unidad coherente. Se simularon escenarios de uso real, cubriendo los flujos de trabajo completos de los diferentes perfiles de usuario (estudiante, asesor, comité, administrador). Esto incluyó pruebas de inicio de sesión, registro de proyectos, asignación de asesores, revisión y retroalimentación de proyectos, aprobación/rechazo, gestión de usuarios y roles, y generación de reportes. Se prestó especial atención a la navegación, la consistencia de la interfaz de usuario y la respuesta del sistema bajo diferentes condiciones.

4. **Pruebas de Aceptación de Usuario (UAT - User Acceptance Testing):** Esta fue una fase crítica donde usuarios finales reales (estudiantes, docentes asesores y miembros del comité de tesis) interactuaron directamente con el sistema. El objetivo fue validar que el sistema no solo funcionara técnicamente, sino que también satisficiera sus necesidades operativas y de negocio en un entorno lo más cercano posible al real. Se proporcionaron escenarios de uso y se les pidió a los usuarios que realizaran tareas específicas, observando su interacción y recopilando su retroalimentación. Esta retroalimentación fue invaluable para identificar problemas de usabilidad, flujos de trabajo confusos o funcionalidades que no se alineaban completamente con sus expectativas, permitiendo realizar ajustes y mejoras antes del despliegue final.

5. **Pruebas de Rendimiento:** Aunque el alcance de un proyecto de tesis puede tener limitaciones en pruebas de rendimiento a gran escala, se realizaron pruebas básicas para evaluar la capacidad de respuesta del sistema bajo cargas de trabajo típicas. Se monitorearon los tiempos de carga de páginas, la velocidad de procesamiento de formularios y la eficiencia de las consultas a la base de datos. Se utilizaron herramientas de desarrollo de navegadores para analizar el rendimiento del frontend y se observó el consumo de recursos del servidor durante operaciones clave. El objetivo fue asegurar que el sistema ofreciera una experiencia fluida y no presentara latencias excesivas en las operaciones más frecuentes.

6. **Pruebas de Seguridad:** Se realizaron pruebas básicas de seguridad para identificar vulnerabilidades comunes. Esto incluyó la verificación de la protección contra inyección SQL, ataques de Cross-Site Scripting (XSS), y la correcta implementación de la autenticación y autorización de usuarios. Se validó que solo los usuarios con los roles adecuados pudieran acceder a ciertas funcionalidades y que los datos sensibles estuvieran protegidos. Laravel, con sus características de seguridad integradas, proporcionó una base robusta, pero se realizaron verificaciones adicionales para asegurar que no se introdujeran vulnerabilidades durante el desarrollo personalizado.

Durante la ejecución de estas pruebas, se documentaron meticulosamente los casos de prueba, los pasos para su reproducción, los resultados esperados y los resultados obtenidos. Cualquier desviación entre el resultado esperado y el real fue registrada como un defecto. Para la gestión de estos defectos, se utilizó un sistema simple de seguimiento que permitía registrar la descripción del error, su severidad, la prioridad, el módulo afectado y el estado (abierto, en progreso, resuelto, cerrado). Este proceso iterativo de detección, reporte, corrección y re-prueba fue fundamental para la mejora continua del sistema, asegurando que cada error identificado fuera abordado y verificado antes de avanzar a la siguiente etapa.

7.2 Análisis de Resultados

El análisis de los resultados obtenidos durante la fase de pruebas es un componente crítico que permite evaluar la calidad del sistema web desarrollado, identificar áreas de mejora y determinar su preparación para la implementación final. Tras la meticulosa elaboración y ejecución del plan de pruebas, se recopiló una vasta cantidad de datos relacionados con el comportamiento del sistema, la detección de defectos y la retroalimentación de los usuarios. Este análisis no solo cuantifica la efectividad del proceso de pruebas, sino que también proporciona una visión cualitativa sobre la robustez, usabilidad y cumplimiento de los requisitos del sistema para la gestión y aceptación de proyectos de tesis.

Los resultados de las pruebas unitarias y de integración revelaron una alta tasa de éxito en la funcionalidad de los componentes individuales y en la interacción entre los módulos principales. Las pruebas unitarias, ejecutadas con **PHPUnit** para el backend de Laravel y **Jest/Vue Test Utils** para el frontend de Vue.js, permitieron identificar y corregir un número significativo de errores lógicos y de sintaxis en las primeras etapas del desarrollo. Esto validó la eficacia de la estrategia de pruebas tempranas, reduciendo la propagación de defectos a fases posteriores. Los errores encontrados en esta etapa fueron principalmente de tipo funcional, relacionados con la lógica de validación de datos, el manejo de excepciones

en las operaciones de base de datos y la correcta manipulación del estado en los componentes de la interfaz de usuario. La rápida resolución de estos problemas contribuyó a la estabilidad general del sistema.

En las pruebas de sistema, que abarcaron los flujos de trabajo completos, se identificaron algunos defectos de mayor complejidad, principalmente relacionados con la integración de diferentes módulos y la gestión de permisos de usuario. Por ejemplo, se detectaron casos donde un usuario con un rol específico podía acceder a funcionalidades que no le correspondían, o donde la transición entre diferentes vistas no era tan fluida como se esperaba. Estos defectos fueron clasificados como de severidad media y alta, y se les asignó una alta prioridad para su corrección. La mayoría de estos problemas se resolvieron mediante ajustes en la lógica de autorización del backend de Laravel (utilizando sus políticas y gates) y refinamientos en el enrutamiento y la navegación del frontend de Vue.js.

La **retroalimentación de las Pruebas de Aceptación de Usuario (UAT)** fue particularmente enriquecedora. Los usuarios finales (estudiantes, asesores y miembros del comité) proporcionaron perspectivas valiosas sobre la usabilidad y la alineación del sistema con sus procesos diarios. Se recibieron comentarios positivos sobre la interfaz de usuario intuitiva, la claridad de los formularios de envío de proyectos y la facilidad para revisar y dar retroalimentación. Sin embargo, también se identificaron áreas de mejora. Por ejemplo, algunos usuarios sugirieron optimizar la visualización de los estados de los proyectos para una comprensión más rápida, o añadir filtros adicionales en las listas de proyectos. Estas sugerencias, aunque no siempre representaban defectos críticos, fueron consideradas para futuras iteraciones o mejoras menores que se implementaron antes del despliegue. La capacidad de incorporar esta retroalimentación en el ciclo de desarrollo demostró la flexibilidad de la metodología ágil y la orientación del sistema hacia las necesidades reales del usuario.

En cuanto al **rendimiento del sistema**, las pruebas básicas indicaron que el sistema respondía de manera adecuada bajo cargas de trabajo típicas. Los tiempos de carga de las páginas y la ejecución de las operaciones clave (como el envío de un proyecto o la aprobación de una propuesta) se mantuvieron dentro de rangos aceptables. Las consultas a la base de datos MySQL fueron optimizadas para asegurar una recuperación eficiente de la información, y el uso de Vue.js en el frontend contribuyó a una experiencia de usuario reactiva y fluida, minimizando las recargas completas de página. Aunque no se realizaron pruebas de estrés a gran escala, el diseño de la arquitectura y la elección de tecnologías robustas sugieren una buena base para una escalabilidad futura.

Las **pruebas de seguridad** confirmaron que el sistema implementaba medidas adecuadas para protegerse contra vulnerabilidades comunes. La validación de entradas en el backend de Laravel, la sanitización de datos y la implementación de un sistema de autenticación y autorización basado en roles, contribuyeron a un entorno seguro. No se detectaron vulnerabilidades críticas como inyección SQL o XSS durante las pruebas, lo que indica una implementación cuidadosa de las mejores prácticas de seguridad en el desarrollo.

En resumen, el análisis de los resultados de la fase de pruebas permitió concluir que el sistema web para la gestión y aceptación de proyectos de tesis ha alcanzado un nivel de madurez y estabilidad considerable. Se logró una alta cobertura de los requisitos funcionales, y los defectos identificados fueron gestionados y resueltos de manera efectiva. La retroalimentación de los usuarios finales fue fundamental para refinar la usabilidad y asegurar que el sistema no solo funcionara correctamente, sino que también fuera práctico y eficiente para quienes lo utilizarían diariamente. Si bien siempre existen oportunidades para mejoras continuas, los resultados de esta fase confirman que el sistema está listo para su implementación y despliegue, cumpliendo con los objetivos planteados en esta investigación y ofreciendo una solución robusta y confiable a la problemática de la gestión de proyectos de tesis en la carrera de Ingeniería en Informática.

8. CONCLUSIONES

La culminación de la presente investigación y el desarrollo del sistema web para la gestión y aceptación de proyectos de tesis en la carrera de Ingeniería en Informática ha permitido alcanzar los objetivos planteados inicialmente, ofreciendo una solución robusta y eficiente a una problemática que históricamente ha afectado la fluidez y transparencia de los procesos académicos. El diagnóstico situacional previo reveló una gestión manual y descentralizada, caracterizada por la lentitud en la tramitación, la falta de un seguimiento claro de las propuestas y la propensión a errores administrativos, lo que generaba frustración tanto en estudiantes como en docentes y autoridades académicas. La implementación de este sistema no solo ha subsanado estas deficiencias, sino que ha establecido un precedente para la modernización de los procesos universitarios, demostrando el potencial transformador de la tecnología en el ámbito educativo.

El **objetivo general** de diseñar, desarrollar e implementar un sistema web integral que automatice y optimice el ciclo de vida de las propuestas de tesis ha sido plenamente satisfecho. Desde la concepción inicial del proyecto, se priorizó la creación de una plataforma intuitiva, segura y escalable, capaz de gestionar todas las etapas, desde la presentación de la propuesta por parte del estudiante, pasando por la revisión y retroalimentación de los tutores y el comité evaluador, hasta la aceptación final y el seguimiento del progreso. La arquitectura del sistema, basada en el patrón **Modelo-Vista-Controlador (MVC)**, y la selección de tecnologías de vanguardia como PHP con Laravel para el backend, MySQL para la gestión de la base de datos y Vue.js para el frontend, garantizaron una base sólida para un desarrollo ágil y una experiencia de usuario óptima. Esta combinación tecnológica no solo facilitó la construcción de un sistema funcional, sino que también aseguró su mantenibilidad y adaptabilidad a futuras necesidades.

En relación con los **objetivos específicos**, se lograron avances significativos. El primer objetivo, **analizar los procesos actuales de gestión de proyectos de tesis**, se cumplió a través de un exhaustivo diagnóstico situacional que identificó los puntos críticos y las áreas de mejora en el flujo de trabajo existente. Este análisis fue fundamental para comprender las necesidades reales de los usuarios y para diseñar un sistema que abordara directamente las ineficiencias. Se documentaron los roles de los actores involucrados (estudiantes, tutores, comité académico, coordinadores) y las interacciones entre ellos, lo que permitió modelar un flujo de trabajo digital que replicara y mejorara el proceso manual.

El segundo objetivo, **diseñar la arquitectura y los módulos del sistema web**, se materializó en la creación de diagramas de casos de uso, diagramas de clases, diagramas de secuencia y prototipos de interfaz de usuario. Estos artefactos de diseño sirvieron como planos detallados para la construcción del sistema, asegurando que cada componente estuviera alineado con los requisitos funcionales y no funcionales. El diseño modular permitió una clara separación de responsabilidades, facilitando el desarrollo paralelo y la integración de nuevas funcionalidades en el futuro. Se prestó especial atención a la usabilidad y la experiencia del usuario (UX), garantizando que la interfaz fuera clara, accesible y fácil de navegar para todos los perfiles de usuario.

El tercer objetivo, **desarrollar e implementar el sistema web utilizando tecnologías adecuadas**, se llevó a cabo mediante un enfoque de desarrollo iterativo y incremental. La fase de desarrollo implicó la codificación del backend para la lógica de negocio, la gestión de usuarios y roles, la persistencia de datos y la seguridad, así como el desarrollo del frontend para la interacción del usuario. La implementación incluyó la configuración del entorno de producción, el despliegue de la aplicación y la migración de datos iniciales, asegurando que el sistema estuviera operativo y accesible para los usuarios finales. Las pruebas exhaustivas, incluyendo pruebas unitarias, de integración y de aceptación, confirmaron la funcionalidad y estabilidad del sistema, validando que cumplía con las especificaciones de diseño y los requisitos del usuario.

Finalmente, el cuarto objetivo, **evaluar la funcionalidad y la usabilidad del sistema implementado**, se logró a través de pruebas de usuario y la recopilación de retroalimentación. Los resultados de estas evaluaciones demostraron que el sistema no solo es funcional y cumple con sus propósitos, sino que también es percibido como una herramienta útil y fácil de usar por los estudiantes y el personal académico. La automatización de tareas repetitivas, la centralización de la información y la mejora en la comunicación entre los actores han reducido significativamente los tiempos de respuesta y han aumentado la transparencia del proceso de gestión de tesis. La capacidad de los estudiantes para presentar sus propuestas en línea, recibir retroalimentación oportuna y seguir el estado de su proyecto en tiempo real ha mejorado drásticamente su experiencia académica.

Las **contribuciones de esta tesis** son múltiples y significativas. En primer lugar, se ha proporcionado una solución tecnológica concreta y operativa que resuelve una problemática real y apremiante en el ámbito universitario, mejorando la eficiencia administrativa y la calidad de la gestión académica. En segundo lugar, el proyecto sirve como un caso de estudio práctico para la aplicación de metodologías de desarrollo de software ágiles y el uso de un stack tecnológico moderno en la creación de sistemas web

complejos. En tercer lugar, la investigación contribuye al cuerpo de conocimiento en el área de la ingeniería de software y la gestión de proyectos, ofreciendo un modelo de sistema que puede ser adaptado y replicado en otras instituciones educativas con necesidades similares. La documentación detallada del proceso de diseño, desarrollo y pruebas proporciona una guía valiosa para futuros proyectos de desarrollo de software en entornos académicos.

A pesar de los logros alcanzados, es importante reconocer las **limitaciones** inherentes a cualquier proyecto de esta envergadura. El alcance del sistema se centró específicamente en la gestión y aceptación de proyectos de tesis, dejando fuera otras fases del proceso de titulación, como la defensa de la tesis o la gestión de tribunales, que podrían ser integradas en futuras iteraciones. Asimismo, la implementación inicial se realizó en un contexto específico, y aunque el diseño es modular, su adaptabilidad a entornos con regulaciones o flujos de trabajo significativamente diferentes podría requerir ajustes considerables. La dependencia de la infraestructura tecnológica existente en la institución también representa una limitación, ya que el rendimiento y la disponibilidad del sistema están sujetos a la calidad y estabilidad de dicha infraestructura.

En cuanto al **trabajo futuro**, se vislumbran varias líneas de investigación y desarrollo que podrían enriquecer aún más el sistema. Una extensión natural sería la integración de módulos adicionales para la gestión completa del proceso de titulación, incluyendo la asignación de tribunales, la programación de defensas y la gestión de calificaciones finales. Otra área de mejora podría ser la implementación de funcionalidades de análisis de datos y generación de informes avanzados, que permitan a las autoridades académicas obtener información valiosa sobre las tendencias en los proyectos de tesis, los tiempos de aprobación y el rendimiento de los tutores. La incorporación de inteligencia artificial para la sugerencia de tutores basada en el tema de la tesis o para la detección de plagio en las propuestas iniciales también representa una dirección prometedora. Finalmente, la mejora continua de la interfaz de usuario y la experiencia del usuario, a través de la retroalimentación constante y la aplicación de nuevas tendencias en diseño web, aseguraría que el sistema se mantenga relevante y efectivo a largo plazo.

9. RECOMENDACIONES

Tras la exitosa culminación del diseño, desarrollo e implementación del Sistema Web para la Gestión y Aceptación de Proyectos de Tesis, y habiendo validado su funcionalidad y eficiencia a través de un riguroso proceso de pruebas, es imperativo formular una serie de recomendaciones estratégicas. Estas recomendaciones buscan no solo optimizar el uso y la sostenibilidad del sistema a largo plazo, sino también potenciar su impacto en la mejora continua de los procesos académicos de la carrera de Ingeniería en Informática. El sistema ha demostrado ser una solución robusta a la problemática de la gestión manual y descentralizada, tal como se diagnosticó en las fases iniciales de esta investigación, y su implementación representa un avance significativo hacia la digitalización y transparencia.

Las siguientes recomendaciones se estructuran en diversas áreas, abarcando desde mejoras funcionales y técnicas para el propio sistema, hasta directrices para su integración institucional y sugerencias para futuras líneas de investigación. Cada una de ellas se fundamenta en las experiencias y hallazgos obtenidos durante todo el ciclo de vida del proyecto, desde la conceptualización hasta la fase de pruebas y conclusiones, buscando maximizar el valor añadido del sistema y asegurar su evolución adaptativa a las necesidades cambiantes del entorno universitario.

9.1 Recomendaciones para el Sistema Web

Para asegurar la **sostenibilidad, escalabilidad y mejora continua** del Sistema Web para la Gestión y Aceptación de Proyectos de Tesis, se proponen las siguientes recomendaciones técnicas y funcionales:

- **Implementación de Módulos de Notificaciones Avanzadas:** Aunque el sistema ya incorpora notificaciones básicas, se recomienda desarrollar un módulo de notificaciones más sofisticado que permita la personalización de alertas por parte de los usuarios (estudiantes, docentes, administradores). Esto incluiría la configuración de preferencias de notificación (correo electrónico, SMS, notificaciones push dentro del sistema), así como la integración de recordatorios automáticos para fechas límite de entrega, revisiones pendientes y estados de aprobación. Una gestión proactiva de las notificaciones reducirá la necesidad de seguimiento manual y mejorará la eficiencia comunicativa entre todos los actores involucrados en el proceso de tesis.

- **Desarrollo de un Módulo de Generación de Informes y Estadísticas Detalladas:** Actualmente, el sistema gestiona una gran cantidad de datos sobre proyectos de tesis. Se recomienda la creación de un módulo robusto para la generación de informes personalizados y la visualización de estadísticas clave. Esto permitiría a la administración académica obtener información valiosa sobre el rendimiento de los proyectos, tiempos promedio de aprobación, áreas temáticas más populares, desempeño de los tutores, y tasas de éxito. Estos datos son cruciales para la toma de decisiones estratégicas, la asignación de recursos y la identificación de áreas de mejora en el plan de estudios o en la gestión de la investigación.

- **Integración con Sistemas Académicos Existentes:** Para optimizar la experiencia del usuario y evitar la duplicidad de datos, se recomienda explorar la integración del sistema con otras plataformas académicas utilizadas por la universidad, como el Sistema de Gestión Académica (SGA) o plataformas de e-learning. Esta integración podría facilitar la sincronización automática de datos de estudiantes y docentes, la validación de requisitos académicos para la presentación de proyectos, y la publicación de resultados de tesis en repositorios institucionales. La interoperabilidad es clave para construir un ecosistema digital cohesivo y eficiente.

- **Mejoras en la Interfaz de Usuario (UI) y Experiencia de Usuario (UX):** Aunque la interfaz actual es funcional, se sugiere realizar auditorías periódicas de UI/UX con usuarios reales para identificar puntos de fricción y oportunidades de mejora. Esto podría incluir la simplificación de flujos de trabajo complejos, la optimización de la navegación, la implementación de guías interactivas o tutoriales, y la adaptación a principios de diseño más modernos y accesibles. Una interfaz intuitiva y una experiencia de usuario fluida son fundamentales para la adopción y satisfacción a largo plazo del sistema.

- **Implementación de un Sistema de Control de Versiones para Documentos:** Para los documentos adjuntos a los proyectos de tesis (propuestas, avances, borradores), se recomienda integrar un sistema de control de versiones. Esto permitiría a estudiantes y tutores rastrear los cambios realizados en los documentos, revertir a versiones anteriores si es necesario, y mantener un historial claro de las revisiones. Esta funcionalidad es vital para la colaboración y para asegurar la integridad y trazabilidad del proceso de redacción de la tesis.

- **Fortalecimiento de la Seguridad y Auditoría:** Continuar con la implementación de las mejores prácticas de seguridad informática, incluyendo auditorías de seguridad periódicas, pruebas de penetración y la actualización constante de las dependencias y librerías del sistema. Además, se recomienda desarrollar un módulo de auditoría más detallado que registre todas las acciones significativas realizadas por los usuarios dentro del sistema, facilitando la trazabilidad de eventos y la resolución de posibles disputas o errores administrativos.

- **Optimización del Rendimiento y Escalabilidad:** A medida que el número de usuarios y proyectos de tesis aumente, será crucial monitorear y optimizar continuamente el rendimiento del sistema. Esto podría implicar la optimización de consultas a la base de datos, la implementación de técnicas de caching, la refactorización de código crítico y la evaluación de arquitecturas de despliegue que permitan una escalabilidad horizontal. Asegurar que el sistema pueda manejar una carga creciente sin degradación del rendimiento es esencial para su éxito a largo plazo.

9.2 Recomendaciones para la Institución y la Carrera de Ingeniería en Informática

La implementación exitosa del sistema web no solo depende de su robustez técnica, sino también de una **adecuada gestión institucional y adaptación de procesos**. Las siguientes recomendaciones están dirigidas a la carrera de Ingeniería en Informática y a la institución en general:

- **Capacitación Continua y Soporte Técnico:** Es fundamental establecer un programa de capacitación continuo para todos los usuarios del sistema (estudiantes, docentes, tutores, miembros del comité de tesis y personal administrativo). Estas capacitaciones deben cubrir desde el uso básico hasta funcionalidades avanzadas, y deben ser actualizadas periódicamente. Adicionalmente, se debe garantizar un canal de soporte técnico eficiente y accesible para resolver dudas e incidencias que puedan surgir durante el uso del sistema, lo que fomentará la adopción y reducirá la resistencia al cambio.

- **Revisión y Actualización del Reglamento de Tesis:** Con la introducción de un sistema automatizado, es crucial revisar y, si es necesario, actualizar el reglamento de tesis existente. El reglamento debe reflejar los nuevos flujos de trabajo digitales, los roles y responsabilidades dentro del sistema, los plazos

automatizados y los procedimientos para la presentación y evaluación de proyectos a través de la plataforma. Esta armonización entre la normativa y la herramienta tecnológica es vital para la coherencia y validez legal del proceso.

- **Promoción del Uso y Beneficios del Sistema:** La institución debe llevar a cabo campañas de comunicación interna para promover activamente el uso del sistema y destacar sus beneficios en términos de eficiencia, transparencia y reducción de la carga administrativa. Es importante que todos los actores comprendan cómo el sistema simplifica sus tareas y contribuye a un proceso de tesis más fluido y justo. La percepción positiva del sistema es clave para su aceptación generalizada.

- **Asignación de Recursos para Mantenimiento y Evolución:** Se recomienda asignar un presupuesto y personal técnico dedicado para el mantenimiento preventivo, correctivo y evolutivo del sistema. Esto incluye la aplicación de parches de seguridad, la corrección de errores, la implementación de nuevas funcionalidades solicitadas por los usuarios y la adaptación a cambios tecnológicos. Un sistema sin un soporte adecuado corre el riesgo de volverse obsoleto o inoperable en poco tiempo.

- **Establecimiento de un Comité de Gestión del Sistema:** Crear un comité multidisciplinario compuesto por representantes de la administración académica, docentes, estudiantes y personal técnico. Este comité sería responsable de supervisar el funcionamiento del sistema, recopilar retroalimentación de los usuarios, priorizar nuevas funcionalidades y asegurar que el sistema continúe alineado con los objetivos estratégicos de la carrera y la institución.

- **Fomento de la Cultura de la Digitalización:** La implementación de este sistema es un paso importante hacia la digitalización de los procesos académicos. Se recomienda que la institución continúe fomentando una cultura de innovación y adopción tecnológica en todos sus ámbitos, explorando nuevas herramientas y soluciones que puedan optimizar otros procesos administrativos y educativos, mejorando así la eficiencia general y la experiencia de la comunidad universitaria.

9.3 Recomendaciones para Futuras Investigaciones

El presente proyecto de tesis ha sentado una base sólida para la gestión digital de proyectos de tesis, pero abre la puerta a múltiples **líneas de investigación futuras** que podrían expandir su alcance y funcionalidad. Se sugieren las siguientes áreas para futuras exploraciones académicas y de desarrollo:

- **Integración de Inteligencia Artificial para la Asignación de Tutores y Temas:** Investigar la aplicación de algoritmos de inteligencia artificial y aprendizaje automático para optimizar la asignación de tutores a proyectos de tesis, basándose en la especialidad del docente, la carga de trabajo y la temática del proyecto. Esto podría incluir el análisis semántico de las propuestas para sugerir tutores más adecuados o incluso para identificar solapamientos o brechas en las áreas de investigación.

- **Desarrollo de un Módulo de Colaboración en Tiempo Real:** Explorar la implementación de funcionalidades de colaboración en tiempo real dentro del sistema, permitiendo a estudiantes y tutores trabajar simultáneamente en documentos, realizar anotaciones conjuntas y comunicarse a través de chats integrados. Esto mejoraría significativamente la interacción y el proceso de revisión de los proyectos de tesis.

- **Análisis Predictivo del Éxito de Proyectos de Tesis:** Investigar la posibilidad de utilizar los datos históricos del sistema para desarrollar modelos predictivos que identifiquen factores de riesgo en los proyectos de tesis. Esto permitiría a la administración intervenir tempranamente para ofrecer apoyo adicional a los estudiantes que muestren indicadores de posible retraso o dificultad, mejorando las tasas de finalización exitosa.

- **Expansión del Sistema a Otros Procesos Académicos:** Estudiar la viabilidad de adaptar la arquitectura y funcionalidades del sistema para gestionar otros procesos académicos que requieran aprobación y seguimiento, como proyectos de investigación de pregrado, prácticas profesionales o solicitudes de becas. Esto consolidaría una plataforma unificada para la gestión de diversos trámites universitarios.

- **Implementación de Tecnologías Blockchain para la Certificación de Tesis:** Investigar la aplicación de la tecnología blockchain para la certificación inmutable de la autoría y la originalidad de las tesis, así como para la emisión de diplomas digitales. Esto podría aumentar la seguridad y la confianza en la validez de los títulos académicos emitidos por la institución.

- **Estudio de Usabilidad y Adopción en Diferentes Contextos Universitarios:** Realizar estudios comparativos de usabilidad y factores de adopción del sistema en otras universidades o facultades, para identificar patrones y adaptar el diseño a diversas necesidades institucionales y culturales. Esto contribuiría a la generalización y mejora del sistema en un contexto más amplio.

En síntesis, las recomendaciones presentadas buscan asegurar que el Sistema Web para la Gestión y Aceptación de Proyectos de Tesis no solo cumpla con su propósito inicial de automatización, sino que también se convierta en una herramienta dinámica y evolutiva, capaz de adaptarse a los desafíos futuros y de seguir aportando valor significativo a la comunidad académica de la carrera de Ingeniería en Informática y, potencialmente, a la institución en su conjunto. La inversión en su mantenimiento, mejora y la exploración de nuevas funcionalidades basadas en la investigación, garantizarán su relevancia y eficacia a largo plazo.

10. REFERENCIAS

La presente sección compila las fuentes bibliográficas y documentales que han sustentado el desarrollo de esta tesis, proporcionando el marco teórico, metodológico y tecnológico necesario para el diseño e implementación del sistema web. Las referencias han sido seleccionadas por su relevancia académica, su autoridad en los campos de la ingeniería de software, el desarrollo web, la gestión de proyectos y la administración de bases de datos, así como por su aplicabilidad directa a la problemática y soluciones propuestas en este estudio. La correcta citación y referenciación de estas obras garantiza el rigor académico y la trazabilidad del conocimiento empleado.

Se ha seguido estrictamente el formato de la American Psychological Association (APA), 7ª edición, para la presentación de todas las entradas bibliográficas, asegurando uniformidad y claridad en la identificación de las fuentes. Esta adherencia a las normas APA facilita la verificación y profundización en los temas abordados por parte de futuros investigadores y lectores interesados en la materia.

- Pressman, R. S., & Maxim, B. R. (2020). **Software Engineering: A Practitioner's Approach** (9th ed.). McGraw-Hill Education.
- Sommerville, I. (2021). **Software Engineering** (11th ed.). Pearson Education.
- Larman, C. (2004). **Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development** (3rd ed.). Prentice Hall.
- Fowler, M. (2002). **Patterns of Enterprise Application Architecture**. Addison-Wesley Professional.
- Beck, K., & Andres, C. (2004). **Extreme Programming Explained: Embrace Change** (2nd ed.). Addison-Wesley Professional.
- Schwaber, K., & Sutherland, J. (2020). **The Scrum Guide: A Guide to Scrum**. Scrum.org.
- Freeman, E., Robson, E., Bates, B., & Sierra, K. (2004). **Head First Design Patterns**. O'Reilly Media.
- McConnell, S. (2004). **Code Complete: A Practical Handbook of Software Construction** (2nd ed.). Microsoft Press.

- PHP Documentation. (n.d.). **The PHP Manual**. Recuperado de https://www.php.net/manual/en/
- Laravel Documentation. (n.d.). **Laravel: The PHP Framework For Web Artisans**. Recuperado de https://laravel.com/docs
- Vue.js Documentation. (n.d.). **Vue.js: The Progressive JavaScript Framework**. Recuperado de https://vuejs.org/guide/
- MySQL Documentation. (n.d.). **MySQL Reference Manual**. Recuperado de https://dev.mysql.com/doc/
- Welling, L., & Thomson, L. (2009). **PHP and MySQL Web Development** (4th ed.). Addison-Wesley Professional.
- Zandstra, M. (2007). **PHP Objects, Patterns, and Practice** (3rd ed.). Apress.
- Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1994). **Design Patterns: Elements of Reusable Object-Oriented Software**. Addison-Wesley Professional.
- Krug, S. (2014). **Don't Make Me Think, Revisited: A Common Sense Approach to Web Usability** (3rd ed.). New Riders.
- Nielsen, J., & Loranger, H. (2006). **Prioritizing Web Usability**. New Riders.
- Project Management Institute. (2017). **A Guide to the Project Management Body of Knowledge (PMBOK® Guide)** (6th ed.). Project Management Institute.
- ISO/IEC/IEEE 29119-1:2013. (2013). **Software and systems engineering — Software testing — Part 1: Concepts and definitions**. International Organization for Standardization.
- IEEE Std 829-1998. (1998). **IEEE Standard for Software Test Documentation**. Institute of Electrical and Electronics Engineers.