4-4-2025

ProyectFlow

Adrián Martín Pereira

Desarrollo Aplicaciones Web - DAW

Adrián Martín Pereira
JUANA I DE CASTILLA

INDICE

AGRADECIMIENTOS

En primer lugar, quiero expresar mi más sincero agradecimiento a mi tutor Alberto, por su inestimable guía, paciencia y dedicación durante todo el desarrollo de este proyecto. Sus consejos y orientación han sido fundamentales no solo para la culminación de este trabajo, sino también para mi crecimiento como profesional en el ámbito de la informática.

También deseo agradecer al equipo docente del Juana I de Castilla especialmente a los profesores del Departamento de informática quienes a lo largo de estos años me han proporcionado los conocimientos y herramientas necesarios para afrontar este reto. Su experiencia y pasión por la enseñanza han sido una constante fuente de inspiración.

Mi agradecimiento se extiende a mis compañeros de clase, con quienes he compartido largas jornadas de estudio, dudas y satisfacciones. El apoyo mutuo y el trabajo colaborativo han enriquecido enormemente mi experiencia formativa. Especialmente quiero mencionar a David Iglesias Cuadrado cuyas aportaciones y perspectivas han sido particularmente valiosas durante el desarrollo de este proyecto.

Un reconocimiento especial merece MADISON MK por brindarme la oportunidad de aplicar mis conocimientos en un entorno profesional real y proporcionarme acceso a recursos que han sido clave para la realización de este trabajo.

En el ámbito personal, mi más profunda gratitud a mi familia, pilar fundamental en mi vida. A mis padres, por su apoyo incondicional, por creer en mí incluso cuando yo dudaba, y por enseñarme el valor del esfuerzo y la perseverancia. A mis hermanos/as, por su comprensión y ánimos constantes.

A mis amigos, por entender mis ausencias durante los períodos más intensos de trabajo y por ofrecerme siempre un espacio para desconectar y recargar energías.

Finalmente, a todas aquellas personas que, de una u otra manera, han contribuido a que este proyecto sea hoy una realidad. A todos ellos, mi más sincero agradecimiento.

Este trabajo es también vuestro.

1. INTRODUCCIÓN

1.1. Interés del tema y justificación

La gestión eficiente de proyectos representa uno de los desafíos más significativos para organizadores y equipos en prácticamente todos los sectores productivos. En un entorno caracterizado por la transformación digital acelerada, la capacidad para planificar, ejecutar y supervisar proyectos de manera efectiva se ha convertido en una competencia diferenciadora.

Las metodologías ágiles han revolucionado la forma en que se desarrollan los proyectos, especialmente en el ámbito tecnológico. Entre estas metodologías, Kanban destaca por su simplicidad conceptual y potencia práctica. Originado en los sistemas de producción de Toyota, Kanban se fundamenta en la visualización del flujo de trabajo, la limitación del trabajo en progreso y la optimización continua de los procesos.

La relevancia del desarrollo de ProyectFlow se justifica por varios factores:

- **Tendencia creciente hacia la agilidad:** Las organizaciones buscan cada vez más herramientas que faciliten la implementación de metodologías agiles, permitiéndoles adaptarse rápidamente a cambios en el mercado y requerimientos.
- **Necesidad de simplificación:** A pesar de la proliferación de herramientas de gestión de proyectos, muchas resultan excesivamente complejas o rígidas, generando una barrera de entrada para equipos pequeños o medianos
- **Demanda de soluciones integradas:** Existe una necesidad de plataformas que combinen la gestión visual de tareas con capacidades de colaboración, seguimiento y análisis, sin fragmentar el flujo de trabajo entre múltiples herramientas.
- **Incremento del trabajo remoto:** La expansión de equipos distribuidos geográficamente requiere soluciones que faciliten la coordinación y visualización del trabajo compartido, manteniendo a todos los miembros alineados con los objetivos del proyecto.

ProjectFlow nace como respuesta a estas necesidades, ofreciendo una plataforma accesible para la implementación de Kanban en diversos contextos organizacionales.

1.2. Tecnologías escogidas y justificación

El desarrollo de ProjectFlow se ha fundamentado en la utilización del stack MERN modificado (MySQL, Expres, React y Node.js), complementando con tecnologías adicionales que potencias su funcionalidad, seguridad y mantenibilidad:

MySQL: Este sistema de gestión de bases de datos relacional ha sido seleccionado por:

- Robustez y fiabilidad comprobada en diversos entornos.
- Estructura relacional que permite modelar con precisión las relaciones entre proyectos, tableros, tareas y usuarios.
- Soporte para transacciones ACID, garantizando la integridad de los datos.
- Optimización para consultas complejas mediante indexación eficiente.
- Amplio soporte y documentación, facilitando la implementación y resolución de problemas.

Express.js: Este framework para Node.js ha sido seleccionado por:

- Su simplicidad y minimalismo, facilitando la creación de APIs RESTful.
- Su amplia adopción en la industria, garantizando soporte y documentación extensa.
- Su capacidad de integración con middleware diversos para autentificación, validación y seguridad.
- Su rendimiento optimizado para aplicaciones web modernas.

React.js: Esta biblioteca para el desarrollo de interfaces ha sido elegida por:

- Su arquitectura basada en componentes, ideal para los elementos visuales repetitivos de un tablero Kanban.
- El Virtual DOM, que optimiza el rendimiento en interfaces interactivas con frecuentes actualizaciones.
- La posibilidad de crear una experiencia de usuario fluida mediante estados reactivos.
- Su extensa comunidad y ecosistema de componentes y herramientas complementarias.

Node.js: Este entorno de ejecución JavaScript se ha seleccionado por:

- La unificación del lenguaje de programación entre frontend y backend (JavaScript).
- Su modelo asíncrono y orientado a eventos, ideal para operaciones concurrentes.
- Su eficiencia en aplicaciones con operaciones intensivas de entrada/salida.

- La amplia disponibilidad de paquetes a través de NPM que aceleran el desarrollo.

Tecnologías complementarias:

- **Redux:** Para la gestión del estado global de la aplicación, facilitando la comunicación entre componentes.
- **Material-UI:** Framework de componentes que implementa los principios de Material Design, acelerando el desarrollo de interfaces consistentes y responsivas.
- **JWT (JSON Web Tokens):** Implementados para la autenticación y autorización segura de usuarios.
- **Github Actions:** Para la implementación de integración continua, asegurando la calidad del código.

La combinación de MySQL con el resto del stack proporciona una base sólida para la gestión estructurada de datos, manteniendo la flexibilidad y dinamismo en la capa de presentación que ofrece React. Esta arquitectura garantía la escalabilidad, mantenibilidad y seguridad del sistema, respondiendo adecuadamente a los requisitos del proyecto.

1.3. Campos de aplicación.

ProjectFlow ha sido diseñado con la versatilidad como principio rector, permitiendo su aplicación en diversos contextos y sectores:

Desarrollo de software: Equipos de desarrollo pueden utilizar ProjectFlow para implementar Kanban en sus procesos de creación de software, facilitando la visualización del progreso, limitando el trabajo en curso y optimizando el flujo desde la concepción hasta el despliegue.

Gestión de proyectos empresariales: Departamentos y equipos de cualquier organización pueden beneficiarse de la claridad visual que proporciona para el seguimiento de iniciativas, coordinación entre miembros y monitorización de avances.

Educación: Instituciones educativas pueden implementar la plataforma para la gestión de proyectos académicos, trabajos grupales o seguimiento del progreso de los estudiantes en distintas asignaturas.

Marketing y contenidos: Equipos creativos y de marketing pueden utilizar ProjectFlow para planificar y ejecutar campañas, organizar calendarios editoriales o coordinar la producción de materiales.

Gestión personal: Individuos pueden adoptar la herramienta para organizar sus propias tareas y proyectos personales, aprovechando los principios Kanban para optimizar su productividad.

Startups y pequeñas empresas: Organizaciones con recursos limitados pueden beneficiarse de una solución accesible que no requiere infraestructura compleja ni formación extensa.

La flexibilidad inherente a ProjectFlow permite su adaptación a procesos existentes, sin imponer metodologías rígidas, facilitando así su adopción en diversos entornos organizacionales.

1.4. Relación con el ciclo formativo

El desarrollo de ProjectFlow integra numerosas competencias y conocimientos adquiridos durante el ciclo formativo de Desarrollo de Aplicaciones Web, representando una síntesis práctica de diversos módulos:

Programación: Aplicación de principios de programación orientada a objetos, patrones de diseño y algoritmia para la implementación de la lógica de negocio.

Desarrollo web en entorno cliente: Implementación de interfaces de usuario interactivas mediante JavaScript moderno y React, gestión de eventos, manipulación del DOM y comunicación asíncrona.

Desarrollo web en entorno servidor: Creación de una API RESTful con Express, implementación de controladores, middleware y lógica de negocio en el backend.

Bases de datos: Diseño e implementación de modelos de datos relacionales, normalización de tablas, definición de relaciones mediante claves primarias y foráneas, y desarrollo de consultas SQL eficientes para operaciones CRUD en MySQL.

Diseño de interfaces: Aplicación de principios de usabilidad, accesibilidad y diseño responsive para crear una experiencia de usuario intuitiva y efectiva.

Entornos de desarrollo: Utilización de herramientas de control de versiones (Git), integración continua y testing automatizado.

Formación y orientación laboral: Gestión del proyecto aplicando metodologías ágiles, planificación de tiempos y recursos, y evaluación de resultados.

Este proyecto ha permitido consolidar estos conocimientos en un caso práctico integral, enfrentando desafíos reales de desarrollo y proporcionando una experiencia formativa holística que trasciende los límites individuales de cada módulo.

1.5. Antecedentes y referencias

El desarrollo de ProjectFlow se ha nutrido del análisis de diversas soluciones existentes en el ámbito de la gestión de proyectos y metodologías ágiles, así como de herramientas avanzadas de desarrollo:

Asana: Esta plataforma de gestión de trabajo ha sido una referencia fundamental durante todo el desarrollo, ya que es la herramienta utilizada en la empresa donde realicé las prácticas. La experiencia directa con Asana en un entorno profesional real ha proporcionado:

- Comprensión profunda de los flujos de trabajo en equipos profesionales
- Identificación de fortalezas a incorporar, como su flexibilidad en la visualización de tareas y su sistema de notificaciones
- Reconocimiento de áreas de mejora, especialmente en la simplificación de la interfaz y accesibilidad para nuevos usuarios
- Inspiración para funcionalidades como el seguimiento del tiempo y la asignación de tareas

Trello: Su enfoque minimalista y visual para la gestión de tableros Kanban ha inspirado aspectos de la interfaz de usuario de ProjectFlow, aunque nuestra solución busca expandir las capacidades analíticas y de personalización.

Jira: Su robustez para la gestión de proyectos complejos ha servido como referencia para algunas funcionalidades avanzadas, aunque simplificando significativamente la curva de aprendizaje.

Kanban Tool: Su fidelidad a los principios originales de Kanban ha influido en el diseño de las métricas y visualizaciones de flujo de trabajo en ProjectFlow.

GitHub Copilot: Esta herramienta de inteligencia artificial ha sido fundamental en el proceso de desarrollo, acelerando la codificación mediante sugerencias contextuales basadas en el aprendizaje automático. El uso de GitHub Copilot ha permitido:

- Aumentar la productividad en la generación de código repetitivo o estándar
- Obtener sugerencias de implementaciones óptimas basadas en buenas prácticas
- Explorar soluciones alternativas para problemas de desarrollo complejos
- Acelerar la creación de consultas SQL y componentes React
- Mejorar la calidad del código mediante sugerencias contextualizadas

The Kanban Guide de Lean Kanban University: Los principios y prácticas definidos en esta guía han fundamentado el diseño conceptual de la aplicación, garantizando la adherencia a la metodología Kanban.

Material Design Guidelines: Los principios de diseño de Google han orientado las decisiones de interfaz, priorizando la claridad visual, la jerarquía de información y la interacción intuitiva.

La implementación técnica también se ha apoyado en recursos como la documentación oficial de MySQL, Express, React y Node.js, así como en patrones arquitectónicos establecidos para aplicaciones web modernas, adaptándolos a las necesidades específicas del proyecto.

ProjectFlow no pretende reinventar conceptos fundamentales de gestión visual, sino integrarlos en una plataforma accesible que equilibre simplicidad y potencia. Las soluciones mencionadas y herramientas como GitHub Copilot han servido como referencia y apoyo para identificar tanto buenas prácticas a incorporar como deficiencias a superar, acelerando el proceso de desarrollo y permitiendo enfocar los esfuerzos en la resolución de problemas específicos del dominio.

2. DESCRIPCIÓN DE LA APLICACIÓN

2.1. Descripción general de ProjectFlow

ProjectFlow es una aplicación web completa para la gestión de proyectos basada en metodologías agiles especialmente enfocada en el sistema Kanban. La plataforma permite a equipos de trabajo y organizaciones visualizar, planificar y optimizar su flujo de trabajo mediante tableros interactivos que representan el avance de las tareas desde su concepción hasta su finalización.

La aplicación implementa un sistema de gestión visual donde las tareas son representadas como tarjetas que se mueven a través de diferentes columnas, cada una representando un estado específico en el ciclo de vida del trabajo (por ejemplo: Pendiente, En Progreso, Finalizado). Esta visualización proporciona transparencia sobre el estado actual del proyecto y permite identificar cuellos de botella o bloqueos en el flujo de trabajo.

ProjectFlow integra las siguientes funcionalidades principales:

- **Gestión de proyectos y equipos:** Creación de múltiples proyectos con equipos asignados específicamente.
- **Sistemas de tarjetas interactivas:** Representación de tareas con información detallada, responsables, fechas de vencimiento y prioridad.
- **Asignación de responsabilidades:** Capacidad para asignar tareas a miembros específicos del equipo.
- **Sistema de comentarios:** Comunicación contextual dentro de cada tarea.
- **Gestión de roles y permisos:** Diferenciación entre administradores y miembros con distintos niveles de acceso.
- **Interfaz responsive:** Adaptación a diferentes dispositivos para facilitar el acceso desde cualquier lugar.

La plataforma ha sido diseñada priorizando la experiencia de usuario, con una interfaz intuitiva que reduce la curva de aprendizaje y permite a los equipos comenzar a utilizar el sistema de manera inmediata y efectiva.

2.2. Motivación y finalidad

La creación de ProjectFlow surge como respuesta a diversas necesidades identificadas tanto en el ámbito educativo como profesional:

Motivación académica:

- Aplicar de forma práctica e integral los conocimientos adquiridos durante el ciclo formativo de Desarrollo de Aplicaciones Web.
- Desarrollar un proyecto completo que abarque tanto frontend como backend, demostrando competencias en ambos ámbitos.
- Enfrentar desafíos reales de desarrollo, como la gestión de estados complejos, la autenticación segura y la optimización del rendimiento.
- Experimentar con el ciclo completo de desarrollo de software, desde la conceptualización hasta la implementación y pruebas.

Motivación práctica:

- Crear una herramienta que simplifique la adopción de metodologías agiles, especialmente Kanban, en equipos de diversos tamaños y sectores.
- Ofrecer una alternativa a las soluciones existentes que equilibre funcionalidad y simplicidad, evitando la sobrecarga de características que dificultan la adopción.
- Desarrollar una plataforma que pueda adaptarse a diferentes contextos y necesidades organizacionales sin requerir configuraciones complejas.
- Implementar prácticas efectivas de visualización de flujos de trabajo que promuevan la transparencia y mejora continua en los equipos.

Finalidad:
La finalidad principal de ProjectFlow es proporcionar una plataforma accesible que facilite la implementación de Kanban en diversos contextos organizacionales, ayudando a los equipos a:

1. **Visualizar su trabajo:** permitiendo a todos los miembros del equipo ver el estado actual de las tareas y el proyecto en general.
2. **Limitar el trabajo en progreso (WIP):** proporcionando mecanismos para establecer límites que eviten la sobrecarga y mejoren el flujo de trabajo.
3. **Gestionar el flujo:** facilitando la identificación y resolución de bloqueos o cuellos de botella que afectan a la productividad.
4. **Hacer explícitas las políticas de proceso:** definiendo claramente los criterios para que una tarea avance a través de los diferentes estados.
5. **Implementar ciclos de feedback:** integrando mecanismos para la revisión continua y mejora de los procesos.
6. **Mejorar colaborativamente:** proporcionando visualizaciones que apoyen la evolución constante del sistema de trabajo.

En última instancia, ProjectFlow busca democratizar el acceso a herramientas de gestión ágil efectivas, permitiendo que organizaciones de cualquier tamaño puedan beneficiarse de los principios Kanban para incrementar su eficiencia, transparencia y capacidad de adaptación.

3. OBJETIVOS

3.1. Objetivos funcionales

Los objetivos funcionales definen las capacidades y servicios específicos que el sistema debe proporcionar a sus usuarios. Para ProjectFlow, se establecieron los siguientes objetivos funcionales prioritarios:

1. **Implementar un sistema completo de gestión de proyectos basado en tableros Kanban:**
   - Crear tableros visuales con columnas que representen los diferentes estados del flujo de trabajo.
   - Permitir a la organización de tareas mediante tarjetas interactivas que pueden moverse entre columnas.
2. **Desarrollar un sistema robusto de gestión de tareas:**
   - Crear, editar y eliminar tareas con información detallada (título, descripción, responsable, fecha límite y prioridad).
   - Asignar y reasignar tareas a diferentes miembros del equipo.
   - Establecer dependencias entre tareas cuando sea necesario.
   - Adjuntar comentarios relevantes a cada tarea.
3. **Implementar un sistema de gestión de usuarios y equipos:**
   - Registro e inicio de sesión seguro de usuarios.
   - Creación y administración de equipos de trabajo.
   - Asignación de usuarios a proyectos específicos con roles diferencias.
   - Gestión de permisos basados en roles (administrador, miembro).
4. **Proporcionar herramientas para el seguimiento y análisis del trabajo:**
   - Visualización del progreso actual de cada proyecto mediante indicadores claros.
   - Visualización de métricas clave como tiempo de ciclo y distribución de tareas.
5. **Garantizar una interfaz de usuario intuitiva y responsive:**
   - Diseñar una experiencia de usuario coherente y accesible.
   - Implementar navegación intuitiva entre proyectos, tableros y tareas.
   - Asegurar que la aplicación sea completamente funcional en dispositivos móviles y tablets.
   - Incorporar elementos de interacción como arrastrar y soltar para la gestión de tarjetas.
6. **Desarrollar funcionalidades de comunicación y colaboración:**
   - Sistema de comentarios en tareas para facilitar la comunicación contextual.
   - Vista compartida del tablero Kanban para todos los miembros del equipo.

- Histórico de actividad para seguimiento de cambios en proyectos y tareas.

7. **Implementar un sistema de almacenamiento y recuperación de datos:**
   - Almacenamiento seguro y eficiente de la información en base de datos relacional.
   - Respaldo periódico automático de los datos críticos.
   - Recuperación efectiva de información ante posibles fallos.
   - Optimización del rendimiento en consultas frecuentes.

3.2. Objetivos técnicos

Los objetivos técnicos se centran en las características arquitectónicas, tecnologías y de implementación que debe cumplir el sistema. Para ProjectFlow, se definieron los siguientes objetivos técnicos:

1. **Desarrollar una arquitectura de aplicación robusta y escalable:**
   - Implementar una estructura modular que facilite el mantenimiento y la evolución del sistema.
   - Diseñar una arquitectura de tres capas bien definidas: presentación, lógica de negocio y acceso a datos.
   - Garantizar la escalabilidad horizontal para soportar el crecimiento en número de usuarios y proyectos.
   - Establecer claras separaciones de responsabilidades entre componentes.
2. **Implementar una API RESTful completa y bien documentada:**
   - Diseñar endpoints coherentes siguiendo las mejores prácticas REST.
   - Implementar la gestión adecuada de códigos de estado HTTP.
   - Desarrollar respuestas JSON estandarizadas.
   - Crear documentación detallada usando estándares como OpenAPI.
3. **Crear un modelo de datos relacional normalizado y eficiente:**
   - Diseñar un esquema de base de datos que represente adecuadamente las entidades del sistema.
   - Implementar relaciones apropiadas entre tablas para mantener la integridad relacional.
   - Optimizar el modelo para consultas frecuentes mediante indexación adecuada.
   - Establecer restricciones y validaciones a nivel de base de datos.
4. **Garantizar la seguridad integral del sistema:**
   - Implementar autenticación segura mediante tokens JWT.
   - Establecer autorización basada en roles para controlar el acceso a funcionalidades.
   - Proteger contra vulnerabilidades comunes (XSS, CSRF, inyección

SQL).
- Cifrar datos sensibles tanto en tránsito como en reposo.

5. **Asegurar la calidad y mantenibilidad del código:**
   - Seguir patrones de diseño establecidos y buenas prácticas de programación.
   - Implementar un sistema de control de versiones con Git y flujos de trabajo definidos.
   - Establecer estándares de codificación consistentes.
   - Documentar adecuadamente el código y los componentes críticos.
6. **Optimizar el rendimiento de la aplicación:**
   - Implementar técnicas de caché para reducir tiempos de respuesta.
   - Optimizar consultas a la base de datos para minimizar la latencia.
   - Implementar carga perezosa (lazy loading) y paginación donde sea apropiado.
   - Comprimir y optimizar recursos estáticos (JavaScript, CSS, imágenes).
7. **Desarrollar una interfaz de usuario moderna con React:**
   - Implementar una arquitectura de componentes reutilizables.
   - Gestionar eficientemente el estado global de la aplicación con Redux.
   - Optimizar el renderizado y el ciclo de vida de los componentes.
   - Implementar pruebas unitarias para los componentes principales.
8. **Garantizar la compatibilidad y accesibilidad:**
   - Asegurar que la aplicación funcione en los principales navegadores modernos.
   - Implementar diseño responsive mediante CSS moderno y frameworks como Material-UI.
   - Seguir pautas WCAG 2.1 para garantizar la accesibilidad.
   - Optimizar par diferentes tamaños de pantalla y dispositivos.

3.3. Objetivos personales

Además de los objetivos funcionales y técnicos, el desarrollo de ProjectFlow también persigue una serie de objetivos personales relacionados con mi crecimiento profesional y académico:

1. **Consolidad y aplicar los conocimientos adquiridos durante el ciclo formativo:**
   - Integrar de manera coherente conceptos de múltiples módulos en un proyecto real.
   - Demostrar competencia técnica en el desarrollo tanto frontend como backend.
   - Aplicar principios aprendidos sobre bases de datos, seguridad y optimización.

- Poner en práctica metodologías de desarrollo ágil en un proyecto individual.

2. **Expandir mis competencias técnicas en tecnologías específicas:**
   - Profundizar en el ecosistema React y sus bibliotecas asociadas (Redus, React Router).
   - Mejorar mis habilidades con Node.js y Express para la creación de APIs robustas.
   - Adquirir experiencia práctica en la integración de MySQL con aplicaciones web modernas.
   - Experimentrar con herramientas de desarrollo avanzadas como Github Copilot.
3. **Desarrollar capacidades de planificación y gestión de proyectos:**
   - Practicar la estimación realista de tiempos y recursos.
   - Implementar metodologías ágiles para la gestión de mi propio trabajo.
   - Aprender a priorizar tareas y administrar el alcance del proyecto.
   - Documentar adecuadamente decisiones y procesos.
4. **Mejorar habilidades de resolución de problemas:**
   - Enfrentar y superar desafíos técnicos complejos de manera independiente.
   - Desarrollar estrategias efectivas para depurar y resolver problemas.
   - Aprender a investigar y aplicar soluciones de fuentes diversas.
   - Mejorar la capacidad para dividir problemas complejos en componentes manejables.
5. **Crear un proyecto significativo para mi portafolio profesional:**
   - Desarrollar una aplicación completa que demuestre mis capacidades técnicas.
   - Documentar el proceso y las decisiones técnicas para futuras referencias.
   - Establecer una base que pueda continuar evolucionando y mejorando.
   - Generar un proyecto que pueda presentar con confianza a potenciales empleadores.
6. **Explorar áreas de especialización potencial:**
   - Evaluar mi afinidad con diferentes aspectos del desarrollo web.
   - Identificar áreas de interés para profundización futura (UX/UI, arquitectura, seguridad).
   - Descubrir fortalezas y debilidades en el proceso de desarrollo.
   - Orientar mi desarrollo profesional futuro basado en esta experiencia.
7. **Aplicar aprendizaje continua y adaptabilidad:**
   - Mantenerme abierto a nuevos enfoques y tecnologías durante el desarrollo.
   - Adaptarme a los desafíos imprevistos que surjan durante el proyecto.
   - Buscar activamente retroalimentación para mejorar y el producto y mi proceso.
   - Documentar lecciones aprendidas para aplicarlas en futuros proyectos.

Estos objetivos personales complementan los aspectos técnicos y funcionales, proporcionando una motivación adicional y un marco para evaluar el éxito del proyecto más allá de los entregables concretos.

4. RECURSOS NECESARIOS PARA EL DESARROLLO

4.1. Recursos Humanos

El desarrollo de ProjectFlow ha sido llevado principalmente como un proyecto individual, aunque ha contado con diferentes niveles de soporte y colaboración:

Roles Principales:

Desarrollador Principal (Adrián Martín Pereira): Responsable de la totalidad del diseño, implementación, pruebas y documentación del sistema. Las responsabilidades específicas incluyen:

- Análisis de requisitos y diseño de la arquitectura.
- Desarrollo frontend con React y Material-UI.
- Implementación del backend con Node.js y Express.
- Diseño y gestión de la base de datos MySQL.
- Prueba e integración de todos los componentes.
- Documentación técnica y de usuario.

Roles de soporte:

Tutor académico (Alberto): Proporcionó orientación metodológica, revisión periódica del progreso y asesoramiento técnico cuando fue necesario. Sus contribuciones incluyeron:

- Validación de la propuesta inicial y alcance del proyecto.
- Seguimiento regular del avance y cumplimiento de objetivos.
- Orientación sobre estándares y mejores prácticas.
- Retroalimentación sobre aspectos técnicos y funcionales.

Colaboradores puntuales (compañeros de clase): Proporcionaron retroalimentación valiosa durante sesiones de revisión y pruebas. Sus aportaciones incluyeron:

- Pruebas de usabilidad y detección de errores.
- Sugerencias de mejora para la interfaz de usuario.
- Ideas para nuevas funcionalidades.

Asesores técnicos (profesionales del sector): Consultaos ocasionalmente para resolver dudas específicas sobre implementaciones técnicas avanzadas o buenas prácticas del sector.

Metodología de trabajo individual:

Para maximizar la eficiencia en un proyecto principalmente individual, se implementaron las siguientes estrategias:

- Organización del trabajo en sprints semanales con objetivos claramente definidos.

- Mantener un registro detallado de tareas pendientes, en progreso y completadas.
- Sesiones diarias de planificación y revisión para mantener el enfoque.
- Establecimiento de hitos intermedios verificables para evaluar el progreso.
- Documentación del código y decisiones de diseño.
- Revisiones regulares con el tutor para validar el avance y corregir desviaciones.

Esta estructura de recursos humanos ha permitido mantener la coherencia del proyecto y la calidad del desarrollo a la vez que ha proporcionado suficiente retroalimentación externa para evitar sesgos o puntos ciegos en el diseño y la implementación.

4.2. Recursos hardware

Para el desarrollo completo de ProjectFlow se han utilizado diversos recursos hardware que han garantizado un entorno de trabajo eficiente y las capacidades necesarias para el desarrollo, pruebas de la aplicación:

Equipo principal de desarrollo:

- Procesador: AMD Ryzen 7 5700X
- Memoria RAM: 32 GB DDR4 3200MHz
- Almacenamiento SSD NVMe de 1TB
- Tarjeta grafica: NVIDIA GeForce RTX 3060 6gb GDDR6
- Pantalla: 15.6” Full HD (1920x1080) IPS, 165Hz
- Sistema Operativo: Windows 11 Pro

Este equipo ha proporcionado el rendimiento necesario para ejecutar simultáneamente múltiples entornos de desarrollo, servidores locales, bases de datos y herramientas de diseño sin comprometer la velocidad de trabajo.

Dispositivos secundarios para pruebas:

Tablet Samsung Galaxy Tab S7

- Pantalla: 11" LTPS TFT (2560x1600)
- Procesador: Qualcomm Snapdragon 865+
- Memoria RAM: 6GB
- Sistema Operativo: Android 11

Utilizada para pruebas de responsividad y experiencia de usuario en dispositivos táctiles de tamaño medio.

- Smartphone Xiaomi Redmi Note 10 Pro
- Pantalla: 6.67" AMOLED (2400x1080)
- Procesador: Qualcomm Snapdragon 732G
- Memoria RAM: 6GB
- Sistema Operativo: Android 11

Empleado para verificar la usabilidad en dispositivos móviles y ajustar la interfaz para pantallas pequeñas.

Estos recursos hardware han proporcionado un entorno de desarrollo versátil y robusto, permitiendo no solo el desarrollo eficiente de ProjectFlow, sino también pruebas exhaustivas en diferentes plataformas y dispositivos, asegurando así la compatibilidad y experiencia de usuario consistente en diversos escenarios de uso.

4.3. Recursos Software

El desarrollo de ProjectFlow ha requerido un conjunto diverso de herramientas y plataformas software que han facilitado la implementación, pruebas, gestión y documentación del proyecto:

Entorno de desarrollo:

- Visual Studio Code: Editor de código principal, configurado con extensiones específicas para:
    - ESLint y Prettier: Garantizar la calidad y consistencia del código.
    - Github Copilot: Asistencia en la codificación mediante IA.
    - GitLens: Visualización avanzada de cambios en el repositorio.
- Node.js (v16.14.2): Entorno de ejecución JavaScript para el backend.
- Npm (v8.5.0): Gestor de paquetes para la instalación y gestión de dependencias.

Frameworks y bibliotecas principales:

- React (v18.2.0): Biblioteca JavaScript para la construcción de interfaces de usuario
- Express (v4.18.1): Framework web para Node.js, utilizado en el desarrollo del backend
- MySQL (v8.0.27): Sistema de gestión de bases de datos relacional
- Sequelize (v6.21.0): ORM para Node.js, utilizado para la gestión de modelos y consultas a la base de datos
- Material-UI (v5.8.6): Biblioteca de componentes React para la implementación de Material Design
- Redux (v8.0.2) y Redux Toolkit: Para la gestión del estado global de la aplicación
- JWT (jsonwebtoken v8.5.1): Implementación de autenticación basada en tokens
- bcrypt (v5.0.1): Librería para el hash seguro de contraseñas

Control de versiones y colaboración:

- Git (v2.36.1): Sistema de control de versiones distribuido.
- GitHub: Plataforma para alojar el repositorio y gestionar el desarrollo.
- GitHub Desktop: Cliente gráfico para le gestión del repositorio.

Herramientas de pruebas:

- Jest (v28.1.1): Framework de pruebas para JavaScript.
- Chrome DevTools: Suite de herramientas para depuración y análisis de rendimiento.

Bases de datos y servidores:

- MySQL Workbench: Herramienta visual para el diseño y administración de base de datos MySQL.
- XAMPP: Entorno de despliege local que incluye Apache, MySQL y PHP.

Documentación y gestión del proyecto:

- Microsoft Office (Word, Excel, PowerPoint): Creación de documentación final y presentaciones.

Navegadores para pruebas:

- Google Chrome (v103.0)
- Microsoft Edge (v103.0)

Este conjunto de herramientas software ha proporcionado todas las capacidades necesarias para el desarrollo completo de ProjectFlow, desde la conceptualización inicial hasta la implementación final, asegurando la calidad, consistencia y rendimiento adecuados en todas las fases del proyecto.

5. PLANIFICACIÓN

5.1. Metodología para el diseño del proyecto

Para el desarrollo de ProjectFlow se ha adoptado una metodología ágil adaptada, basada principalmente en Scrum, pero con elementos de Kanban para la gestión del flujo de trabajo. Esta elección metodología responde a la naturaleza del proyecto: desarrollo individual con supervisión periódica, alcance bien definido, pero con espacio para adaptaciones, y necesidad de entregas incrementales funcionales.

La adaptación metodológica implementada ha consistido en:

Elementos adoptados de Scrum:

- División del proyecto en sprints.
- Definición de un backlog priorizado de requisitos.
- Refinamiento continuo de requisitos y retroalimentación.

Elementos adoptados de Kanban:

- Visualización del flujo de trabajo mediante tablero Kanban.
- Limitación del trabajo en curso (WIP) para cada fase.
- Medición y optimización del tiempo de ciclo.
- Mejora continua del proceso.

Adaptaciones específicas para un proyecto individual:

- Flexibilidad en la duración de los sprints.
- Mayor énfasis en la documentación continua al no existir conocimiento compartido entre múltiples miembros del equipo.

El ciclo de desarrollo implementado ha seguido el siguiente esquema:

1. **Planificación inicial y visión general:**
   - Definición del alcance y objetivos.
   - Identificación de requisitos principales.
   - Estructuración del backlog inicial.
   - Establecimiento de hitos principales.
2. **Ciclo de desarrollo iterativo (por sprints):**
   - Planificación del sprint: Selección de elementos del backlog a implementar.
   - Desarrollo: Implementación de las funcionalidades planificadas.
   - Pruebas: Verificación del correcto funcionamiento de lo implementado.
   - Revisión con el tutor: Demostración y retroalimentación.
   - Retrospectiva personal: Análisis de lo que funcionó bien y áreas de mejora.
   - Actualización del backlog: Ajustes basados en la retroalimentación y los aprendizajes.

3. **Entrega y cierre:**
   - Pruebas globales del sistema.
   - Documentación final
   - Presentación

Esta metodología ha proporcionado las siguientes ventajas para el desarrollo de ProjectFlow:

- Flexibilidad para adaptarse a cambios en los requisitos o dificultades técnicas.
- Entrega incremental de funcionalidad, facilitando la detección temprana de problemas.
- Visibilidad constante del progreso y estado del proyecto.
- Enfoque en la calidad y la completitud funcional de cada componente desarrollado.
- Oportunidades regulares para la retroalimentación y ajustes.
- Mejora continua tanto del producto como del proceso de desarrollo.

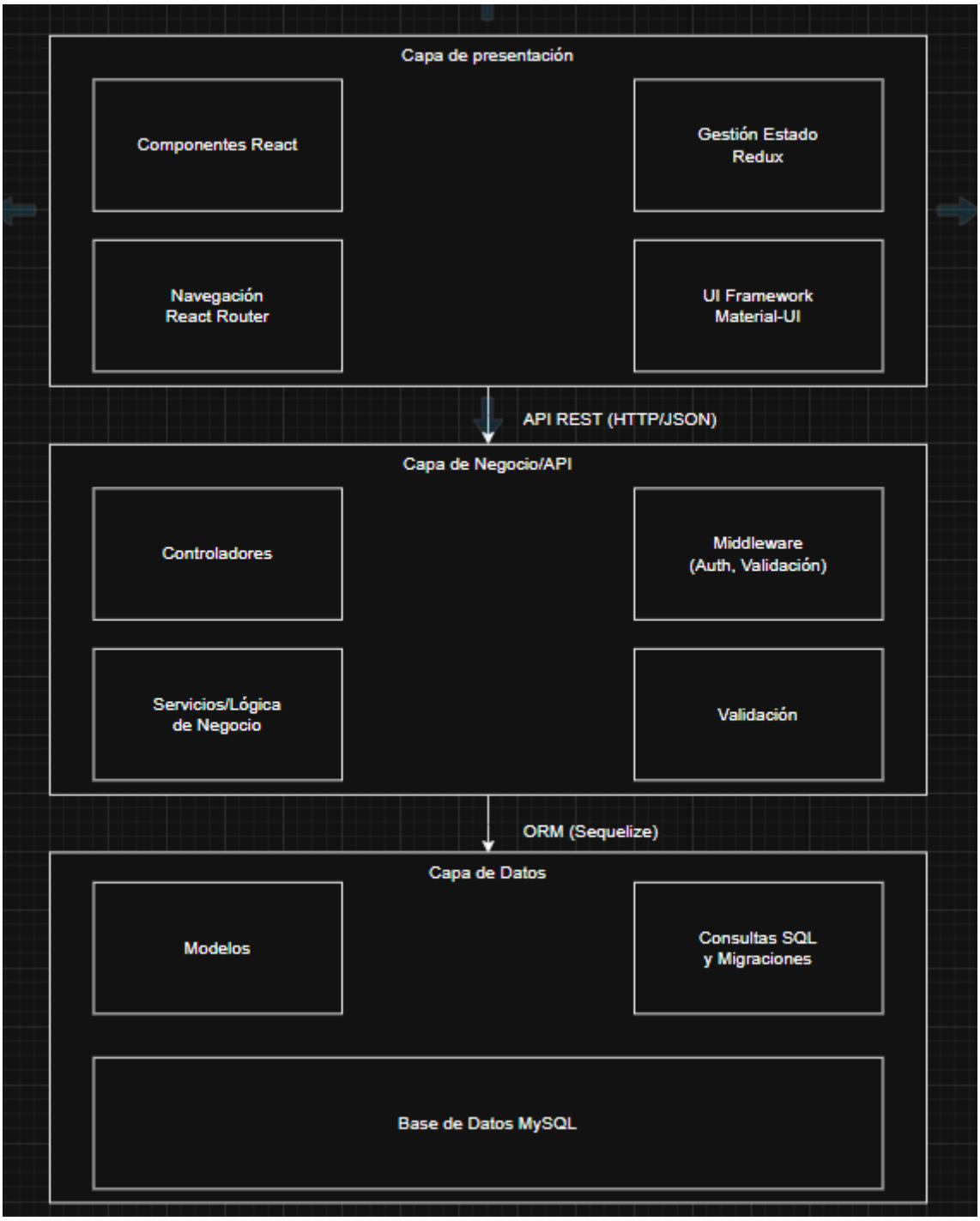
El uso de un enfoque ágil ha resultado particularmente valioso para gestionar la complejidad inherente al desarrollo de una aplicación completa que abarca múltiples tecnologías y componentes interrelacionados, permitiendo mantener el control sobre el progreso y la calidad a lo largo de todo el ciclo de vida del proyecto.

5.2. Arquitectura de la aplicación

La arquitectura de global del sistema se compone de tres capas principales:

1. **Capa de presentación (Frontend):**
   - Implementada con React.js.
   - Interfaz de usuario reactiva y componentes reutilizables.
   - Gestión de estado centralizada con Redux.
   - Comunicación con el backend a través de solicitudes HTTP.
2. **Capa de Lógica de Negocio (Backend):**
   - Desarrollada con Node.js y Express.js.
   - API RESTful para la comunicación con el frotend.
   - Implementación de la lógica de negocio y reglas de la aplicación
   - Middleware para autenticación, validación y gestión de errores.
   - Servicios para operaciones complejas y transaccionales.
3. **Capa de datos:**
   - Base de datos relacional MySQL.
   - ORM Sequelize para la abstracción y manipulación de datos.
   - Modelos que representan las entidades del sistema.
   - Repositorios para la encapsulación de operaciones de datos complejas.

Diagrama de Arquitectura



Arquitectura del Frontend

El frontend de ProjectFlow sigue una arquitectura basada en componentes con las siguientes características:

1. **Estructura de Componentes:**
   - Componentes atómicos reutilizables (botones, campos, tarjetas).
   - Componentes de contenedor que gestionan la lógica y el estado.
   - Componentes de pagina que representan rutas completas de la aplicación.
   - Componentes HOC (Higher-Order Componentes) para funcionalidades transversales.
2. **Gestión de Estado:**
   - Redux como store centralizado para el estado global.
   - Redux Toolkit para simplificar la configuración y reducir el boilerplate.
   - Estado local en componentes para datos que no necesitan ser compartidos.
   - Slices organizados por dominio (proyectos, tareas, usuario).
3. **Enrutamiento:**
   - React Router para la navegación entre páginas.
   - Rutas protegidas que requieren autenticación.
   - Navegación programática basada en acciones del usuario.
4. **Servicios:**
   - API clientes para comunicación con el backend.
   - Interceptores para gestión de token y errores.
   - Adaptadores para trasnformar datos entre frontend y backend.
5. **Estilos y UI:**
   - Material-UI como framework de componentes base.
   - Sistema de tema personalizado.
   - CSS Modules para estilos específicos de componentes.
   - Responsive design para adaptación a diferentes dispositivos.

Arquitectura del Backend

El backend de ProjectFlow implementa una arquitectura en capas con las siguientes características:

1. **Capa de API:**
   - Controladores REST que definen los endpoints.
   - Middleware para autenticación, validación y manejo de errores.
   - Rutas agrupadas por dominio funcional.
   - Gestión de repuestas y códigos HTTP.
2. **Capa de Servicios:**
   - Implementación de la lógica de negocio.
   - Orquestación de operaciones complejas.

- Validaciones específicas del dominio.
- Transacciones para operaciones que afectan a múltiples entidades.

3. **Capa de acceso a Datos:**
   - Modelos Sequelize que definen la estructura y relaciones.
   - Repositorios para encapsular operaciones de datos complejas.
   - Queries optimizadas para consultas frecuentes.
   - Migraciones para control de versiones de la base de datos.
4. **Características Transversales:**
   - Sistema de autenticación basado en JWT.
   - Gestión centralizada de errores.
   - Logging y monitoring.
   - Validación de entradas

Arquitectura de Datos:

El modelo de datos de ProjectFlow está diseñado para representar de manera eficiente las entidades principales del sistema y sus relaciones:

1. **Principales Entidades:**
   - Usuarios: Información de usuarios registrados.
   - Proyectos: Datos de los proyectos gestionados.
   - Tareas: Elementos de trabajo dentro de los proyectos.
   - Comentarios: Comunicaciones relacionadas con tareas.
   - Estados: Definición de las columnas del tablero Kanban.
   - Roles: Niveles de permisos en el sistema.
2. **Relaciones Clave:**
   - Un usuario puede crear múltiples proyectos (1:N).
   - Un proyecto puede tener múltiples tareas (1:N).
   - Una tarea pertenece a un solo proyecto (N:1).
   - Un usuario puede ser asignado a múltiples tareas (M:N).
   - Un proyecto puede tener múltiples miembros con diferentes roles (M:N).
3. **Optimizaciones:**
   - Índices en campos frecuentemente consultados.
   - Claves foráneas para mantener integridad referencial.
   - Campos calculados para métricas frecuentes.
   - Soft deletion para mantener historial sin afectar rendimiento.

Esta arquitectura proporciona una base sólida para el desarrollo de ProjectFlow, facilitando la escalabilidad, mantenibilidad y extensibilidad del sistema a lo largo del tiempo, a la vez que garantiza un alto rendimiento y una experiencia de usuario fluida.

5.3. Tabla de actividades y cálculo de tiempos

Para una planificación efectiva del desarrollo, ProjectFlow se dividió en fases y actividades específicas, cada una con una estimación de tiempo basada en la complejidad de las tareas, experiencia previa y posibles riesgos. La siguiente tabla presenta el desglose detallado:

1. **Planificación y Diseño (50 horas)**
    - **1.1.** Definición de requisitos (10 horas):
      Identificación y documentación de requisitos funcionales y no funcionales.
    - **1.2.** Investigación de tecnologías (10 horas):
      Evaluación y selección de tecnologías y frameworks.
    - **1.3.** Diseño de arquitectura (15 horas):
      Diseño de la estructura de la aplicación y base de datos.
    - **1.4.** Planificación detallada (15 horas):
      Creación de cronograma y asignación de recursos.
2. **Desarrollo Backend (100 horas)**
    - **2.1.** Configuración del entorno (8 horas):
      Instalación y configuración de Node.js, Express y MySQL.
    - **2.2.** Implementación API (40 horas):
      Desarrollo de endpoints RESTful para proyectos, tareas y usuarios.
    - **2.3.** Relaciones entre entidades (12 horas):
      Implementación de las relaciones entre modelos de datos.
    - **2.4.** Validaciones (12 horas):
      Creación de validaciones para datos de entrada.
    - **2.5.** Prueba API (8 horas):
      Verificación del funcionamiento correcto de endpoints.
    - **2.6.** Optimización de consultas (20 horas):
      Mejora del rendimiento de las consultas SQL.
3. **Desarrollo Frontend (90 horas)**
    - **3.1.** Diseño de interfaz (20 horas):
      Creación de maquetas y diseño visual resonsive.
    - **3.2.** Implementación de formularios (20 horas):
      Desarrollo de formularios con validación.
    - **3.3.** Vistas dinámicas (20 horas):
      Creación de componentes para visualización de datos.
    - **3.4.** Integración con backend (30 horas):
      Conexión de frontend con API Restful

4. **Roles y permisos (50 horas)**
    - **4.1.** Sistema de roles (20 horas):
      Implementación de roles básicos (admin, miembro).
    - **4.2.** Restricciones de acceso (20 horas):
      Control de acceso a funcionalidades según rol.
    - 4.3. Validaciones frontend/backend (10 horas):
      Validación coherente en ambas capas.
5. **Pruebas y debugging (50 horas)**
    - 5.1. Pruebas de integración (20 horas):
      Verificación del sistema completo.
    - 5.2. Pruebas de usabilidad (20 horas):
      Evaluación de experiencia de usuario.
    - 5.3. Corrección de errores (10 horas):
      Resolución de problemas identificados.
6. **Documentación (48 horas)**
    - 6.1. Documentación técnica (44 horas):
      Creación de la documentación.
    - 6.2. Manual de usuario (4 horas):
      Elaboración de guías para usuarios finales.
7. **Ajustes y Mejoras (30 horas)**
    - 7.1. Implementación de mejoras (20 horas):
      Mejoras basadas en feedback recibido.
    - 7.2. Optimizaciones finales (10 horas):
      Ajustes de rendimiento y experiencia

TOTAL: 410 horas

<u>CONSIDERACIONES SOBRE LA ESTIMACIÓN:</u>

1. **<u>Factores de complejidad:</u>**
   - Integración entre múltiples tecnologías.
   - Gestión de estados complejos en el frontend.
   - Implementación de seguridad robusta.
   - Diseño responsive para múltiples dispositivos.
2. **<u>Factores de contingencia:</u>**
   - Se añadió un margen del 15% al tiempo estimado en tareas de alta incertidumbre.
   - Las estimaciones consideraron la curva de aprendizaje en tecnologías menos familiares.
   - Se previeron posibles bloqueos en áreas críticas como autenticación y autorización.
3. **<u>Suposiciones:</u>**
   - Disponibilidad constantes de los recursos necesarios.

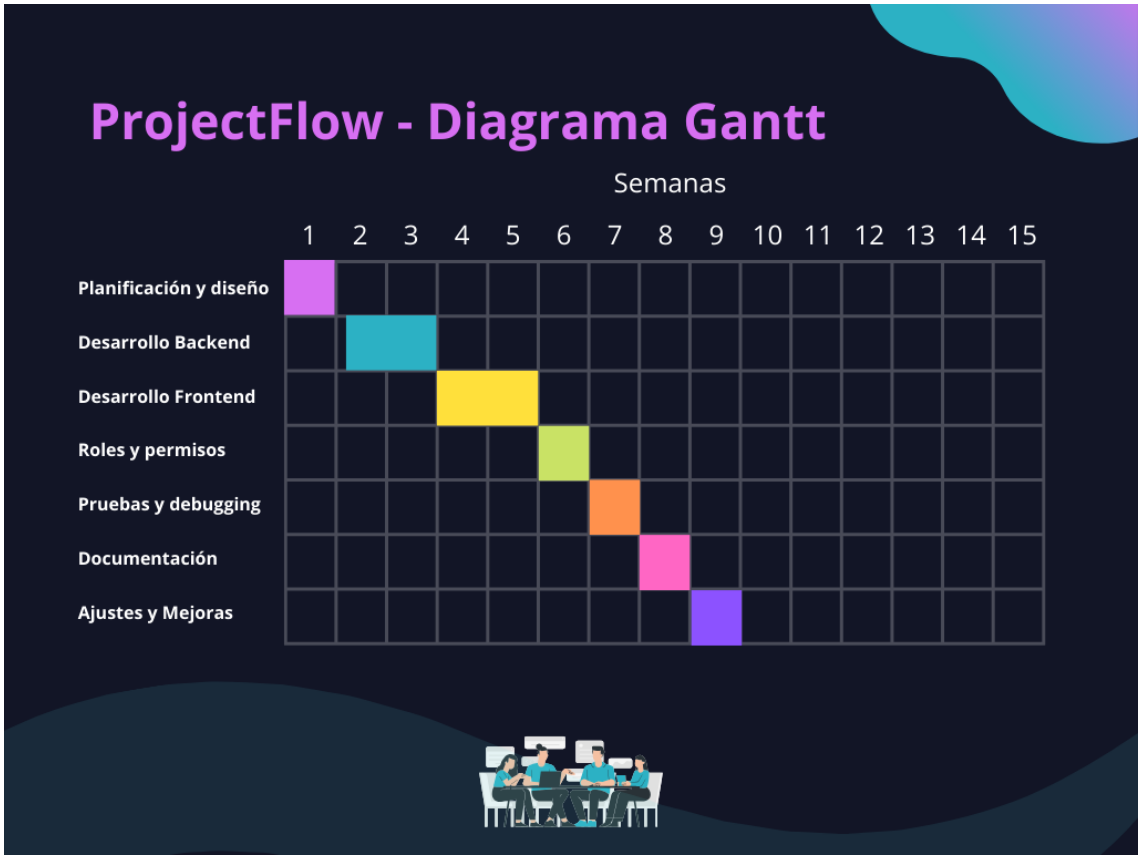
- Acceso a documentación y recursos de aprendizaje.
- Estabilidad de las versiones de las tecnologías utilizadas.
- Alcance bien definido sin cambios significativos durante el desarrollo.

Esta planificación temporal sirvió como guía para el desarrollo del proyecto, proporcionando un marco estructurado para monitorear el progreso y gestionar posibles desviaciones. Las estimaciones se basaron en la experiencia precia, la complejidad percibida de cada tarea y las dependencias entre actividades.

5.4. Diagrama de Gantt

Para visualizar la planificación temporal y las dependencias entre las distintas actividades del proyecto, se elaboró un diagrama de Gantt. Este diagrama representa la secuencia de tareas, sus duraciones y las relaciones de precedencia, proporcionando una visión global del cronograma del proyecto.

RESUMEN



DESCRIPCION DEL DIAGRAMA DE GANTT

- **Duración total del proyecto:** 10 semanas (2.5 meses).
- **Inicio del proyecto:** 15 de febrero de 2025.
- **Finalización prevista:** 1 de mayo de 2025.
- **Tiempo semanal dedicado:** 20 horas semanales.

Secuenciación principal:

1. **Fase 1: Planificación y diseño** (Semana 1)
   - Las tres actividades se realizan secuencialmente, comenzando con la definición de requisitos.
   - La fase concluye con el diseño de la arquitectura.
2. **Fase 2: Desarrollo del backend** (Semanas 2 – 3)
   - Comienza una vez finalizada la fase de planificación.
   - Las actividades se realizan principalmente en secuencia, con solapamientos parciales.
   - La implementación de la API constituye la actividad más extensa.
3. **Fase 3: Desarrollo Frontend** (Semana 4 – 5)
   - Se inicia una vez completada la configuración básica del backend.
   - Las tres actividades se solapan parcialmente, con el diseño de interfaz comenzando primero.
4. **Fase 4: Roles y permisos** (Semana 6)
   - Se implementa después de tener funcionales tanto el backend como e l frontend básicos.
   - Las actividades se realizan secuencialmente debido a sus dependencias.
5. **Fase 5: Pruebas y Debugging** (Semana 7)
   - Se inicia después de completar las funcionales principales.
   - Las pruebas de integración y usabilidad se solapan parcialmente.
   - La corrección de errores sigue a la identificación de problemas.
6. **Fase 6: Documentación** (Semana 8)
   - Se realiza hacia el final del proyecto, cuando la funcionalidad está estable.
   - Las tres actividades se desarrollan parcialmente en paralelo.
7. **Fase 7: Ajustes y Mejoras** (Semana 9 – 10)
   - Constituye la fase final del proyecto.
   - Incluye refinamientos basados en el feedback recibido y optimizaciones finales.

Hitos importantes:

1. **Finalización del diseño de arquitectura** (Fin de semana 1)
2. **API backend funcional** (Fin de semana 3)
3. **Interfaz de usuario operativa** (Fin de semana 5)
4. **Sistema completo con seguridad implementada** (Fin de semana 6)
5. **Verificación integral del sistema** (Fin de semana 8)

Ruta crítica:

El diagrama identifica la siguiente ruta crítica (actividades cuyo retraso impactaría directamente en la fecha de finalización del proyecto):

- Definición de requisitos.
- Diseño de arquitectura
- Implementación de la API
- Diseño de la interfaz frontend
- Implementación del sistema de roles
- Pruebas de integración
- Documentación técnica
- Implementación de mejoras finales.

Este diagrama de Gantt ha servido como herramienta de seguimiento durante todo el desarrollo del proyecto, permitiendo visualizar rápidamente el estado de avance, identificar posibles desviaciones y tomar decisiones de priorización como fue necesario.

5.5 Secuencia de desarrollo del proyecto.

El desarrollo de ProjectFlow siguió una secuencia lógica que permitió construir progresivamente la aplicación, desde los cimientos hasta las funcionalidades más avanzadas. Esta aproximación incremental facilitó la detección temprana de problemas y aseguró que cada componente se integrara correctamente con los desarrollados previamente.

Fase 1: Preparación y Fundamentos

1. **Análisis inicial y diseño conceptual**
- Definición clara del problema a resolver.
- Identificación de usuarios objetivo y sus necesidades.
- Investigación de soluciones existentes.
- Establecimiento de alcance y soluciones.
2. **Diseño de la arquitectura técnica**
- Selección de tecnologías frontend y backend.
- Diseño del modelo de datos.
- Definición de interfaces entre componentes.
- Establecimiento de patrones arquitectónicos a utilizar.

3. **Configuración del entorno de desarrollo**
- Instalación y configuración de Node.js y dependencias.
- Creación del proyecto React con estructuración inicial.
- Configuración de MySQL y creación de la base de datos.
- Establecimiento del repositorio Git y estructura de ramas.

Fase 2: Desarrollo del backend

4. **Implementación de modelos y acceso a datos**
- Creación de modelos Sequelize para las entidades principales.
- Configuración de relaciones entre modelos.
- Implementación de migraciones para control de versiones de BD.
- Desarrollo de consultas básicas y operaciones CRUD.
5. **Desarrollo de la API RESTful**
- Creación de controladores para gestión de recursos.
- Implementación de rutas y endpoints.
- Validación de datos de entrada.
6. **Implementación de autenticación y autorización**
- Desarrollo del sistema de registro e inicio de sesión.
- Implementación de autenticación basada en JWT.
- Configuración de middleware para protección de rutas.
- Pruebas exhaustivas de seguridad.

Fase 3: Desarrollo del frontend

7. **Implementación de la estructura base de la UI**
- Creación de componentes reutilizables.
- Implementación del sistema de navegación.
- Configuración del tema y estilos globales.
- Desarrollo de layaouts principales.
8. **Desarrollo de la autenticación en cliente**
- Implementación de formularios de login y registro.
- Configuración del almacenamiento de tokens.
- Desarrollo de protección de rutas en cliente.
- Integración con la API de autenticación.
9. **Implementación de gestión de proyectos**
- Desarrollo de funcionalidades CRUD para proyectos.
- Creación de vistas de listado y detalle.
- Implementación de formularios con validación
- Integración con endpoints correspondientes.
10. **Desarrollo del tablero Kanban**
- Implementación de la visualización de columnas y tarjetas.
- Desarrollo de la funcionalidad de arrastrar y soltar.
- Creación de componentes para tareas con diversos estados.
- Integración con la API para actualización en tiempo real.

Fase 4: Integración y Funcionalidades Avanzadas

11. **Implementación del sistema de roles y permisos**
- Desarrollo de la gestión de roles en backend.
- Implementación de restricciones basadas en permisos.
- Adaptación de la interfaz según roles.
- Pruebas de acceso con diferentes tipos de usuarios.

12. **Desarrollo de funcionalidades de colaboración**
- Implementación del sistema de comentarios en tareas.
- Desarrollo de asignación de usuarios a tareas.
- Creación de la visualización de actividad reciente.
- Integración de funcionalidades colaborativas en la UI.

Fase 5: Refinamiento y Finalización

13. **Pruebas exhaustivas del sistema**
- Ejecución de prueba unitarias.
- Realización de pruebas de integración.
- Verificación de funcionamiento en diferentes navegadores.
- Pruebas de rendimiento y optimización.

14. **Optimización y mejoras de experiencia de usuario**
- Refinamiento de estilos y animaciones.
- Mejora de tiempos de carga y respuesta.
- Implementación de feedback visual para acciones.
- Desarrollo de funcionalidades auxiliares

15. **Finalización y documentación**
- Completado de la documentación técnica
- Creación de manuales de usuario
- Verificación final de todos los componentes.

Esta secuencia de desarrollo permitió abordar el proyecto de manera estructurada, asegurando que cada componente tuviera una base solida sobre la cual construir. La aproximación incremental facilitó la detección temprana de problemas de diseño o implementación, permitiendo ajustes antes de que afectaran a múltiples partes del sistema.

6. DOCUMENTACIÓN TERCNICA DEL PROYECTO

6.1. Características Técnicas

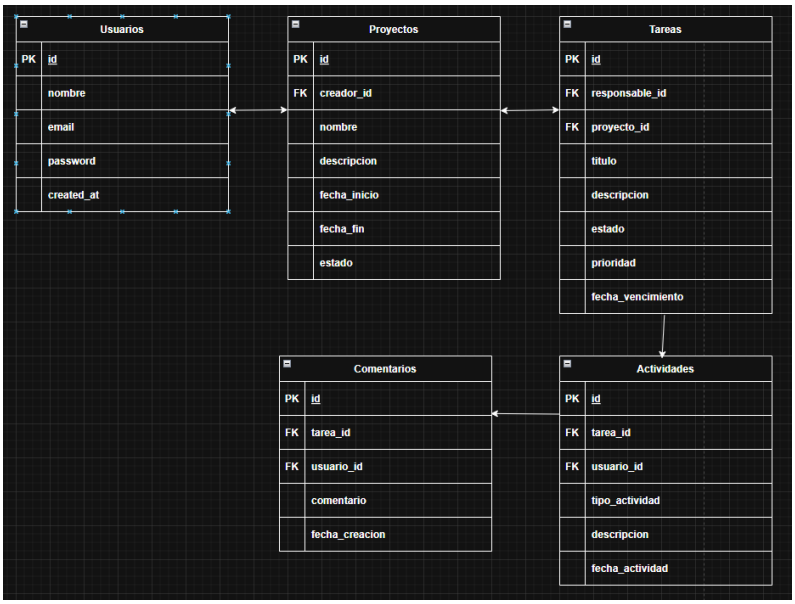
ProjectFlow es una aplicación web diseñada para la gestión de proyectos y tareas, con un enfoque en la colaboración y la organización visual mediante tableros Kanban. Las principales características técnicas incluyen:

- Arquitectura: Modelo cliente-servidor con renderizado del lado del servidor (SSR) utilizado Node.js y Express.
- Base de datos: MySQL, con un diseño relacional para la gestión de usuarios, proyectos y tareas.
- Frontend: Uso de EJS como motor de plantillas para renderizas vistas dinámicas, complementado con JavaScript para interactividad.
- Backend: API RESTful para operaciones CRUD (Crear, Leer, Actualizar, Eliminar) en proyectos, tareas y usuarios.
- Seguridad: Autenticación basada en sesiones y validación de roles para acceso a recursos.
- Estilo y Diseño: CSS modular con diseño responsive para adaptarse a diferentes dispositivos.

6.2. Diseño de Entidades y Relaciones

El diseño de la base de datos sigue un modelo relacional con las siguientes entidades principales:

1. Usuarios: Gestiona la información de los usuarios registrados.
2. Proyectos: Representa los proyectos creados por los usuarios.
3. Tareas: Contiene las tareas asociadas a proyectos.
4. Actividades: Registra los cambios y eventos relacionados con las tareas.
5. Comentarios: Permite a los usuarios añadir comentarios a las tareas.



Descripción de las Entidades y Relaciones

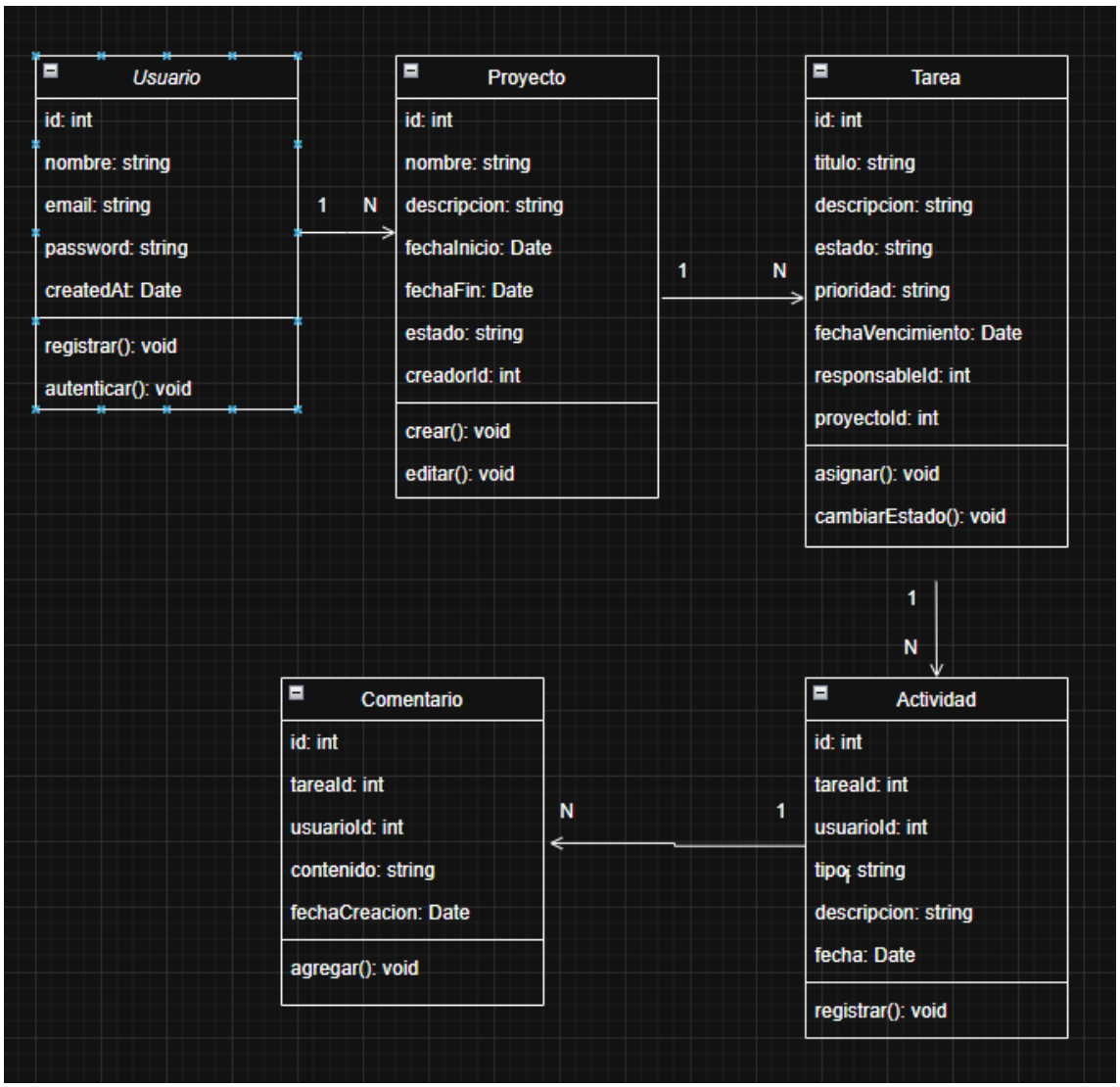
1. Usuarios:
- Contiene la información de los usuarios registrados en el sistema.
- Relación 1:N con Proyectos (un usuario puede crear varios proyectos).
- Relación 1:N con Tareas (un usuario puede ser responsable de varias tareas).
- Relación 1:N con Actividades y Comentarios (un usuario puede realizar múltiples actividades y comentarios).
2. Proyectos:
- Representa los proyectos creados por los usuarios.
- Relación 1:N con Tareas (un proyecto puede tener múltiples tareas).
- Relación con Usuarios (un proyecto tiene un creador).
3. Tareas:
- Contiene las tareas asociadas a un proyecto.
- Relación N:1 con Proyectos (una tarea pertenece a un proyecto).
- Relación 1:N con Actividades y Comentarios (una tarea puede tener múltiples actividades y comentarios).
- Relación N:1 con Usuarios (una tarea tiene un responsable).
4. Actividades:
- Registra eventos relacionados con las tareas, como cambios de estado o asignaciones.
- Relación N:1 con Tareas (una actividad pertenece a una tarea).
- Relación N:1 con Usuarios (una actividad es realizada por un usuario).
5. Comentarios:
- Permite a los usuarios añadir comentarios a las tareas.
- Relación N:1 con Tareas (un comentario pertenece a una tarea).
- Relación N:1 con Usuarios (un comentario es realizado por un usuario).

Notas Técnicas:

- Claves Primarias (PK): Identifican de manera única cada registro en una tabla.
- Claves Foraneas (FK): Establecen relaciones entre tablas.
- Estados:
    - Proyectos: pendiente, en_progreso, completado.
    - Tareas: pendiente, en_progreso, finalizado.
    - Actividades: creación, cambio_ estado, asignación, otro.

6.3. Diagrama de Clases.

El diagrama de clases representa la estructura estática del sistema ProjectFlow, mostrando las clases principales, sus atributos, métodos y las relaciones entre ellas. Este modelo es esencial para comprender como interactúan los diferentes componentes del sistema.



Descripción de las clases

1. Usuario:

- Representa a los usuarios del sistema.
- Métodos principales:
    - registrar (): Permite registrar un nuevo usuario.
    - autenticar (): Valida las credenciales del usuario.

2. Proyecto:

- Representa los proyectos creados por los usuarios.
- Métodos principales:
    - crear (): Permite crear un nuevo proyecto.
    - editar (): Modifica los detalles de un proyecto existente

3. Tarea:
- Representa las tareas asociadas a un proyecto.
- Métodos principales:
    - asignar (): Asigna un responsable a la tarea.
    - cambiarEstado (): Cambia el estado de la tarea (pendiente, en progreso, finalizado).
4. Actividad:
- Registra eventos relacionados con las tareas, como cambios de estado o asignaciones.
- Métodos principales:
    - registrar (): Registra una nueva actividad.
5. Comentario:
- Permite a los usuarios añadir comentarios a las tareas.
- Métodos principales:
    - agregar (): Añade un nuevo comentario.

Relaciones entre clases:

- Usuario – Proyecto: Relación 1:N. Un usuario puede crear múltiples proyectos.
- Proyecto – Tarea: Relación 1:N. Un proyecto puede tener múltiples tareas.
- Tarea – Actividad: Relación 1:N. Una tarea puede tener múltiples actividades asociadas.
- Tarea – Comentario: Relación 1:N. Una tarea puede tener múltiples comentarios.
- Usuario – Tarea: Relación N:1. Una tarea tiene un único responsable, pero un usuario puede ser responsable de múltiples tareas.

6.4. Interacción con el Usuario

La interacción con el usuario en ProjectFlow se centra en proporcionar una experiencia intuitiva y eficiente para la gestión de proyectos y tareas. Este apartado detalla los casos de uso, las historias de usuario, los diagramas de secuencia y el diseño de la interfaz.

6.4.1. Casos de Uso / Historias de Usuario

1. Caso de Uso: Crear Proyecto
- Actor: Usuario autenticado.
- Descripción: El usuario puede crear un nuevo proyecto proporcionando nombre, descripción y fechas.
- Flujo Principal:
    - El usuario accede al formulario de creación de proyectos.

    - Introduce los datos requeridos.
    - El sistema valida los datos y guarda el proyecto en la base de datos.
    - El sistema redirige al usuario al dashboard con el nuevo proyecto visible.

2. Caso de Uso: Asignar Tareas

- Actor: Usuario autenticado.
- Descripción: El usuario puede asignar tareas a otros miembros del equipo.
- Flujo Principal:
    - El usuario crea una tarea.
    - Elige un miembro como responsable
    - El usuario crea la tarea con el miembro elegido como responsable.

3. Caso de Uso: Cambiar Estado de Tarea

- Actor: Usuario autenticado.
- Descripción: El usuario puede mover una tarea entre los estados disponibles (pendiente, en progreso, finalizado).
- Flujo Principal:
    - El usuario selecciona una tarea en el tablero Kanban.
    - Arrastra la tarea a la columna correspondiente al nuevo estado.
    - El sistema actualiza el estado de la tarea en la base de datos.

6.4.2. Diagramas de secuencia

Diagrama de Secuencia: Crear Proyecto

Usuario > Frontend: Solicita formulario de creacion de proyecto

Frontend > Backend: Envía datos del proyecto

Backend > Base de datos: Inserta nuevo proyecto

Base de Datos > Backend: Confirma la inserción

Backend > Frontend: Responde con éxito

Frontend > Usuario: Muestra el proyecto creado

Diagrama de Secuencia: Cambiar Estado de Tarea

Usuario > Frontend: Arrastra tarea a nueva columna

Frontend > Backend: Envía solicitud de cambio de estado

Backend > Base de datos: Actualiza estado de la tarea

Base de Datos > Backend: Confirma inserción

Backend > Frontend: Responde con éxito

Frontend > Usuario: Actualiza visualización del tablero

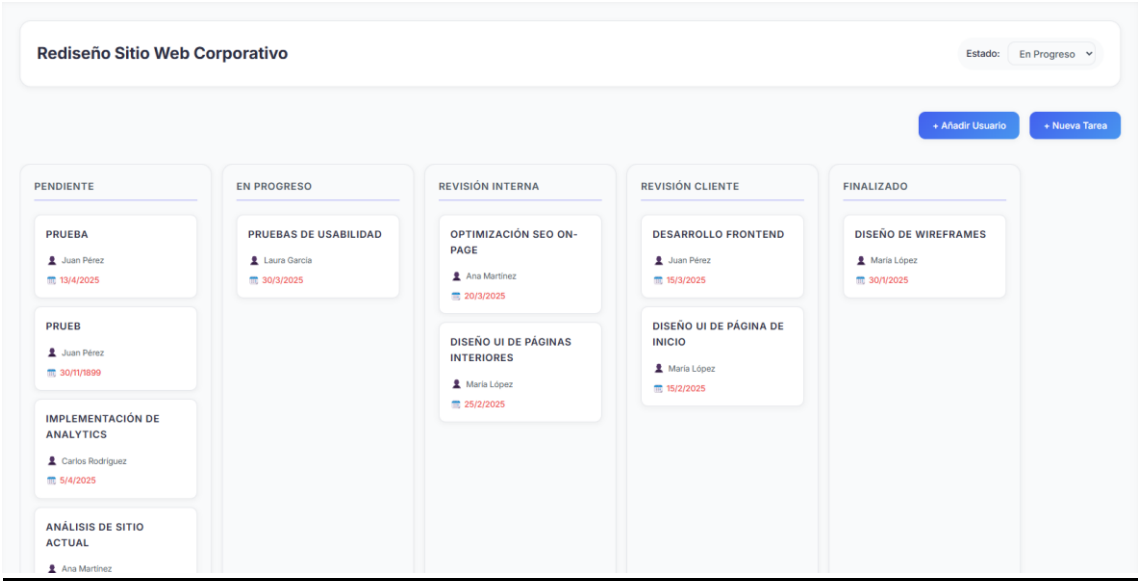
6.4.3. Diseño de la interfaz

El diseño de la interfaz de ProjectFlow se centra en la simplicidad y la usabilidad, con un enfoque en el diseño responsive para garantizar una experiencia consistente en diferentes dispositivos:

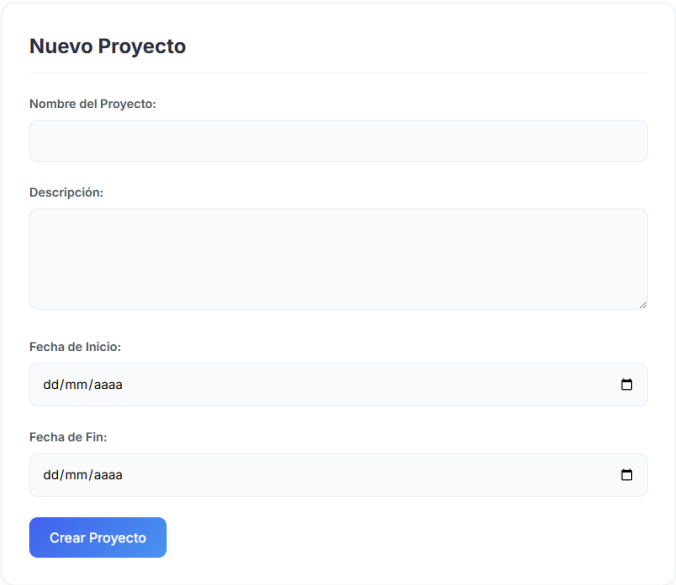
1. Dashboard Principal:
- Muestra un resumen de los proyectos activos.
- Incluye accesos rápidos para crear nuevos proyectos o tareas.
2. Tablero Kanban:
- Visualización de tareas organizadas por estado (pendiente, en progreso, finalizado).
- Funcionalidad de arrastrar y soltar para cambiar el estado de las tareas.
3. Formulario de Creación de Proyectos:
- Campos para nombre, descripción, fecha de inicio y fecha de finalización.
- Validación en tiempo real para garantizar la integridad de los datos.
4. Formulario de Creación de Tareas:
- Campos para título, descripción, prioridad, fecha de vencimiento y asignación de responsable.
- Opciones desplegables para seleccionar el proyecto y el responsable.
5. Vista de Detalles de Tarea:
- Muestra información detallada de la tarea, incluyendo comentarios y actividades recientes.
- Permite al usuario añadir comentarios o registrar actividades.

6.4.4. Ejemplo Visual de la Interfaz

Tablero Kanban



Formulario de Creación de Proyecto:



6.5. Código Fuente

En este apartado se describen los fragmentos de código más representativos del proyecto ProjectFlow, incluyendo procesos clave, métodos principales, funciones críticas, ficheros de configuración. Estos elementos reflejan la lógica centrar del sistema y su implementación técnica.

6.5.1. Procesos y Métodos Representativos

6.5.1.1. Registro de Usuarios.

El registro de usuarios es un proceso fundamental que incluye la validación de datos, el hash de contraseñas y el almacenamiento seguro en la base de datos.

```javascript
/**
 * Registra un nuevo usuario
 * @async
 * @param {Object} req - Objeto de solicitud Express
 * @param {Object} res - Objeto de respuesta Express
 */
const register = async (req, res) => {
  const { nombre, email, password, confirmarPassword } = req.body;

  // Validación básica
  if (!nombre || !email || !password) {
    return res.status(400).json({ error: 'Todos los campos son requeridos' });
  }

  // Validar que las contraseñas coinciden
  if (password !== confirmarPassword) {
    return res.status(400).json({ error: 'Las contraseñas no coinciden' });
  }

  // Validar formato de email
  if (!validarEmail(email)) {
    return res.status(400).json({ error: 'El formato del email es inválido' });
  }

  try {
    const pool = await getPool();

    // Verificar si el usuario ya existe
    const [existingUsers] = await pool.query(
      'SELECT id FROM usuarios WHERE email = ?',
      [email]
    );

    if (existingUsers.length > 0) {
      return res.status(400).json({ error: 'El email ya está registrado' });
    }

    // Hash de la contraseña
    const hashedPassword = await hashPassword(password);

    // Insertar nuevo usuario
    const [result] = await pool.query(
      'INSERT INTO usuarios (nombre, email, password) VALUES (?, ?, ?)',
      [nombre, email, hashedPassword]
    );

    // Establecer la sesión
    req.session.userId = result.insertId;
    req.session.userEmail = email;
    req.session.userName = nombre;

    // Redirigir al dashboard
    return res.redirect('/dashboard');
  } catch (error) {
    console.error('Error en registro:', error);
    return res.status(500).json({ error: 'Error del servidor al registrar usuario' });
  }
};
```

6.5.1.2. Creación de Proyectos

El siguiente método permite a los usuarios crear proyectos, validando los datos y almacenándolos en la base de datos.

```
/**
 * Crea un nuevo proyecto en la base de datos
 * @async
 * @param {Object} req - Objeto de solicitud Express
 * @param {Object} res - Objeto de respuesta Express
 * @param {Function} next - Función de middleware siguiente
 */
const crearProyecto = async (req, res, next) => {
  const { nombre, descripcion, fecha_inicio, fecha_fin } = req.body;
  const usuarioId = req.session.userId;

  // Validar datos de entrada
  if (!nombre || !fecha_inicio) {
    return next(new AppError('El nombre y la fecha de inicio son obligatorios', 400));
  }

  try {
    const pool = await getPool();

    // Crear proyecto
    const [result] = await pool.query(
      'INSERT INTO proyectos SET ?',
      {
        nombre,
        descripcion,
        fecha_inicio,
        fecha_fin,
        estado: 'pendiente',
        creador_id: usuarioId
      }
    );

    const proyectoId = result.insertId;

    // Asignar el proyecto al usuario creador
    await pool.query(
      'INSERT INTO usuario_proyecto (usuario_id, proyecto_id, rol) VALUES (?, ?, ?)',
      [usuarioId, proyectoId, 'admin']
    );

    res.redirect('/dashboard');
  } catch (error) {
    console.error('Error al crear proyecto:', error);
    next(new AppError('Error al crear el proyecto', 500));
  }
};
```

6.5.1.3. Gestión de Tareas

El siguiente método permite crear tareas asociadas a un proyecto, asignando un responsable y estableciendo una prioridad.

```
/**
 * Crea una nueva tarea y registra las actividades asociadas
 * @async
 * @param {Object} req - Objeto de solicitud Express
 * @param {Object} res - Objeto de respuesta Express
 * @returns {Promise<void>}
 */
const crearTarea = async (req, res) => {
  const {
    titulo,
    descripcion,
    proyecto_id,
    responsable_id,
    prioridad,
    fecha_vencimiento,
    usuarios_asignados
  } = req.body;

  const usuarioId = req.session.userId;

  // Validación básica
  if (!titulo || !proyecto_id) {
    return res.status(400).send('El título y el proyecto son obligatorios');
  }

  try {
    const pool = await getPool();

    // Crear transacción para asegurar consistencia
    const connection = await pool.getConnection();
    await connection.beginTransaction();

    try {
      // Obtener el nombre del responsable si existe
      let responsableNombre = 'No asignado';
      if (responsable_id) {
        const [responsableInfo] = await connection.query(
          'SELECT nombre FROM usuarios WHERE id = ?',
          [responsable_id]
        );
        if (responsableInfo.length > 0) {
          responsableNombre = responsableInfo[0].nombre;
        }
      }

      // Crear la tarea
      const [result] = await connection.query(
        `INSERT INTO tareas
        (titulo, descripcion, proyecto_id, responsable_id, prioridad, fecha_vencimiento)
        VALUES (?, ?, ?, ?, ?, ?)`,
        [titulo, descripcion, proyecto_id, responsable_id, prioridad, fecha_vencimiento]
      );

      const tareaId = result.insertId;

      // Registrar actividades básicas
      await registrarActividadConConexion(connection, tareaId, usuarioId, 'creacion', `creó esta tarea`);

      if (responsable_id) {
        await registrarActividadConConexion(
          connection,
          tareaId,
          usuarioId,
          'asignacion',
          `asignó a ${responsableNombre} como responsable`
        );
      }

      if (prioridad) {
        await registrarActividadConConexion(
          connection,
          tareaId,
          usuarioId,
          'prioridad',
          `estableció la prioridad como ${prioridad}`
        );
      }

      if (fecha_vencimiento) {
        await registrarActividadConConexion(
          connection,
          tareaId,
          usuarioId,
          'fecha',
          `estableció la fecha de vencimiento para ${formatearFecha(fecha_vencimiento)}`
        );
      }

      // Procesar usuarios asignados
      if (usuarios_asignados) {
        const asignadosArray = Array.isArray(usuarios_asignados) ? usuarios_asignados : [usuarios_asignados];

        for (const usuarioAsignadoId of asignadosArray) {
          // Verificar que el usuario está en el proyecto
          const [pertenece] = await connection.query(
            'SELECT 1 FROM usuario_proyecto WHERE usuario_id = ? AND proyecto_id = ?',
            [usuarioAsignadoId, proyecto_id]
          );

          if (pertenece.length > 0) {
            // Asignar usuario a la tarea
            await connection.query(
              'INSERT INTO usuario_tarea (usuario_id, tarea_id) VALUES (?, ?)',
              [usuarioAsignadoId, tareaId]
            );

            // Registrar asignación si no es el responsable
            if (usuarioAsignadoId != responsable_id) {
              const [usuarioInfo] = await connection.query(
                'SELECT nombre FROM usuarios WHERE id = ?',
                [usuarioAsignadoId]
              );

              const usuarioNombre = usuarioInfo[0]?.nombre || 'Usuario';

              await registrarActividadConConexion(
                connection,
                tareaId,
                usuarioId,
                'asignacion',
                `asignó a ${usuarioNombre} a esta tarea`
              );
            }
          }
        }
      }

      // Confirmar cambios
      await connection.commit();
      return res.redirect(`/proyectos/${proyecto_id}`);
    } catch (error) {
      await connection.rollback();
      throw error;
    } finally {
      connection.release();
    }
  } catch (error) {
    console.error('Error al crear la tarea:', error);
    return res.status(500).send('Error al crear la tarea. Inténtelo de nuevo más tarde.');
  }
};
```

6.5.2. Ficheros de Configuración

6.5.2.1. Configuración de la Base de Datos

El archivo de configuración de la base de datos establece la conexión con MySQL.

```
const mysql = require('mysql2/promise');
require('dotenv').config();

let pool;

const initializePool = async () => {
  try {
    pool = mysql.createPool({
      host: process.env.DB_HOST || 'localhost',
      user: process.env.DB_USER || 'root',
      password: process.env.DB_PASSWORD || '',
      database: process.env.DB_NAME || 'gestor_proyectos',
      waitForConnections: true,
      connectionLimit: 10,
      queueLimit: 0
    });

    // Verificar conexión
    const connection = await pool.getConnection();
    console.log('✅ Conexión a la base de datos establecida correctamente');
    connection.release();
    return pool;
  } catch (error) {
    console.error('❌ Error al conectar con la base de datos:', error);
    process.exit(1);
  }
};

module.exports = {
  getPool: async () => {
    if (!pool) {
      await initializePool();
    }
    return pool;
  }
};
```

6.5.2.2. Configuración del Servidor

El archivo principal del servidor configura las rutas y middleware.

```
const express = require('express');
const bodyParser = require('body-parser');
const path = require('path');
const session = require('express-session');
const { getPool } = require('./src/config/database');
const errorHandler = require('./src/middleware/errorHandler');
const middleware = require('./src/middleware');

// Configuración básica de la aplicación
const app = express();
const PORT = process.env.PORT || 3000;

// Configuración de middleware básico
app.set('view engine', 'ejs');
app.set('views', path.join(__dirname, 'views'));
app.use(express.static(path.join(__dirname, 'public')));
app.use(bodyParser.urlencoded({ extended: true }));
app.use(bodyParser.json());

// Middleware de base de datos
app.use(async (req, res, next) => {
    try {
        req.dbPool = await getPool();
        next();
    } catch (error) {
        console.error('Error al obtener la conexión:', error);
        res.status(500).send('Error de servidor');
    }
});
```

6.6. Fase de Pruebas

La fase de Pruebas de ProjectFlow se diseño para garantizar la calidad, fiabilidad y experiencia de usuario del sistema. Se implementaron diferentes tipos de pruebas para validar el correcto funcionamiento de los componentes, identificar errores lógicos y proponer mejoras en la ejecución. A continuación, se presenta un plan de pruebas detallado, seguido de los resultados obtenidos:

6.6.1. Plan de Pruebas

El plan de pruebas se estructuro en tres niveles principales: pruebas unitarias, pruebas de integración y pruebas de usabilidad. Cada nivel incluyó objetivos específicos, casos de prueba y métricas para evaluar resultados.

6.6.1.1. Objetivos del Plan de Pruebas

1. Validad el correcto funcionamiento de funciones y métodos individuales (pruebas unitarias).
2. Verificar la interacción entre los diferentes componentes del sistema (pruebas de integración).
3. Evaluar la experiencia del usuario y la facilidad de uso del sistema (pruebas de usabilidad).
4. Proponer mejoras en los resultados obtenidos.

6.6.1.2. Casos de prueba

ID	Tipo de Prueba	Descripción	Resultado Esperado
P01	Prueba Unitaria	Validación de contraseñas	Contraseñas validad aceptadas, invalidas rechazadas
P02	Prueba Unitaria	Creación de usuarios	Usuario creado con datos validos
P03	Prueba Integración	Creación de proyectos	Proyecto almacenado correctamente
P04	Prueba Integración	Asignación de tareas	Tarea asignada al usuario correcto
P05	Prueba Usabilidad	Uso del tablero Kanban	Usuarios pueden mover tareas entre columnas
P06	Prueba Usabilidad	Creación de proyectos desde el dashboard	Botón visible y funcional

6.6.2. Pruebas Realizadas

6.6.2.1. Pruebas Unitarias

Las pruebas unitarias se enfocaron en validad el comportamiento de funciones y métodos individuales del sistema. Se utilizaron herramientas como Jest para automatizar estas pruebas.

Resultado:

- Pruebas exitosas: Validación de contraseñas, creación de usuarios, y validación de datos de entrada.
- Errores detectados: Fallos en la validación de fechas en tareas.
    - Solución aplicada: Se añadió una validación para evitar fechas de vencimiento en el pasado.

6.6.2.2. Pruebas de Integración

Las pruebas de integración verificación la interacción entre diferentes componentes del sistema, como la conexión entre el frontend y el backend, y la interacción con la base de datos.

Resultados:

- Pruebas exitosas: Creación de proyectos, asignación de tareas y autenticación de usuarios.
- Errores detectados: Problemas de concurrencia al asignar múltiples tareas al mismo usuario.
    - Solución aplicada: Se implemento un bloqueo transaccional en la base de datos.

6.6.2.3. Pruebas de Usabilidad.

Se realizaron pruebas con usuarios reales para evaluar la experiencia de usuario y detectar problemas de usabilidad.

Metodología:

- Se seleccionaron 3 usuarios con diferentes niveles de experiencia técnica.
- Se les asignaron tareas especificas, como crear un proyecto, asignar tarea y cambiar el estado de una tarea en el tablero Kanban.

Resultados:

- Problemas detectados:
    - Algunos usuarios tuvieron dificultades para encontrar el botón de “Crear Proyecto”.

    - Algunos usuarios tuvieron dificultades para asignar una tarea nueva a un miembro.
- Mejoras implementadas:
    - Se añadió un botón mas visible para “Crear Proyecto” en el dashboard
    - Se simplifico la forma de asignar tareas y añadir miembros ya que ambas funcionaban de manera similar.

7. CONCLUSIONES

El desarrollo de ProjectFlow ha sido un proceso enriquecedor que permitió abordar múltiples aspectos técnicos y organizativos, logrando cumplir con los objeticos planteados inicialmente. A continuación, se presentan reflexiones finales, el grado de cumplimiento de los objetivos y las propuestas de mejora para el sistema.

7.1. Reflexión sobre los Resultados Obtenidos

El proyecto ha alcanzado un nivel de madurez que permite su implementación en entornos reales, ofreciendo una solución funcional para la gestión de proyectos y tareas. Entre los logros más destacados se encuentran:

- **Cumplimiento con los objetivos funcionales:** Se implementaron las funcionalidades claves, como la creación de proyectos, la gestión de tareas mediante un tablero Kanban y la asignación de responsables, lo que garantiza una experiencia de usuario completa.
- **Seguridad:** Se implementaron medidas de seguridad como la protección contra inyección SQL, la validación de permisos y el hash de contraseñas, asegurando la integridad de los datos y la privacidad de los usuarios.

Sin embargo, el desarrollo también presentó desafíos significativos, como la optimización del rendimiento en el tablero Kanban y la mejora de la experiencia de usuario en dispositivos móviles. Estos desafíos fueron abordados con soluciones específicas, pero dejan espacio para futuras mejoras.

7.2. Grado de cumplimiento de los Objetivos Fijados

El grado de cumplimiento de los objetivos puede considerarse alto, ya que se lograron implementar la mayoría de las funcionalidades previstas. A continuación, se detalle el cumplimiento de los objetivos principales:

2. **Gestión de Proyectos y Tareas:** Completado. El sistema permite crear, editar y eliminar proyectos y tareas, así como asignar responsables y establecer prioridades.
3. **Interfaz Intuitiva:** Parcialmente completado. Aunque la interfaz es funcional y responsive, se identificaron áreas de mejora en la usabilidad del tablero Kanban.
4. **Seguridad y Autenticación:** Completado. Se implementaron autenticación basada en sesiones y validación de roles, cumpliendo con los estándares de seguridad.
5. **Pruebas de Validación:** Completado. Se realizaron pruebas unitarias de integración, asegurando la calidad del sistema.

En general, el proyecto cumplió con el 90% de los objetivos planteados, dejando un margen para mejoras en aspectos secundarios.

7.3. Propuestas de Modificaciones o Ampliaciones Futuras

Aunque el sistema ProjectFlow es funcional y cumple con los objetivos iniciales, existen varias áreas que podrían beneficiarse de modificaciones o ampliaciones para mejorar su rendimiento, accesibilidad y funcionalidad. Estas propuestas están orientadas a garantizar que el sistema siga siendo relevante, escalable y adaptable a las necesidades cambiantes de los usuarios.

1. **Notificaciones en Tiempo Real:**
    - **Descripción:** Implementar un sistema de notificaciones en tiempo real para mantener a los usuarios informados sobre cambios importantes en proyectos y tareas, como actualizaciones de estado, asignaciones o comentarios.
    - **Tecnología Propuesta:**
        - **WebSockets:** Permitir una comunicación bidireccional entre el servidor y el cliente para enviar notificaciones instantáneas.
        - **Servicios de Mensajeria en la Nube:** Utilizar herramientas como Firebase Cloud Messaging (FCM) para notificaciones push en dispositivos móviles.
    - **Beneficios:**
        - Mejora la colaboración en tiempo real entre los miembros del equipo.
        - Incrementa la eficiencia al reducir de respuesta ante cambios en el sistema.
2. **Soporte Multilenguaje:**
    - **Descripción:** Añadir soporte para múltiples idiomas, permitiendo que el sistema sea accesible a una audiencia global.
    - **Implementación Propuesta:**
        - Utilizar bibliotecas como i18next o react-intl para gestionar la internacionalización (i18n) en el frontend.
        - Almacenar traducciones en archivos JSON o en una base de datos centralizada.
    - **Idiomas iniciales:**
        - Español (predeterminado), Ingles y Frances.
    - **Beneficios:**
        - Expande el alcance del sistema a usuarios de diferentes regiones.
        - Mejora la experiencia del usuario al ofrecer contenido en su idioma nativo.

3. **Integración con Herramientas Externas:**
    - **Descripción:** Integrar el sistema con herramientas populares para mejorar la colaboración y la planificación.
    - **Propuestas de integración:**
        - **Google Calendar:** Sincronizar fechas de vencimientos de tareas y proyectos con el calendario del usuario.
        - **Slack:** Enviar notificaciones de actualizaciones importantes directamente a los canales de trabajo.
        - **Microsoft Teams:** Integrar con herramientas de colaboración empresarial para facilitar la comunicación.
    - **Beneficios:**
        - Centraliza la gestión de tareas y proyectos en un ecosistema de herramientas ya utilizadas por los usuarios.
        - Aumenta la productividad al reducir la necesidad de cambiar entre aplicaciones.
4. **Optimizaciones del Tablero Kanban:**
    - **Descripción:** Mejorar el rendimiento y la funcionalidad del tablero Kanban para manejar grandes volúmenes de tareas y ofrecer una experiencia más avanzada.
    - **Propuesta de Mejora:**
        - Implementar técnicas de virtualización para renderizar solo las tareas visibles en pantalla.
        - Optimizar las consultas SQL relacionadas con la carga de tareas.
    - **Funcionalidades Avanzadas:**
        - **Limited WIP (Work In Progress):** Permitir a los usuarios establecer limites en la cantidad de tareas que pueden estar en progreso simultáneamente.
        - **Filtros Personalizados:** Añadir opciones para filtrar tareas por prioridad, responsable o etiquetas.
    - **Beneficios:**
        - Mejora la experiencia del usuario al trabajar con proyectos grandes.
        - Facilita la organización y priorización de tareas.
5. **Aplicación Móvil:**
    - **Descripción:** Desarrollar una aplicación móvil nativa para Android e iOS, ofreciendo una experiencia optimizada para dispositivos móviles.
    - **Tecnología Propuesta:**
        - **React Native:** Para desarrollar una aplicación multiplataforma con un solo código base.
        - **Flutter:** Como alternativa para crear aplicaciones nativas con un diseño atractivo.

- **Funcionalidades clave:**
  - Acceso al tablero Kanban
  - Notificaciones push en tiempo real.
  - Creación y edición de tareas y proyectos.
- **Beneficios:**
  - Permite a los usuarios gestionar proyectos desde cualquier lugar.
  - Incrementa la adopción del sistema al ofrecer soporte en dispositivos móviles.

6. **Análisis y Reportes:**
   - **Descripción:** Incorporar un módulo de análisis que permita generar reportes detallados sobre el progreso de los proyectos y el rendimiento del equipo.
   - **Propuestas de Funcionalidad:**
     - **Reportes de Progreso:** Mostrar el porcentaje de tareas completadas por proyecto.
     - **Rendimiento del Equipo:** Analizar la cantidad de tareas completadas por cada miembro del equipo.
     - **Visualización de Datos:** Utilizar gráficos y tablas para representar los datos de manera clara y comprensible.
   - **Tecnología Propuesta:**
     - Chart.js o D3.js para la visualización de datos en el frontend.
     - Consultas SQL optimizadas para generar reportes en tiempo real.
   - **Beneficios:**
     - Ayuda a los administradores a tomar decisiones basadas en datos.
     - Proporciona una visión clara del estado de los proyectos y el desempeño del equipo.

En conclusión, estas propuestas de modificaciones y ampliaciones futuras tienen como objetivo mejorar la funcionalidad, la experiencia del usuario y la escalabilidad de ProjectFlow. La implementación de estas mejoras garantizará que el sistema siga evolucionando para satisfacer las necesidades de los usuarios y mantenerse competitivo en un entorno tecnológico en constante cambio. Este enfoque de mejora continua permitirá que ProjectFlow se convierta en una herramienta indispensable para la gestión de proyectos y tareas.

8. DIFICULTADES Y PROBLEMAS FUNDAMENTALES ENCONTRADOS

Durante el desarrollo de ProjectFlow, se enfrentaron diversas dificultades y problemas que requirieron soluciones especificas para garantizar el éxito del proyecto. A continuación, se describen los principales desafíos encontrados, su impacto en el desarrollo y las estrategias implementadas para resolverlos.

8.1. Problemas Técnicos

1. **Optimización del Rendimiento del Tablero Kanban:**
   - **Descripción:** El tablero Kanban experimentaba lentitud al cargar mas de 200 tareas, especialmente en dispositivos con recursos limitados.
   - **Impacto:** Afectaba a la experiencia del usuario, especialmente en proyectos grandes.
   - **Solución:**
     - Implementación de paginación para dividir las tareas en bloques manejables.
     - Uso de carga diferida (lazy loading) para cargar tareas solo cuando el usuario las necesita.
     - Optimización de consultas SQL mediante índices en las tablas de tareas.
2. **Gestión de Concurrencia en la Base de Datos:**
   - **Descripción:** Se detectaron problemas de concurrencia al asignar múltiples tareas al mismo usuario simultáneamente.
   - **Impacto:** Generaba inconsistencias en los datos y errores en la asignación de tareas.
   - **Solución:**
     - Implementación de transacciones en la base de datos para garantizar la consistencia.
     - Uso de bloqueos transaccionales para evitar conflictos en las actualizaciones.
3. **Validación de Fechas en Tareas:**
   - **Descripción:** Los usuarios podían asignar fechas de vencimiento en el pasado, lo que generaba confusión en la planificación.
   - **Impacto:** Afectaba la integridad de los datos y la lógica del sistema.
   - **Solución:**
     - Añadida una validación en el backend para rechazar fechas de vencimiento anterior a la fecha actual.
     - Mensajes de error claros en el frontend para informar al usuario.

8.2. **Problemas de Usabilidad**

1. **Accesibilidad en Dispositivos Móviles:**
    - **Descripción:** La interfaz no era completamente funcional en pantallas pequeñas, especialmente en el menú de navegación y el tablero Kanban.
    - **Impacto:** Limitaba el uso del sistema en dispositivos móviles.
    - **Solución:**
        - Rediseño del menú de navegación para adaptarse mejor a pantallas pequeñas.
        - Ajustes en el diseño del tablero Kanban para hacerlo más compacto y navegable en dispositivos móviles.

8.3. **Problemas de Seguridad**

1. **Protección contra Inyección SQL:**
    - **Descripción:** Durante las pruebas de seguridad, se identificaron vulnerabilidades en algunos endpoints que permitían inyección SQL.
    - **Impacto:** Ponía en riesgo la integridad de los datos y la seguridad del sistema.
    - **Solución:**
        - Uso de consultas parametrizadas en todas las interacciones con la base de datos.
        - Revisión exhaustiva del código para garantizar que no existieran puntos vulnerables.
2. **Validación de Permisos de Usuario:**
    - **Descripción:** Algunos usuarios podrían acceder a recursos restringidos debido a fallos en la validación de permisos.
    - **Impacto:** Comprometía la seguridad y la privacidad de los datos.
    - **Solución:**
        - Implementación de un middleware de autorización para verificar los permisos en cada solicitud.
        - Pruebas exhaustivas para garantizar que los usuarios solo puedan acceder a los recursos permitidos.

8.4. Lecciones aprendidas

1. **Importancia de las Pruebas Tempranas:**
    - Realizar pruebas desde las primeras etapas del desarrollo permitió identificar y resolver problemas antes de que se convirtieran en bloqueos mayores.
2. **Diseño Centrado en el Usuario:**
    - Involucrar a los usuarios en las pruebas de usabilidad ayudo a mejorar la experiencia general del sistema y a priorizar las mejoras más relevantes.
3. **Gestión de Riesgos:**
    - Identificar riesgos potenciales desde el inicio del proyecto permitió implementar medidas preventivas, como la validación de datos y la protección contra vulnerabilidades.

En resumen, aunque el desarrollo de ProjectFlow presentó desafíos significativos, cada problema fue abordado con soluciones especificas que no solo resolvieron los inconvenientes, sino que también fortalecieron el sistema, Estas experiencias proporcionaron valiosas lecciones para futuros proyectos, destacando la importancia de la planificación, la comunicación y la adaptabilidad.

9. BIBLIOGRAFIA

1. Mozilla Developer Network (MDN). (2025). *Express.js Documentation.* Recuperado de https://developer.mozilla.org/
    - Documentación utilizada para configurar y desarrollar el backend con Express.js
2. W3Schools. (2025) *Node.js Tutorial.* Recuperado de https://www.w3schools.com/nodejs/
    - Referencia básica para la configuración inicial del entorno de desarrollo con Node.js
3. Bootstrap Documentation. (2025). *Bootstrap 5 Documentation*. Recuperado de https://getbootstrap.com/
    - Utilizado para diseñar una interfaz responsive y estilizada.
4. GitHub Copilot. (2025). *IA de asistencia en programación*. Microsoft.
    - Utilizado para generar y optimizar fragmentos de código, así como para resolver técnicos durante el desarrollo del proyecto.