PONTIFICIA UNIVERSIDAD CATÓLICA DEL ECUADOR

FACULTAD DE HÁBITAT, INFRAESTRUCTURA Y CREATIVIDAD

CARRERA DE: INGENIERÍA EN SISTEMAS DE INFORMACIÓN



Trabajo de Titulación

Tema:

EVALUACIÓN DEL IMPACTO DE UNA PLATAFORMA WEB DE GESTIÓN DE PROYECTOS EN LA EFICIENCIA OPERATIVA Y EL CUMPLIMIENTO DE TAREAS DE ESTUDIANTES UNIVERSITARIOS

AUTOR:

Patricio Nicolás Oña Ledesma

TUTOR:

Ing. Francisco Rodríguez Clavijo

QUITO DM, JUNIO DE 2026

DEDICATORIA

Dedico esta tesis, en primer lugar, a Dios por darme salud, fortaleza y la oportunidad de llegar hasta este momento muy importante en mi vida. A mis padres, por su apoyo incondicional, por cada esfuerzo realizado, por haber confiado en mí y por ayudarme en momentos difíciles. Todo lo que he logrado hasta hoy es resultado de su sacrificio, consejos y amor. A mi hermana, por su cariño, su compañía y por estar presente en este proceso de una u otra manera, por cada una de sus palabras de ánimo y momentos compartidos también han sido parte importante de este camino. A mi familia, por acompañarme a lo largo de esta etapa, por motivarme a seguir adelante y por recordarme siempre la importancia de no rendirme. A mis docentes, quienes con sus enseñanzas y guía aportaron de manera significativa a mi formación académica y personal. Finalmente, me dedico este trabajo por la constancia, paciencia y esfuerzo invertido para convertir una idea en un proyecto real, superando dificultades y aprendiendo en cada etapa del proceso.

AGRADECIMIENTO

Quiero agradecer a todas las personas que me acompañaron durante esta etapa y que, de distintas maneras, hicieron posible la culminación de este trabajo de titulación. Agradezco a Dios por permitirme llegar a este momento, por darme la calma para afrontar los distintos retos del proceso y no detenerme ante los obstáculos. A mis padres, por sostenerme con su ejemplo, paciencia y apoyo permanente a lo largo de mi formación. A mi hermana y familia, por estar presentes, escucharme, animarme todos los días de mayor presión y brindarme tranquilidad para continuar. A mis docentes, por compartir sus conocimientos y aportar con orientaciones valiosas que fortalecieron mi formación académica y el desarrollo de este proyecto. También agradezco a la Pontificia Universidad Católica del Ecuador por haberme brindado el espacio de aprendizaje y crecimiento profesional que hoy me permite cerrar esta etapa con satisfacción. Finalmente, agradezco a todas las personas que, de una u otra manera, contribuyeron con su tiempo, apoyo o palabras de aliento durante esta etapa.

RESUMEN

Este trabajo tiene como objetivo diseñar e implementar NovaTask, una aplicación web orientada a la organización académica de estudiantes universitarios de la Pontificia Universidad Católica del Ecuador (PUCE). Este desarrollo surge ante la necesidad de contar con una herramienta digital que permita centralizar la gestión de tareas, proyectos, materias y eventos académicos, ya que, en muchos casos, estas actividades se administran de forma dispersa mediantes agendas o notas. Para ello se desarrolló una plataforma web utilizando React y Tailwind CSS en el frontend, Node.js y Express en el backend, y PostgreSQL con Prisma para la persistencia de datos. Entre sus principales funcionalidades se incluyen la gestión de proyectos y tareas, el calendario académico, plantillas y analíticas de avances. Como resultado, se obtuvo un MVP funcional que establece una base sólida para futuras etapas de validación y mejora del sistema.

Palabras claves: Organización académica. Aplicación web, estudiantes, gestión de tareas

ABSTRACT

This work aims to design and implement NovaTask, a web application for the academic organization of students at the Pontifical Catholic University of Ecuador (PUCE). This development arose from the need for a digital tool to centralize the management of tasks, projects, courses, and academic events, as these activities are typically managed in a scattered manner using calendars or notes. To this end, a web platform was developed using React and Tailwind CSS on the frontend, Node.js and Express on the backend, and PostgreSQL with Prisma for data persistence. Its main functionalities include project and task management, an academic calendar, templates, and progress analysis. The result is a functional MVP that establishes a solid foundation for future validations and system improvements.

Keywords: Academic organization, web application, students, task management

CONTENIDO

DEDICATORIA ..... **II**
AGRADECIMIENTO ..... **III**
RESUMEN ..... **IV**
ABSTRACT ..... **V**
CAPÍTULO I: INTRODUCCIÓN **1**
1.1 Planteamiento del problema **1**
1.2 Justificación **2**
1.3 Objetivos **3**
1.3.1 Objetivo general **3**
1.3.2 Objetivos específicos **3**
1.4 Alcance y delimitación del trabajo **3**
1.5 Metodología de trabajo adoptada **4**
1.6 Estado actual del proyecto **4**
CAPÍTULO II: FUNDAMENTACIÓN TEÓRICA **7**
2.1 Aplicaciones web orientadas a la organización académica **7**
2.2 Arquitectura cliente-servidor y API REST **7**
2.3 Diseño centrado en el usuario, escaneo visual y enfoque responsive **8**
2.4 Interfaces tipo SaaS, tema oscuro y consistencia visual **8**
2.5 React y la construcción de interfaces basadas en componentes **9**
2.6 Vite como herramienta de desarrollo y construcción **9**
2.7 Tailwind CSS como estrategia de estilos utilitarios **10**
2.8 Navegación, estado global y comportamiento del cliente **10**
2.9 Bibliotecas para interacción visual: animación, arrastre y analíticas **11**
2.10 Node.js y Express en el desarrollo del backend **11**
2.11 PostgreSQL y Prisma para persistencia y modelado relacional **12**
2.12 JWT y bcrypt en autenticación propia **12**
2.13 Buenas prácticas aplicadas al proyecto **13**
CAPÍTULO III: ANÁLISIS Y DISEÑO DE LA APLICACIÓN **15**
3.1 Análisis del problema y del usuario objetivo **15**
3.2 Necesidades identificadas **15**

3.3 Requerimientos funcionales....16
3.4 Requerimientos no funcionales....18
3.5 Navegación y flujo general del sistema....19
3.6 Arquitectura general del sistema....19
3.7 Diseño de la base de datos....20
3.8 Prototipado y diseño de interfaz....22
3.9 Flujo funcional del MVP....22
3.10 Diseño de despliegue y continuidad operativa....23
CAPÍTULO IV: DESARROLLO E IMPLEMENTACIÓN DE NOVATASK....24
4.1 Preparación del entorno de desarrollo....24
4.2 Evolución del proyecto desde el prototipo inicial....25
4.3 Implementación del frontend....26
4.3.1 Pantalla de acceso e identificación del usuario....26
4.3.2 Dashboard como centro operativo....27
4.3.3 Gestión de proyectos....28
4.3.4 Kanban, vista de lista, filtros, etiquetas y subtareas....29
4.3.5 Calendario académico....31
4.3.6 Analíticas y lectura del avance....33
4.3.7 Plantillas para acelerar nuevos proyectos....34
4.3.8 Perfil, periodos, materias y agenda....35
4.3.9 Navegación lateral, topbar y menú de cuenta....36
4.3.10 Responsive, tema claro y tema oscuro....37
4.4 Implementación del backend....38
4.4.1 Estructura del servidor....38
4.4.2 Autenticación y control de acceso....38
4.4.3 Servicios por módulo....39
4.4.4 Validaciones, seguridad y manejo de errores....39
4.5 Implementación de la base de datos....40
4.6 Registro de actividad y consistencia operativa....41
4.7 Experiencia de usuario, feedback y calidad visual....41
4.8 Diseño experimental propuesto....42

4.9 Pruebas realizadas y validación funcional 43
4.10 Despliegue y publicación del sistema 45
4.11 Dificultades encontradas y decisiones tomadas 45
4.12 Estado actual y trabajo pendiente 46
CAPÍTULO V: CONCLUSIONES Y RECOMENDACIONES 47
5.1 Conclusiones 47
5.2 Recomendaciones 47
ANEXOS 48
ANEXO A. Capturas web sugeridas para completar la tesis 48
ANEXO B. Resumen de entidades y su papel dentro del sistema 49
ANEXO C. Variables de entorno relevantes 50
ANEXO D. Variables e indicadores para el grupo experimental y el grupo control 51
ANEXO E. Checklist sugerido para la siguiente fase 51
ANEXO F. Repositorio del sistema en GitHub 51
ANEXO G. Sistema desplegado en Vercel 52
BIBLIOGRAFÍAS ..... ¡Error! Marcador no definido.

ÍNDICE DE FIGURAS

Figura 1. Captura de la pantalla de login 26
Figura 2. Espacio para captura de la pantalla final de login y registro 27
Figura 3. Captura del dashboard principal 28
Figura 4. Captura del módulo de proyectos 29
Figura 5. Captura del tablero Kanban 30
Figura 6. Captura de la vista de lista 31
Figura 7. Captura del calendario mensual 32
Figura 8. Captura del calendario semanal 33
Figura 9. Captura de la página de analíticas 34
Figura 10. Captura del módulo de plantillas 35
Figura 11. Captura del perfil de usuario 36
Figura 12. Captura del menú de cuenta y acciones del perfil 37
Figura 13. Captura del tema claro 38

ÍNDICE DE TABLAS

Tabla 1. Estado actual del proyecto NovaTask 5
Tabla 2. Marco tecnológico adoptado en NovaTask ..... **¡Error! Marcador no definido.**
Tabla 3. Requerimientos funcionales principales 16
Tabla 4. Requerimientos no funcionales 18
Tabla 5. Organización técnica del backend 20
Tabla 6. Entidades principales del modelo de datos 21
Tabla 7. Estructura general del repositorio NovaTask 24
Tabla 8. Resumen de módulos y endpoints del backend 40
Tabla 9. Distribución de la muestra simulada 42
Tabla 10. Plan general de seguimiento por cinco semanas 42
Tabla 11. Casos de prueba funcionales ejecutados 43
Tabla 12. Registro sugerido de capturas web 48
Tabla 13. Entidades funcionales de NovaTask 49
Tabla 14. Variables de entorno para operación del proyecto 50
Tabla 15. Matriz básica de seguimiento comparativo 51

CAPÍTULO I: INTRODUCCIÓN

Durante los últimos semestres, la carga académica de un estudiante universitario se ha vuelto cada vez más compleja y fragmentada. Ya no se trata solo de asistir a clases y entregar tareas puntuales. En una misma semana pueden coincidir avances de tesis, proyectos grupales, exposiciones, evaluaciones parciales, lecciones y trámites administrativos. El problema muchas veces no radica en la falta de voluntad, sino en la dificultad para tener una visión clara, actualizada y confiable de todo lo que se debe hacer.

En la práctica, muchos estudiantes terminan repitiendo su información en diferentes lugares: una agenda física, notas en el celular, hojas de cálculo, archivos de textos y aplicaciones variadas. Esta dispersión genera duplicidad de información, pérdida de texto y una sensación constante de desorden. Cuando una fecha cambia o una tarea depende de otra, reconstruir el panorama completo se vuelve una tarea adicional que consume tiempo y energía.

Es en este contexto que surge NovaTask: una plataforma web pensada específicamente para estudiantes universitarios. La propuesta no busca ser una herramienta genérica más, sino una solución práctica que permita centralizar proyectos, tareas, materias, eventos y plazos en un solo lugar, adaptada a la realidad académica de la PUCE.

1.1 Planteamiento del problema

El problema central que motiva esta tesis es el siguiente: los estudiantes universitarios, particularmente aquellos en semestres avanzados o que desarrollan trabajos de titulación, carecen de una herramienta integrada, intuitiva y adaptada a sus necesidades que les permita organizar su trabajo académico de forma clara, visual y persistente.

Aunque existen aplicaciones comerciales consolidadas, muchas están orientadas a entornos empresariales, presentan interfaces complejas, requieren pago para funcionalidades completas o exigen una curva de aprendizaje elevada. Por otro lado, las soluciones improvisadas que usan los estudiante como agendas, notas sueltas o hojas de cálculo solo resuelven parte del problema y suelen generar inconsistencias cuando la carga académica aumenta.

Como consecuencia, es común observar olvido de plazos, dificultad para priorizar, acumulación de pendientes y un aumento en los niveles de estrés. Esta fragmentación afecta no solo la productividad, sino también la percepción de control sobre el propio trabajo académico con lo que surge la siguiente pregunta principal:

¿Cómo diseñar e implementar una plataforma web que integre de forma coherente proyectos, tareas, calendario y seguimiento académico en una sola experiencia?

Preguntas secundarias:

¿Qué funcionalidades resultan realmente útiles y prácticas para un estudiante universitario que maneja entregas, materias y actividades paralelas?

¿De qué manera una arquitectura desacoplada con frontend, backend y base de datos puede sostener el crecimiento futuro del sistema manteniendo su usabilidad?

¿Qué decisiones de experiencia de usuario favorecen una mayor adopción de la herramienta tanto en escritorio como en dispositivos móviles?

1.2 Justificación

La realización de este trabajo de titulación responde a una necesidad real y cotidiana que enfrentan los estudiantes universitarios. Gestionar múltiples asignaturas, proyectos grupales, entregas y actividades extracurriculares genera una carga importante de organización que, en la mayoría de los casos, se resuelve de forma improvisada y dispersa. NovaTask surge como una herramienta pensada específicamente para este contexto, con el objetivo de centralizar toda la información académica en un solo lugar, ofreciendo una experiencia visual, intuitiva y adaptada a la realidad universitaria.

Desde el punto de vista académico, este proyecto permite integrar y aplicar de manera práctica gran parte de los conocimientos adquiridos durante la carrera: análisis de requerimientos, diseño de interfaces, desarrollo de frontend y backend, modelado de la base de datos, autenticación segura y despliegue de aplicaciones web. El trabajo busca dejar un producto funcional que pueda servir como base para futuras mejoras y validaciones con usuarios reales.

Además, NovaTask tiene potencial para contribuir al campo de las herramientas de apoyo académico, especialmente en universidades donde las soluciones comerciales no siempre se adaptan a las necesidades locales. Al combinar desarrollo técnico con una propuesta de evaluación de impacto, este proyecto no solo entrega una plataforma, sino también una base sólida para analizar cómo las herramientas digitales pueden mejorar la eficiencia y el bienestar de los estudiantes.

1.3 Objetivos

1.3.1 Objetivo general

Diseñar, implementar y proponer un esquema de evaluación para NovaTask, una plataforma web orientada a la organización académica de estudiantes universitario, integrando gestión de proyectos, tareas, calendario y analíticas dentro de una arquitectura moderna.

1.3.2 Objetivos específicos

- Analizar las necesidades reales de organización académica de los estudiantes y traducirlas en requerimientos funcionales.
- Diseñar una arquitectura desacoplada que facilite el mantenimiento y la escalabilidad futura.
- Implementar autenticación propia mediante correo y contraseña, utilizando JWT y cifrado de contraseñas con bcrypt.
- Desarrollar los módulos principales del MVP: dashboard, proyectos, kanban, calendario, analíticas, plantillas y perfil académico.
- Diseñar un modelo de datos con PostgreSQL y Prisma, considerando relaciones, restricciones, índices y trazabilidad de actividad.
- Incorporar buenas prácticas de experiencia de usuario, incluyendo diseño responsive y tema claro/oscuro.
- Preparar el sistema para su publicación en la web y proponer un diseño experimental para evaluar su impacto.

1.4 Alcance y delimitación del trabajo

El alcance del proyecto abarca el desarrollo completo de un MVP funcional. Esto incluye el diseño e implementación del frontend en React con Vite y Tailwind CSS, el backend en Node.js y Express, la base de datos en PostgreSQL con Prisma, la autenticación propia con JWT y bcrypt y la puesta en producción inicial mediante GitHub, Render y Vercel. También forman parte del alcance los módulos académicos que dan identidad al sistema: periodos, materias, eventos, plantillas y seguimiento visual del avance.

No obstante, existen límites claros. NovaTask no se integra todavía con sistemas institucionales de la universidad, no sincroniza calendarios externos, no incorpora trabajo

colaborativo multiusuario en tiempo real y no dispone de una aplicación móvil nativa. Estas delimitaciones permiten mantener un MVP coherente y técnicamente viable dentro del alcance de una tesis de grado.

1.5 Metodología de trabajo adoptada

Para el desarrollo de NovaTask se adoptó una lógica de trabajo iterativa, cercana al prototipado evolutivo. La razón fue sencilla: desde el inicio no se buscó producir un documento teórico desconectado del producto, sino avanzar por capas, validando continuamente cada módulo. En la práctica, esto significó comenzar con una idea general del flujo académico, traducirla en prototipos visuales, convertir esas pantallas en componentes React, conectar luego la lógica de negocio en el backend y, finalmente, estabilizar la persistencia con PostgreSQL y Prisma.

Esta metodología resultó adecuada porque el proyecto cambió mientras se construía. Algunas decisiones visuales iniciales, como privilegiar únicamente el modo oscuro, se revisaron más adelante al detectarse la necesidad de un modo claro mejor trabajado. Del mismo modo, ciertas pantallas crecieron respecto de la idea original: el perfil terminó incorporando gestión de periodos, materias y eventos; el calendario integró tanto tareas como eventos; y el módulo de proyectos pasó de un simple tablero a un espacio con filtros, etiquetas, subtareas y vista de lista.

Otro aspecto importante fue la resolución incremental de problemas reales. Durante el proceso aparecieron dificultades con la conexión local a PostgreSQL, bloqueos del cliente de Prisma en Windows y varios detalles visuales que hacían ver inestable la interfaz. Lejos de detener el proyecto, esos incidentes se integraron al trabajo como parte del aprendizaje técnico. Se habilitaron alternativas transitorias, como un modo demostración para seguir probando la interfaz, y luego se retornó a la persistencia definitiva. Esa flexibilidad fue útil porque permitió avanzar sin romper el ritmo del proyecto.

1.6 Estado actual del proyecto

Al momento de redactar estos capítulos, NovaTask se encuentra en un estado de MVP funcional y publicado. El sistema ya superó la etapa de prototipo básico y cuenta con un backend funcional, base de datos relacional, autenticación segura, módulos principales conectados y despliegue público en internet. Esto permite que la siguiente fase es centre principalmente en la

validación con usuarios reales, el afinamiento de detalles, la ampliación de pruebas y la consolidación de la parte teórica.

El repositorio presenta una organización clara tanto en el frontend como el backend, incluye documentación técnica, scripts para despliegue, datos de prueba iniciales y archivos de configuración para distintos entornos. Además, existe una versión del backend publicada en Render y del frontend en Vercel, junto con configuraciones adicionales. Este conjunto de entregables demuestra que el trabajo no se limitó al entorno local y ya fue preparado para funcionar en un entorno web real.

En términos de producto, el avance más importante radica en que ya existe una experiencia de usuario coherente de principio a fin. El usuario puede ingresar desde una pantalla de login clara, acceder a un dashboard con métricas relevantes, gestionar sus proyectos, mover tareas, revisar su agenda, consultar analíticas, aplicar plantillas y ajustar su perfil. Esa continuidad funcional es lo que convierte a NovaTask en una base sólida y creíble para sostener el resto de la tesis.

Tabla 1. *Estado actual del proyecto NovaTask*

Componente	Situación actual	Observaciones
Frontend	Implementado y publicado	Incluye autenticación, dashboard, proyectos, calendario, analíticas, plantillas y perfil.
Backend	Implementado y publicado	API REST con Express, validaciones, JWT, bcrypt, manejo de errores y servicios por módulo.
Base de datos	Implementada	PostgreSQL con Prisma, migraciones, seed y relaciones completas.
Autenticación	Operativa	Registro, login, refresh, sesión persistente y protección de rutas privadas.

Experiencia de usuario	Operativa	Validaciones visuales, estados vacíos, loaders, responsive y tema claro y oscuro.
Despliegue	Operativo	Frontend en Vercel, backend en Render y repositorio versionado en GitHub.

Nota. Elaboración propia

CAPÍTULO II: FUNDAMENTACIÓN TEÓRICA

2.1 Aplicaciones web orientadas a la organización académica

Una aplicación web es, fundamentalmente, un sistema al que se puede accedere desde cualquier navegador sin necesidad de instalaciones complicadas. En el ámbito universitario esto representa una ventaja importante, ya que permite a los estudiantes consultar y actualizar su información desde diferentes dispositivos y lugares. No dependen de un solo equipo ni de configuraciones específicas, lo que favorece un uso más continuo y práctico.

Cuando una aplicación web se diseña para apoyar la organización personal, su valor va más allá de simplemente almacenar datos. Debe ayudar a estructurarlos, priorizarlos y convertirlos en acciones concretas. No es lo mismo tener una lista de pendientes que contar con un sistema capaz de agrupar tareas por proyecto, vincularlas con fechas, mostrar progreso y ofrecer una agenda integrada. NovaTask se concibe precisamente como un apoyo real al trabajo académico del estudiante, más que como un simple repositorio de notas.

Esta aproximación coincide con la tendencia actual de las interfaces web modernas, que buscan responder a necesidades específicas del usuario mediante flujos sencillos, estados visibles y persistencia confiable de la información

2.2 Arquitectura cliente-servidor y API REST

La arquitectura cliente-servidor sigue siendo una de las formas más prácticas para desarrollar este tipo de aplicaciones. El cliente, en este caso el frontend, se encarga de la interacción con el usuario, la navegación y la presentación visual de los datos. El servidor, por su parte, procesa las solicitudes, valida información, aplica reglas de negocio y se comunica con la base de datos. Esta separación facilita el mantenimiento, ya que se pueden hacer cambios en la interfaz sin afectar la lógica interna, o mejorar el backend sin tener que rehacer todas las pantallas.

Para conectar ambas capas se utilizó una API REST. Esto significa que el sistema expone rutas claras para operaciones como registrar usuarios, iniciar sesión, crear proyectos, actualizar tareas, cargar el calendario o consultar analíticas. El enfoque CRUD resulta natural para este tipo de sistemas, porque la mayoría de las interacciones del usuario pueden expresarse mediante estas operaciones básicas.

La ventaja de esta decisión no es solo técnica. También ordena el proyecto a nivel conceptual. Cada módulo tiene un contrato claro: la interfaz solicita, el backend valida y responde, y la base de datos persiste. De esta forma, el crecimiento futuro del sistema puede darse por capas, sin convertir el código en una estructura difícil de sostener.

2.3 Diseño centrado en el usuario, escaneo visual y enfoque responsive

Un error frecuente en proyectos académicos es concentrarse demasiado en la tecnología y dejar la experiencia del usuario como algo secundario. En NovaTask se buscó evitar eso desde el principio. El objetivo no era crear muchas pantallas, sino construir un recorrido claro: entrar, entender rápidamente el estado del trabajo, localizar un proyecto, modificar tareas y revisar fechas sin fricción. Esto obligó a pensar en jerarquías visuales, textos breves, tarjetas legibles, estados vacíos comprensibles y acciones visibles.

Otro aspecto importante fue la adaptación responsive. La herramienta debía funcionar bien tanto en computadoras como en computadoras y teléfonos. Tailwind CSS, a través de su sistema de utilidades y puntos de quiebre, ofrece una base práctica para ajustar la composición según el tamaño de pantalla sin duplicar estilos completos. Esa decisión fue valiosa porque permitió mantener una sola base visual y, aun así, reorganizar paneles, columnas y tarjetas cuando el espacio disponible cambia.

El enfoque responsive no se asumió solo como una exigencia de presentación. En realidad, es una forma de respetar el contexto del usuario. Un estudiante puede revisar pendientes desde el laboratorio, desde su computador personal o desde el teléfono mientras se moviliza. Si la interfaz falla en uno de esos escenarios, la utilidad general del sistema disminuye, por más correcto que sea el backend.

2.4 Interfaces tipo SaaS, tema oscuro y consistencia visual

NovaTask adoptó una estética inspirada en aplicaciones SaaS modernas: composición limpia, superficies bien diferenciadas, información resumida y navegación intuitiva. Esto no significa copiar interfaces comerciales, sino de tomar principios que ya demostraron ser efectivos: composición modular, superficies bien diferenciadas, información resumida en tarjetas, navegación lateral, acciones primarias visibles y micro transiciones suaves. En este tipo de

entornos, el usuario no necesita jugar con la interfaz cada vez que entra; reconoce patrones y se concentra en su tarea.

El proyecto comenzó priorizando el modo oscuro como base visual principal. Esa decisión respondía a una intención estética clara, pero también a la lectura prolongada de paneles con mucha densidad de información. Más adelante se añadió un modo claro porque el sistema también debía responder a preferencias de uso distintas. Lograr que ambos temas se vieran equilibrados requirió ajustes cuidadosos en contrastes, sombras y superficies.

2.5 React y la construcción de interfaces basadas en componentes

React se eligió como base del frontend porque facilita la construcción por componentes y estados bien definidos. Esta forma de trabajo resulta conveniente cuando la interfaz incluye módulos que se repiten, como tarjetas, modales, badges, pestañas, loaders o barras de navegación. En lugar de repetir código en cada pantalla, se pueden crear piezas independientes que luego se ensamblan según sea necesario.

En NovaTask esta decisión fue evidente desde las primeras pantallas. La barra lateral, las tarjetas métricas, los modales, los componentes de estado y las alertas visuales fueron diseñados como piezas independientes. Eso no solo organiza mejor el código. También vuelve más rápido el mantenimiento, porque un ajuste en un componente puede reflejarse en varios lugares sin reescribir la interfaz completa.

Desde una perspectiva académica, React también aporta claridad metodológica: obliga a pensar la interfaz como una composición de responsabilidades acotadas. Esa idea es muy valiosa para un trabajo de tesis porque facilita explicar qué hace cada parte, cómo se comunica con las demás y dónde se toman ciertas decisiones de diseño o de estado.

2.6 Vite como herramienta de desarrollo y construcción

Vite se eligió porque simplifica el entorno de desarrollo del frontend. En proyectos pequeños o medianos, la experiencia de trabajo mejora considerablemente cuando el servidor se inicia rápidamente, la recarga de cambios es casi inmediata y no se requiere una configuración complicada ni muchos ajustes manuales. Esa facilidad fue una de las razones principales para optar por Vite en lugar de otras herramientas más pesadas.

Su utilidad no se limitó al tiempo de inicio. También facilitó el proceso de publicación del frontend, ya que genera un paquete optimizado y listo para desplegar en plataformas como Vercel con una configuración bastante sencilla (Vite, s. f.). En el contexto del proyecto, este tipo de decisiones resultan especialmente valiosas porque reduce la brecha entre el desarrollo local y la versión final, que se pone en funcionamiento, dos etapas que suelen sentirse separadas pero que en realidad forman parte del mismo ciclo.

2.7 Tailwind CSS como estrategia de estilos utilitarios

Tailwind CSS se utilizó como la principal herramienta para definir el estilo visual del proyecto. En lugar de crear hojas de estilo largas y específicas para cada pantalla, esta herramienta permite construir todo el diseño como composición, espaciado, color, tipografía y responsive mediante clases pequeñas y claras. Esto facilitó iterar con mayor rapidez, sobre todo cuando el proyecto empezó a pulir detalles de distribución, contraste y adaptación a diferentes tamaños de pantalla.

Otra ventaja importante fue la consistencia. Como NovaTask trabaja con una paleta de colores definida y un conjunto limitado de patrones visuales como paneles, tarjetas, chips, botones, inputs, badges, rejillas, Tailwind ayudó a mantener un diseño uniforme sin que el código CSS se volviera desordenado o difícil de mantener. Además, su integración con Vite fue muy sencilla, lo que simplificó considerablemente el entorno de desarrollo.

2.8 Navegación, estado global y comportamiento del cliente

En una aplicación con varias pantallas, la navegación no puede resolverse simplemente con enlaces sueltos. Es necesario contar con una estructura clara que distinga entre rutas públicas y privadas, además de manejar estados compartidos y las preferencias del usuario. React Router se encargó de la navegación entre las diferentes vistas, mientras que Zustand se utilizó para almacenar información como el tema activo, la vista de proyectos o la vista del calendario.

Esta combinación resultó muy práctica, ya que evita sobrecargar el proyecto con soluciones complejas de estado global cuando la necesidad real es concreta. El sistema no requiere un almacén enorme para todo; solo necesita recordar unas pocas decisiones del usuario y sincronizar ciertos valores entre módulos. Desde el punto de vista de la experiencia, esto permite una navegación

continua: cuando el usuario regresa a la aplicación y encuentra la interfaz tal como dejó, con el mismo tema y en la vista que estaba utilizando.

2.9 Bibliotecas para interacción visual: animación, arrastre y analíticas

Una de las funcionalidades centrales de NovaTask es la posibilidad de mover tareas entre diferentes estados de forma intuitiva. Para lograrlo se requería una solución confiable de drag and drop. La biblioteca dnd-kit resultó adecuada porque se integra muy bien con React y permite manejar tanto el arrastre como la lógica de reordenamiento sin imponer estructuras demasiado rígidas. En este proyecto, esto fue especialmente importante para sostener el tablero kanban y mantener el orden de las tareas dentro de cada columna.

Framer Motion se utilizó para añadir transiciones y microanimaciones suaves. Su aporte no fue solo decorativo. En paneles, modales y mensajes de confirmación, estas pequeñas animaciones ayudan a comunicar cambios de estado y evitan que la interfaz se sienta brusca o estática. Cuando el usuario abre un modal, cierra un menú o confirma una acción, esos detalles refuerzan la sensación de fluidez y continuidad en la experiencia.

Para el módulo de analíticas se eligió Recharts, una librería de gráficos construida sobre componentes React. Esto permitió mostrar métricas semanales, distribución por proyecto y resúmenes de avance de manera coherente con el resto del diseño. En una herramienta orientada a la organización académica, la visualización de datos no es un lujo: ayuda a transformar información dispersa en patrones claros y fáciles de interpretar.

2.10 Node.js y Express en el desarrollo del backend

Del lado del servidor, Node.js resultó una opción muy conveniente porque permitió usar el mismo lenguaje tanto en el frontend como el backend. Aunque no es un requisito obligatorio, esta consistencia aporta fluidez al desarrollo, ya que reduce el esfuerzo mental de cambiar constantemente de paradigma. Node.js se presenta como un entorno de ejecución de JavaScript orientado a crear aplicaciones de red rápidas y escalables (Node.js, s. f.), una característica que resultó más que suficiente para un backend de API como el de NovaTask.

Sobre esa base se utilizó Express, uno de los frameworks más prácticos para estructurar rutas, middlewares y controladores en proyectos web basados en REST API. Express fue especialmente valioso porque mantiene la arquitectura clara y explícita: hay una capa de entrada

para las rutas, otra para validaciones, controladores ligeros y servicios donde reside realmente la lógica de negocio. Esta organización facilita explicar el sistema y también extenderlo sin tener que reescribir grandes partes del código.

Esta decisión demostró su utilidad cuando el proyecto creció más allá de la autenticación y los proyectos básicos. Gracias a la misma estructura, fue posible incorporar sin dificultad los módulos de materias, calendario, analíticas, plantillas y registro de actividad manteniendo siempre una organización estable.

2.11 PostgreSQL y Prisma para persistencia y modelado relacional

PostgreSQL fue seleccionado como motor de base de datos principalmente por su madurez, estabilidad y excelente soporte para modelos relacionales. En un sistema como NovaTask, donde existen usuarios, proyectos, tareas, subtareas, etiquetas, periodos académicos, materias, eventos y registros de actividad, las relaciones entre las distintas entidades representan un aspecto fundamental. No era suficiente almacenar información de forma aislada; era necesario garantizar la integridad de los datos, permitir consultas eficientes por estado o fecha, y manejar eliminaciones en cascada cuando corresponde.

Prisma se incorporó como ORM para facilitar la comunicación entre la capa de aplicación y el modelo de la base de datos. Su aporte en este proyecto fue doble: por un lado, permite definir el esquema de forma clara y versionarlo mediante migraciones; por otro, mejora considerablemente la legibilidad del código del backend al generar consultas expresivas y consistentes. En lugar de escribir las consultas SQL complejas directamente en la lógica de negocio, el proyecto cuenta con un modelo de datos tipado y fácil de mantener.

Esta combinación también resultó muy útil para la documentación. El esquema generado por Prisma ofrece un resumen ordenado de entidades, enums, claves foráneas, índices y restricciones de forma, lo que facilita explicar el diseño de la base de datos y justificar la necesidad de entidades adicionales como refresh tokens, plantillas, periodos académicos o registro de actividad para sostener la funcionalidad real del MVP.

2.12 JWT y bcrypt en autenticación propia

Uno de los criterios importantes del proyecto fue evitar depender de servicios externos de autenticación. La intención era mantener el control total sobre usuarios, sus credenciales, las

sesiones y las políticas internas del sistema. Por eso se optó por una estrategia clásica y suficientemente robusta para un MVP: una autenticación propia con correo electrónico y contraseña, utilizando tokens JWT para gestionar las sesiones y bcrypt para proteger las contraseñas.

JSON Web Token ofrece un formato compacto y seguro para transmitir información entre cliente y el servidor, y es ampliamente utilizado en escenarios de autenticación y autorización. En NovaTask se empleó un access token para las solicitudes autenticadas y un refresh token para sostener la continuidad de la sesión. Esta separación mejora la organización del flujo de acceso y deja abierta la posibilidad de reforzar políticas de expiración o revocación en futuras versiones.

Por su parte, bcrypt sigue siendo una de las mejores opciones cuando se necesita almacenar contraseñas de forma segura. Fue diseñado como un esquema adaptable al aumento de la capacidad computacional. La decisión de hashear las contraseñas en lugar de guardarlas en texto plano no fue una mejora opcional; se consideró una práctica básica y necesaria para tratar las credenciales con seriedad, incluso en el contexto de un proyecto académico.

2.13 Buenas prácticas aplicadas al proyecto

La fundamentación técnica de NovaTask no se limitó a elegir librerías populares. También implicó incorporar varias buenas prácticas que, aunque no siempre son visibles para el usuario, sostienen la calidad y robustez del sistema. Entre ellas destacan la validación de entradas con Zod, el manejo centralizado de errores, la configuración de CORS por origen específico, el uso de Helmet para cabeceras de seguridad y una clara separación del backend en rutas, controladores, servicios, middlewares, utilidades y validadores.

Desde el punto de vista del usuario, otras prácticas relevantes incluyeron el uso de skeleton loaders durante las cargas, mensajes visuales de confirmación, estados vacíos bien diseñados, protección de rutas privadas y un tratamiento claro de errores o acciones completadas. Estas decisiones mejoran notablemente la percepción de calidad porque el sistema acompaña al usuario en lugar de dejarlo frente a pantallas en blanco, silencios inesperados o mensajes confusos.

Finalmente, se incorporó trazabilidad mediante registros de actividad. Esto refuerza la idea de que la aplicación no solo guarda datos; sino que también construye una memoria operativa del trabajo realizado. En un contexto académico, este seguimiento resulta muy útil para reconocer los avances y entender con mayor claridad el ritmo de trabajo del estudiante.

Tabla 2. *Marco tecnológico adoptado en NovaTask*

Capa	Tecnologías principales	Aporte dentro del proyecto
Frontend	React, Vite, Tailwind CSS, React Router, Zustand	Estructura de interfaz, rutas, estilos utilitarios, estado global y construcción del cliente.
Interacción visual	Framer Motion, dnd-kit, Recharts	Animaciones suaves, tablero kanban con arrastre y visualización de métricas.
Backend	Node.js, Express, Zod	API REST, validaciones, separación de responsabilidades y orquestación del dominio.
Seguridad	JWT, bcrypt, Helmet, CORS	Autenticación propia, cifrado de contraseñas y protección básica del servidor.
Persistencia	PostgreSQL, Prisma	Modelo relacional, migraciones, restricciones, seed y consultas desde el backend.
Publicación	GitHub, Render, Vercel	Versionamiento y despliegue del backend y frontend en entornos públicos.

Nota. Elaboración propia

CAPÍTULO III: ANÁLISIS Y DISEÑO DE LA APLICACIÓN

3.1 Análisis del problema y del usuario objetivo

El análisis de NovaTask partió de una observación sencilla pero importante: el estudiante universitario necesita una herramienta que le ayude a organizar un trabajo heterogéneo, no solo listas aisladas de tareas. Existen actividades que pertenecen a materias, otras a proyectos largos y algunas responsabilidades personales. Algunas tienen fecha fija; otras dependen avances parciales, unas breves y otras requieren varias sesiones y subtareas. Esta variedad hace que una simple lista de pendientes se vuelva insuficiente muy rápido.

A diferencia de un gestor empresarial, donde varias personas comparten tableros, roles y permisos complejos, en este caso el usuario principal es una sola persona que busca control personal y claridad visual. Por eso se enfocó en un estudiante que administra su propio espacio de trabajo. Este usuario necesita entrar rápidamente, reconocer prioridades, decidir qué hacer en el día y revisar cómo se distribuye su carga sin tener que navegar por menús complicados.

Desde esa perspectiva, el sistema debía resolver al menos cuatro problemas concretos: centralizar información, reducir la pérdida de contexto, facilitar la priorización y ofrecer retroalimentación visual del avance. Si alguna funcionalidad no aportaba claramente a alguno de esos puntos, era preferible dejarla fuera del MVP.

3.2 Necesidades identificadas

El levantamiento funcional se construyó alrededor de necesidades prácticas reales. No se partió de una lista abstracta de funciones deseables, sino de situaciones recurrentes que viven los estudiantes: olvidar una fecha importante, no saber qué tarea sigue a continuación, perder de vista qué proyecto tiene más carga, necesitar una plantilla para comenzar rápido o querer revisar el estado general del semestre en una sola pantalla. Este enfoque ayudó a priorizar las funciones realmente útiles y a evitar un crecimiento desordenado del producto.

Las necesidades principales identificadas fueron:

- Registrar proyectos académicos con título, descripción, color y fecha límite.
- Crear tareas con prioridad, estado, fecha, subtareas y etiquetas.
- Mover tareas entre estados sin perder su orden interno.
- Vincular proyectos con materias y periodos académicos.

- Visualizar tareas y eventos en calendario semanal y mensual.
- Mostrar métricas de progreso, carga semanal y distribución por proyecto.
- Empezar nuevos proyectos a partir de plantillas reutilizables.
- Recibir retroalimentación al completar tareas y registrar actividad relevante.
- Mantener una experiencia usable en diferentes tamaños de pantalla.

3.3 Requerimientos funcionales

Los requerimientos funcionales se definieron como la base operativa del MVP. El objetivo fue cubrir el recorrido completo del usuario, desde el acceso hasta la revisión de resultados. La tabla siguiente resume los requerimientos principales considerados durante el diseño y la implementación.

Tabla 3. *Requerimientos funcionales principales*

Requerimiento	Descripción resumida
Registro de usuario	El sistema debe permitir crear una cuenta con nombre, correo y contraseña.
Inicio de sesión	El usuario debe autenticarse mediante correo y contraseña.
Persistencia de sesión	El sistema debe sostener la sesión y renovar el acceso con refresh token.
Protección de rutas	Las pantallas privadas solo deben estar disponibles para usuarios autenticados.
CRUD de proyectos	El usuario debe crear, editar y eliminar proyectos académicos.
CRUD de tareas	Cada proyecto debe permitir crear, editar y eliminar tareas.

Estados de trabajo	Las tareas deben manejar los estados Por hacer, En curso, Revisión y Hecho.
Drag and drop	El usuario debe poder mover tareas entre columnas del tablero.
Subtareas	Cada tarea debe admitir checklist de subtareas.
Etiquetas	El usuario debe crear etiquetas y asignarlas a las tareas.
Filtros y búsqueda	La vista de tareas debe filtrar por texto, estado, prioridad, etiqueta y temporalidad.
Calendario	La plataforma debe mostrar tareas y eventos en vistas semanal y mensual.
Analíticas	El sistema debe presentar métricas visuales de progreso y carga.
Plantillas	La aplicación debe permitir crear proyectos nuevos desde plantillas base.
Perfil académico	El usuario debe gestionar periodo, materias, eventos y preferencias visuales.
Registro de actividad	Las acciones relevantes deben quedar registradas en un log interno.
Despliegue web	El proyecto debe quedar preparado para ejecutarse

	localmente y publicarse en internet.

Nota. Elaboración propia

3.4 Requerimientos no funcionales

Los requerimientos no funcionales ayudaron a traducir las expectativas de calidad en decisiones concretas. Aunque el usuario no ve directamente una validación con Zod, un índice en base de datos o una política CORS, esos elementos son los que hacen que la experiencia sea confiable, segura y mantenible.

Tabla 4. *Requerimientos no funcionales*

Criterio	Aplicación en NovaTask
Usabilidad	Interfaz clara, navegación lateral, formularios comprensibles y retroalimentación inmediata.
Responsive	Adaptación del layout a escritorio, laptop, tablet y móvil.
Seguridad	Hash de contraseñas, JWT, Helmet, control CORS y protección de rutas.
Mantenibilidad	Separación por capas en backend y componentes reutilizables en frontend.
Escalabilidad	Arquitectura desacoplada y base de datos preparada para nuevos módulos.
Persistencia	Todos los datos deben quedar almacenados en PostgreSQL a través de Prisma.

Disponibilidad	El sistema debe poder ejecutarse localmente y publicarse en servicios web.
Consistencia visual	Paleta definida, tarjetas coherentes, jerarquía tipográfica y experiencia uniforme.
Trazabilidad	Registro de actividad y evidencia del cambio de estados relevantes.

Nota. Elaboración propia

3.5 Navegación y flujo general del sistema

La navegación de NovaTask se diseñó buscando un recorrido de baja fricción. La página principal es la pantalla de acceso. Una vez autenticado, el usuario llega al dashboard, que funciona como centro de lectura del sistema. Desde el menú lateral puede moverse fácilmente a proyectos, listas de tareas, calendario, analíticas, plantillas y perfil. Esta estructura busca que el usuario nunca se sienta perdido: siempre sabe dónde está y cómo volver a la vista que necesita.

A nivel de flujo, la aplicación sigue una lógica sencilla: primero se identifica el usuario, luego se carga su espacio de trabajo, después se consultan los módulos y, finalmente, se guardan los cambios sobre la misma fuente de verdad. El hecho de que el calendario, el dashboard y las analíticas consuman datos del mismo backend evita inconsistencias entre pantallas.

3.6 Arquitectura general del sistema

La arquitectura general del sistema se organiza en tres capas principales. En la capa de presentación se encuentra el frontend desarrollado con React y Vite, cuya responsabilidad es mostrar la información, gestionar la navegación, validar algunos campos básicos y coordinar la interacción con el usuario. En la capa de aplicación se ubica el backend con Express, donde residen las rutas, controladores, middlewares, servicios y validadores. Finalmente, la capa de persistencia corresponde a PostgreSQL, administrado mediante Prisma.

Esta arquitectura se eligió porque permite crecer de forma ordenada. Un cambio en el dashboard no obliga a modificar la base de datos; una mejora en la autenticación no requiere reescribir el calendario; y una nueva consulta en analíticas se puede agregar desde la capa de

servicios sin alterar el resto del sistema. Esta separación también facilita explicar el proyecto, ya que cada capa tiene responsabilidades bien definidas.

Adicionalmente, se contempló una capa transversal de soporte que incluye seguridad, manejo de errores, validación, logging y despliegue. Aunque estas funciones no son visibles para el usuario, son fundamentales parte el comportamiento global del sistema.

Tabla 5. *Organización técnica del backend*

Carpeta	Responsabilidad principal	Ejemplo dentro del proyecto
routes	Definir rutas y enlazarlas con controladores	auth.routes.js, project.routes.js, analytics.routes.js
controllers	Recibir la request y delegar a servicios	auth.controller.js, task.controller.js
services	Aplicar reglas de negocio	task.service.js, template.service.js, dashboard.service.js
middleware	Ejecutar lógica transversal	auth.middleware.js, error.middleware.js, validate.middleware.js
validators	Validar payloads de entrada	auth.validator.js, task.validator.js, template.validator.js
utils	Reutilizar funciones y conexión a Prisma	jwt.js, password.js, prisma.js, AppError.js

Nota. Elaboración propia .

3.7 Diseño de la base de datos

El diseño de la base de datos comenzó con el modelo mínimo necesario para proyectos, tareas, subtareas, etiquetas y actividad. Sin embargo, a medida que avanzaba el desarrollo de NovaTask, fue necesario extender ese esquema inicial para incluir entidades que responden mejor al contexto académico: periodos, materias, eventos, refresh tokens, plantillas y tareas de plantillas. Esta ampliación no fue un desvío, sino una consecuencia lógica de querer que el sistema represente de forma más fiel la organización real del estudiante.

El esquema actual utiliza identificadores UUID como claves primarias y define relaciones explícitas entre entidades. Los proyectos pertenecen a un usuario y pueden vincularse opcionalmente con una materia. Las tareas dependen de un proyecto y pueden tener subtareas y etiquetas. Los eventos pueden relacionarse tanto con materias como con proyectos. Los registros de actividad permiten conservar memoria de acciones relevantes y los refresh tokens ayudan a sostener la sesión sin obligar al usuario a iniciar sesión con frecuencia.

Desde una perspectiva de normalización, el diseño evita duplicar información innecesaria y privilegia relaciones consistentes. Por ejemplo, las etiquetas se modelan por usuario y se enlazan a tareas mediante una tabla intermedia. De esa forma, una misma etiqueta puede reutilizarse muchas veces sin duplicarse y sin comprometer la integridad del modelo.

Tabla 6. *Entidades principales del modelo de datos*

Entidad	Propósito	Relaciones clave
users	Almacena la cuenta principal del usuario.	Se relaciona con proyectos, tags, periodos, materias, eventos, logs y tokens.
academic_terms	Representa periodos académicos como semestres o ciclos.	Pertenece a users y se relaciona con subjects.
subjects	Agrupa materias o asignaturas del estudiante.	Pertenece a users y puede asociarse a projects y events.
projects	Contiene proyectos académicos o personales.	Pertenece a users y puede vincularse a subjects.
tasks	Modela tareas individuales con estado y prioridad.	Pertenece a projects y se relaciona con subtasks y tags.
subtasks	Descompone tareas en pasos menores.	Pertenece a tasks.
tags	Clasifica tareas por tema, materia o prioridad adicional.	Pertenece a users y se vincula a tasks por task_tags.

events	Registra clases, exámenes, reuniones y hitos.	Pertenece a users y puede enlazarse a subjects y projects.
activity_logs	Registra acciones importantes del sistema.	Pertenece a users y referencia entidades operativas.
refresh_tokens	Sostiene la persistencia de sesión.	Pertenece a users.
templates	Guarda estructuras base reutilizables.	Se relaciona con template_tasks.

Nota. Elaboración propia

3.8 Prototipado y diseño de interfaz

Antes de consolidar el frontend en React, se trabajó con prototipos visuales en Figma y con versiones estáticos en HTML, CSS y JavaScript. Esta fase fue clave porque permitió definir el tono visual sin comprometer todavía la lógica de negocio. En ese momento se establecieron varios rasgos que se mantuvieron: tarjetas sobrias, menú lateral, modo oscuro como base, bordes redondeados y un estilo inspirados en herramientas modernas como Linear, Notion o Trello, pero adaptado al uso académico.

La pantalla de acceso fue una de las primeras en diseñarse con detalle, ya que representa la puerta de entrada a toda la experiencia. Posteriormente se avanzó sobre en el dashboard y la vista de proyectos, que terminaron siendo los espacios con mayor densidad funcional. A lo largo del desarrollo se respetó la identidad original, pero se profesionalizaron los componentes: mejor distribución del contenido, jerarquías más claras, mejoras responsive, menú de cuenta, loaders, estados vacíos y un modo claro más equilibrado.

Este trabajo de prototipado no fue un paso aislado, sino un puente importante entre la idea inicial y la implementación real. Muchas decisiones que hoy se ven en la aplicación como el uso de chips, tarjetas métricas, vistas modulares y transiciones suaves surgieron en esa etapa y se fueron refinando conforme el sistema crecía.

3.9 Flujo funcional del MVP

El flujo funcional del MVP se diseñó para que cada módulo alimente al siguiente de forma natural. Una vez autenticado, el usuario llega al dashboard y obtiene una visión general del trabajo

pendiente. Desde allí puede abrir proyectos y gestionar tareas. Las tareas con fecha alimentan el calendario. A su vez, el conjunto de tareas y estados alimenta las analíticas. Finalmente, el módulo de plantillas permite acelerar la creación de nuevos proyectos, mientras que el perfil académico complementa el contexto con materias, periodos y eventos.

Esta continuidad fue una decisión deliberada. En lugar de crear pantallas desconectadas, se buscó que la aplicación funcionara como un sistema integrado. Si un usuario cambia el estado de una tarea a Hecho, ese movimiento no debe quedarse solo en el tablero: sino que también debe reflejarse en las métricas del dashboard, en las analíticas y en el registro de actividad.

1. El usuario crea su cuenta o inicia sesión.
2. El backend valida credenciales y emite tokens de acceso y refresh.
3. El frontend protege las rutas internas y carga la información del usuario.
4. El dashboard consulta métricas, actividad reciente, eventos cercanos y carga semanal.
5. El usuario crea un proyecto y posteriormente sus tareas asociadas.
6. Las tareas pueden ordenarse por estado, prioridad, fecha, etiqueta y búsqueda libre.
7. Al cambiar de estado, el backend persiste el nuevo orden y actualiza el tablero.
8. Las tareas con fecha y los eventos académicos se integran a la agenda temporal.
9. Las analíticas reutilizan la misma base de datos para construir gráficos y resúmenes.
10. Cuando una tarea pasa a completada, el sistema actualiza métricas y deja constancia de la acción en el registro interno.

3.10 Diseño de despliegue y continuidad operativa

Desde temprano se pensó en que el proyecto pudiera salir del entorno local. El frontend se preparó para Vercel; el backend, para Render; y la base de datos, para PostgreSQL local o servicios administrados. Este diseño reduce la brecha entre desarrollo y la publicación, y deja una base real para la fase siguiente de validación con usuarios.

CAPÍTULO IV: DESARROLLO E IMPLEMENTACIÓN DE NOVATASK

4.1 Preparación del entorno de desarrollo

La implementación de NovaTask se llevó a cabo en un entorno de desarrollo orientado a la productividad y la continuidad. Se trabajó principalmente con Visual Studio Code como editor, Git y GitHub para el control de versiones, Node.js para el ecosistema tanto del frontend como del backend, PostgreSQL como base de datos y Prisma como ORM. Además, se utilizaron herramientas complementarias como pgAdmin y clientes SQL para revisar conexiones, tablas y datos durante las pruebas locales.

Una parte importante de esta preparación fue dejar scripts claros para iniciar el sistema. El backend incluye comandos para generar el cliente Prisma, aplicar migraciones, poblar la base de datos con datos de prueba y levantar el servidor. El frontend, por su parte, quedó configurado para desarrollo, construcción y vista previa. Aunque esto puede parecer un detalle operativo, en el contexto de una tesis tiene gran valor: evita que el proyecto dependa exclusivamente de un solo equipo y mejora la reproducibilidad.

También se prepararon archivos de variables de entorno y configuraciones específicas para distintos destinos de despliegue. Esto permitió separar la lógica del proyecto de los datos sensibles, algo especialmente importante al contar ya con una versión publicada en servicios externos.

Tabla 7. *Estructura general del repositorio NovaTask*

Ruta	Contenido principal	Uso dentro del proyecto
frontend/	Aplicación React	Pantallas, componentes, estado visual, navegación y consumo de API.
backend/	API Express	Autenticación, reglas de negocio, validaciones, Prisma y servicios.
database/	Esquema SQL	Versión de apoyo para documentación y trazabilidad del modelo relacional.

db/	Copia de apoyo	Material adicional relacionado con el diseño de datos.
docs/	Documentación técnica	Arquitectura, flujo MVP, diseño de base y contrato de API.
.github/	Automatización	Workflow de publicación para GitHub Pages y soporte al versionamiento.
render.yaml y vercel.json	Configuración de despliegue	Publicación del backend en Render y del frontend en Vercel.

Nota. Elaboración propia

4.2 Evolución del proyecto desde el prototipo inicial

NovaTask no surgió como una plataforma completa desde el principio. El proceso comenzó con el análisis del documento base de la titulación y la definición de una propuesta que fuera útil, defendible y técnicamente realizable. En esa etapa se redactaron avances iniciales, se organizaron ideas del MVP y se generó documentación sobre el flujo de usuario, la arquitectura y la base de datos.

Posteriormente llegó la fase de prototipado visual. Se crearon pantallas de referencia en Figma y versiones estáticas en HTML, CSS y JavaScript que ayudaron a definir el tono de la interfaz. En ese momento se optó por una estética moderna tipo SaaS con modo oscuro como base. Más adelante, ese diseño no se reemplazó, sino que se profesionalizó: se respetó la idea original y se refinaron tarjetas, navegación, formularios, modales y transiciones.

Una vez consolidado el prototipo, se pasó a implementarlo en React con Vite, incorporando posteriormente Tailwind CSS, React Router, Zustand, Framer Motion, Recharts y dnd-kit. En paralelo se desarrolló el backend con Express, Prisma y PostgreSQL. Esto permitió que el proyecto evolucionara de una maqueta visual a una aplicación con autenticación, persistencia real y lógica de negocio. Finalmente, se estabilizó la publicación mediante GitHub, Render y Vercel.

Hubo también una etapa intermedia en la que se activó un modo de demostración con datos simulados. Esto ocurrió mientras se resolvían algunos problemas locales de conexión con

PostgreSQL. Lejos de ser un retroceso, esa decisión permitió mantener el ritmo de desarrollo del frontend mientras se consolidaba la persistencia definitiva.

4.3 Implementación del frontend

4.3.1 Pantalla de acceso e identificación del usuario

La aplicación se configuró para que la ruta principal sea la pantalla de login. Esta fue una corrección importante respecto a versiones anteriores, donde algunos enlaces redirigían directamente al dashboard. En la versión actual, el usuario entra primero a una pantalla clara de inicio de sesión o registro, con validaciones básicas de nombre, correo y contraseña, opción de mostrar u ocultar la clave y mensajes claros cuando algo falla o cuando el acceso se completa correctamente.

La pantalla de acceso también cumple un rol de presentación del producto. Además del formulario, explica brevemente qué ofrece NovaTask y resume sus beneficios en tarjetas simples. Desde el punto de vista de la experiencia de usuario, esto ayuda a contextualizar la plataforma sin abrumar al visitante con demasiada información técnica.

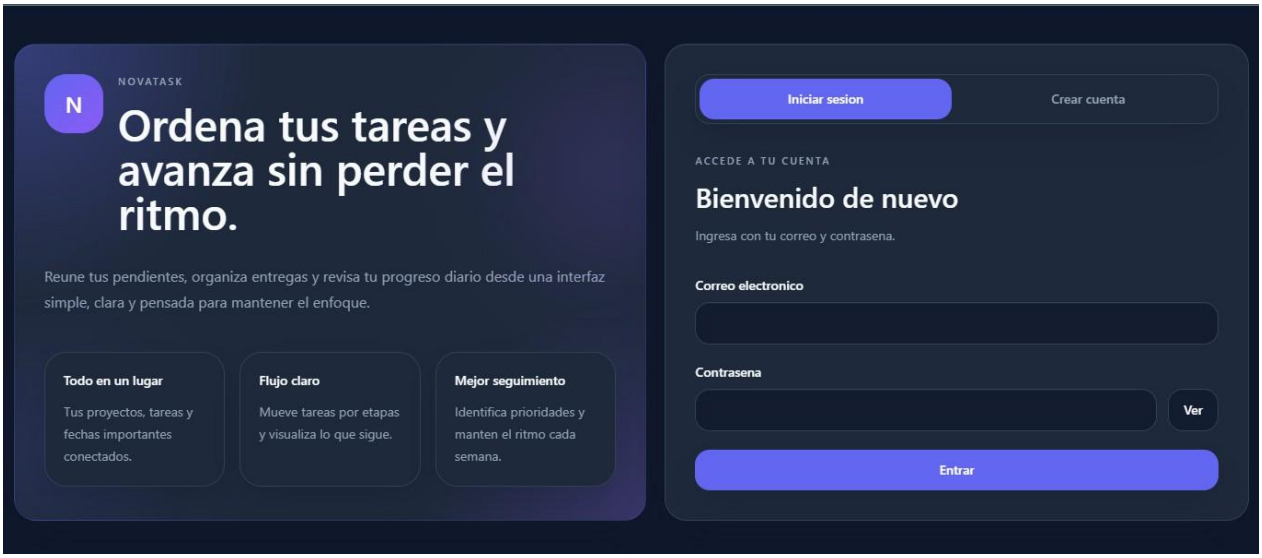


Figura 1. Captura de la pantalla de login

Elaboración propia.



Figura 2. Espacio para captura de la pantalla final de login y registro

Elaboración propia.

4.3.2 Dashboard como centro operativo

El dashboard fue concebido como el punto de lectura principal del sistema. No solo muestra métricas generales, sino también un resumen del trabajo inmediato, accesos directos, materias activas, próximos eventos, carga semanal, entregas cercanas, resumen por proyecto y actividad reciente. La idea fue que el usuario pueda entender en pocos segundos cómo está su semana y qué requiere atención inmediata.

En las primeras versiones, algunos paneles presentaban problemas de superposición, contraste o jerarquía visual. A medida que avanzó el trabajo, se fueron corrigiendo esos detalles, especialmente en el topbar y en los bloques del dashboard. Se ajustaron capas visuales, sombras, comportamiento del menú de cuenta y armonía de las tarjetas. También se pulió la experiencia en modo claro, que inicialmente era menos equilibrada que el modo oscuro.

El resultado actual es un dashboard más coherente y cercano a un producto real: resume sin saturar, orienta sin esconder acciones y sirve como entrada efectiva a los demás módulos.



Figura 3. Captura del dashboard principal

Elaboración propia

4.3.3 Gestión de proyectos

La vista de proyectos organiza el trabajo por contenedores principales. Cada proyecto puede tener título, descripción, color, materia vinculada y fecha límite. Este módulo se reforzó bastante durante el desarrollo, porque una de las primeras limitaciones detectadas 0era que no se veía nada o no se podía crear contenido con claridad. La solución pasó por revisar la carga de datos, los modales de creación y edición, y la lógica de refresco del detalle activo.

Hoy el módulo permite crear, editar y eliminar proyectos con persistencia real en la base de datos. También ofrece una lectura visual del progreso por proyecto, resúmenes rápidos y un espacio interno para operar tareas. Este comportamiento es importante porque convierte al proyecto en la unidad organizativa del trabajo académico y evita que el tablero se convierta en una lista dispersa sin contexto.

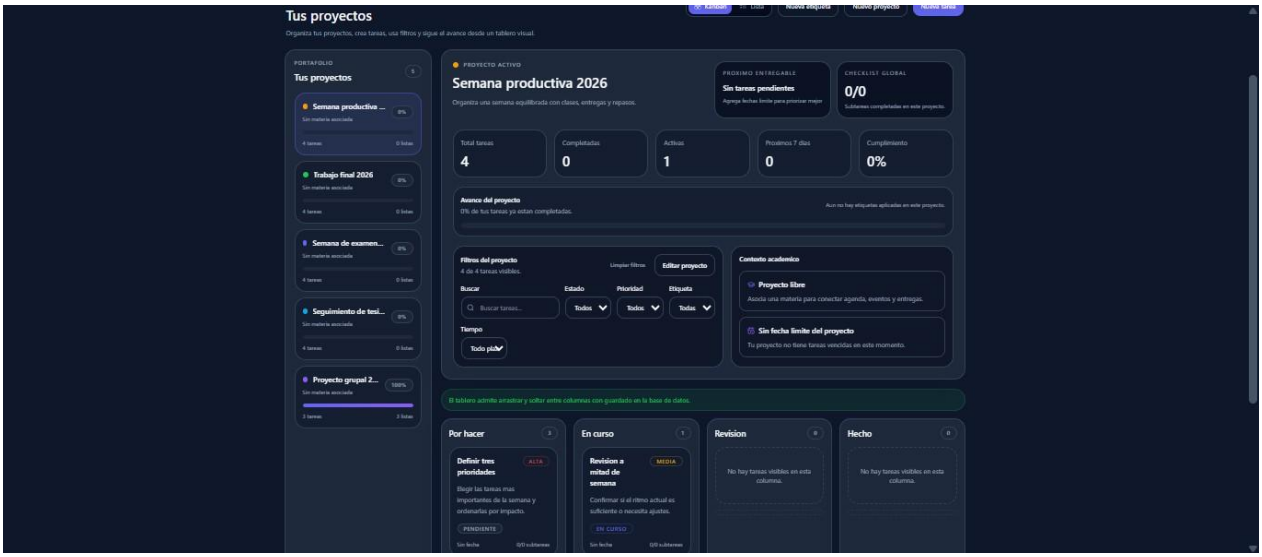


Figura 4. Captura del módulo de proyectos

Elaboración propia.

4.3.4 Kanban, vista de lista, filtros, etiquetas y subtareas

El kanban es uno de los módulos más representativos de NovaTask. Se implementó con dnd-kit para permitir arrastrar tareas entre las columnas Por hacer, En curso, Revisión y Hecho. Cada movimiento se refleja visualmente y luego se persiste en PostgreSQL mediante una llamada al backend. Para el usuario, esto reduce considerablemente la fricción de seguimiento: no tiene que editar manualmente el estado en un formulario cada vez que avanza.

Además del tablero, se incorporó una vista de lista con filtros por estado, prioridad, etiqueta, búsqueda libre y temporalidad. Esta segunda representación resulta útil cuando se necesita revisar el conjunto completo de tareas sin depender solo de la lectura por columnas. El proyecto también permite crear etiquetas personalizadas, agregar subtareas tipo checklist y abrir modales de edición que concentran información y acciones relevantes.

Un detalle funcional importante es el manejo del cierre de ciclo de la tarea. Cuando una actividad pasa a estado Hecho, el frontend actualiza el tablero y prepara el cambio para que también se refleje en dashboard, las analíticas y el registro interno. Esto refuerza la sensación de avance y evita que el cierre de una tarea quede aislado en una sola pantalla.

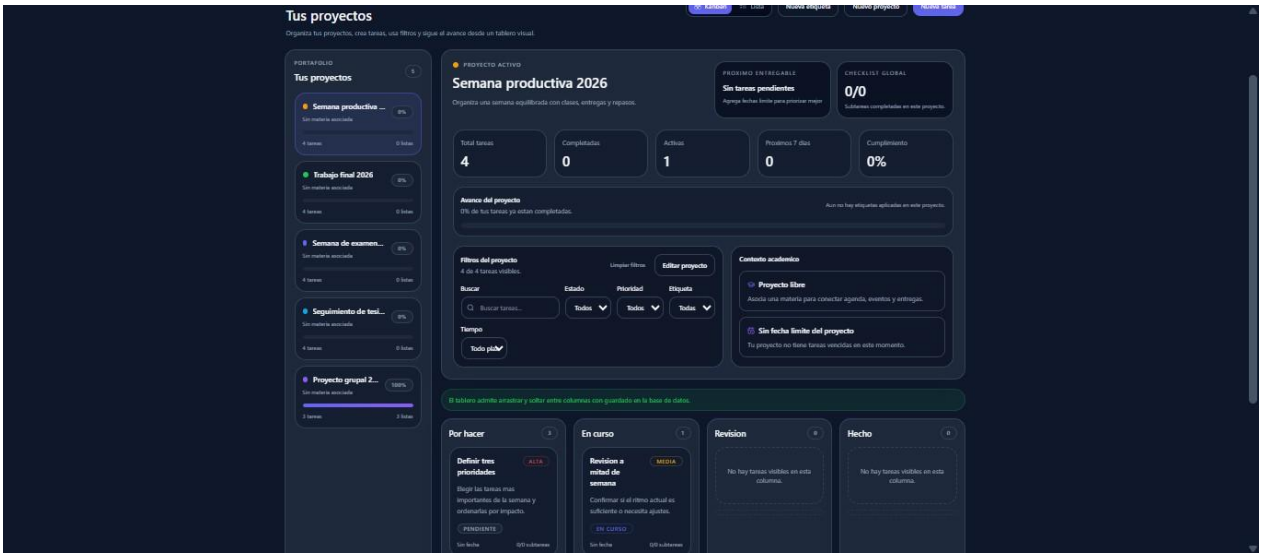


Figura 5. Captura del tablero Kanban

Elaboración propia.

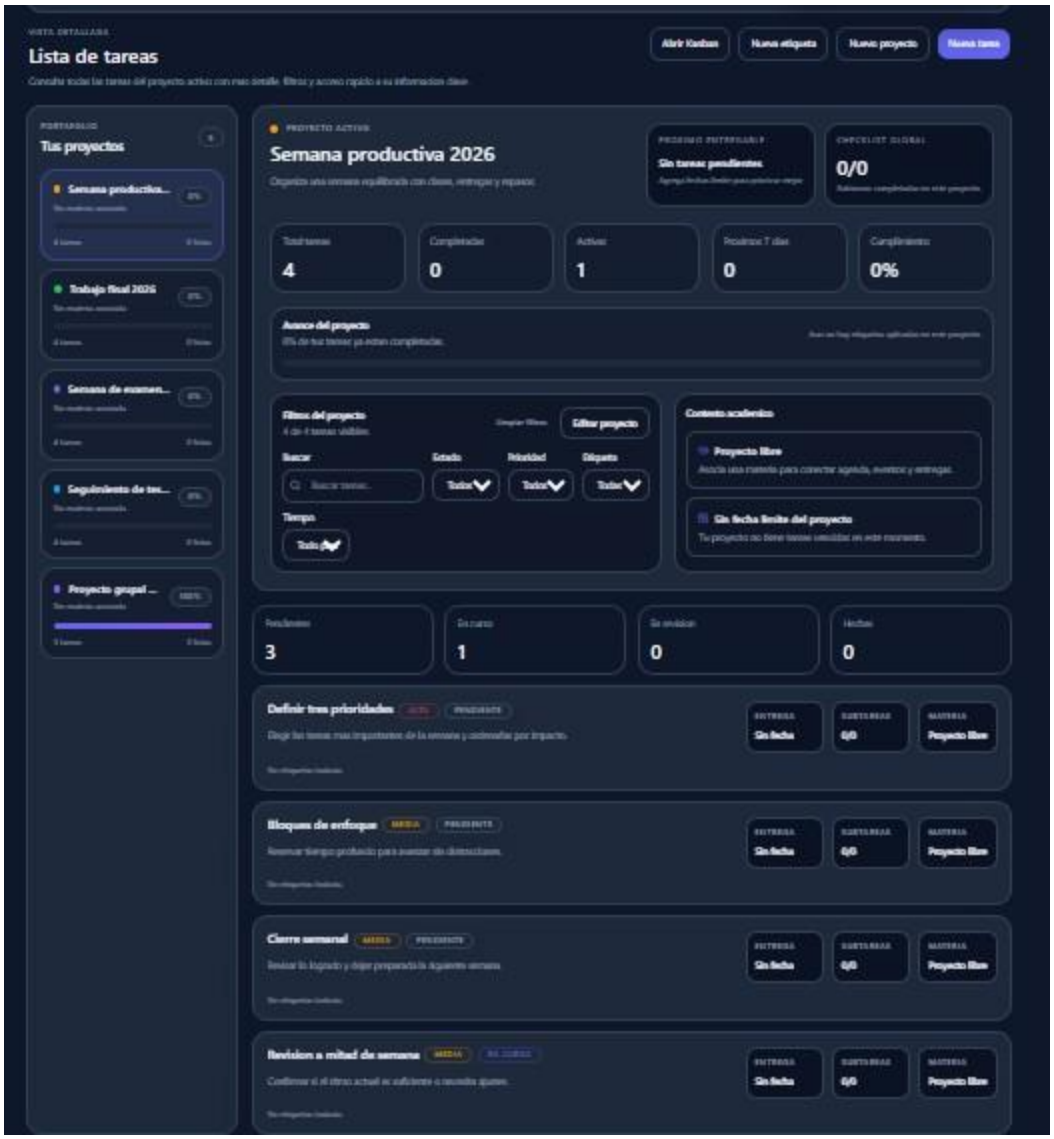


Figura 6. Captura de la vista de lista

Elaboración propia.

4.3.5 Calendario académico

El calendario integra dos tipos de información: tareas con fecha límite y eventos académicos creados desde el perfil. El módulo ofrece una vista mensual y otra semanal. La mensual permite revisar la distribución general de actividades; la semanal ayuda a leer la carga inmediata y seleccionar días específicos con más detalle.

Esta integración fue importante porque evita que la agenda quede desconectada del trabajo real. Una entrega no es solo una fecha aislada, sino una tarea vinculada a un proyecto, con

prioridad, estado y subtareas. Al reunir esa información en el calendario, NovaTask ofrece una lectura más útil de la agenda y reduce la necesidad de duplicar registros en otra plataforma.

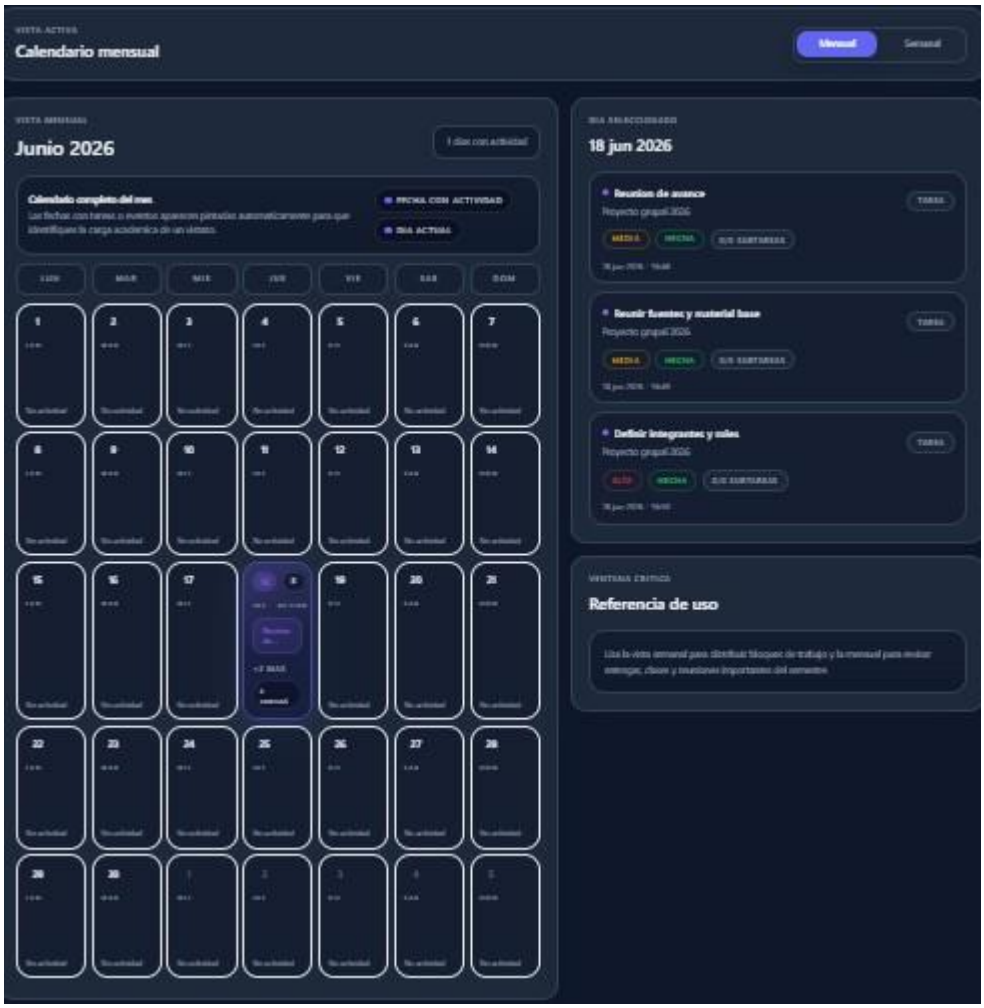


Figura 7. Captura del calendario mensual

Elaboración propia.

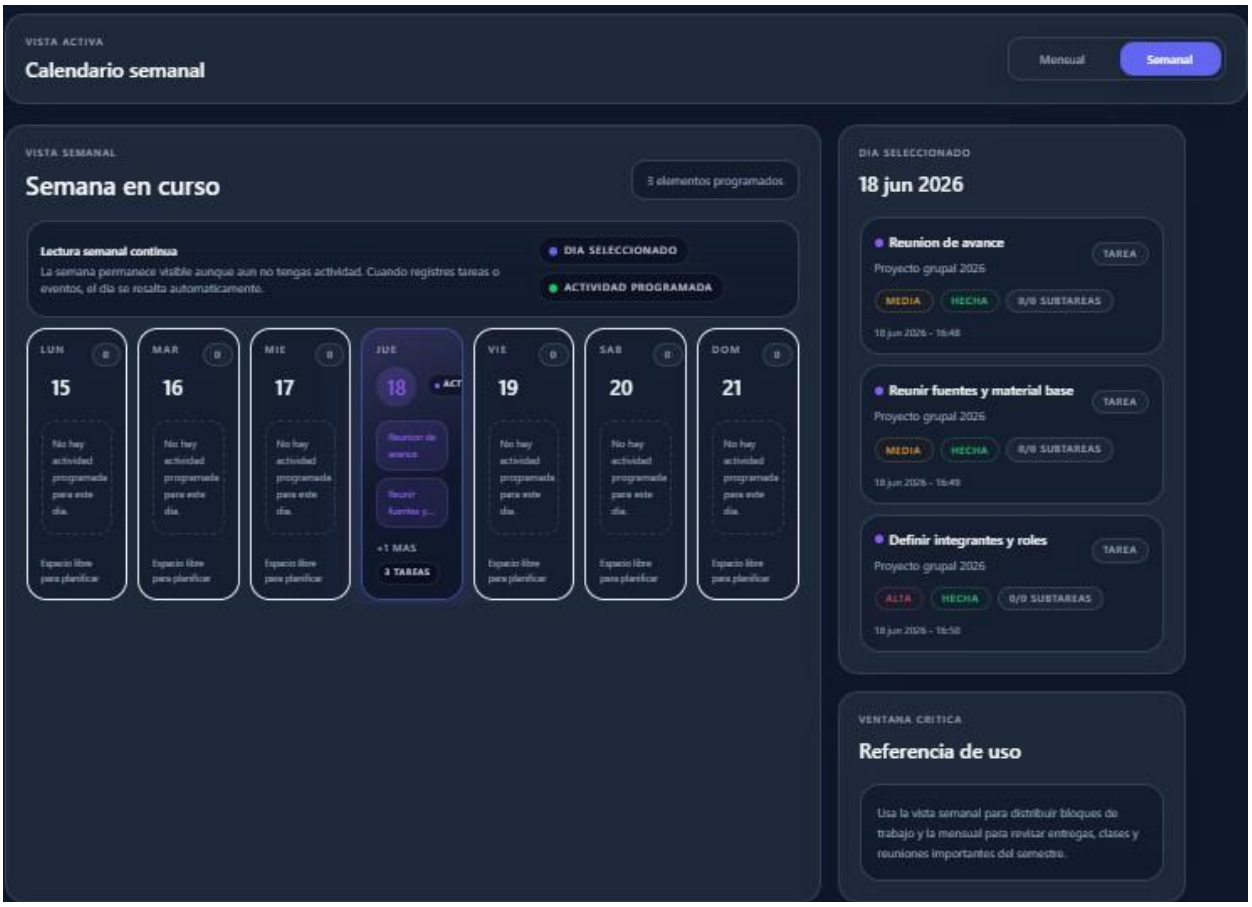


Figura 8. Captura del calendario semanal

Elaboración propia.

4.3.6 Analíticas y lectura del avance

El módulo de analíticas se desarrolló con Recharts y presenta un resumen de cumplimiento, tareas por proyecto, tendencia semanal, radar operativo y comparación de avance entre proyectos. El propósito de este espacio no fue generar reportes complejos, sino ofrecer una lectura visual rápida sobre la carga real del usuario y el comportamiento reciente del trabajo.

Una mejora reciente consistió en adaptar mejor la experiencia del módulo al tema claro. Algunos gráficos, tooltips y superficies estaban demasiado pensados para el entorno oscuro. Por eso se ajustaron colores de ejes, fondos de tooltip, bloques internos y barras de progreso para que el resultado se viera correcto cuando el usuario cambia de tema.



Figura 9. Captura de la página de analíticas

Elaboración propia.

4.3.7 Plantillas para acelerar nuevos proyectos

Las plantillas nacieron como una forma de evitar que cada nuevo proyecto empezara desde cero. En la versión actual, el sistema incluye estructuras iniciales listas para usar, por ejemplo, para semanas de exámenes, proyectos grupales o trabajo final/titulación. Cuando el usuario aplica una plantilla, el backend crea un proyecto nuevo y replica las tareas base definidas para ella.

Este módulo tiene valor funcional como metodológico. Desde el punto de vista del usuario, reduce tiempo de configuración. Se demuestra que el sistema ya puede reutilizar conocimiento operativo y no solo guardar tareas aisladas. Es decir, NovaTask empieza a comportarse como una plataforma que acumula y reaprovecha estructuras de trabajo.

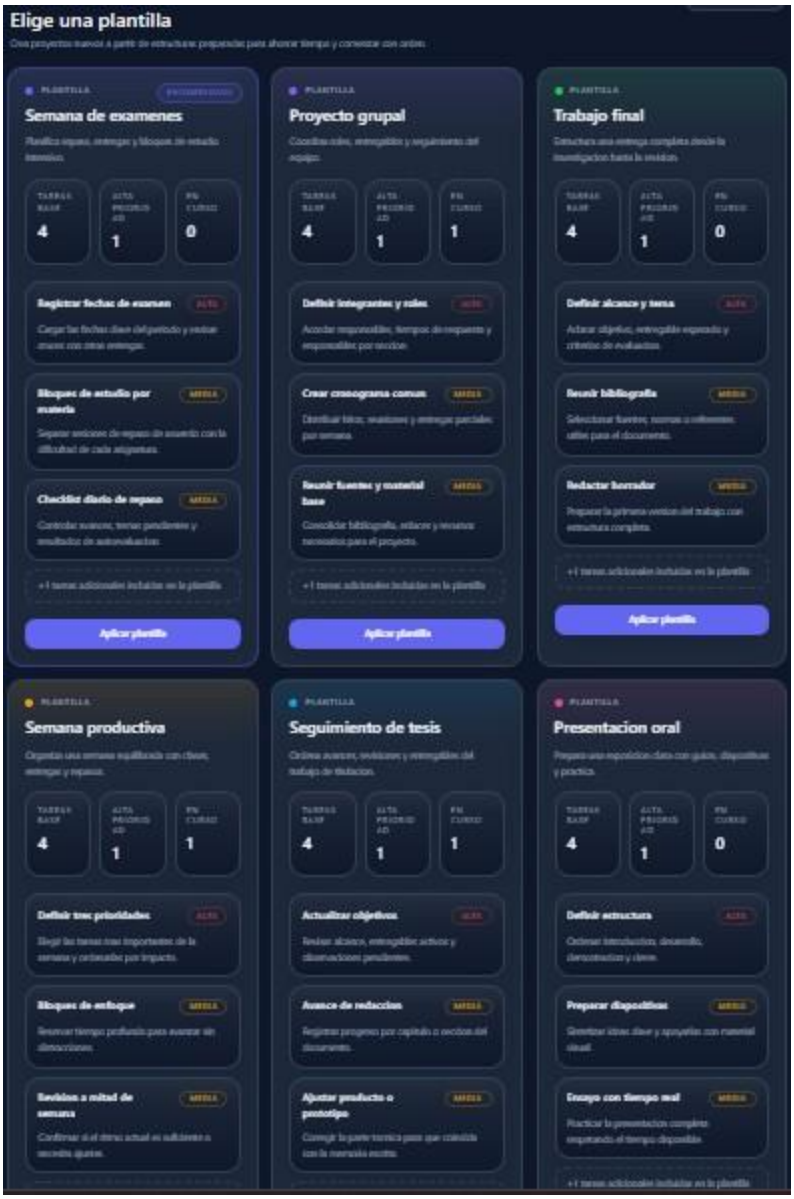


Figura 10. Captura del módulo de plantillas

Elaboración propia.

4.3.8 Perfil, periodos, materias y agenda

El perfil terminó convirtiéndose en un módulo más amplio de lo previsto inicialmente. Además de editar nombre y avatar, permite registrar periodos académicos, materias y eventos. También ofrece el cambio entre modo oscuro y claro. Esta ampliación fue coherente con la lógica del producto: si NovaTask pretende ser una herramienta académica y no solo un tablero genérico,

necesita entender qué materias cursa el usuario, qué periodo está activo y qué eventos forman parte de su agenda.

Este módulo se organizó mediante pestañas para separar cuenta, información académica y agenda. El usuario puede crear y eliminar periodos, asociar materias a esos periodos, registrar docente y color por materia, y crear eventos vinculados opcionalmente con una materia o un proyecto. El resultado es una experiencia más contextualizada, que acerca el sistema a la vida real del estudiante.

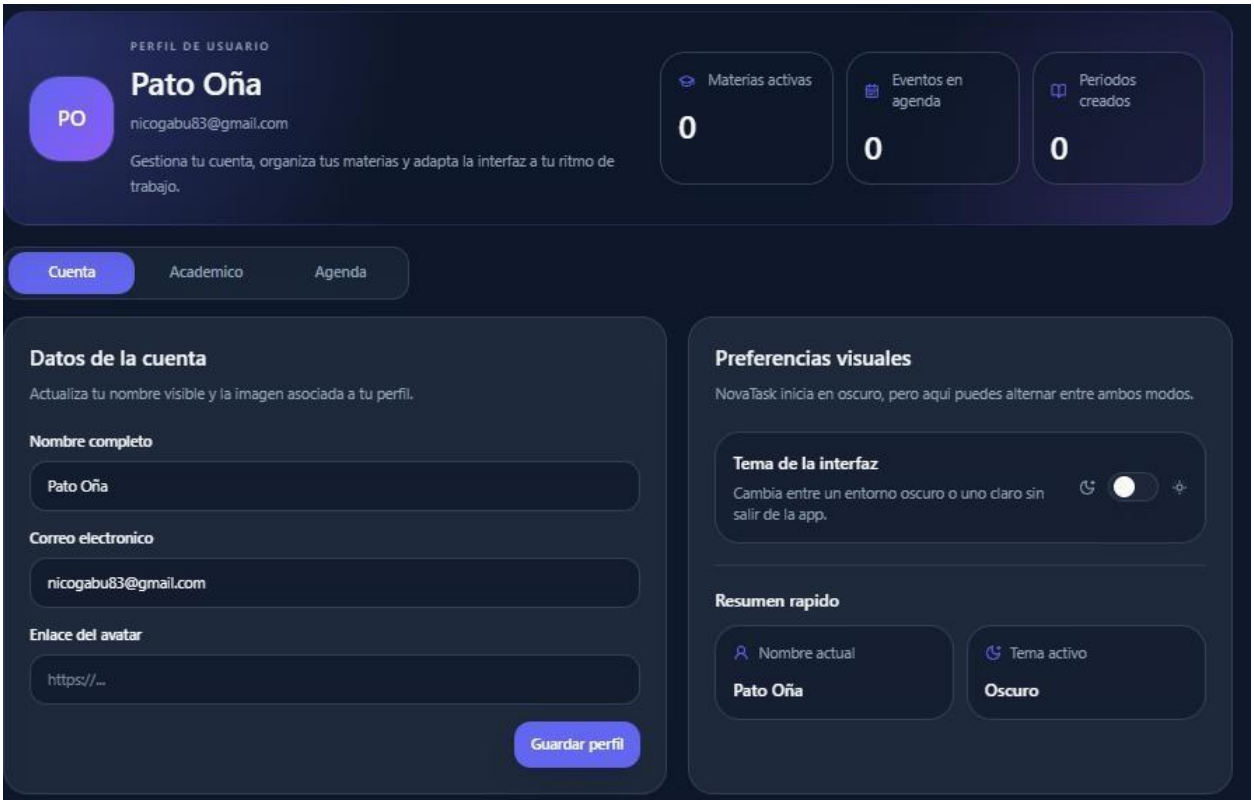


Figura 11. Captura del perfil de usuario

Elaboración propia

4.3.9 Navegación lateral, topbar y menú de cuenta

La navegación se estructuró a partir de una barra lateral moderna y una barra informativa. La barra lateral concentra el acceso a módulos y mantiene visible el contexto general. El topbar, por su parte, presenta la fecha, el nombre de la vista y una tarjeta de cuenta desde la cual el usuario puede editar su perfil o cerrar sesión.

Durante el refinamiento visual aparecieron varios problemas de capas y superposición: el menú de cuenta quedaba por detrás de otros bloques, algunas tarjetas invadían espacios del topbar y ciertos contenedores no respetaban bien la jerarquía visual. Corregir estos detalles fue una parte importante del trabajo porque un producto puede estar técnicamente correcto y aun así transmitir descuido si los elementos flotantes no se comportan bien.



Figura 12. Captura del menú de cuenta y acciones del perfil

Elaboración propia

4.3.10 Responsive, tema claro y tema oscuro

Otro frente de trabajo importante fue la adaptación a distintos tamaños de pantalla. Varias pantallas, especialmente dashboard, proyectos y calendario, se revisaron para evitar cortes, solapamientos o tarjetas difíciles de leer en resoluciones intermedias. Esto implicó ajustar rejillas, alturas, márgenes, breakpoints y comportamiento de paneles laterales.

En paralelo, se realizó una mejora intensa del tema claro. El usuario reportó que el modo claro se veía poco equilibrado, lo que obligó a revisar fondos, relieve de tarjetas, tooltips, tabs, botones, badges, mensajes breves y superficies internas. El objetivo fue que el cambio de tema no se sintiera como un agregado secundario, sino como una opción real de uso. Esta iteración representa bien la lógica del proyecto: no solo agregar funciones, sino elevar la calidad percibida de lo ya construido.

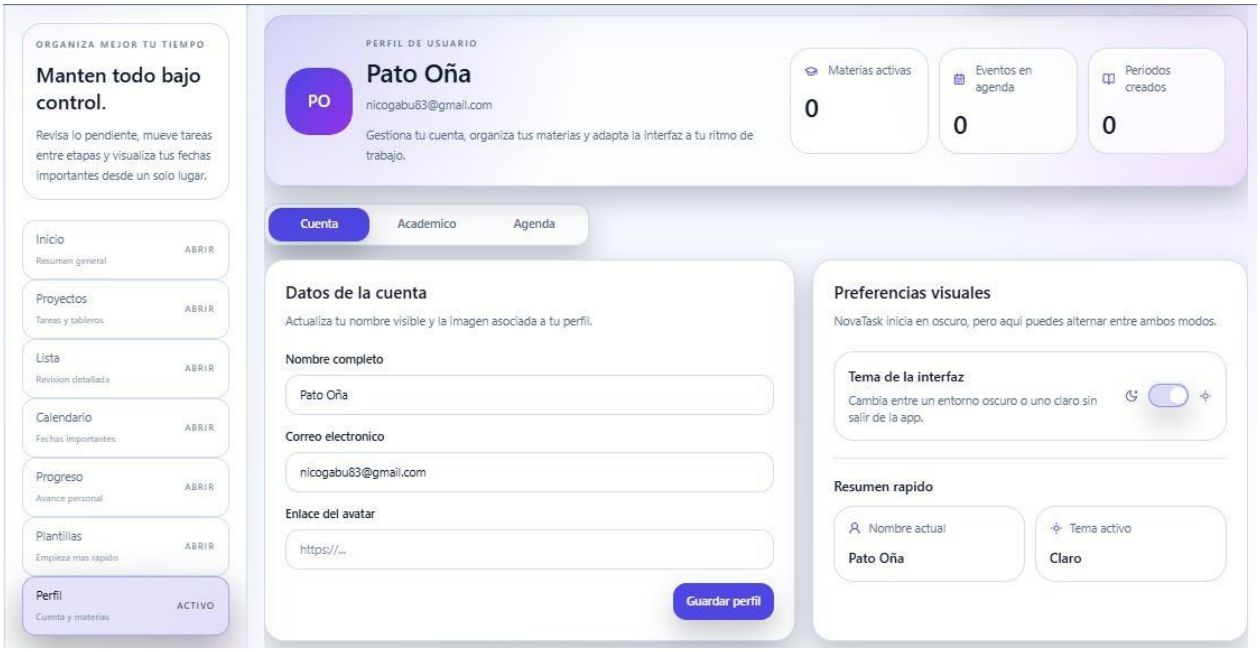


Figura 13. Captura del tema claro

Elaboración propia.

4.4 Implementación del backend

4.4.1 Estructura del servidor

El backend se organizó en una estructura clásica pero clara: rutas, controladores, servicios, middlewares, validadores, utilidades y configuración. Esta forma de trabajo ayudó a que la lógica de negocio no quedara escondida dentro de archivos grandes y mezclados. Las rutas definen los puntos de entrada; los controladores reciben la solicitud; los servicios concentran la lógica; y los middlewares resuelven validación, autenticación o manejo de errores.

Esta arquitectura fue importante para sostener el crecimiento del proyecto. Al principio solo existían necesidades de autenticación y proyectos, pero después se añadieron tareas, etiquetas, plantillas, calendario, analíticas y módulo académico. Como la estructura ya estaba bien delimitada, fue posible expandir el sistema sin reescribir la base.

4.4.2 Autenticación y control de acceso

El módulo de autenticación implementa registro, login, consulta del usuario actual, refresh de sesión y logout. Las contraseñas se almacenan hasheadas con bcrypt y el backend emite dos

tipos de token: uno de acceso y otro de refresh. Este último, además, se guarda en la base de datos para facilitar revocación y trazabilidad.

La protección de rutas privadas se resuelve mediante un middleware que verifica el token, recupera el usuario y bloquea la operación si no existe autenticación válida. Esta capa fue fundamental porque todo el modelo de datos se organiza por usuario: proyectos, tareas, etiquetas, periodos, materias y eventos pertenecen a una cuenta concreta y no deben mezclarse entre sí.

4.4.3 Servicios por módulo

Cada conjunto de funciones relevantes del sistema quedó encapsulado en un servicio. Existen servicios para autenticación, proyectos, tareas, calendario, analíticas, plantillas, actividad y académicos. Esta decisión permite que el backend no sea una colección de endpoints aislados, sino un sistema con responsabilidades distribuidas de forma lógica.

El servicio de tareas es uno de los más completos porque maneja creación, edición, eliminación, movimiento entre columnas y gestión de subtareas. El servicio de dashboard consolida datos de varios módulos. El servicio de calendario unifica tareas y eventos. El de analíticas procesa el estado general del trabajo. Y el módulo académico articula periodos, materias y agenda. En conjunto, estos servicios convierten la API en una pieza central del producto y no solo en un intermediario entre formularios y tablas.

4.4.4 Validaciones, seguridad y manejo de errores

Para validar entradas se utilizó Zod. Esto permite definir reglas claras en esquemas por módulo y evitar que payloads incompletos o mal formados lleguen directamente a la lógica de negocio. Los errores se canalizan por un middleware centralizado que responde con mensajes más consistentes, lo cual también mejora la integración con el frontend.

A nivel de seguridad básica se configuró Helmet para cabeceras de protección y CORS para permitir únicamente orígenes autorizados, incluyendo el entorno local y el frontend publicado. Aunque NovaTask todavía es un MVP, este conjunto de decisiones responde a una idea importante: la calidad técnica no empieza cuando el sistema ya es masivo; empieza desde las primeras versiones funcionales.

Tabla 8. *Resumen de módulos y endpoints del backend*

Módulo	Endpoints principales	Propósito
Auth	/api/auth/register, /login, /refresh, /me, /logout	Gestionar acceso, sesión y usuario autenticado.
Projects	/api/projects y /api/projects/:id	Crear, listar, editar y eliminar proyectos.
Tasks	/api/tasks, /api/tasks/:id, /move, /subtasks	Administrar tareas, moverlas, crear subtareas y tags asociados.
Dashboard	/api/dashboard/overview	Consolidar métricas y resúmenes del panel principal.
Calendar	/api/calendar	Unificar tareas y eventos para la agenda temporal.
Analytics	/api/analytics/overview	Calcular indicadores para la visualización gráfica.
Templates	/api/templates y /api/templates/:id/apply	Listar plantillas y crear proyectos desde ellas.
Academics	/api/academics/overview, /terms, /subjects, /events	Gestionar el contexto académico del usuario.

Nota. Elaboración propia

4.5 Implementación de la base de datos

La base de datos se implementó con PostgreSQL y Prisma. El esquema principal define enums para prioridad, estado, tipo de evento y tipo de entidad de actividad. También modela relaciones con eliminación en cascada cuando corresponde y crea índices en campos de consulta frecuente, como usuario, proyecto, estado o fecha. Estas decisiones ayudan a mantener integridad y preparan el sistema para crecer sin necesidad de rehacer el modelo.

Prisma aportó además una capa práctica de migraciones y un seed con datos de apoyo. Gracias a eso, el proyecto puede levantarse con mayor rapidez en un entorno nuevo y sostener una experiencia de prueba más realista. Este punto fue especialmente útil durante el desarrollo, ya que

permitió poblar el sistema con proyectos, tareas, eventos y plantillas sin crear todo manualmente en cada reinicio del entorno.

4.6 Registro de actividad y consistencia operativa

Más que añadir funciones visibles, una parte importante del trabajo fue asegurar que los cambios hechos por el usuario quedaran coherentes en todo el sistema. Cuando se crea, edita o mueve una tarea, el backend registra la operación en activity_logs y el frontend actualiza las vistas relacionadas. Esto evita la sensación de que cada módulo va por su lado.

El registro de actividad cumple una función sencilla pero útil: dejar huella de acciones como creación de proyectos, edición de tareas, cambios de estado y actualización de subtareas. En una herramienta académica esto ayuda a reconstruir el avance y también sirve para futuras fases de análisis. No reemplaza una auditoría formal, pero sí aporta una memoria básica del uso.

En la práctica, esta consistencia operativa fue clave para que el sistema se sintiera confiable. Una plataforma de organización pierde valor si el usuario duda de si el cambio quedó guardado o no. Por eso se cuidó que la persistencia en base de datos, la actualización visual y el registro interno mantengan la misma lógica.

4.7 Experiencia de usuario, feedback y calidad visual

La calidad de NovaTask no depende únicamente de su stack técnico. Una parte importante del trabajo estuvo en pequeños detalles de experiencia de usuario. Se añadieron skeleton loaders durante las cargas, mensajes breves para comunicar acciones, mensajes de error legibles, estados vacíos que orientan al usuario cuando todavía no hay información y confirmaciones visuales para acciones clave.

También se corrigieron varios puntos de fricción reportados durante el proceso: problemas en el registro cuando la pantalla quedaba congelada, tarjetas que se superponían, menús que quedaban detrás de otros paneles, textos con caracteres extraños y un tema claro poco agradable. Resolver estos aspectos fue esencial porque, en conjunto, son los que hacen que la aplicación se sienta estable y cuidada.

Esta parte del desarrollo deja una lección importante para la tesis: una interfaz no se vuelve profesional solo por verse moderna. También necesita comportarse bien, responder con claridad y mantenerse consistente en distintos escenarios de uso.

4.8 Diseño experimental propuesto

Para la fase de validación se plantea un diseño experimental con 40 estudiantes universitarios. La muestra se divide en dos grupos de 20. El grupo experimental trabajará con NovaTask durante cinco semanas, mientras que el grupo control continuará usando sus formas habituales de organización, como cuadernos, agendas físicas, notas del celular, hojas de cálculo o recordatorios personales. La intención de esta propuesta no es forzar un resultado, sino establecer una comparación razonable entre dos formas de organizar la carga académica.

La lógica de esta comparación es sencilla. Si NovaTask realmente aporta a la organización académica, el grupo experimental debería mostrar una percepción mayor de control, mejor seguimiento de entregas y menos sensación de desorden al finalizar las cinco semanas. El grupo control funciona como punto de referencia para no atribuir cualquier mejora simplemente al paso del tiempo o al entusiasmo inicial por participar en la prueba.

Tabla 9. *Distribución de la muestra simulada*

Grupo	Cantidad	Condición de trabajo	Duración
Experimental	20 estudiantes	Uso continuo de NovaTask para organizar materias, proyectos, tareas y agenda.	5 semanas
Control	20 estudiantes	Uso de métodos tradicionales de organización académica sin NovaTask.	5 semanas

Nota. Elaboración propia

Tabla 10. *Plan general de seguimiento por cinco semanas*

Semana	Grupo experimental	Grupo control	Evidencia a registrar
1	Inducción, creación de cuenta y carga inicial de	Levantamiento de la rutina actual con agenda, cuaderno, notas o Excel.	Encuesta inicial y descripción del método de organización.

	materias, proyectos y tareas.		
2	Uso diario de dashboard, proyectos y kanban.	Seguimiento habitual sin NovaTask.	Cantidad de tareas planificadas y entregas de la semana.
3	Uso combinado de calendario y vista de lista.	Organización tradicional con revisión semanal.	Puntualidad en entregas y sensación de orden.
4	Aplicación de plantillas y ajuste del perfil académico.	Continuidad del método tradicional.	Tiempo percibido para organizar la semana.
5	Cierre de uso, revisión de analíticas y encuesta final.	Encuesta final del grupo control.	Comparación final de percepción y cumplimiento.

Nota. Elaboración propia

4.9 Pruebas realizadas y validación funcional

Antes de pasar a una comparación con estudiantes, se consolidó una validación técnica sobre el flujo completo del usuario. Se verificó el registro e inicio de sesión, la creación de proyectos, la persistencia de tareas, el movimiento en kanban, la creación de subtareas, el uso del calendario, la aplicación de plantillas, la edición del perfil y la navegación entre pantallas. Además, se comprobó el build del frontend y la disponibilidad del backend a través de su ruta pública de salud.

A nivel de persistencia, también se revisó que la información quedara guardada correctamente en las tablas de PostgreSQL. Este punto fue sensible porque, durante el proceso, era necesario asegurarse de que los cambios no quedaran solo en memoria, sino efectivamente almacenados en la base de datos. El backend actual responde a esa necesidad.

Tabla 11. *Casos de prueba funcionales ejecutados*

Acción validada	Resultado esperado

Registro de usuario	La cuenta se crea, se guarda en users y el usuario puede iniciar sesión.
Inicio de sesión	Se emiten tokens válidos y el usuario accede al dashboard.
Creación de proyecto	El proyecto aparece en interfaz y se registra en projects.
Creación de tarea	La tarea queda visible en el proyecto y persistida en tasks.
Movimiento en kanban	La tarea cambia de columna y actualiza estado/posición en base.
Subtareas	Las subtareas se crean, editan y eliminan correctamente.
Calendario	Las tareas con fecha y los eventos se muestran en la vista temporal.
Plantillas	Aplicar una plantilla genera un proyecto con tareas base asociadas.
Tema visual	El usuario puede cambiar entre modo oscuro y claro sin romper layout.
Consistencia entre módulos	Al cerrar una tarea, el cambio se refleja en tablero, dashboard, analíticas y registro.
Despliegue	El frontend compila y el backend responde en /api/health.

Nota. Elaboración propia

4.10 Despliegue y publicación del sistema

El proyecto quedó conectado con GitHub como repositorio principal, lo que facilita el control de cambios, el respaldo y la publicación. A partir de ese repositorio se desplegó el backend en Render y el frontend en Vercel. Esta decisión fue muy conveniente porque ambos servicios ofrecen una integración simple para proyectos Node/React y permiten demostrar que la plataforma ya no depende exclusivamente del entorno local.

El backend publicado responde en la ruta de salud /api/health y el frontend se encuentra accesible desde una URL pública. Además, el repositorio conserva algunas configuraciones y archivos de apoyo para despliegue en Docker. Aunque no todos esos destinos son la ruta principal de publicación, sí muestran que la solución fue pensada con portabilidad y continuidad.

Este punto es relevante porque una buena parte del valor de NovaTask está en haber llegado a una versión realmente ejecutable. El despliegue no es un anexo opcional del proyecto: es una prueba de madurez técnica.

4.11 Dificultades encontradas y decisiones tomadas

El desarrollo no estuvo libre de dificultades. Uno de los primeros bloqueos apareció con la conexión local a PostgreSQL y la interacción con herramientas como pgAdmin. En ciertos momentos el usuario podía navegar por la interfaz, pero no tenía certeza de que los datos quedaran almacenados. Esto obligó a revisar variables de entorno, servicio local, rutas de conexión y validación real en base de datos.

Más adelante surgió un problema con Prisma en Windows: al ejecutar prisma generate apareció un error relacionado con el binario del motor de consulta. La solución implicó revisar procesos activos y regenerar el cliente en condiciones más limpias. Aunque fue un incidente técnico puntual, refleja un aprendizaje frecuente en proyectos reales: no todo el trabajo de ingeniería está en escribir módulos; también está en estabilizar el entorno.

Otra dificultad apareció en el comportamiento visual. Se reportaron repetidamente problemas como tarjetas que quedaban atrás, menús que se ocultaban detrás de otros bloques y un tema claro poco equilibrado. Estas observaciones llevaron a una ronda fuerte de pulido: revisión de z-index, sombras, fondos, overlays, tabs, badges, botones y superficies. Si se observa la historia del proyecto, esta etapa representa una profesionalización del producto ya existente.

También hubo que corregir detalles que parecían menores, pero afectaban bastante la experiencia: caracteres extraños en algunos textos, redirecciones que enviaban directo al dashboard, formularios que se quedaban pensando sin feedback claro y módulos que no refrescaban de inmediato después de guardar cambios. Resolver esto obligó a revisar manejo de estado, sesiones, normalización de textos y sincronización entre frontend y backend.

4.12 Estado actual y trabajo pendiente

En su estado actual, NovaTask puede considerarse una plataforma académica web funcional y suficientemente avanzada para iniciar una fase de validación con estudiantes. Tiene frontend y backend conectados, base de datos operativa, autenticación propia, módulos clave integrados y despliegue real. Esto le da un peso importante como trabajo de titulación porque demuestra implementación real y no solo intención de diseño.

Sin embargo, el proyecto todavía tiene espacio para crecer. El siguiente tramo debería concentrarse en pruebas con usuarios reales, retroalimentación sobre usabilidad, mejora de algunos textos y formularios, ampliación del módulo de analíticas y consolidación del esquema experimental planteado para los 40 estudiantes. También sería deseable fortalecer una estrategia formal de pruebas automáticas, algo que aún no se ha convertido en el foco principal del proyecto.

Aun con ese trabajo pendiente, la base actual ya es suficientemente consistente para sostener el desarrollo posterior y, al mismo tiempo, servir como evidencia técnica sólida dentro de la tesis.

CAPÍTULO V: CONCLUSIONES Y RECOMENDACIONES

5.1 Conclusiones

La primera conclusión es que el problema de organización académica sí puede abordarse de manera concreta mediante una plataforma web específica y no necesariamente con herramientas genéricas o improvisadas. NovaTask demuestra que, cuando el diseño parte de un contexto real de uso, es posible construir un producto más pertinente, incluso sin llegar todavía a una versión final cerrada.

En segundo lugar, la combinación tecnológica elegida resultó adecuada para el alcance. React, Vite y Tailwind facilitaron una interfaz modular y visualmente consistente; Node.js y Express permitieron estructurar una API clara; y PostgreSQL con Prisma ofrecieron una base relacional coherente para sostener proyectos, tareas, materias, eventos y trazabilidad. El stack no solo funcionó técnicamente, sino que también fue comprensible y mantenible a lo largo del proceso.

También se concluye que el tablero Kanban con arrastre, las plantillas reutilizables, las métricas visuales y la integración con el perfil académico fueron las más valiosas. Estas herramientas responden directamente a problemas reales como la pérdida de contexto y la dificultad para priorizar.

Se concluye que la experiencia de usuario es una parte crítica del resultado final. Varias de las iteraciones más importantes no consistieron en agregar funciones, sino en corregir comportamientos visuales, superposiciones, mensajes, estados vacíos y armonía del modo claro. Esto refuerza una idea central de la ingeniería de software aplicada: un producto útil no solo debe funcionar, también debe sentirse estable, claro y confiable.

Finalmente, el proyecto confirma que la tesis puede dejar un entregable que transciende el documento escrito. NovaTask ya cuenta con repositorio, despliegue público, backend activo y una base de datos diseñada para seguir creciendo.

5.2 Recomendaciones

En el plano funcional, sería recomendable ampliar la integración del calendario con servicios externos y profundizar la lectura del progreso semanal. Esto reforzaría la utilidad

cotidiana del sistema y permitiría que NovaTask acompañe mejor la planificación real del estudiante.

Desde la perspectiva académica, también sería valioso documentar la experiencia de uso con métricas simples: tiempo de aprendizaje, frecuencia de ingreso, percepción de utilidad, número de tareas completadas o claridad visual del dashboard. Esta información enriquecería mucho la fase experimental propuesta y permitiría argumentar con más fuerza el impacto del sistema.

Por último, se recomienda mantener la misma lógica que guio el proyecto hasta ahora: extender la solución sin romper su coherencia. NovaTask ha avanzado bien porque no intentó resolver todo a la vez. Conservar esa disciplina será clave para que las siguientes versiones sigan siendo útiles, defendibles y sostenibles.

ANEXOS

ANEXO A. Capturas web sugeridas para completar la tesis

Los espacios marcados a lo largo del documento permiten incorporar evidencia visual del proyecto. Se recomienda utilizar capturas recientes de la versión desplegada o del entorno local funcionando con datos reales. En la siguiente tabla se resumen solo las pantallas web que conviene insertar.

Tabla 12. *Registro sugerido de capturas web*

Elemento sugerido	Ubicación dentro del documento
Pantalla de login y registro	Capítulo IV, sección 4.3.1
Dashboard principal	Capítulo IV, sección 4.3.2
Vista general de proyectos	Capítulo IV, sección 4.3.3
Tablero kanban	Capítulo IV, sección 4.3.4
Vista de lista con filtros	Capítulo IV, sección 4.3.4
Calendario mensual	Capítulo IV, sección 4.3.5

Calendario semanal	Capítulo IV, sección 4.3.5
Módulo de analíticas	Capítulo IV, sección 4.3.6
Plantillas disponibles	Capítulo IV, sección 4.3.7
Perfil académico	Capítulo IV, sección 4.3.8
Menú de cuenta	Capítulo IV, sección 4.3.9
Tema claro	Capítulo IV, sección 4.3.10

Nota. Elaboración propia

ANEXO B. Resumen de entidades y su papel dentro del sistema

Tabla 13. *Entidades funcionales de NovaTask*

Entidad	Uso específico dentro de la solución
users	Define la cuenta principal, credenciales cifradas, nombre y avatar del usuario.
projects	Organiza el trabajo por contenedores amplios, vinculados opcionalmente con materias.
tasks	Representa la unidad operativa básica del trabajo, con prioridad, estado y fecha.
subtasks	Permite descomponer una tarea grande en pasos menores y medibles.
tags	Agrega clasificación libre por tema, contexto o tipo de actividad.
academic_terms	Sitúa el trabajo dentro de un periodo concreto del calendario académico.
subjects	Conecta el producto con la realidad universitaria del usuario.

events	Amplía el sistema más allá de tareas, incorporando clases, exámenes y reuniones.
activity_logs	Conserva trazabilidad de acciones internas del sistema.
refresh_tokens	Sostiene la persistencia de sesión y el flujo de refresh.
templates y template_tasks	Permiten generar proyectos nuevos a partir de estructuras predefinidas.

Nota. Elaboración propia

ANEXO C. Variables de entorno relevantes

Tabla 14. *Variables de entorno para operación del proyecto*

Variable	Finalidad	Ámbito
DATABASE_URL	Cadena de conexión a PostgreSQL.	Backend
PORT	Puerto de ejecución del servidor.	Backend
CLIENT_URL / CLIENT_URLS	Orígenes permitidos para CORS.	Backend
JWT_ACCESS_SECRET	Firma del token de acceso.	Backend
JWT_REFRESH_SECRET	Firma del refresh token.	Backend
VITE_API_URL	URL base del backend consumido por el frontend.	Frontend
VITE_DEMO_MODE	Permite trabajar con datos simulados si se requiere una fase demostrativa.	Frontend
VITE_ROUTER_MODE y VITE_APP_BASE_PATH	Ajustan el comportamiento de rutas según el despliegue.	Frontend

Nota. Elaboración propia

ANEXO D. Variables e indicadores para el grupo experimental y el grupo control

Tabla 15. *Matriz básica de seguimiento comparativo*

Dimensión	Cómo observarla	Qué se espera comparar
Cumplimiento	Relación entre tareas planificadas y tareas entregadas cada semana.	Si el grupo experimental sostiene mejor el cierre de tareas.
Puntualidad	Número de entregas realizadas dentro de la fecha prevista.	Si NovaTask ayuda a anticipar plazos con mayor claridad.
Percepción de orden	Encuesta breve al inicio y al final de la prueba.	Si el estudiante siente mayor control sobre su carga académica.
Tiempo de planificación	Minutos aproximados invertidos en organizar la semana.	Si la plataforma reduce el esfuerzo inicial de organizarse.
Uso del sistema o método	Frecuencia semanal de revisión del tablero, agenda o método tradicional.	Si el hábito de organización se vuelve más constante.

Nota. Elaboración propia

ANEXO E. Checklist sugerido para la siguiente fase

- Ejecutar pruebas con estudiantes reales y registrar hallazgos de uso.
- Medir claridad del dashboard y efectividad de la vista calendario.
- Revisar consistencia de formularios y validaciones en móvil.
- Evaluar nuevas plantillas según carreras o tipos de entregable.
- Definir una estrategia de pruebas automatizadas para backend y frontend.
- Revisar si conviene integrar calendarios externos en una etapa posterior.
- Profundizar la fase teórica final con resultados de uso, discusión y análisis comparativo.

ANEXO F. Repositorio del sistema en GitHub

Enlace: https://github.com/Patricio-Ona/NovaTask

ANEXO G. Sistema desplegado en Vercel

Enlace: https://nova-task-eight.vercel.app/

BIBLIOGRAFÍAS

Home | Scrum Guides. (s. f.). https://scrumguides.org/

Installing with Vite - Installation. (s. f.). Tailwind CSS. https://tailwindcss.com/docs/installation/using-vite

Jones, M., Bradley, J., & Sakimura, N. (2015). *JSON Web Token (JWT)*. https://doi.org/10.17487/rfc7519

Node.js — Run JavaScript everywhere. (s. f.). https://nodejs.org/es

Prisma Data, Inc. (s. f.). *Get started with Prisma*. Prisma Documentation. https://www.prisma.io/docs

React. (s. f.). https://react.dev/

Tailwind CSS - Rapidly build modern websites without ever leaving your HTML. (s. f.). Tailwind CSS. https://tailwindcss.com/

Vercel. (2026, 26 mayo). *Vercel documentation*. Vercel Documentation. https://vercel.com/docs

Vite. (s. f.). Vitejs. https://vite.dev/