**UNIVERSIDAD CONTINENTAL**

**FACULTAD DE INGENIERÍA**

**ESCUELA ACADÉMICO PROFESIONAL DE INGENIERÍA DE SISTEMAS E INFORMÁTICA**



**PROYECTO**

**“Implementación de una plataforma de Gestión de Proyectos Colaborativos con Integración de Inteligencia Artificial SMARTASK”**

# PORTADA

**PRESENTADO POR:**

| **APELLIDOS Y NOMBRES** | **CÓDIGO** |
| --- | --- |
| **Hernández Calzada Laura Patricia** | **72276440** |
| **Picon Mendoza Dany** | **75497467** |
| **Picon Mendoza Juan** | **75988801** |
| **Gonzales Orihuela Johonny** | **78104318** |
| **Ramos Canchomania Malher** | **74543214** |

**ASESOR:**

**DANIEL GAMARRA MORENO**

**HUANCAYO – PERÚ**

**2025**

# LISTA DE CONTENIDO

[PORTADA 1](#_heading=h.kg0z6rx9fi7o)

[LISTA DE CONTENIDO 2](#_heading=h.ged06jym0keq)

[LISTA DE TABLAS 7](#_heading=h.5zf4n2mozxl4)

[LISTA DE FIGURAS 8](#_heading=h.pq5gffc5azqo)

[CAPÍTULO 1 9](#_heading=h.86wqyen83z5h)

[PLANTEAMIENTO DEL ESTUDIO 9](#_heading=h.l0uang4twfa)

[1.1.](#_heading=h.u1ih3f698ldo) Aspectos Generales de la Empresa 9

[1.1.1.](#_heading=h.glt29dz2z0a) Organigrama 9

[1.1.2.](#_heading=h.gzjbv2r929r9) Misión y visión 9

[1.2.](#_heading=h.rpcuoxwu3g0d) Diagnóstico del Problema 9

[1.3.](#_heading=h.atrmiso5dt0g) Procesos de la Empresa 9

[1.4.](#_heading=h.ickk4hqfs1gd) Oportunidad Encontrada 9

[1.5.](#_heading=h.qpgmuga9ewwq) Detalles del Proyecto 9

[1.5.1.](#_heading=h.bbhqor39jwbh) Solución planteada 9

[1.5.2.](#_heading=h.6uv6z0o5l1cg) Objetivos generales 10

[CAPÍTULO 2 11](#_heading=h.bnk9wr2umt5i)

[ESTUDIO DE FACTIBILIDAD 11](#_heading=h.s8entmebdr7h)

[2.1.](#_heading=h.9kgx60m8bc4q) Alternativas de Solución 11

[2.2.](#_heading=h.b7rkxdb3k07p) Factibilidad Técnica 11

[2.2.1.](#_heading=h.tizl3zyhcrrq) Hardware: Servidor 11

[2.3.](#_heading=h.9q27hjw1xjrz) Factibilidad Económica 12

[2.3.1.](#_heading=h.3j2gknucicxv) Gastos generales 12

[2.4.](#_heading=h.o75hm8dymphs) Factibilidad Operacional 12

[2.4.1.](#_heading=h.ud5ovpz40h1m) Actividad del sistema de ventas 13

[2.4.1.1.](#_heading=h.dargxmjb2lf) Fase de análisis. 13

[2.4.1.2.](#_heading=h.ix6xdclegcy4) Fase de diseño y desarrollo. 13

[2.4.1.3.](#_heading=h.ugflbm8fg3x7) Fase de transición. 14

[2.4.1.4.](#_heading=h.m0a06y5jg254) Fase de implementación. 14

[CAPÍTULO 3 15](#_heading=h.g0leip1e02u7)

[ANÁLISIS DE REQUERIMIENTOS 15](#_heading=h.d5v93s9ld95i)

[3.1.](#_heading=h.f2qmvsz6lt6v) Metas del Sistema de Información 15

[3.2.](#_heading=h.q6g207tk9lx) Requisitos del Sistema 15

[3.2.1.](#_heading=h.p5aln8s2ci88) Requerimientos funcionales 15

[3.2.2.](#_heading=h.6gzw6pdck62c) Requerimientos no funcionales 16

[3.3.](#_heading=h.a3zpxetwnqyj) Identificación de Actores del Sistema 17

[3.3.1.](#_heading=h.kxlaeuakyp1e) Administrador 17

[3.3.2.](#_heading=h.62vo1ohd2b2g) Usuario 17

[CAPÍTULO 4 18](#_heading=h.hz3p5j1q3fgl)

[PLANIFICACIÓN DEL PROYECTO 18](#_heading=h.amuazf6ycxe7)

[4.1.](#_heading=h.she0rtk6ssht) Definición de Roles de Trabajo 18

[4.1.1.](#_heading=h.se39tfrpli2f) Product owner 18

[4.1.2.](#_heading=h.mtqf0s3ovr9u) Scrum master 18

[4.1.3.](#_heading=h.5ic7c1ljbdol) Team member 18

[4.1.4.](#_heading=h.ywai5cpgfcdg) Tester 18

[4.2.](#_heading=h.g5zy7iion32h) Product Backlog 19

[4.3.](#_heading=h.jynk769krq5l) Sprint Backlog 20

[4.3.1.](#_heading=h.hhjqizdqb1j1) Sprint 1 20

[4.3.2.](#_heading=h.wco9ul4p01j) Sprint 2 21

[4.3.3.](#_heading=h.rox1fwc7wdir) Sprint 3 21

[4.3.4.](#_heading=h.x9tihvc8jyjj) Sprint 4 22

[4.4.](#_heading=h.mxduv4ko8yb4) Planificación de Sprints 22

[4.4.1.](#_heading=h.1f6xukldezxy) Historias de usuario 22

[4.5.](#_heading=h.fu9e1rwm0739) Cronograma de Actividades 24

[4.6.](#_heading=h.bwj1gi4630q6) Gestión de Riesgos 25

[CAPÍTULO 5 26](#_heading=h.h8zct2mtcjum)

[DISEÑO DEL SISTEMA DE INFORMACIÓN 26](#_heading=h.i7qyoz3htxcq)

[5.1.](#_heading=h.uigox6xn36hz) Diseño de Diagramas UML 26

[5.1.1.](#_heading=h.xmj4vwhv4t7w) Diagramas de casos de uso 26

[5.1.2.](#_heading=h.70lvhas6yjcx) Diagramas de secuencia 26

[5.1.3.](#_heading=h.u9c7lfwhfvux) Diagramas de colaboración 26

[5.1.4.](#_heading=h.r4lo4hbfbd8p) Diagramas de clases 26

[5.2.](#_heading=h.4uppog88zue) Diseño de Base de Datos 26

[5.2.1.](#_heading=h.yj61x94fiony) Diseño conceptual (E/R) 26

[5.2.2.](#_heading=h.uxryuk74dtnf) Diseño lógico 26

[5.2.3.](#_heading=h.twc2fgf2nxte) Diseño físico 26

[5.2.4.](#_heading=h.1fnfc1gdmpcv) Modelado de base de datos 26

[5.3.](#_heading=h.7rfmfcyttt9) Diseño de Interfaces Básicas 26

[5.3.1.](#_heading=h.7b7l16tcv1m) Acceso login 27

[5.3.2.](#_heading=h.lme024fhdu5q) Interfaz … 27

[**CAPÍTULO 6** 28](#_heading=h.qs2hcrjvhiq2)

[CODIFICACIÓN DEL SOFTWARE 28](#_heading=h.mipbhuqkzstx)

[6.1.](#_heading=h.oxn4irvsyum0) Desarrollo del Sprint 1 28

[6.1.1.](#_heading=h.kf18ylhm42r) Sprint planning 28

[6.1.2.](#_heading=h.aczyrigecshz) Sprint backlog 28

[6.1.3.](#_heading=h.7an8hoe85ze2) Historias de usuarios 28

[6.1.4.](#_heading=h.uowler9s8wqy) Taskboard 28

[6.1.5.](#_heading=h.4vp9yalf9vhk) Daily scrum 28

[6.1.6.](#_heading=h.btchtm35k6qb) Sprint review 28

[6.1.7.](#_heading=h.6qrdvygoxgfh) Criterios de aceptación 28

[6.1.8.](#_heading=h.z2a6opm25p92) Resultados del sprint 28

[6.1.8.1.](#_heading=h.v8v3sxjici6o) Evidencias. 28

[6.1.8.2.](#_heading=h.crk0c7lb2v0q) Prueba de desarrollo. 28

[6.1.8.3.](#_heading=h.z5wby6ir6f98) …. 29

[6.1.9.](#_heading=h.g03dhg8nwvav) Sprint retrospective 29

[6.2.](#_heading=h.b2srnqfh0xy6) Desarrollo del Sprint 2 29

[6.2.1.](#_heading=h.rwm63do9p195) Sprint planning 29

[6.2.2.](#_heading=h.2wdxte1a06xz) Sprint backlog 29

[6.2.3.](#_heading=h.okrx7vi5yksl) Historias de usuarios 29

[6.2.4.](#_heading=h.bz6703y68t45) Taskboard 29

[6.2.5.](#_heading=h.wpuwgxlscl1i) Daily scrum 29

[6.2.6.](#_heading=h.ebn5f24jhqui) Sprint review 29

[6.2.7.](#_heading=h.8fge1n3240hq) Criterios de aceptación 29

[6.2.8.](#_heading=h.yuywc98f1rdu) Resultados del sprint 29

[6.2.8.1.](#_heading=h.uxe9yvve36nx) Evidencias. 29

[6.2.8.2.](#_heading=h.ux5oqkk9ows2) Prueba de desarrollo. 30

[6.2.8.3.](#_heading=h.sncrejjbyxob) …. 30

[6.2.9.](#_heading=h.5zs761s3djb6) Sprint retrospective 30

[6.3.](#_heading=h.pftfaiy5x3qj) Desarrollo del Sprint 3 30

[6.3.1.](#_heading=h.wrt7k5scypl0) Sprint planning 30

[6.3.2.](#_heading=h.ygzy3iqho7ge) Sprint backlog 30

[6.3.3.](#_heading=h.kjy6mrimnit0) Historias de usuarios 30

[6.3.4.](#_heading=h.kixjtgxz355v) Taskboard 30

[6.3.5.](#_heading=h.9bp47j9v9kiz) Daily scrum 30

[6.3.6.](#_heading=h.2tf7f44oi0bm) Sprint review 30

[6.3.7.](#_heading=h.uzbyaimo95ft) Criterios de aceptación 30

[6.3.8.](#_heading=h.ec8wn98mm180) Resultados del sprint 30

[6.3.8.1.](#_heading=h.747sigb3hupx) Evidencias. 31

[6.3.8.2.](#_heading=h.fhql5niv0xg4) Prueba de desarrollo. 31

[6.3.8.3.](#_heading=h.t344q0ns7tt1) …. 31

[6.3.9.](#_heading=h.kqkp085zmymf) Sprint retrospective 31

[6.4.](#_heading=h.9akub8kmmajm) Desarrollo del Sprint 4 31

[6.4.1.](#_heading=h.xdum5u3mcjd7) Sprint planning 31

[6.4.2.](#_heading=h.3ysa3alp894p) Sprint backlog 31

[6.4.3.](#_heading=h.q7lqamhbj3d8) Historias de usuarios 31

[6.4.4.](#_heading=h.ydz996rkn1xo) Taskboard 31

[6.4.5.](#_heading=h.non5xq3mo1mk) Daily scrum 31

[6.4.6.](#_heading=h.w4tmgjtf7tfc) Sprint review 31

[6.4.7.](#_heading=h.9p8wogqk5nor) Criterios de aceptación 31

[6.4.8.](#_heading=h.xviva083trql) Resultados del sprint 32

[6.4.8.1.](#_heading=h.44dl90rf0vec) Evidencias. 32

[6.4.8.2.](#_heading=h.crvwzww7pvko) Prueba de desarrollo. 32

[6.4.8.3.](#_heading=h.jvzd9uv94zbq) …. 32

[6.4.9.](#_heading=h.t903oasncuq8) Sprint retrospective 32

[CONCLUSIONES 33](#_heading=h.38bjsgu9h5s9)

[RECOMENDACIONES 34](#_heading=h.q1rd40giek16)

[ANEXOS 35](#_heading=h.tal36cnbj31m)

[Anexo 01. Manual Técnico 36](#_heading=h.eurbyef068w4)

[Anexo 02. Manual de Usuario 37](#_heading=h.flezqvfx5h7k)

# LISTA DE TABLAS

[Tabla 1. Modelo de CPU - Servidor. 11](#_heading=h.8wswq7t6xtdi)

[Tabla 2. Gastos generales presupuestados. 12](#_heading=h.bijcqf9emgc2)

[Tabla 3. Requerimientos funcionales del sistema 1](#_heading=h.bijcqf9emgc2)9

[Tabla 4. Requerimientos no funcionales del sistema](#_heading=h.bijcqf9emgc2) 20

# LISTA DE FIGURAS

[Figura 1. Recursos presupuestados para el proyecto. 12](#_heading=h.s9bs01s1jy33)

[Figura 2. Product Backlog 21](#_heading=h.s9bs01s1jy33)

[Figura 3. Sprint Backlog. 22](#_heading=h.s9bs01s1jy33)

[Figura 4. Planificación de sprints. 24](#_heading=h.s9bs01s1jy33)

[Figura 5. Cronograma de actividades. 27](#_heading=h.s9bs01s1jy33)

[Figura 6. Gestión de riesgos. 28](#_heading=h.s9bs01s1jy33)

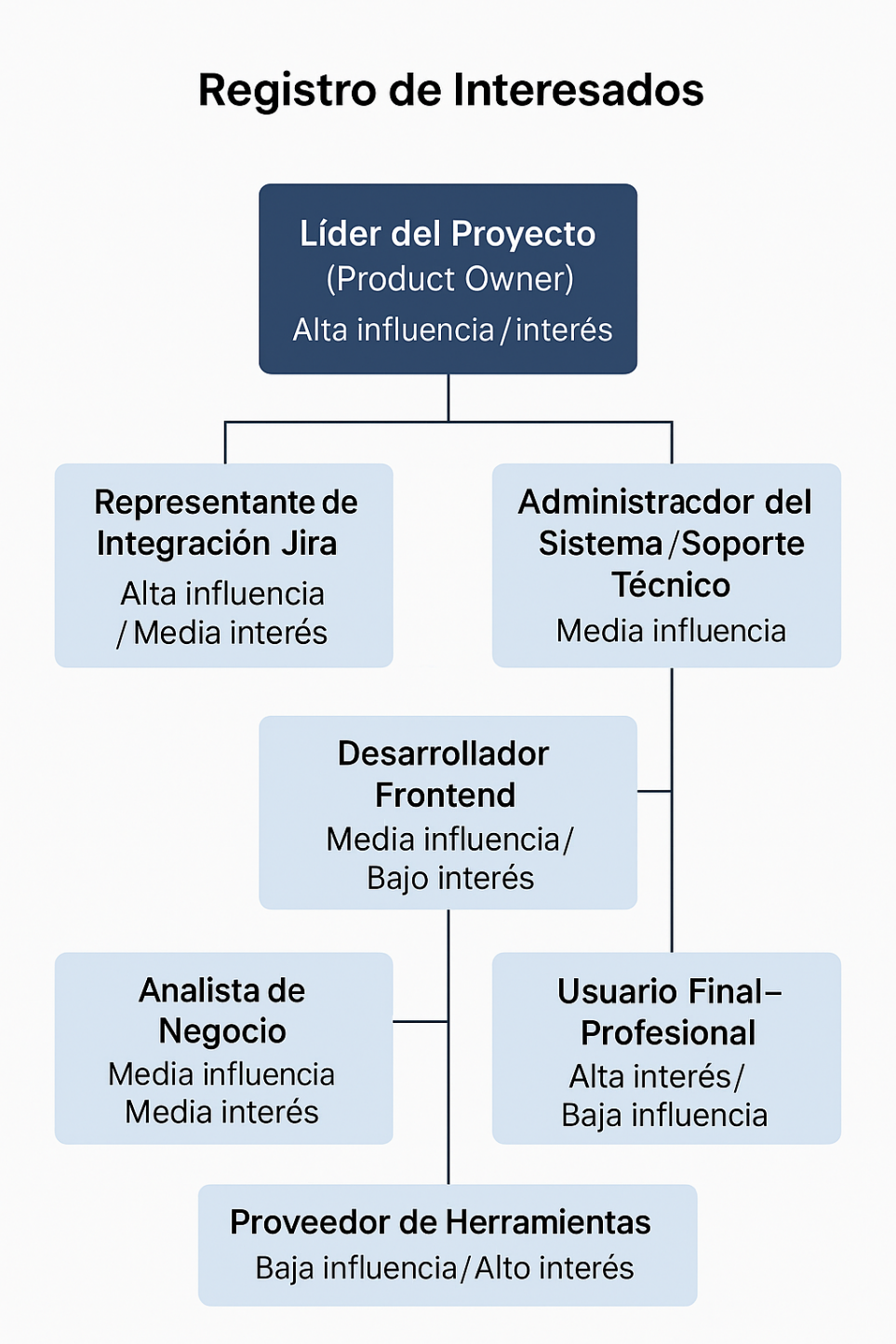
# CAPÍTULO 1

# PLANTEAMIENTO DEL ESTUDIO

## Aspectos Generales de la Empresa

SmartTask es una plataforma web diseñada para ayudar a individuos y equipos a organizar, priorizar y realizar seguimiento de tareas. A través de la inteligencia artificial, se asignan prioridades automáticamente, se generan recordatorios por correo electrónico y se permite la integración con plataformas externas como Jira. Su objetivo es mejorar la productividad y reducir los olvidos y repetir los trabajos en la gestión de actividades cotidianas y de proyectos.

### Organigrama



### Misión y visión

**Misión:** Proveer una solución tecnológica inteligente para la gestión de tareas y proyectos, que impulse la productividad personal y colaborativa mediante herramientas automatizadas y fáciles de usar.

**Visión:** Convertirse en la plataforma de gestión de tareas líder en Latinoamérica, reconocida por su eficiencia, automatización y capacidad de integración.

## Diagnóstico del Problema

En los entornos personales y organizacionales, la falta de herramientas integradas para la gestión de tareas deriva en una baja eficiencia, olvido de actividades, tareas duplicadas o mal priorizadas. El uso de múltiples herramientas desconectadas complica el seguimiento y la coordinación. Esta situación ha motivado el desarrollo de SmartTask, con el fin de consolidar funcionalidades críticas en una sola plataforma.

## Procesos de la Empresa

SmartTask contempla los siguientes procesos:

* Gestión y edición de tareas mediante formularios web (React).
* Priorización automática basada en fechas límite (algoritmos de IA).
* Envío de recordatorios programados por correo electrónico (Nodemailer + Cron Jobs).
* Exportación de tareas a Jira como historias de usuario.
* Autenticación y autorización mediante tokens JWT.
* Almacenamiento y consulta de datos mediante MongoDB.

## Oportunidad Encontrada

El crecimiento de equipos distribuidos y el teletrabajo ha incrementado la necesidad de herramientas flexibles, colaborativas y automatizadas. *SmartTask* surge como una alternativa innovadora al integrar IA, notificaciones y conectividad con Jira, permitiendo a los usuarios una mejor organización de sus tareas.

## Detalles del Proyecto

### Solución planteada

La solución planteada consiste en el desarrollo de *SmartTask*, una plataforma web modular, inteligente y segura para la gestión de tareas personales y colaborativas. Esta solución se caracteriza por integrar funciones clave como:

* **Gestión de tareas** (crear, editar, listar y eliminar tareas) con una interfaz intuitiva y accesible desde cualquier navegador moderno.
* **Priorización automática con IA**, que evalúa la proximidad de las fechas límite para clasificar tareas como “Urgente”, “Prioritaria” o “Normal”.
* **Recordatorios automáticos por correo electrónico**, generados por procesos cron y enviados mediante Nodemailer cuando las tareas están por vencer.
* **Integración con Jira**, permitiendo exportar tareas como historias de usuario para sincronizar con flujos de trabajo de desarrollo ágil.
* **Seguridad básica mediante JWT**, que protege las rutas del backend y asegura que solo usuarios autenticados accedan a información sensible.
* **Escalabilidad y mantenimiento**, gracias a su arquitectura cliente-servidor y al uso de tecnologías modernas como React, Node.js y MongoDB.

Esta solución se desarrolla de forma incremental a través de Productos Mínimos Viables (PMV), permitiendo validar y mejorar cada funcionalidad con retroalimentación temprana de los usuarios.

### Objetivos generales

#### **Objetivo General**

Desarrollar e implementar una plataforma web inteligente de gestión de tareas que permita a los usuarios organizar, priorizar y hacer seguimiento de sus actividades, mediante inteligencia artificial, recordatorios automatizados e integración con sistemas externos, contribuyendo a una mejor productividad y organización personal o grupal.

#### **Objetivos Específicos**

* Diseñar una interfaz amigable y funcional para la creación, edición y visualización de tareas.
* Implementar un algoritmo de priorización automática de tareas basado en fechas límite.
* Desarrollar un sistema de notificaciones automáticas por correo electrónico para tareas próximas a vencer.
* Integrar la plataforma con Jira para exportar tareas como historias de usuario.
* Asegurar la autenticación de usuarios mediante tokens JWT y proteger las rutas privadas del sistema.
* Validar la funcionalidad del sistema mediante pruebas con usuarios reales y despliegue en la nube.

# CAPÍTULO 2

# ESTUDIO DE FACTIBILIDAD

## Alternativas de Solución

Se evaluaron dos caminos principales:

1. **Usar plataformas existentes** como Trello, Notion o Todoist. Aunque funcionales, presentan límites significativos en personalización, inteligencia de priorización e integración directa con herramientas de desarrollo como Jira.
2. **Desarrollar una solución propia (SmartTask)** con las siguientes ventajas:
   * Total flexibilidad y adaptabilidad del sistema a las necesidades reales de los usuarios.
   * Inclusión de algoritmos de IA y lógica de negocio personalizada.
   * Potencial de escalar, modularizar y agregar nuevas funcionalidades sin depender de terceros.

La segunda opción fue seleccionada por su viabilidad a largo plazo y alineamiento con los objetivos del proyecto.

## Factibilidad Técnica

El equipo de desarrollo cuenta con experiencia comprobada en las siguientes tecnologías:

* **Frontend:** React JS para componentes dinámicos e interfaz responsive.
* **Backend:** Node.js con Express para el manejo de rutas, autenticación y lógica del servidor.
* **Base de datos:** MongoDB Atlas, con modelo definido para tareas, usuarios y subtareas.
* **Infraestructura:** Vercel y Heroku para despliegue rápido y gratuito en entornos de pruebas.
* **Automatización:** Cron jobs para tareas programadas (recordatorios) y Nodemailer para el envío de correos.
* **Seguridad:** Implementación de JWT en endpoints críticos y middleware de autenticación.

Los diagramas de base de datos y arquitectura demuestran una organización clara del modelo de información y flujos entre cliente, servidor y servicios externos (correo, Jira API).

### Hardware: Servidor

El CPU que servirá como servidor también, presenta características de Inter Core i7-1820K de 12 núcleos y 24 hilos de ejecución; 128MB de caché, frecuencia base de 3.3Ghz y máxima con turbo de 3.6GHz, capacidad de memoria DDR4 normal, 28 LANES PCI Express, un TDP de 140W y link Speedy de 8GT/s en los PCI Express.

*Tabla 1. Modelo de CPU - Servidor.*

|  | **Características** | **Valores** |
| --- | --- | --- |
| **1** | Número de núcleos | 12 |
| **2** | Memoria caché | 128 MB |
| **3** | Frecuencia mínima | 1.2GHz |
| **4** | Frecuencia base máxima turbo | 3.3GHz |
| **5** | RAM compatible | DDR4 |
| **6** | Memoria ECC compatible | No |
| **7** | Número de PCI | 1x16 + 1x8 + 1x4 |
| **8** | Máximo PCI Express | 8GT/s |
| **9** | TDP | 140w |
| **10** | Disco Duro | 1TB |
| **11** | Tarjeta de red | 1300Mbps – 2.4G |
| **12** | Tarjeta de video | 16GB GDDR5 |

*Fuente. Elaboración propia.*

## Factibilidad Económica

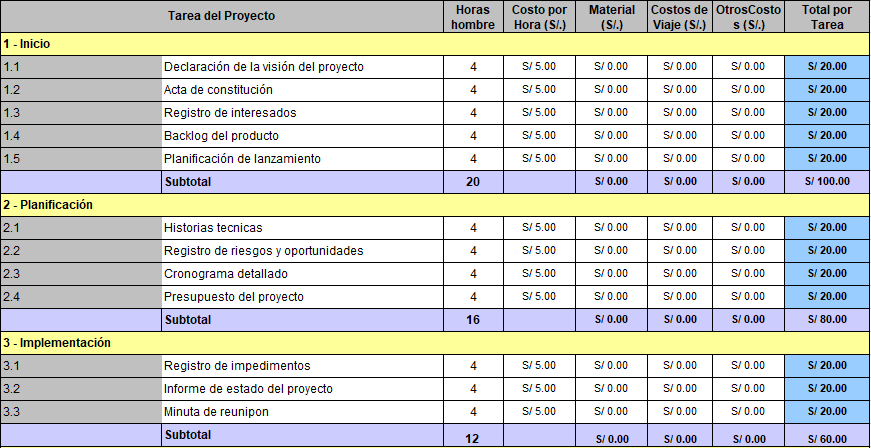
Gracias al uso de herramientas gratuitas y open-source, los costos de infraestructura han sido reducidos. Los recursos principales utilizados incluyen:

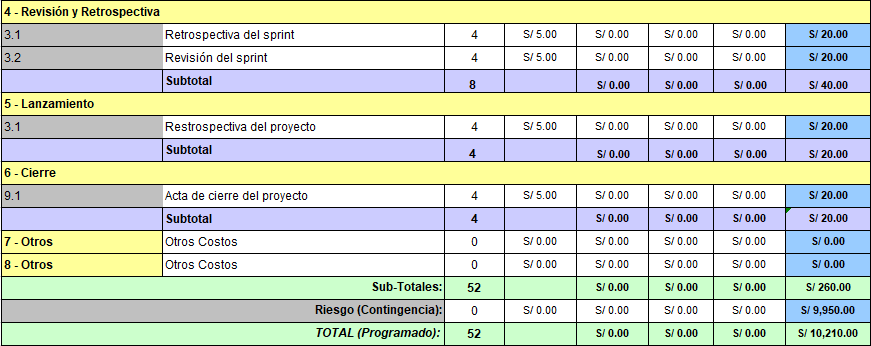
* GitHub (control de versiones)
* MongoDB Atlas (base de datos en la nube)
* Vercel y Heroku (hosting gratuito)
* Nodemailer (servicio SMTP gratuito)

A la fecha, se ha ejecutado un 70% del presupuesto proyectado, manteniéndose dentro del rango esperado sin desviaciones críticas. Esto permite avanzar hacia PMV3 y PMV4 con recursos controlad

### Gastos generales

*Tabla 2. Gastos generales presupuestados.*





*Fuente. Elaboración propia.*

## Factibilidad Operacional

El sistema ha sido probado con usuarios reales en entornos locales y de pruebas. Las observaciones iniciales reportan:

* Interfaz simple y rápida de aprender.
* Flujo lógico en la creación y consulta de tareas.
* Priorización útil y acertada.
* Recordatorios funcionales, aunque con necesidad de mejora en confiabilidad (en proceso).

En resumen, el sistema es operable desde cualquier dispositivo con navegador moderno, no requiere instalación y está diseñado para una adopción rápida sin curva de aprendizaje elevada.

### Sistema de ventas

#### Fase de análisis.

Durante esta fase se identificaron los siguientes puntos críticos para el despliegue y adopción del sistema:

* Público objetivo principal: estudiantes universitarios, docentes investigadores y equipos técnicos de desarrollo.
* Necesidades del usuario: gestión de tareas organizada, visualización de prioridades, recordatorios efectivos, facilidad de uso.
* Expectativas: acceso multiplataforma, interfaz clara, integración con herramientas ya conocidas (como Jira).
* Condiciones de uso: navegador web moderno, conexión a internet estable, cuentas personales seguras.

#### Fase de diseño y desarrollo.

En esta fase se definieron y construyeron los componentes que permitirán escalar y entregar la solución:

* Diseño UI/UX: Interfaz pensada para reducir la curva de aprendizaje y aumentar la interacción frecuente.
* Gestión de usuarios: Registro/login con autenticación JWT y estructura de usuarios en la base de datos.
* Automatización de comunicaciones: Recordatorios por correo electrónico generados automáticamente por el sistema.
* Integración con herramientas externas: API de Jira incorporada para facilitar la adopción por parte de equipos SCRUM.
* Modelo de base de datos: Escalable, con colecciones para tareas, usuarios, subtareas y configuraciones.
* Durante esta fase también se desarrollaron pruebas de aceptación, seguridad y validación de criterios funcionales definidos en las HU del PMV1 y PMV2.

#### Fase de transición.

Esta fase está orientada a preparar al usuario final para la adopción y correcto uso del sistema:

* Despliegue en la nube: Utilización de Vercel para el frontend y Heroku para el backend, accesibles desde cualquier navegador.
* Pruebas piloto: Se invitó a un grupo de usuarios a interactuar con el sistema en versión de pruebas para recopilar feedback.
* Documentación básica: Se creó una guía de usuario (PDF) y videotutoriales que explican el uso de las funciones principales.
* Soporte inicial: Canal de retroalimentación mediante formularios o contacto directo con el equipo de desarrollo.

El objetivo de esta fase es garantizar que los usuarios comprendan el valor de la herramienta, detecten errores menores y generen confianza en el producto.

#### Fase de implementación.

La plataforma se implementará en un entorno operativo real, con las siguientes acciones:

* Acceso público a la URL del sistema: Se asignará dominio para acceso libre por parte de los usuarios objetivo.
* Soporte post-despliegue: Mantenimiento inicial por parte del equipo desarrollador para resolver problemas urgentes.
* Evaluación del uso: Se recolectarán métricas sobre el uso de la plataforma (tareas creadas, sesiones activas, tiempos de respuesta).
* Retroalimentación continua: A través de formularios o encuestas, se registrarán sugerencias y mejoras futuras.

Esta fase completa el ciclo de ventas mediante la entrega funcional del sistema, la validación en campo y la recolección de lecciones aprendidas para iteraciones posteriores (PMV3 y PMV4).

# CAPÍTULO 3

# ANÁLISIS DE REQUERIMIENTOS

## Metas del Sistema de Información

El sistema de información SMARTASK tiene como meta principal optimizar la gestión de tareas personales y colaborativas mediante una plataforma web que combine automatización, priorización inteligente y conectividad con herramientas externas como Jira. Se busca facilitar la organización del trabajo y mejorar la productividad de los usuarios.

## Requisitos del Sistema

### Requerimientos funcionales

*Tabla 3. Requerimientos funcionales del sistema*

| **Código** | **Nombre** | **Descripción** |
| --- | --- | --- |
| **RF01** | Gestión de Tareas | Crear, editar, listar y eliminar tareas. |
| **RF02** | Priorización Inteligente de Tareas | Priorizar tareas automáticamente según la fecha límite. |
| **RF03** | Recordatorios Automatizados | Enviar recordatorios automáticos por correo. |
| **RF04** | Integración con Jira | Exportar tareas a Jira como historias de usuario. |
| **RF05** | Autenticación y Autorización de Usuarios | Autenticación y autorización de usuarios mediante JWT. |
| **RF06** | Validación de Formulario | Validación de campos obligatorios en formularios. |
| **RF07** | Gestión de Subtareas | División de tareas en subtareas. |

### Requerimientos no funcionales

*Tabla 4. Requerimientos no funcionales del sistema*

| **Código** | **Nombre** | **Descripción** |
| --- | --- | --- |
| **RNF01** | Accesibilidad Multinavegador | Accesibilidad desde navegadores modernos. |
| **RNF02** | Escalabilidad del Sistema | Escalabilidad del sistema. |
| **RNF03** | Disponibilidad en la Nube | Alta disponibilidad en la nube (usando Vercel/Heroku). |
| **RNF04** | Seguridad de Autenticación | Seguridad mediante JWT y middleware de autenticación. |
| **RNF05** | Interfaz Intuitiva | Facilidad de uso con una interfaz UI/UX intuitiva. |
| **RNF06** | Compatibilidad con Herramientas Externas | Compatibilidad con plataformas externas como Jira. |

## Identificación de Actores del Sistema

### Administrador

Encargado de gestionar el acceso de usuarios, monitorear el rendimiento de la plataforma y asegurar el cumplimiento de los parámetros operativos. Tiene acceso completo al sistema.

### Usuario

Persona que hace uso de la plataforma para gestionar tareas. Puede crear, editar, eliminar y priorizar tareas. También recibe notificaciones por correo y puede exportar tareas a Jira.

# CAPÍTULO 4

# PLANIFICACIÓN DEL PROYECTO

## Definición de Roles de Trabajo

### Product owner

Responsable de representar los intereses del cliente, definir los requerimientos del producto, y priorizar el Product Backlog.

### Scrum master

Facilita el proceso Scrum, asegura el cumplimiento de sus principios y elimina obstáculos que impidan el avance del equipo.

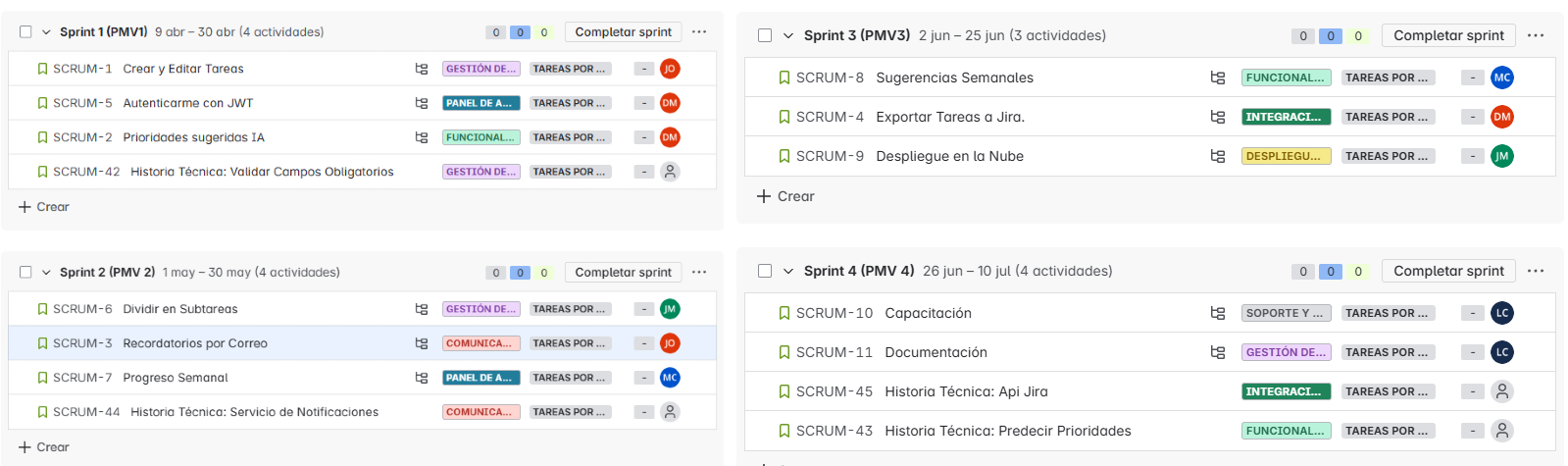
### Team member

Encargado de ejecutar las tareas de desarrollo, diseño, pruebas e integración del sistema. Participa activamente en las ceremonias Scrum.

### Tester

Responsable de ejecutar las pruebas funcionales y no funcionales, reportar errores y validar que el sistema cumpla con los criterios de aceptación definidos.

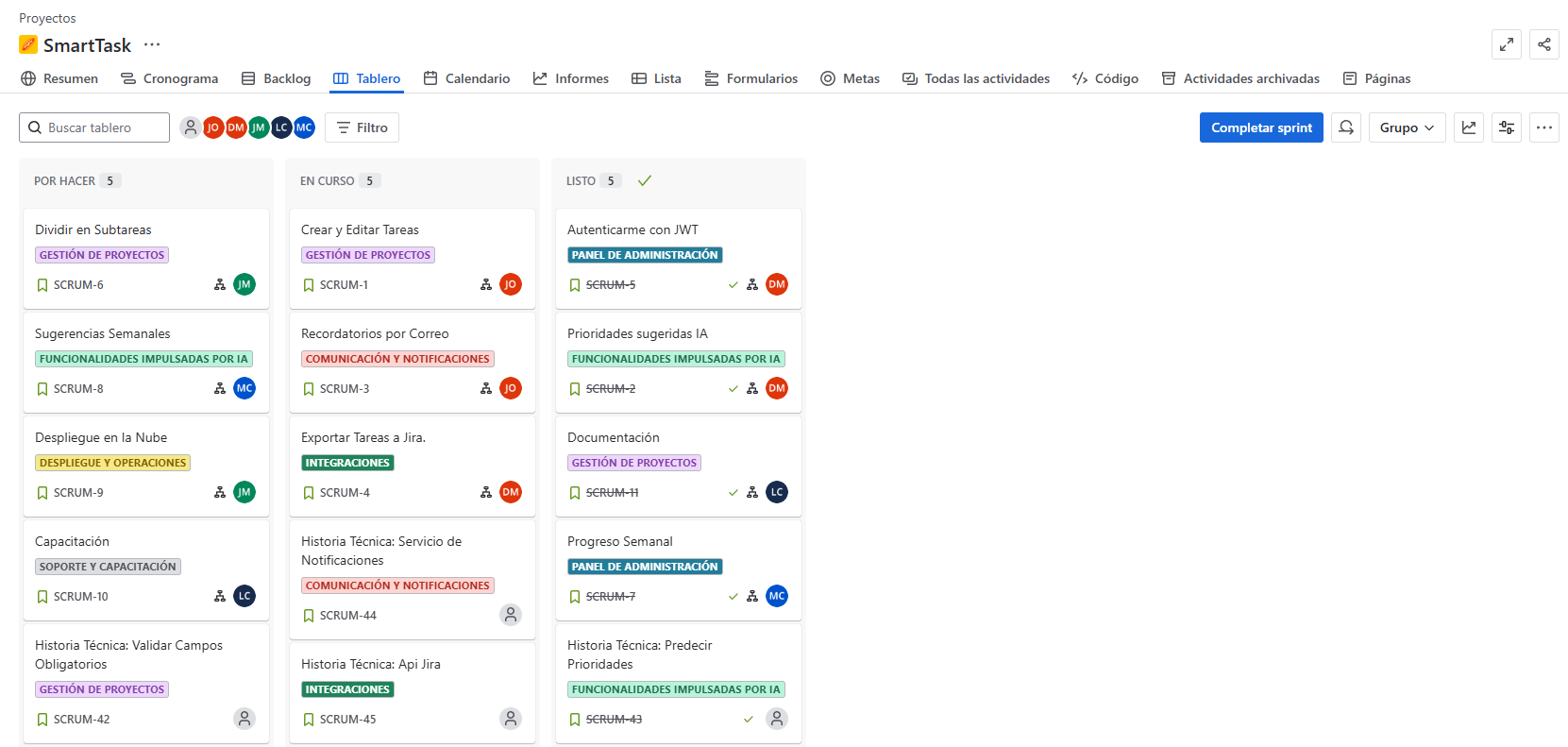
## Product Backlog

*Figura 2. Product Backlog*

[**https://continental-team-lhodxug5.atlassian.net/jira/software/projects/SCRUM/boards/1/backlog?epics=visible&selectedIssue=SCRUM-3&atlOrigin=eyJpIjoiMjg3Yzk0YzliMzhiNDIzOGI3MmQ4YjA3YjQyODg3NzciLCJwIjoiaiJ9**](https://continental-team-lhodxug5.atlassian.net/jira/software/projects/SCRUM/boards/1/backlog?epics=visible&selectedIssue=SCRUM-3&atlOrigin=eyJpIjoiMjg3Yzk0YzliMzhiNDIzOGI3MmQ4YjA3YjQyODg3NzciLCJwIjoiaiJ9)

## Sprint Backlog

*Figura 3. Sprint Backlog*



[**https://continental-team-lhodxug5.atlassian.net/jira/software/projects/SCRUM/boards/1**](https://continental-team-lhodxug5.atlassian.net/jira/software/projects/SCRUM/boards/1)

### Sprint 1

* Crear y editar tareas.
* Autenticarse con JWT.
* Priorización sugerida por IA.
* Validar campos obligatorios.

### Sprint 2

* Dividir en subtareas.
* Recordatorios por correo.
* Progreso semanal.
* Servicio de notificaciones.

### Sprint 3

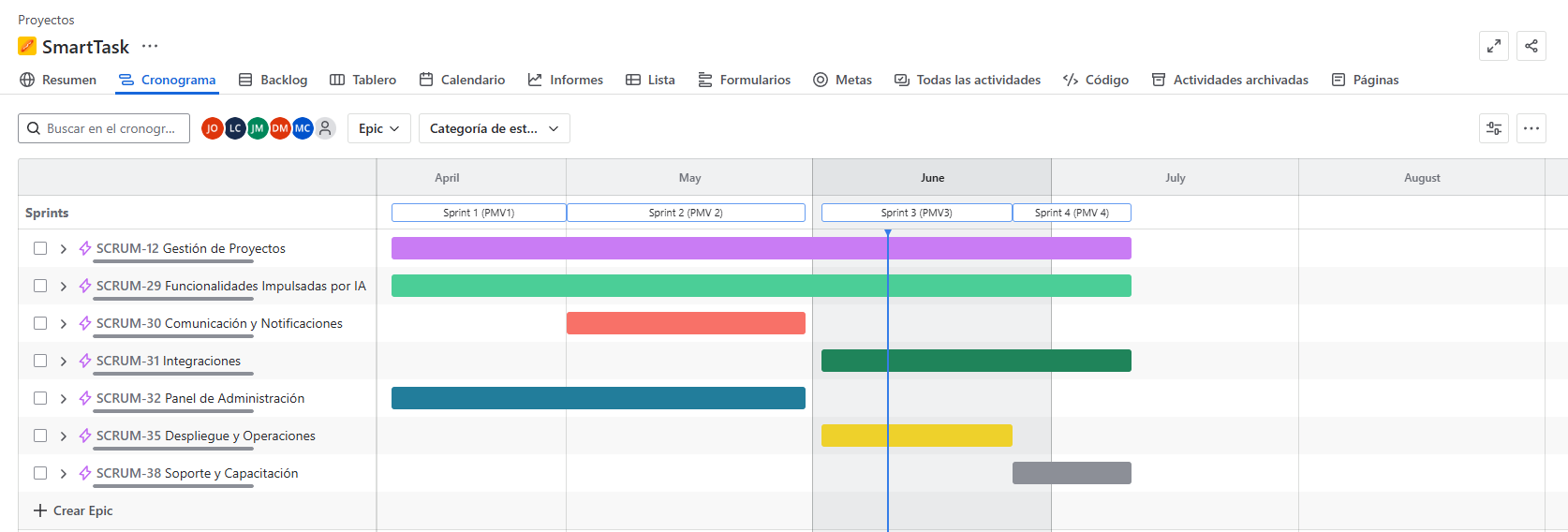
* Sugerencias semanales.
* Exportar tareas a Jira.
* Despliegue en la nube.

### Sprint 4

* Capacitación.
* Documentación.
* Historia técnica: API Jira.
* Historia técnica: Predecir prioridades.

## Planificación de Sprints

*Figura 4. Planificación de Sprints.*



[**https://continental-team-lhodxug5.atlassian.net/jira/software/projects/SCRUM/boards/1/timeline?timeline=MONTHS**](https://continental-team-lhodxug5.atlassian.net/jira/software/projects/SCRUM/boards/1/timeline?timeline=MONTHS)

### Historias de usuario

**HU01- Crear y editar tareas:**

Como usuario, quiero crear y editar tareas para organizar mi trabajo.

**Criterios de Aceptación**

* El sistema permite ingresar título, descripción, fecha límite y categoría para cada tarea.
* Las tareas creadas se guardan en MongoDB y se muestran en la lista de tareas.

**HU02- Autenticarme con JWT:**

Como administrador, quiero autenticarme con JWT para proteger mis datos.

**Criterios de Aceptación**

* El sistema valida credenciales y devuelve un token JWT al iniciar sesión.
* Las rutas protegidas (ej.: /tasks) requieren token válido

**HU03- Prioridades sugeridas IA:**

Como usuario, quiero que la IA sugiera prioridades basadas en fechas límite.

**Criterios de Aceptación**

* La plataforma clasifica automáticamente las tareas como "Urgente", "Prioritaria" o "Normal" según su fecha límite
* Las tareas se ordenan en la vista principal según su prioridad.

**HU04- Validar Campos Obligatorios:**

Como equipo de desarrollo, necesitamos implementar un sistema de validación de datos en el backend para garantizar que las tareas sean creadas correctamente con los campos obligatorios.

**HU05- Recordatorios por Correo:**

Como usuario quiero recibir recordatorios por correo de tareas próximas a vencer

**Criterios de Aceptación**

* El sistema envía un correo 24 horas antes de la fecha límite si la tarea está incompleta.
* Los recordatorios incluyen título de la tarea y enlace para editarla

**HU06- Servicio de Notificaciones:**

Como equipo de desarrollo, necesitamos configurar un servicio de notificaciones por correo que se active automáticamente cuando una tarea esté próxima a vencer.

**HU07- Progreso Semanal:**

Como usuario, quiero ver un panel con mi progreso semanal.

**Criterios de Aceptación**

* El panel muestra métricas como "Tareas completadas" y "Tiempo promedio por tarea".
* Los datos se actualizan en tiempo real al marcar tareas como completadas.

**HU08- Dividir en Subtareas:**

Como usuario, quiero dividir tareas grandes en subtareas para gestionarlas mejor

**Criterios de Aceptación**

* Cada tarea principal permite agregar subtareas con fechas límite independientes.
* El progreso de la tarea principal se actualiza según subtareas completadas

**HU09- Exportar tareas a Jira:**

Cada tarea en SmartTask puede convertirse en una historia de usuario en Jira con un botón.

**Criterios de Aceptación**

* La plataforma clasifica automáticamente las tareas como "Urgente", "Prioritaria" o "Normal" según su fecha límite
* Los campos básicos (título, descripción) se mapean correctamente.

**HU10- Sugerencias Semanales:**

Como usuario, quiero recibir sugerencias semanales para mejorar mi productividad.

**Criterios de Aceptación**

* La IA sugiere hábitos (ej.: "Agrupa tareas similares") basados en mi historial.
* Las sugerencias se muestran cada lunes en un banner destacado.

**HU11- Despliegue en la Nube:**

Como administrador, quiero desplegar la plataforma en un servidor en la nube.

**Criterios de Aceptación**

* La aplicación está accesible vía web en un dominio público (ej.: app.smarttask.com).
* La base de datos MongoDB está configurada en el entorno de producción.

**HU12- API JIRA:**

Como equipo de desarrollo, necesitamos implementar una API que permita la integración de SmartTask con Jira para crear historias de usuario de manera automática desde una tarea.

**HU13- Documentación:**

Como equipo de desarrollo, queremos documentar lecciones aprendidas.

**Criterios de Aceptación**

* Se registran errores críticos y soluciones implementadas.
* El documento está disponible para futuros proyectos.

**HU14- Predecir Prioridades:**

Como equipo de desarrollo, necesitamos integrar un algoritmo de IA para predecir prioridades de tareas en función de las fechas límite y el comportamiento pasado.

**HU15- Capacitación:**

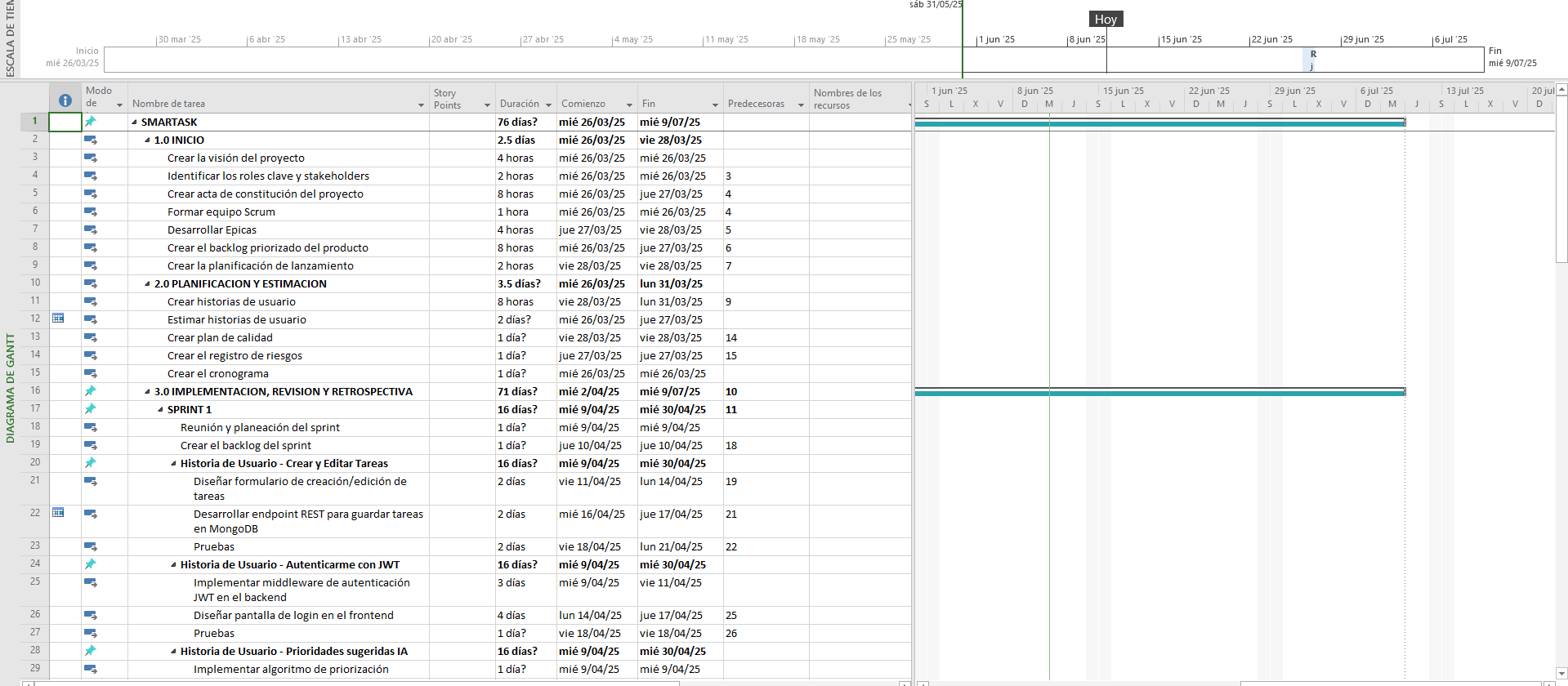
Como usuario final, quiero recibir capacitación sobre cómo usar SmartTask.

**Criterios de Aceptación**

* Se realizan talleres virtuales con demostraciones prácticas.
* Se entrega una guía de usuario en PDF y videotutoriales.

## Cronograma de Actividades

*Figura 5. Cronograma de actividades*

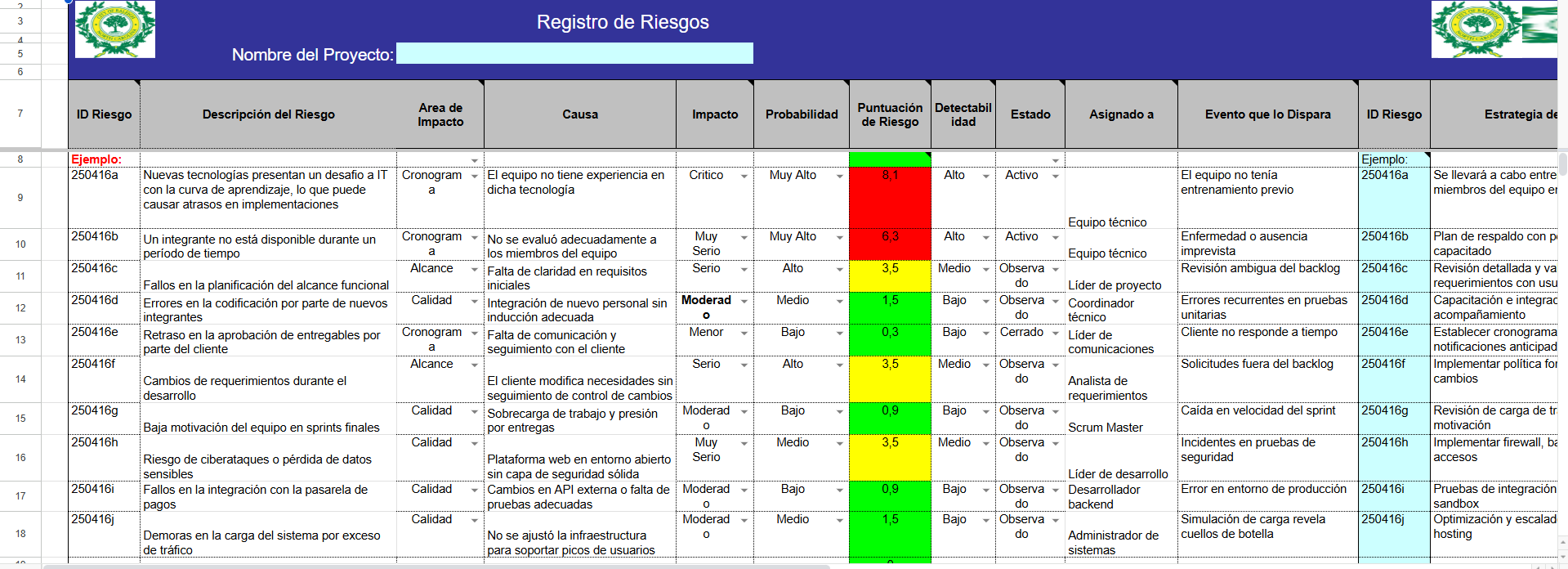


[**https://drive.google.com/drive/folders/1Oz3EyEaJsDYPDzbKWry0HQBDk7TG\_PbI?usp=sharing**](https://drive.google.com/drive/folders/1Oz3EyEaJsDYPDzbKWry0HQBDk7TG_PbI?usp=sharing)

## 

## 

## Gestión de Riesgos

*Figura 6. Gestión de riesgos.*

[**https://docs.google.com/spreadsheets/d/1jM6jkyWUkpWkTHULm5tgp1qbVoaaJ4CbhB3yPie7A-w/edit?usp=sharing**](https://docs.google.com/spreadsheets/d/1jM6jkyWUkpWkTHULm5tgp1qbVoaaJ4CbhB3yPie7A-w/edit?usp=sharing)

# **CAPÍTULO 5**

# DISEÑO DEL SISTEMA DE INFORMACIÓN

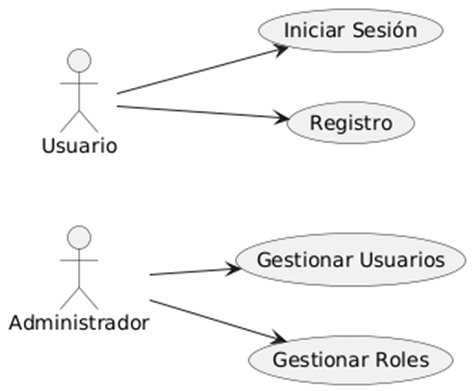
## 5.1. Diseño de Diagramas UML

### 5.1.1. Diagramas de casos de uso:

Los diagramas de casos de uso proporcionan una representación visual de las interacciones entre los usuarios (actores) y el sistema, mostrando cómo los actores utilizan las funcionalidades principales.

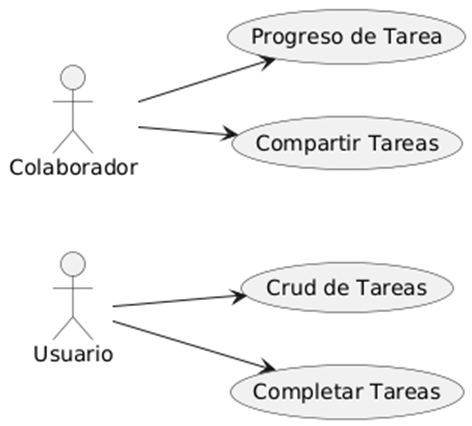
Caso de Uso 01: Gestión de roles y autenticación.

*Permite a los usuarios autenticarse en el sistema y a los administradores gestionar los roles asignados a cada usuario.*

*Figura 7. Caso de Uso 01. Gestión de roles y autenticación.*

Caso de Uso 02: Gestión de tareas.

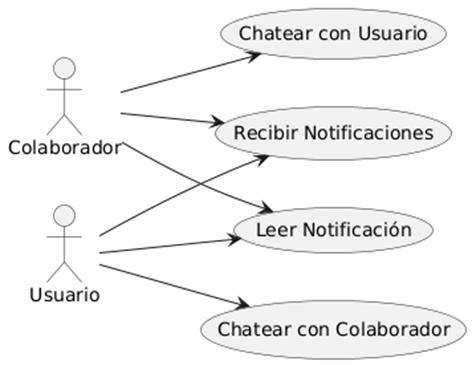
*Incluye la creación, edición, asignación y seguimiento de tareas por parte de los usuarios.*

**

*Figura 8. Caso de Uso 02. Gestión de tareas.*

Caso de Uso 03: Notificaciones y mensajes.

*Permite a los usuarios recibir notificaciones y enviar mensajes dentro del sistema.*

**

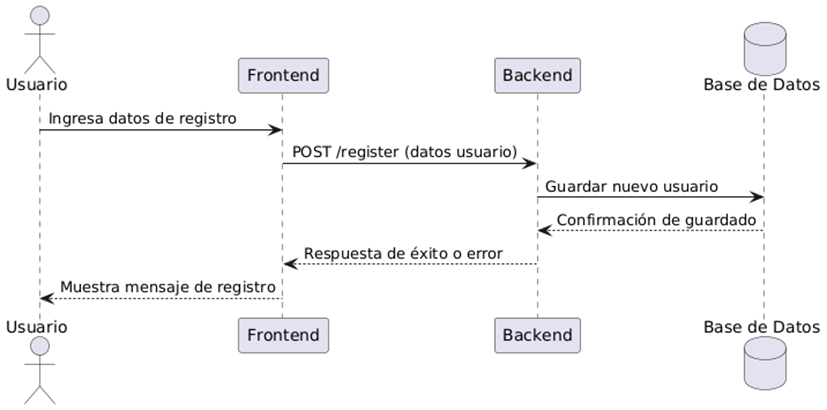
*Figura 9. Caso de Uso 03. Notificaciones y mensajes.*

### 5.1.2. Diagramas de secuencia:

Los diagramas de secuencia ilustran el flujo de mensajes y la interacción temporal entre los objetos del sistema durante la ejecución de procesos clave.

Diagrama de Secuencia 01: Autenticación del usuario.

*Describe el proceso de inicio de sesión, validación de credenciales y generación de sesión.*

**

*Figura 10. Diagrama de secuencia 01. Autenticación del usuario.*

Diagrama de Secuencia 02: Gestión de tareas.

*Muestra la secuencia de acciones para crear, modificar o eliminar una tarea.*

### 

Diagrama de Secuencia 03: Completar tarea.

*Detalla el flujo para marcar una tarea como completada y actualizar su estado.*

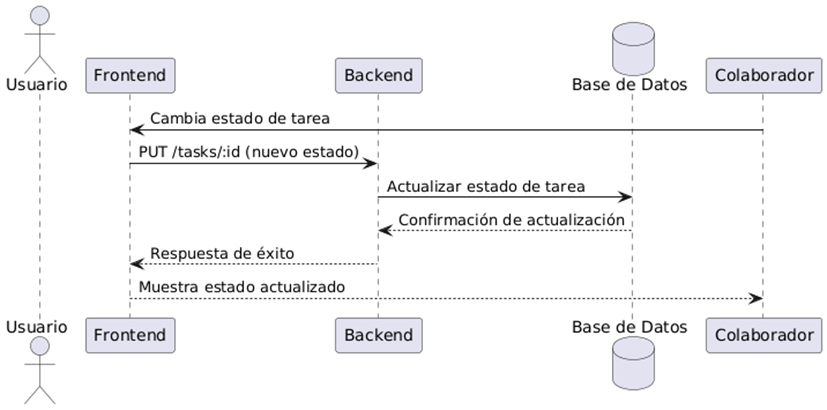
**

Diagrama de Secuencia 04: Notificaciones.

*Explica cómo se generan y envían notificaciones a los usuarios.*

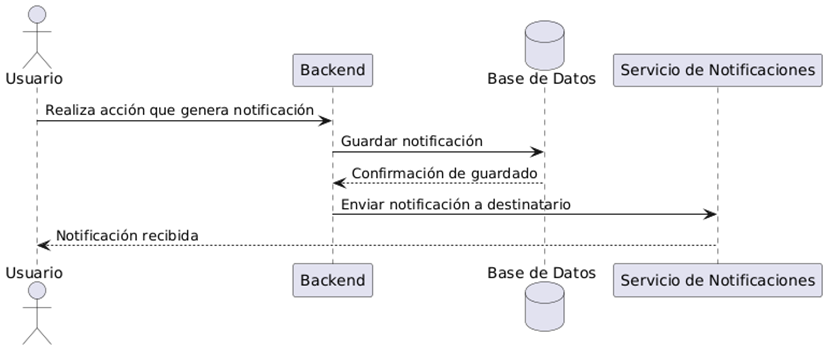
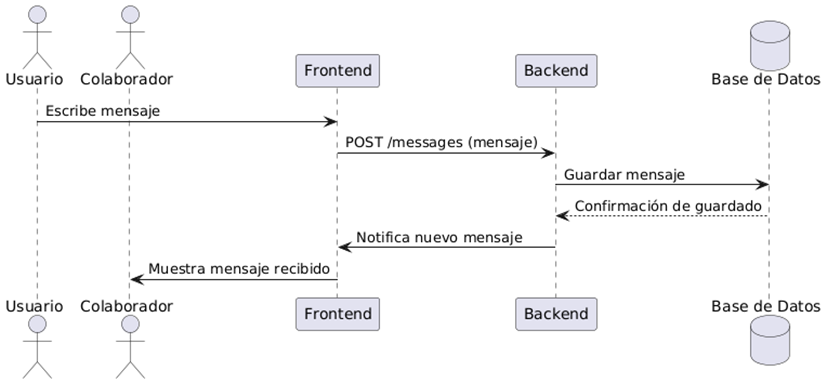
**

Diagrama de Secuencia 05: Mensajes.

*Representa la interacción entre usuarios al enviar y recibir mensajes.*

**

### 5.1.3. Diagramas de colaboración:

Estos diagramas muestran cómo los objetos y roles colaboran entre sí para cumplir con los casos de uso, destacando la organización de las interacciones.

Diagrama de Colaboración 01: Gestión de roles y autenticación.

Describe cómo interactúan el usuario, el sistema de autenticación y el módulo de roles para validar el acceso y asignar permisos.



Diagrama de Colaboración 02: Gestión de tareas.

Ilustra la colaboración entre el usuario, el gestor de tareas y la base de datos para la administración de tareas.



Diagrama de Colaboración 03: Notificaciones.

Muestra la interacción entre el generador de eventos, el sistema de notificaciones y los usuarios receptores.



Diagrama de Colaboración 04: Mensajería interna.

Detalla cómo los usuarios y el sistema de mensajería colaboran para el envío y recepción de mensajes.

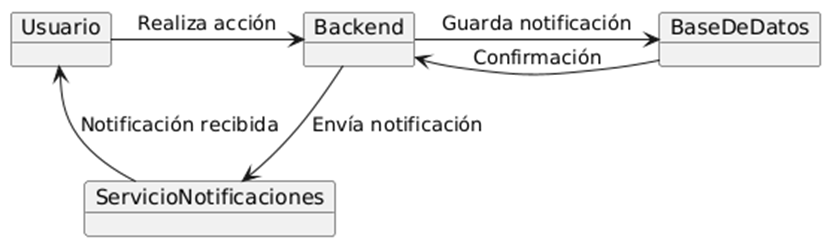
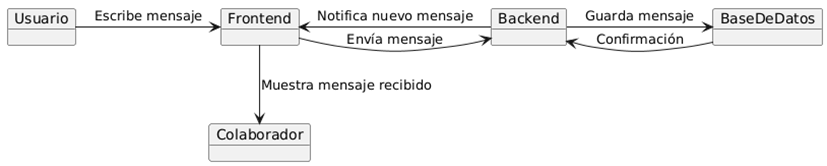


Diagrama de Colaboración 05:

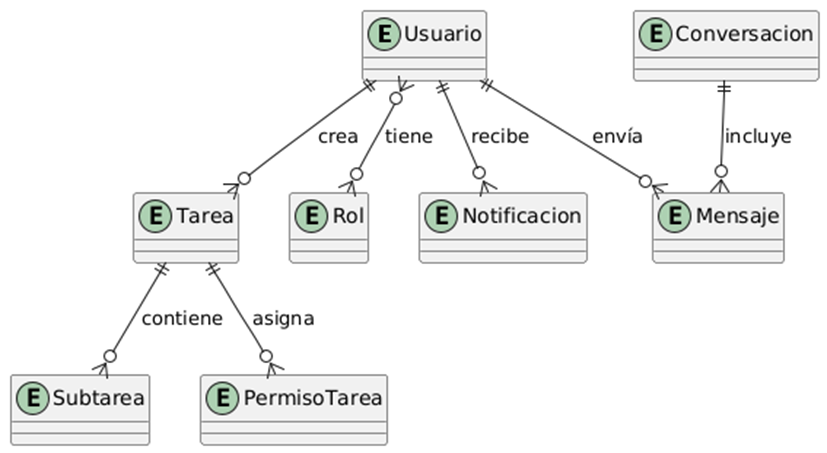
Seguimiento de tareas.

Explica la colaboración entre usuarios, tareas y el sistema de seguimiento para actualizar el progreso y estado de las tareas.



### 5.1.4. Diagramas de clases:

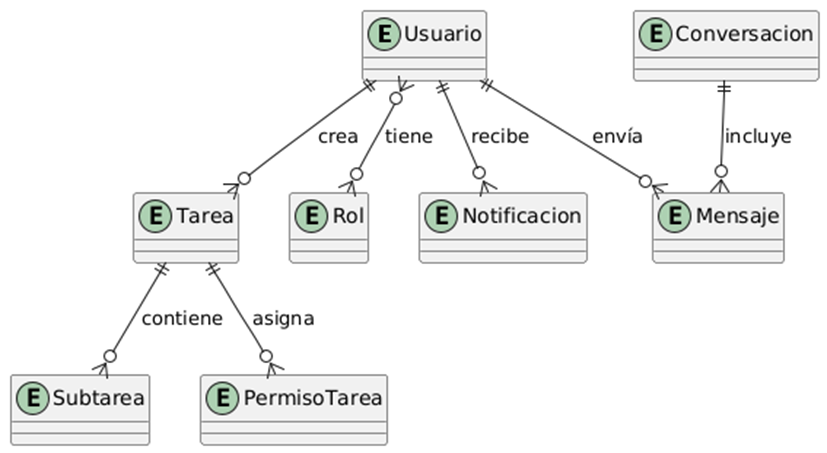
El diagrama de clases describe la estructura estática del sistema, mostrando las clases principales, sus atributos y métodos, así como las relaciones entre ellas.



## 5.2. Diseño de Base de Datos

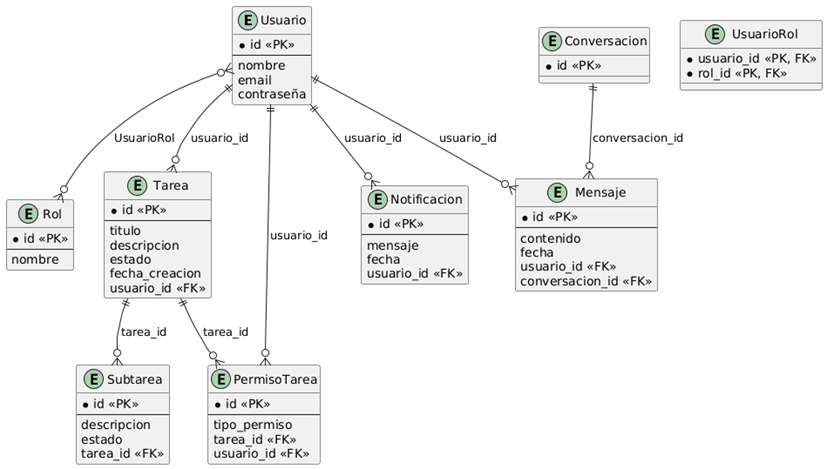
### 5.2.1. Diseño conceptual (E/R):

El modelo Entidad-Relación (E/R) representa las entidades principales del sistema y las relaciones entre ellas, proporcionando una visión abstracta de la estructura de datos.



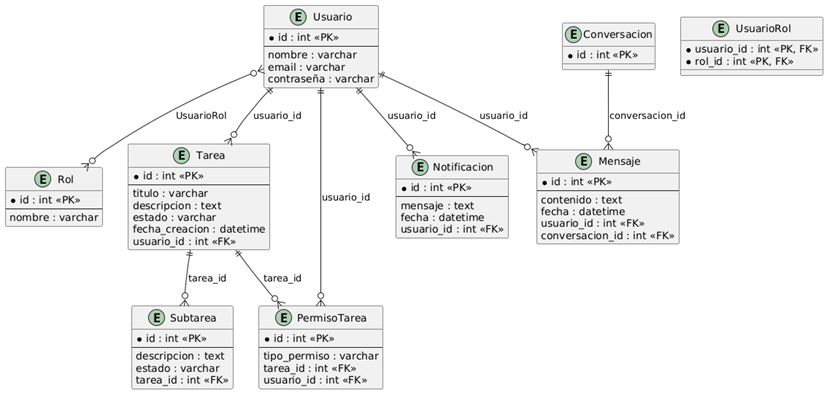
### 5.2.2. Diseño lógico:

En el diseño lógico se definen las tablas normalizadas, especificando claves primarias y foráneas, así como los campos básicos de cada entidad. Este modelo asegura la integridad y eficiencia de la base de datos.



### 5.2.3. Diseño físico:

El diseño físico detalla la implementación técnica de la base de datos, incluyendo la definición de índices, tipos de datos específicos y restricciones para optimizar el rendimiento y la seguridad.



### 5.2.4. Modelado de base de datos

Estrategias de modelado:

5.2.4.1. Normalización:

* Todas las tablas cumplen con la Tercera Forma Normal (3FN).
* Eliminación de datos redundantes
* Uso de claves foráneas para mantener la integridad referencial

5.2.4.2. Desnormalización controlada:

* El campo compartida en la tabla de tareas es un array que contiene información desnormalizada para mejorar el rendimiento en consultas frecuentes.
* El campo permisos en la tabla roles es un objeto JSON que almacena múltiples banderas de permiso

5.2.4.3. Índices:

* Índices en campos de búsqueda frecuente
* Índices compuestos para consultas que filtran por múltiples campos
* Índices para ordenación en campos de fecha

5.2.4.4. Particionamiento:

* Considerar particionamiento por rangos de fechas para tablas de crecimiento rápido como notificaciones
* Particionamiento por hash para distribuir la carga en tablas grandes

5.2.4.5. Estrategia de respaldo:

* Copias de seguridad completas semanales
* Copias incrementales diarias
* Retención de 30 días de respaldos

5.2.4.6. Consideraciones de rendimiento:

* Uso de conexiones persistentes
* Caché de consultas frecuentes
* Paginación en consultas que devuelven muchos registros
* Uso de proyecciones para recuperar solo los campos necesarios

5.2.4.7. Seguridad:

* Contraseñas almacenadas con hash bcrypt
* Tokens JWT para autenticación
* Validación de entrada en todos los endpoints
* Protección contra inyección SQL mediante consultas parametrizadas
* Registro de actividades sensibles

## 5.3. Diseño de Interfaces Básicas

### 5.3.1. Pantalla de Inicio de Sesión (Login):

La interfaz de acceso permite a los usuarios autenticarse en el sistema mediante sus credenciales. Además, ofrece opciones para el registro de nuevos usuarios y la recuperación de contraseña en caso de olvido, garantizando así una experiencia de usuario completa y segura.

Interfaz de usuario gráfica

El contenido generado por IA puede ser incorrecto.

### 5.3.2. Pantalla de Registro de Usuario:

Esta interfaz facilita la creación de nuevas cuentas de usuario. El formulario de registro solicita la información básica necesaria para el alta en el sistema, asegurando la validación de datos y la protección de la información personal.

Captura de pantalla de un celular

El contenido generado por IA puede ser incorrecto.

### 5.3.3. Panel Principal (Dashboard):

El dashboard constituye la vista principal tras el inicio de sesión. En él se presenta un resumen de la información relevante del sistema, accesos directos para la creación de nuevas tareas y enlaces para visualizar las tareas existentes. Su diseño prioriza la usabilidad y el acceso rápido a las funciones más utilizadas.

Interfaz de usuario gráfica, Sitio web

El contenido generado por IA puede ser incorrecto.

### 5.3.4. Vista de Lista de Tareas:

La interfaz de lista de tareas muestra de manera clara y ordenada todas las tareas creadas por el usuario. Incluye información sobre la prioridad, el estado de avance y permite filtrar o buscar tareas específicas, facilitando la gestión eficiente del trabajo pendiente.

Captura de pantalla de un celular

El contenido generado por IA puede ser incorrecto.

### 5.3.5. Vista Detallada de Tarea:

En esta pantalla se presenta la información completa de una tarea individual. El usuario puede editar, eliminar, marcar como completada o compartir la tarea. Además, se ofrece la opción de agregar subtareas, permitiendo una gestión más granular y detallada de las actividades.

Captura de pantalla de un celular

El contenido generado por IA puede ser incorrecto.

### 5.3.6. Interfaz de Seguimiento de Progreso:

Esta sección proporciona herramientas visuales para el seguimiento del rendimiento. Incluye un gráfico de progreso semanal, métricas de tareas completadas, un gráfico de barras de productividad y el porcentaje global de cumplimiento, permitiendo al usuario evaluar su desempeño y planificar mejoras.

Interfaz de usuario gráfica, Aplicación

El contenido generado por IA puede ser incorrecto.

**CAPÍTULO 6**

**CODIFICACIÓN DEL SOFTWARE**

## 6.1. Desarrollo del Sprint 1

### 6.1.1. Sprint planning

Durante la reunión de planificación del Sprint 1, el equipo definió como prioridad desarrollar las funcionalidades más básicas del sistema SmartTask, enfocadas en la creación de tareas, autenticación básica y priorización inteligente mediante reglas de inteligencia artificial. Se planteó trabajar con dos historias de usuario fundamentales para sentar las bases del sistema.

También se determinó que se emplearían herramientas como React para el frontend, Node.js para el backend, MongoDB como base de datos y Jira como plataforma para la gestión de tareas y control del avance del sprint.

### 6.1.2. Sprint backlog

Las tareas incluidas en el backlog de este sprint fueron:

* Implementar el formulario de creación y edición de tareas en el frontend.
* Configurar el endpoint POST /tasks en el backend para guardar las tareas en la base de datos MongoDB.
* Desarrollar un algoritmo de priorización simple basado en fechas límite.
* Ajustar el endpoint GET /tasks para retornar las tareas ordenadas por nivel de prioridad.
* Implementar la autenticación con tokens JWT para asegurar las rutas protegidas.

### 6.1.3. Historias de usuarios

**Épica: Gestión de Proyectos** **HU1:** *Como usuario, quiero crear y editar tareas para organizar mi trabajo.* **Criterios de Aceptación:**

* El sistema permite ingresar título, descripción, fecha límite y categoría para cada tarea.
* Las tareas creadas se guardan correctamente en MongoDB y se muestran en la lista de tareas del usuario.

**Tareas asociadas:**

* Diseñar formulario de creación/edición de tareas utilizando React.
* Desarrollar endpoint REST (POST /tasks) en Node.js para almacenar las tareas.

**Épica: Funcionalidades Impulsadas por IA** **HU2:** *Como usuario, quiero que la IA sugiera prioridades basadas en fechas límite.* **Criterios de Aceptación:**

* La plataforma clasifica automáticamente las tareas como “Urgente”, “Prioritaria” o “Normal” según la cercanía de su fecha límite.
* Las tareas se ordenan en la vista principal de acuerdo con su prioridad asignada.

**Tareas asociadas:**

* Implementar un algoritmo de priorización que asigne etiquetas según la fecha límite (por ejemplo, si faltan 2 días o menos, se marca como “Urgente”).
* Integrar el ordenamiento de tareas en el endpoint GET /tasks, asegurando que se visualicen en orden descendente de prioridad.

### 6.1.4. Taskboard

Para el control del avance del sprint se utilizó la herramienta Jira, en la que se configuró un tablero con las columnas clásicas del flujo ágil: “To Do”, “In Progress” y “Done”. Cada miembro del equipo asumió tareas específicas, y se realizó un seguimiento diario del estado de cada historia de usuario y sus respectivas tareas técnicas.

### 6.1.5. Daily scrum

Se realizaron reuniones diarias (Daily Scrum) de 15 minutos, de manera virtual, utilizando Google Meet. En estas reuniones, cada integrante del equipo informaba brevemente lo que hizo el día anterior, lo que haría ese día y cualquier impedimento encontrado. Esta dinámica permitió identificar rápidamente bloqueos y redistribuir esfuerzos si era necesario.

### 6.1.6. Sprint review

Al final del sprint, se realizó una reunión de revisión (Sprint Review) con todo el equipo, donde se presentaron las funcionalidades desarrolladas. Se validó que las tareas pudieran crearse y editarse correctamente, que el sistema aplicara el algoritmo de priorización y que las tareas aparecieran en el orden esperado. También se verificó que los datos se almacenaran en MongoDB sin inconsistencias.

### 6.1.7. Criterios de aceptación

Los criterios definidos en cada historia de usuario fueron comprobados uno a uno:

• En HU1 se validó la funcionalidad del formulario, el guardado exitoso en la base de datos y la visualización.

• En HU2 se evaluó el algoritmo de clasificación y el ordenamiento de las tareas en pantalla, los cuales funcionaron conforme a lo esperado.

### 6.1.8. Resultados del sprint

#### 6.1.8.1. Evidencias.

Se generaron capturas de pantalla que muestran el formulario de tareas en funcionamiento, así como la lista de tareas ordenada por prioridad. También se documentó el resultado de las pruebas de integración entre el frontend y backend.

#### 6.1.8.2. Prueba de desarrollo.

Se realizaron pruebas unitarias en el backend para validar la funcionalidad del endpoint POST /tasks y la clasificación por prioridad. En el frontend se realizaron pruebas funcionales manuales para verificar el flujo de creación y edición de tareas.

#### 6.1.8.3. Observaciones

Durante el desarrollo se detectó que las fechas ingresadas debían validarse en formato ISO para evitar errores en la comparación por el algoritmo de IA. Esto fue resuelto integrando una función de validación de fechas en el backend.

### 6.1.9. Sprint retrospective

En la retrospectiva del sprint, el equipo resaltó como aspectos positivos la buena coordinación interna, la claridad de las historias y la implementación funcional de todas las tareas del backlog. Como mejora se propuso reforzar la documentación técnica de los endpoints desarrollados y automatizar parte de las pruebas de backend en los siguientes sprints.

## 6.2. Desarrollo del Sprint 2

### 6.2.1. Sprint planning

En este segundo sprint, el equipo priorizó funcionalidades que mejorarían la experiencia del usuario y la integración con herramientas externas. Se decidió implementar los recordatorios automáticos por correo, la exportación de tareas a Jira y un sistema de autenticación seguro utilizando JWT. Estas funcionalidades estaban asociadas a historias de usuario orientadas a la comunicación y seguridad, y a la interoperabilidad con entornos ágiles como Jira.

### 6.2.2. Sprint backlog

• Configurar el servicio de envío de correos con Nodemailer.

• Desarrollar un proceso automático (cron job) que revise tareas próximas a vencer y dispare los recordatorios.

• Integrar la API de Jira para exportar tareas desde SmartTask.

• Implementar el sistema de login con generación y validación de token JWT.

• Proteger las rutas críticas con middleware de autenticación.

### 6.2.3. Historias de usuarios

**Épica: Comunicación y Notificaciones** **HU3:** *Como usuario, quiero recibir recordatorios por correo de tareas próximas a vencer.* **Criterios de Aceptación:**

* El sistema envía un correo 24 horas antes de la fecha límite si la tarea está incompleta.
* El correo incluye el título de la tarea y un enlace para editarla.

**Tareas asociadas:**

* Configurar el servicio de correo con Nodemailer.
* Crear un cron job diario que revise las tareas con fecha límite próxima.

**Épica: Integraciones** **HU4:** *Como líder de equipo, quiero exportar tareas a Jira como historias de usuario.* **Criterios de Aceptación:**

* Cada tarea tiene un botón para ser exportada a Jira.
* Los campos básicos (título y descripción) se transfieren correctamente.

**Tareas asociadas:**

* Integrar API de Jira con endpoint POST /jira/tasks.
* Crear botón en la interfaz que active la exportación.

**Épica: Panel de Administración** **HU5:** *Como administrador, quiero autenticarme con JWT para proteger mis datos.* **Criterios de Aceptación:**

* El sistema valida credenciales y genera un token JWT.
* Las rutas protegidas requieren un token válido para acceder.

**Tareas asociadas:**

* Implementar middleware de autenticación con JWT en el backend.
* Diseñar la pantalla de inicio de sesión en el frontend.

### 6.2.4. Taskboard

Jira fue nuevamente la herramienta empleada para gestionar tareas, con tarjetas vinculadas a cada historia y tareas técnicas. Se añadieron etiquetas específicas como “notificación”, “seguridad” e “integración” para facilitar el seguimiento.

### 6.2.5. Daily scrum

Las reuniones diarias se enfocaron en identificar errores en el cron job y compatibilidad con la API de Jira. Se detectaron retrasos en los envíos de correo en entornos con poca conectividad y se tomaron decisiones rápidas para configurar servidores de prueba alternativos.

### 6.2.6. Sprint review

En la revisión, se demostró el envío de correos automáticos correctamente programados, el funcionamiento del login con JWT y la creación exitosa de historias en Jira a partir de tareas. Las funcionalidades fueron recibidas de forma positiva por el equipo y se sugirieron mejoras en el diseño de los correos enviados.

### 6.2.7. Criterios de aceptación

Las tres historias fueron verificadas de forma satisfactoria:

• En HU3 se validaron los tiempos de envío y el contenido de los correos.

• En HU4 se observó el mapeo correcto de los datos a Jira.

• En HU5 se protegieron correctamente las rutas privadas del sistema.

### 6.2.8. Resultados del sprint

#### 6.2.8.1. Evidencias.

• Correos electrónicos con recordatorios reales recibidos.

• Capturas del token JWT generado y almacenado en el cliente.

• Pantallas de Jira con las tareas exportadas desde SmartTask.

#### 6.2.8.2. Prueba de desarrollo.

• Pruebas de integración de cron y Nodemailer.

• Pruebas funcionales de exportación con cuenta de Jira de prueba.

• Validación manual del middleware de autenticación.

#### 6.2.8.3. Observaciones

El cron job inicialmente presentaba fallos por zona horaria del servidor. Se solucionó configurando TZ y actualizando los registros en UTC. También se mejoró la validación del token en el frontend.

### 6.2.9. Sprint retrospective

El equipo acordó como fortaleza la integración de herramientas externas y la correcta protección del sistema. Se propuso agregar pruebas automatizadas para el cron job y mejorar la interfaz del login en el siguiente sprint.

## 6.3. Desarrollo del Sprint 3

### 6.3.1. Sprint planning

En esta etapa, el objetivo fue enriquecer la experiencia del usuario con funcionalidades adicionales: la creación de subtareas, un panel de métricas de productividad y sugerencias semanales generadas por IA. Estas funcionalidades buscaban transformar la plataforma en una herramienta de gestión más completa e inteligente.

### 6.3.2. Sprint backlog

• Añadir estructura de subtareas al modelo de datos.

• Implementar lógica para calcular el progreso total de tareas compuestas.

• Crear panel de métricas con estadísticas de uso y rendimiento.

• Generar sugerencias semanales basadas en patrones de uso.

### 6.3.3. Historias de usuarios

**Épica: Gestión de Proyectos**

**HU6**: *Como usuario, quiero dividir tareas grandes en subtareas para gestionarlas mejor.*

**Criterios de Aceptación:**

• Se pueden crear subtareas con fechas límite propias.

• El avance de la tarea principal se actualiza automáticamente.

**Épica: Panel de Administración**

**HU7:** *Como usuario, quiero ver un panel con mi progreso semanal.*

**Criterios de Aceptación:**

• Métricas como “tareas completadas” y “tiempo promedio por tarea”.

• Datos en tiempo real, actualizados al completar tareas.

**Épica: Funcionalidades con IA**

**HU8:** *Como usuario, quiero recibir sugerencias semanales para mejorar mi productividad.*

**Criterios de Aceptación:**

• Recomendaciones personalizadas con base en hábitos de uso.

• Banner visible cada lunes con sugerencias claras.

**Tareas Asociadas:**

· Diseñar algoritmo que analice comportamiento del usuario (tiempos, incumplimientos, etc.).

· Crear componente React que muestre las sugerencias y permita marcarlas como leídas.

· Programar ejecución semanal del módulo de IA.

### 6.3.4. Taskboard

Se dividieron tareas entre diseño del panel, backend de métricas y lógica de IA. Cada subtarea técnica fue rastreada en Jira y se asignó responsable directo para cada una.

### 6.3.5. Daily scrum

Se centraron en resolver conflictos entre el cálculo del progreso de subtareas y el renderizado en el panel. También se ajustaron los formatos de los datos métricos para evitar errores de tipo en frontend.

### 6.3.6. Sprint review

El equipo presentó el nuevo panel con gráficas, subtareas en funcionamiento y el banner de sugerencias IA. Las pruebas mostraron que el sistema respondía bien a medida que los usuarios interactuaban con él.

### 6.3.7. Criterios de aceptación

Todas las funcionalidades cumplieron sus criterios. El panel reflejó datos correctos, las subtareas se comportaron de forma esperada y las sugerencias se generaron al iniciar semana.

### 6.3.8. Resultados del sprint

#### 6.3.8.1. Evidencias.

• Panel con gráfico de tareas completadas y tiempos promedio.

• Vista con subtareas y su progreso.

• Captura del banner de sugerencias.

#### 6.3.8.2. Prueba de desarrollo.

• Pruebas manuales de creación, edición y eliminación de subtareas.

• Revisión de los cálculos métricos con datos simulados.

#### 6.3.8.3. Observaciones

Algunos usuarios confundieron subtareas con tareas normales; se propuso diferenciar visualmente ambos tipos.

### 6.3.9. Sprint retrospective

Se reconoció el avance significativo en usabilidad e inteligencia del sistema. Se sugirió optimizar el rendimiento de la carga del panel y usar cache local para los datos estáticos.

## 6.4. Desarrollo del Sprint 4

### 6.4.1. Sprint planning

El sprint final se centró en el despliegue completo del sistema, la creación de materiales de capacitación y el cierre formal del proyecto con la documentación de lecciones aprendidas.

### 6.4.2. Sprint backlog

• Desplegar sistema en Heroku y Vercel.

• Generar manual técnico y guía de usuario en PDF.

• Preparar videotutoriales de uso básico.

• Recopilar feedback y escribir informe de cierre.

### 6.4.3. Historias de usuarios

**Épica: Despliegue y Operaciones** **HU9:** *Como administrador, quiero desplegar la plataforma en un servidor en la nube.* **Criterios de Aceptación:**

* La plataforma debe estar accesible desde un dominio público (ej. Vercel + Heroku).
* La conexión a la base de datos MongoDB Atlas debe funcionar en entorno de producción.

**Tareas Asociadas:**

* Configurar entorno de producción (frontend y backend).
* Validar integridad de funcionalidades tras el despliegue.
* Comprobar tiempos de respuesta y acceso desde dispositivos móviles.

**Épica: Soporte y Capacitación** **HU10:** *Como usuario final, quiero recibir capacitación sobre cómo usar SmartTask.* **Criterios de Aceptación:**

* Se debe entregar una guía de usuario en formato PDF y al menos un videotutorial.
* Realizar una sesión de capacitación virtual o taller grabado.

**Tareas Asociadas:**

* Redactar manual con capturas y descripciones claras de funciones.
* Grabar y editar un videotutorial explicativo.
* Programar una sesión con usuarios finales e implementar formulario de retroalimentación.

**Épica: Gestión del Proyecto** **HU11:** *Como equipo de desarrollo, queremos documentar lecciones aprendidas.* **Criterios de Aceptación:**

* Se debe generar un documento que recoja problemas detectados, soluciones aplicadas y sugerencias para futuras versiones.
* El documento debe estar estructurado y validado por todos los miembros del equipo.

**Tareas Asociadas:**

* Recolectar feedback del equipo y usuarios piloto.
* Sistematizar las experiencias positivas y desafíos encontrados.
* Redactar el informe de cierre del proyecto con enfoque en mejora continua.

### 6.4.4. Taskboard

El taskboard del Sprint 4 fue más documental que técnico. Se dividió por productos (manual, video, taller, despliegue).

### 6.4.5. Daily scrum

Se enfocaron en afinar el despliegue, comprobar la estabilidad del sistema en producción y revisar los contenidos de los tutoriales.

### 6.4.6. Sprint review

El sistema quedó operativo y accesible desde un dominio público. Los materiales fueron entregados y se realizó una sesión de capacitación vía Zoom con usuarios piloto.

### 6.4.7. Criterios de aceptación

Todos los criterios fueron cumplidos. El sistema fue accesible públicamente, los usuarios recibieron material adecuado y el equipo documentó formalmente su proceso.

### 6.4.8. Resultados del sprint

#### 6.4.8.1. Evidencias.

• Capturas del sistema desplegado.

• Manual en PDF y enlaces a videotutoriales.

• Acta de lecciones aprendidas.

#### 6.4.8.2. Prueba de desarrollo.

• Prueba funcional en producción.

• Validación de accesibilidad en distintos dispositivos.

#### 6.4.8.3. Observaciones

Se reportó carga lenta en conexión móvil. Se propuso usar un CDN para mejorar la distribución.

### 6.4.9. Sprint retrospective

El equipo valoró el cumplimiento de los entregables, la utilidad de los materiales y la experiencia de los usuarios piloto. Se planteó iniciar una segunda versión del sistema en una nueva iteración.

**CAPÍTULO 7**

**PRUEBAS DE SOFTWARE**

**7.1. Plan de Pruebas**

El presente capítulo describe el plan y la ejecución de pruebas realizadas para el sistema Smart Task Manager, abarcando tanto el frontend como el backend, así como las pruebas end-to-end automatizadas con Cypress. El objetivo es garantizar la calidad, funcionalidad y robustez del software antes de su despliegue.

**7.1.1. Objetivos de las Pruebas**

* Verificar que cada componente del sistema funcione de acuerdo a los requisitos especificados.
* Detectar errores en la lógica de negocio, interfaz de usuario y comunicación entre módulos.
* Validar la integración entre frontend y backend a través de flujos completos de usuario.

**7.1.2. Tipos de Pruebas Realizadas**

**a) Pruebas Unitarias y de Integración (**npm test**)** Se ejecutaron pruebas unitarias y de integración tanto en el frontend como en el backend utilizando el comando npm testen cada módulo.

* **Frontend:**
  + Framework: Jest y React Testing Library.
  + Alcance: Componentes de React, manejo de eventos, renderizado y validación de la interfaz de usuario.
  + Objetivo: Garantizar que los componentes individuales funcionen correctamente y que las interacciones del usuario generen los resultados esperados.
* **Backend:**
  + Framework: Jest.
  + Alcance: Controladores, rutas, modelos y middleware de la API REST.
  + Objetivo: Verificar la lógica de negocio, la correcta respuesta de los endpoints, la validación de datos y el manejo de errores.

**b) Pruebas End-to-End (Cypress)**

Se implementaron pruebas automatizadas de extremo a extremo utilizando Cypress, permitiendo simular el comportamiento real de un usuario a través de la interfaz gráfica.

* **Flujos cubiertos:**
  + Autenticación de usuarios (inicio y cierre de sesión).
  + Creación, edición y eliminación de tareas.
  + Navegación y validación de la información mostrada en la interfaz.
* **Estructura:**
  + Los archivos de prueba se encuentran en cypress/e2e/.
  + Se utilizan comandos personalizados para automatizar acciones repetitivas como el login.

**7.1.3. Herramientas Utilizadas**

* **Jest:** Para pruebas unitarias y de integración en frontend y backend.
* **React Testing Library:** Para pruebas de componentes React.
* **Cypress:** Para pruebas end-to-end que validan flujos completos de usuario.

**7.1.4. Evidencias de Ejecución y Resultados de Pruebas**

A continuación se presentan los resultados y evidencias obtenidas tras la ejecución de las pruebas unitarias, de integración y end-to-end, realizadas tanto en el frontend como en el backend del sistema Smart Task Manager. Las capturas de pantalla incluidas respaldan la correcta ejecución y superación de los casos de prueba más relevantes.

**7.1.4.1. Pruebas Unitarias y de Integración (**npm test**)**

Las pruebas unitarias y de integración fueron ejecutadas en los módulos frontend y backend utilizando el comando npm test. Estas pruebas tienen como objetivo validar el correcto funcionamiento de los componentes individuales, la lógica de negocio y la interacción entre los diferentes módulos del sistema.

* En el frontend, se utilizaron Jest y React Testing Library para validar componentes de React y la interacción del usuario con la interfaz. Como se observa en la siguiente captura, se ejecutaron dos suites de pruebas, abarcando un total de cuatro pruebas, todas ellas superadas exitosamente:

Texto

El contenido generado por IA puede ser incorrecto.

* ·En la segunda imagen, se muestra el resultado específico de la prueba relacionada con el componente de inicio de sesión, donde se valida que el formulario y los mensajes asociados se muestran correctamente:

Texto

El contenido generado por IA puede ser incorrecto.

* En el backend, las pruebas unitarias e integración se enfocan en la validación de las rutas API, controladores y modelos de datos. En la siguiente captura se observa cómo las pruebas para los módulos de roles y tareas fueron ejecutadas exitosamente, validando operaciones como la creación y recuperación de tareas, así como la gestión de roles de usuario:

Texto

El contenido generado por IA puede ser incorrecto.

El éxito de estas pruebas demuestra la robustez de la lógica de negocio y la correcta integración entre los distintos componentes del sistema.

**7.1.4.2. Pruebas End-to-End con Cypress**

Las pruebas end-to-end (E2E) se realizaron utilizando Cypress, una herramienta que permite simular el comportamiento real de un usuario interactuando con la aplicación a través de la interfaz gráfica. Estas pruebas son fundamentales para asegurar que los flujos críticos de la aplicación funcionan correctamente desde la perspectiva del usuario final.

* **Pruebas de Gestión de Tareas (**task.cy.js**):**

**Interfaz de usuario gráfica

El contenido generado por IA puede ser incorrecto.**

En la siguiente captura se muestra la ejecución de una prueba E2E donde se valida el flujo completo de gestión de tareas. El test automatizado simula el inicio de sesión, la creación de una nueva tarea, la verificación de su aparición en la lista de tareas y otras interacciones relevantes. La consola de Cypress muestra cada paso ejecutado, las peticiones realizadas al backend y las aserciones cumplidas, mientras que la interfaz gráfica refleja el estado actualizado de las tareas:

* **Pruebas de Autenticación (**auth.cy.js**):**

Captura de pantalla de un celular

El contenido generado por IA puede ser incorrecto.

En esta captura se evidencia el flujo de autenticación, donde el test automatizado simula el ingreso de credenciales, la validación de la sesión y la navegación a la vista principal de tareas. Se observa la notificación de “Inicio de sesión exitoso” y la correcta carga de la interfaz personalizada para el usuario autenticado. La consola de Cypress detalla cada paso, incluyendo las peticiones HTTP y las aserciones sobre la URL y el contenido visible:



**7.1.4.3. Análisis de Resultados**

La ejecución exitosa de todas las pruebas unitarias, de integración y end-to-end evidencia que el sistema cumple con los requisitos funcionales y no funcionales establecidos en el plan de pruebas. Los flujos críticos, como la autenticación y la gestión de tareas, han sido validados tanto a nivel de componentes individuales como en la experiencia completa del usuario, asegurando un alto nivel de calidad y confiabilidad en el software desarrollado.

**CONCLUSIONES**

* La implementación de SmartTask ha demostrado ser una solución efectiva para la gestión de tareas personales y colaborativas, integrando inteligencia artificial para la priorización automática y recordatorios automatizados, lo cual mejora significativamente la productividad del usuario.
* El enfoque incremental mediante productos mínimos viables (PMV) permitió validar funcionalidad clave en cada etapa del desarrollo, adaptando la plataforma a las necesidades reales de los usuarios.
* Las pruebas funcionales y de usuario final evidenciaron una interfaz intuitiva, accesible y útil, con buena aceptación en los entornos de prueba. El sistema logró satisfacer los requerimientos funcionales y no funcionales establecidos en el análisis de requerimientos.
* La integración con herramientas externas como Jira amplía el alcance del sistema, haciéndolo apto para equipos de desarrollo ágil, especialmente en entornos académicos y técnicos.
* La factibilidad técnica, económica y operativa fue comprobada, destacando el uso eficiente de tecnologías open source y servicios gratuitos para el desarrollo y despliegue del sistema.

**RECOMENDACIONES**

* Ampliar las pruebas piloto a un grupo más diverso de usuarios, incluyendo profesionales y empresas, para identificar mejoras funcionales y validar el sistema en contextos más exigentes.
* Optimizar el sistema de notificaciones por correo electrónico, enfocándose en su confiabilidad y personalización, a fin de aumentar la eficacia de los recordatorios y alertas.
* Incorporar métricas avanzadas de rendimiento y análisis de productividad, lo cual podría permitir ofrecer recomendaciones más personalizadas mediante IA, basadas en patrones de uso del usuario.
* Explorar futuras integraciones con otras plataformas colaborativas, como Slack, Microsoft Teams o Trello, para posicionar a SmartTask como un hub central de gestión de tareas.
* Establecer una estrategia de soporte y mantenimiento continuo, incluyendo monitoreo, actualizaciones periódicas y recolección de feedback constante para asegurar la evolución del producto.

**ANEXOS**

**ANEXO 01. Manual Técnico**

#### 1. Información General del Sistema

* Nombre del sistema: Sistema Inteligente de Control de Inventario
* Versión: 1.0
* Lenguajes utilizados: JavaScript (React), Node.js
* Base de datos: MongoDB
* Servidor: Express.js
* Plataforma: Web (compatible con navegadores modernos)
* Fecha de implementación: Julio de 2025

#### 2. Requisitos del Sistema

**2.1 Requisitos de Hardware:**

* Procesador: Intel Core i5 o superior
* RAM: 8 GB o más
* Almacenamiento: 20 GB disponibles
* Conexión a Internet estable

**2.2 Requisitos de Software:**

* Sistema Operativo: Windows 10/11, Linux o macOS
* Navegador: Google Chrome, Mozilla Firefox, Microsoft Edge
* Node.js v18 o superior
* MongoDB v6 o superior

#### 3. Instalación del Sistema

**3.1 Instalación del Backend**

1. **Clonar el repositorio:  
    bash  
   git clone https://github.com/lesuho/Informe---Proyectos-2S**

**cd backend-inventario**

1. **Instalar dependencias:  
    bash  
   npm install**
2. **Configurar archivo .env con las variables necesarias (ejemplo):  
    ini  
   PORT=4000**

**DB\_URI=mongodb://localhost:27017/inventario**

**JWT\_SECRET=secreto123**

1. **Iniciar el servidor:  
    bash  
   npm run dev**

**3.2 Instalación del Frontend**

1. **Clonar el repositorio:  
    bash  
   git clone https://github.com/usuario/frontend-inventario.git**

**cd frontend-inventario**

1. **Instalar dependencias:  
    bash  
   npm install**
2. **Ubica el archivo .env.example en la raíz del proyecto, renómbralo como .env, guarda los cambios y asegúrate de que el backend esté en ejecución antes de iniciar el frontend.**
3. **Iniciar la aplicación:  
    bash  
   npm run dev**

#### 4. Estructura del Proyecto

**4.1 Backend**

* **controllers/:** Controladores de cada entidad
* **models/:** Esquemas de la base de datos
* **routes/:** Rutas de las API REST
* **middlewares/:** Autenticación y validaciones
* **config/:** Configuraciones del entorno

**4.2 Frontend**

* **components/:** Componentes reutilizables
* **pages/:** Páginas principales del sistema
* **services/:** Funciones para interactuar con la API
* **hooks/:** Hooks personalizados
* **assets/:** Imágenes y recursos estáticos

#### 5. Base de Datos

**5.1 Modelo de Datos Principal**

* Equipos: ID, nombre, categoría, estado, ubicación, fecha de ingreso
* Usuarios: ID, nombre, correo, rol, contraseña (encriptada)
* Historial de movimientos: ID, equipo, fecha, acción, responsable

**6. Seguridad**

* Autenticación mediante JWT
* Hash de contraseñas con bcrypt
* Control de accesos según roles de usuario (admin, técnico)

#### 7. Copias de Seguridad y Restauración

**7.1 Backup de la base de datos**

**bash**

**mongodump --db smartask --out ./backup**

**7.2 Restauración de la base de datos**

**bash**

**mongorestore --db smartask ./backup/smartask**

#### 8. Mantenimiento del Sistema

* Revisar errores en consola y logs.
* Verificar la integridad de la base de datos cada mes**.**

**8.1. Actualizar dependencias con regularidad:  
 bash**

**npm update**

**ANEXO 02. Manual de Usuario - Aplicación Smart Task**

1. **Introducción**

Smart Task es una aplicación para la gestión de tareas, roles de usuario, notificaciones y chat en tiempo real. Está compuesta por un backend (Node.js/Express) y un frontend (React), además de pruebas automatizadas y tests end-to-end con Cypress.

1. **Requisitos del Sistema**

* Node.js (v14 o superior recomendado)
* npm (v6 o superior)
* Navegador web moderno (Chrome, Firefox, Edge, etc.)

1. **Ejecución de la Aplicación**
   1. Iniciar el Backend

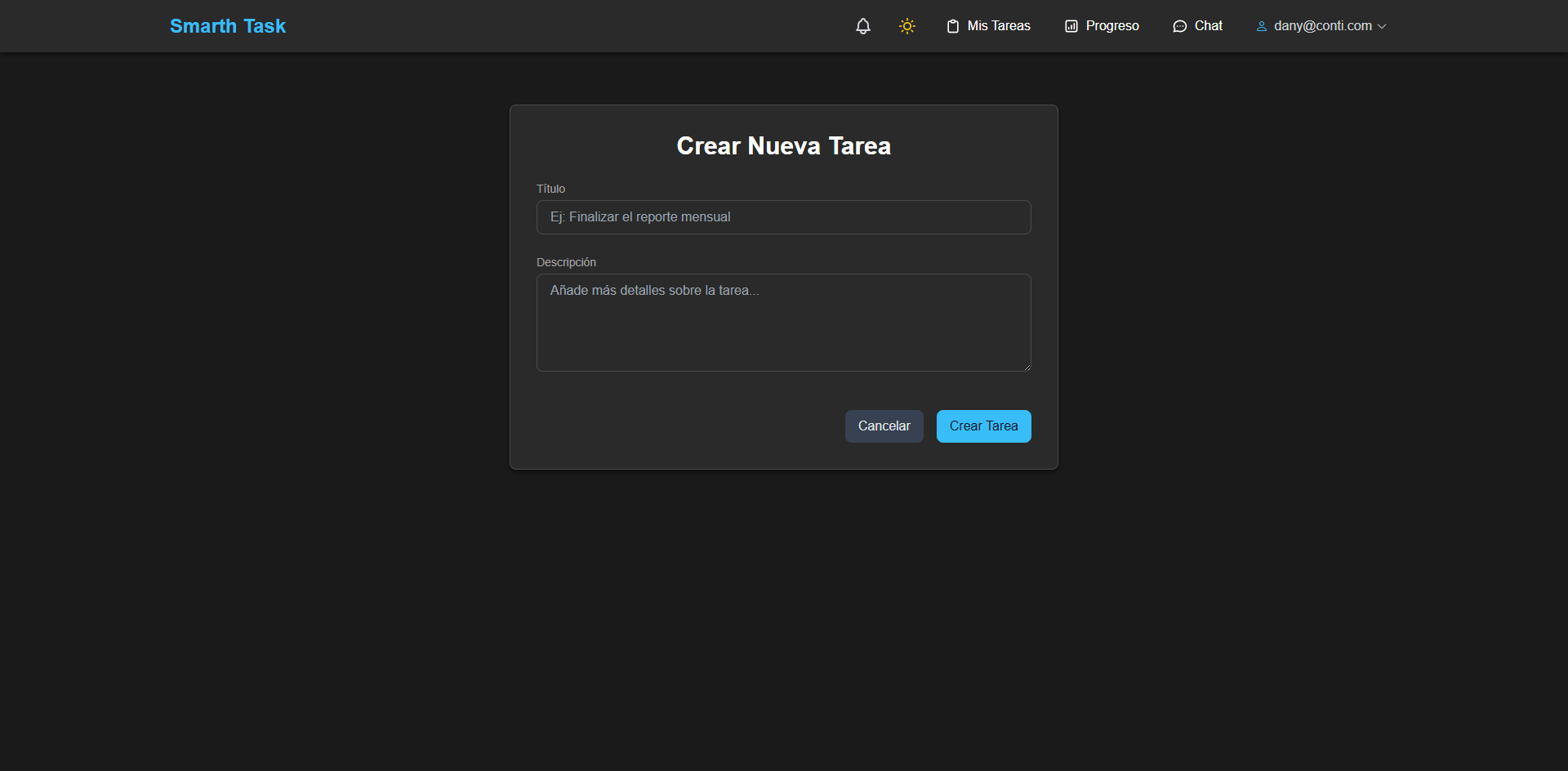
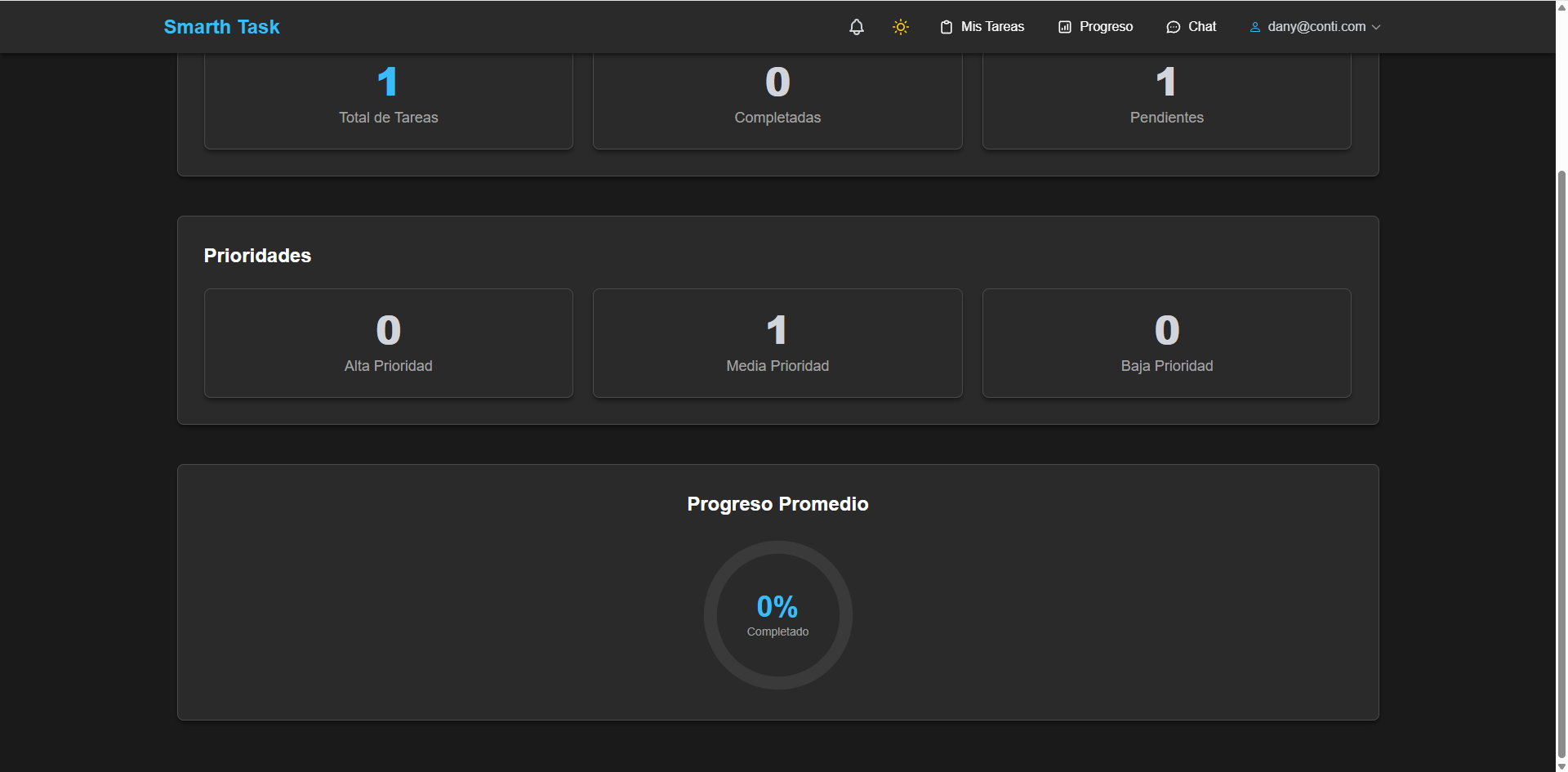
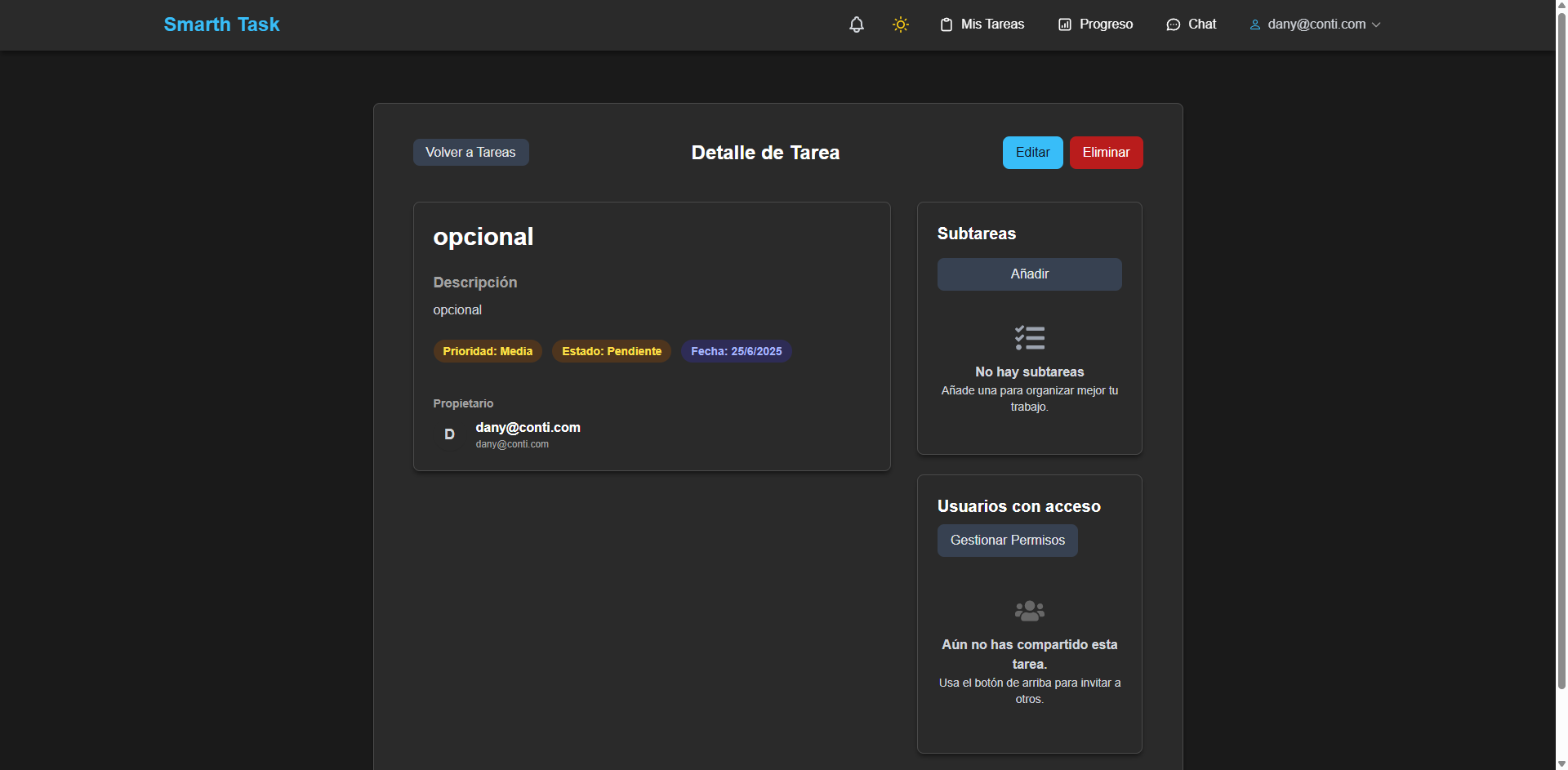
* Desde la carpeta backend:
* npm run dev
  1. Iniciar el Frontend
* Desde la carpeta frontend:
* npm start

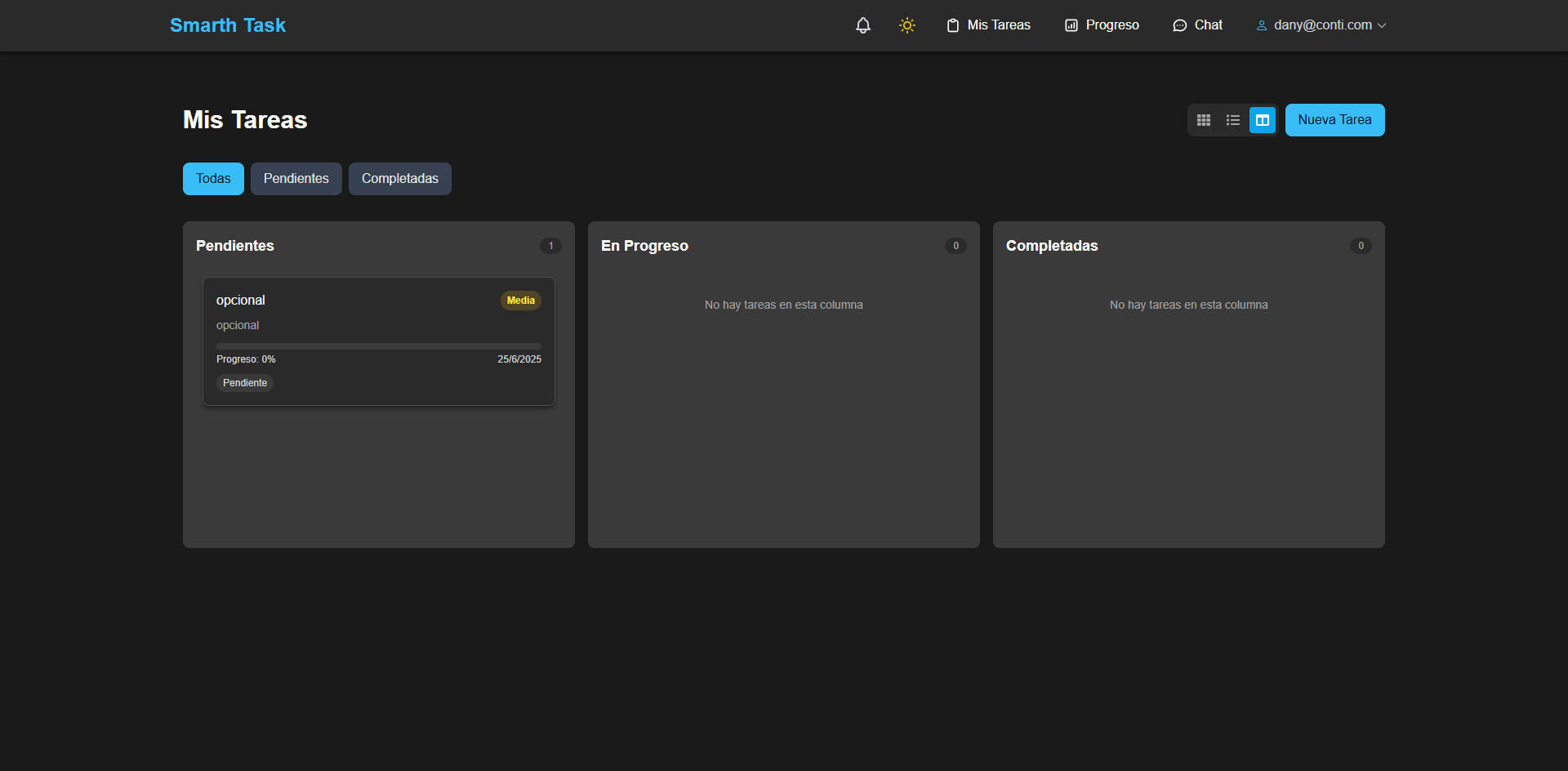
La aplicación estará disponible normalmente en http://localhost:3000.

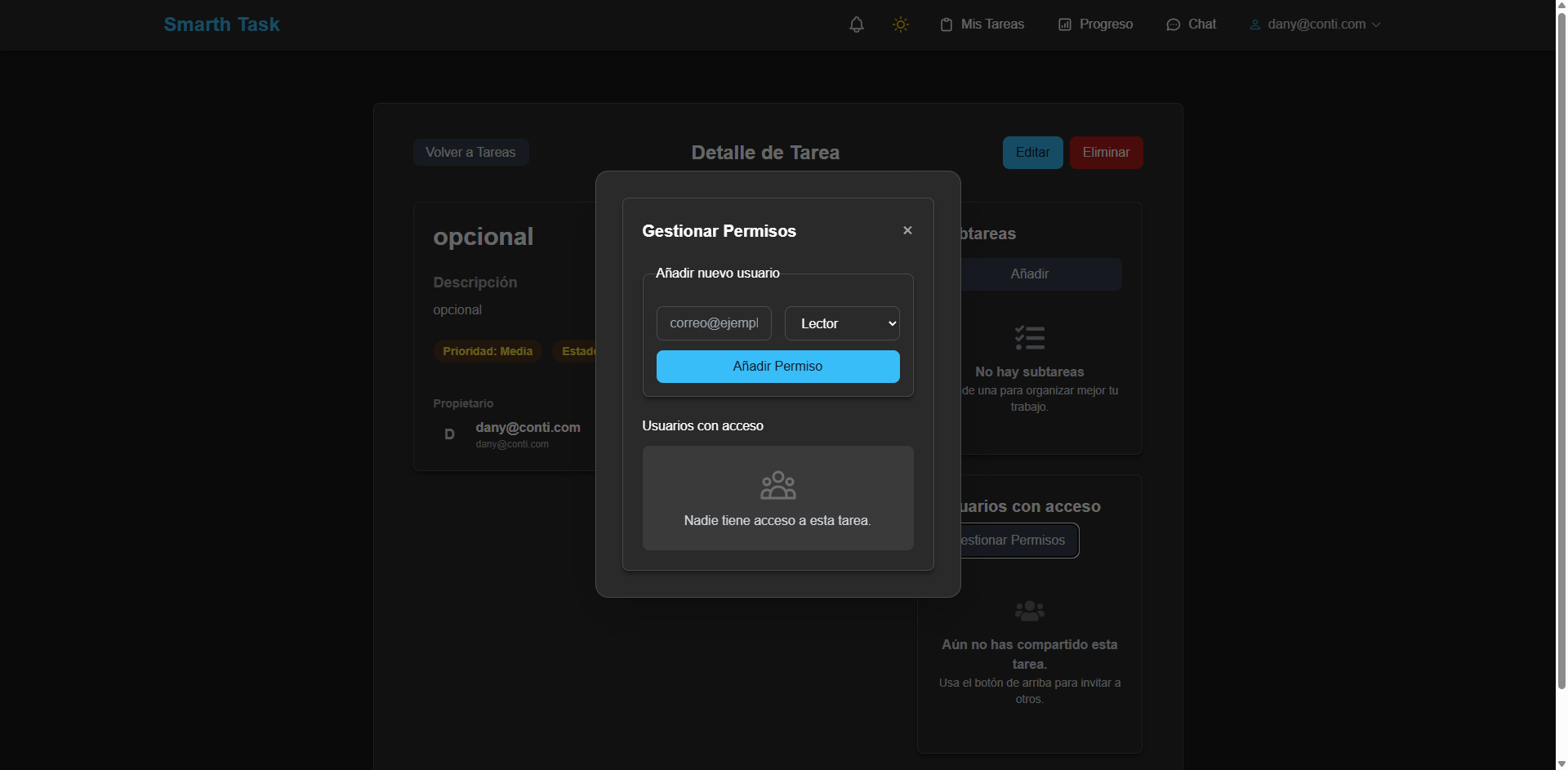
1. **Estructura de la Aplicación**

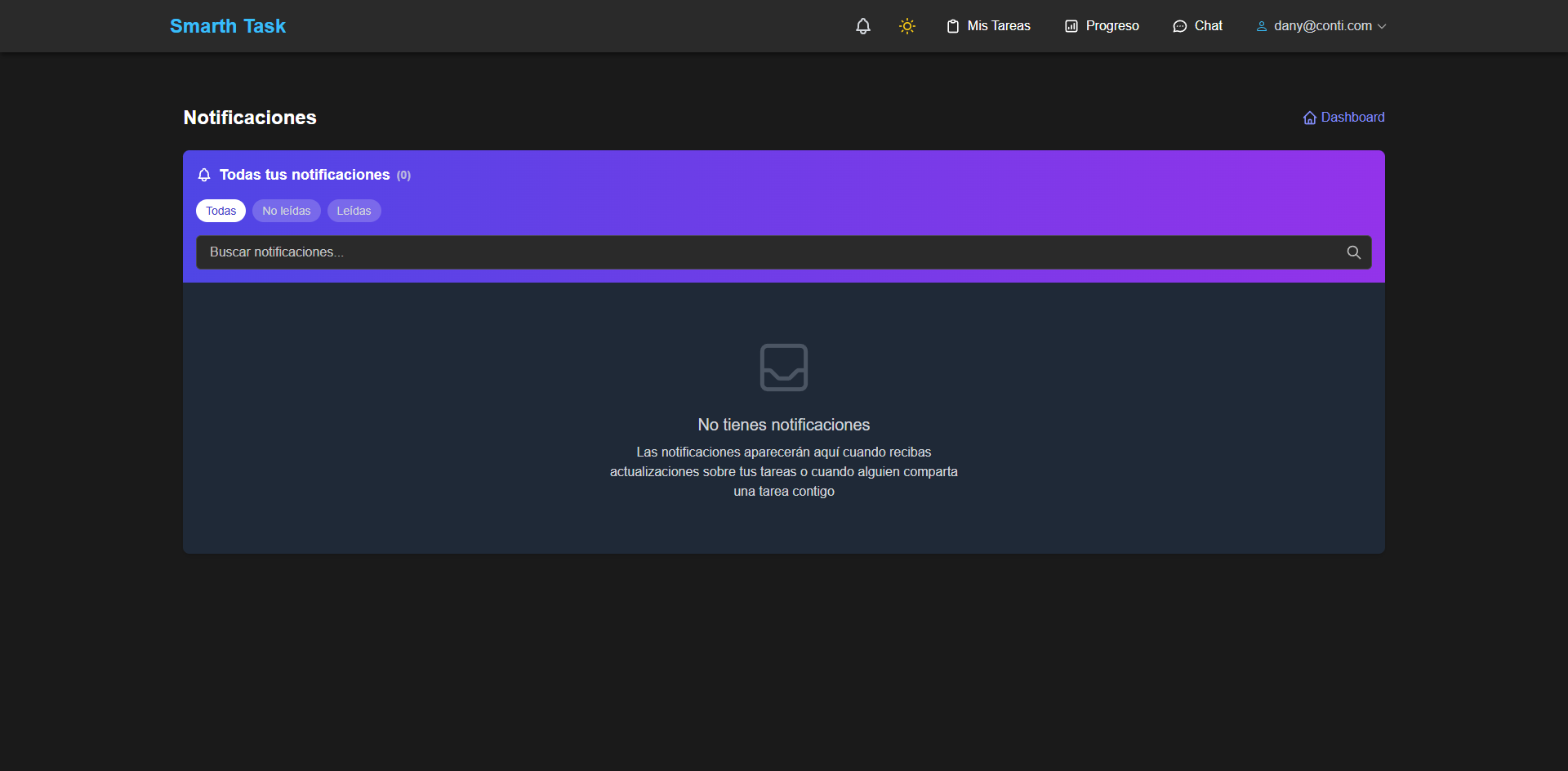
* backend/: Servidor, API REST, lógica de negocio, autenticación, chat, notificaciones, roles, tareas.
* frontend/: Interfaz de usuario (React), componentes, páginas, servicios.
* cypress/: Pruebas end-to-end automatizadas.
* package.json: Dependencias y scripts principales.

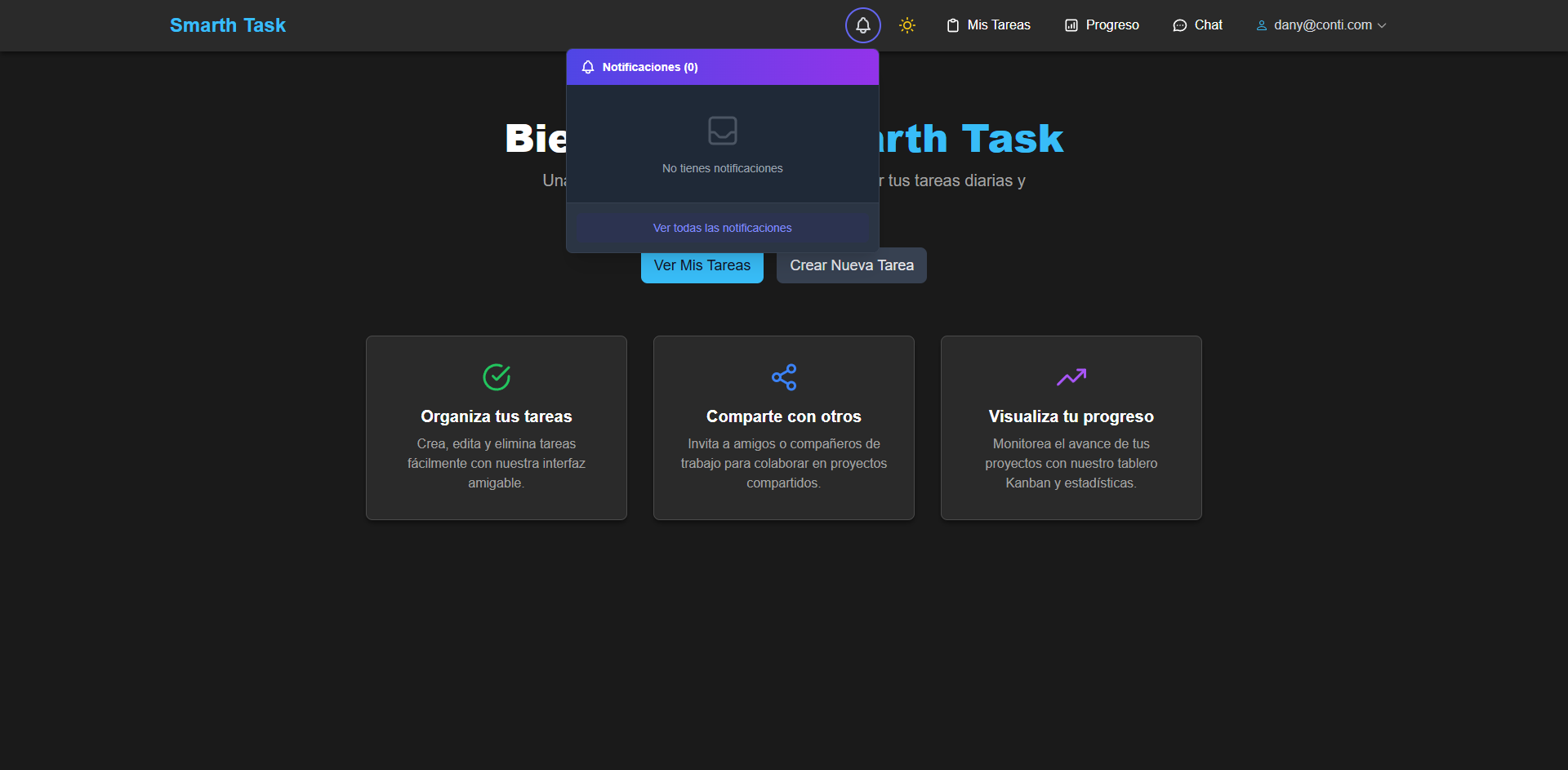
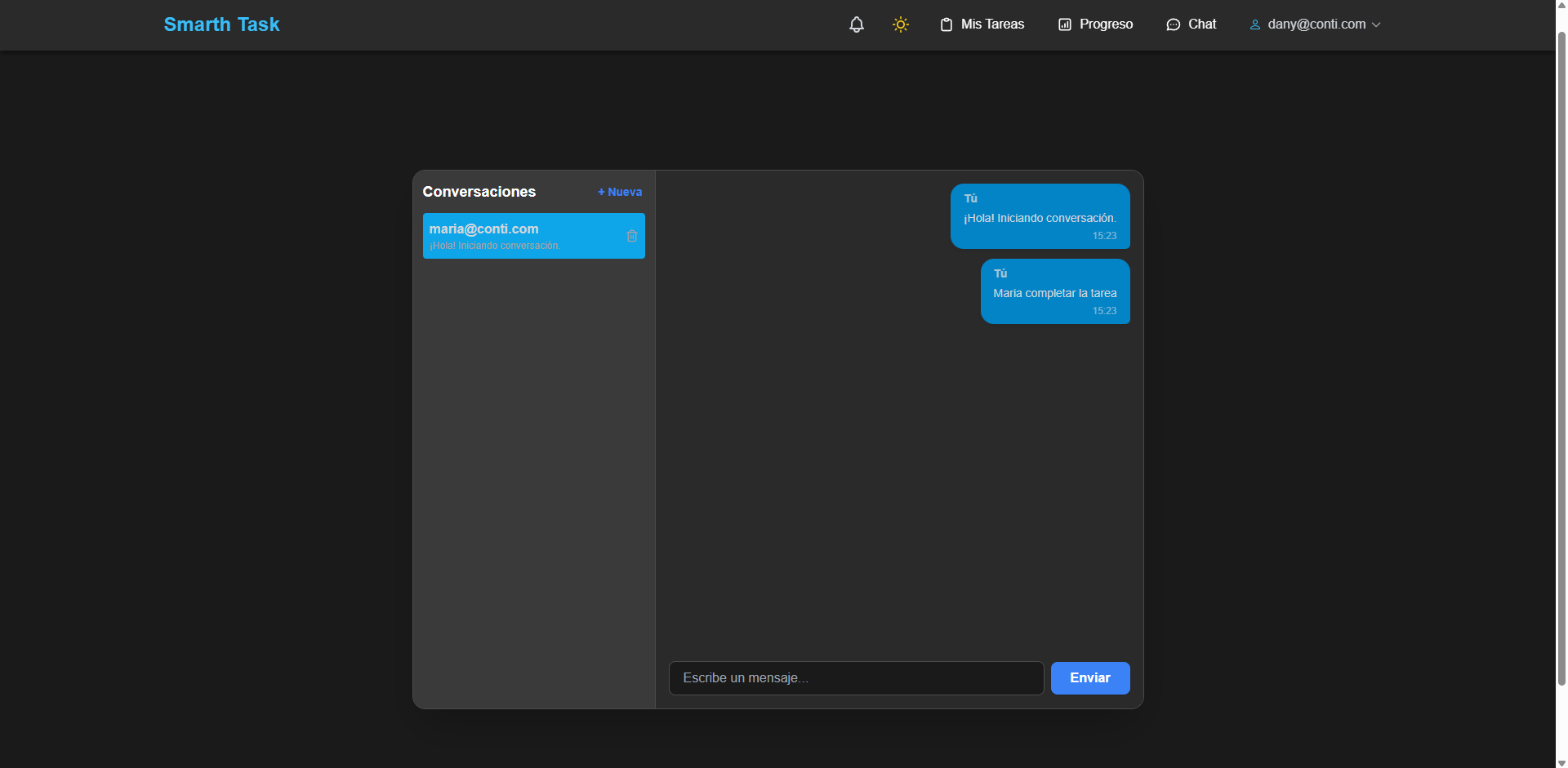
1. **Funcionalidades Principales**
   1. Gestión de Tareas

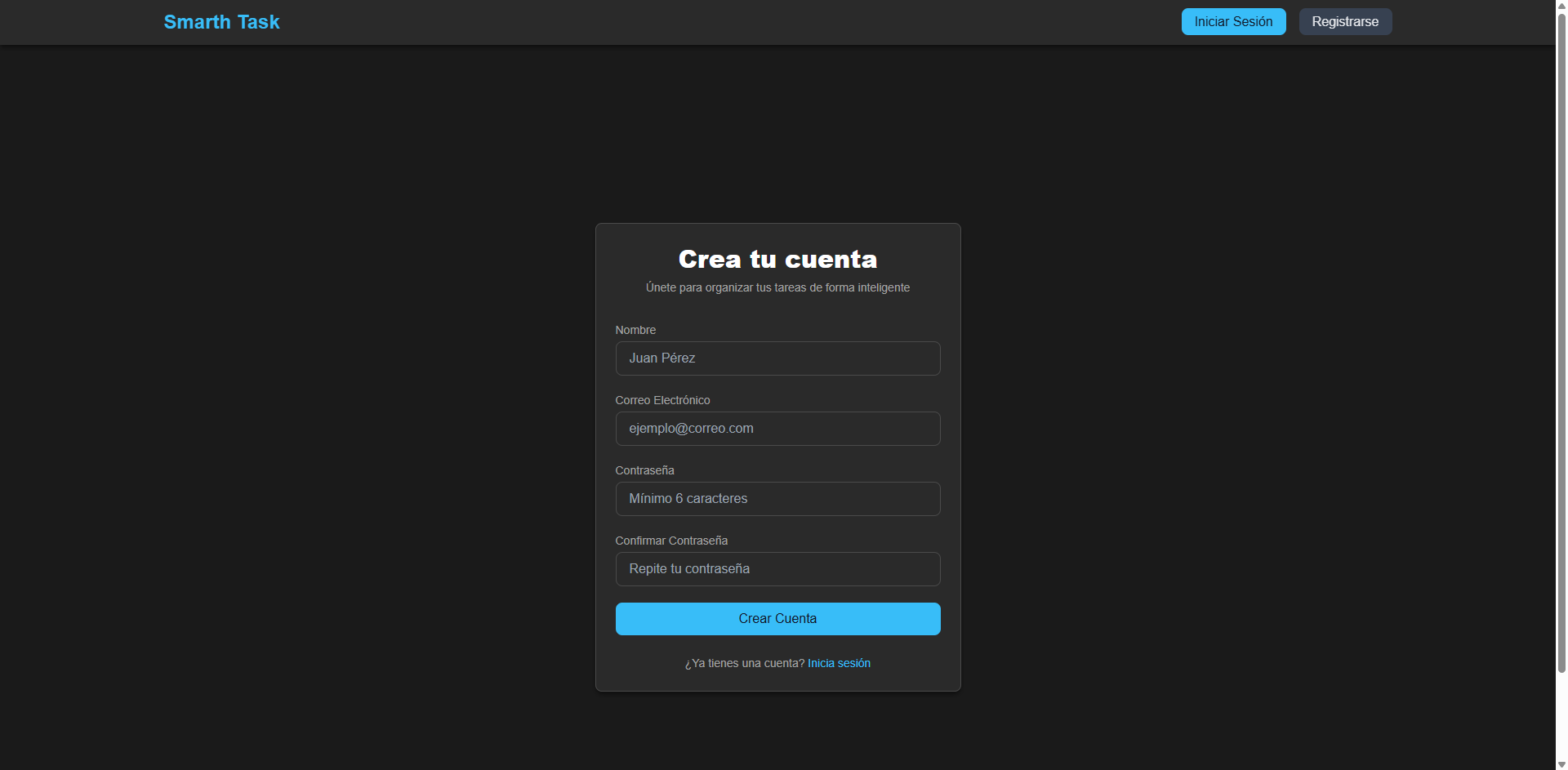
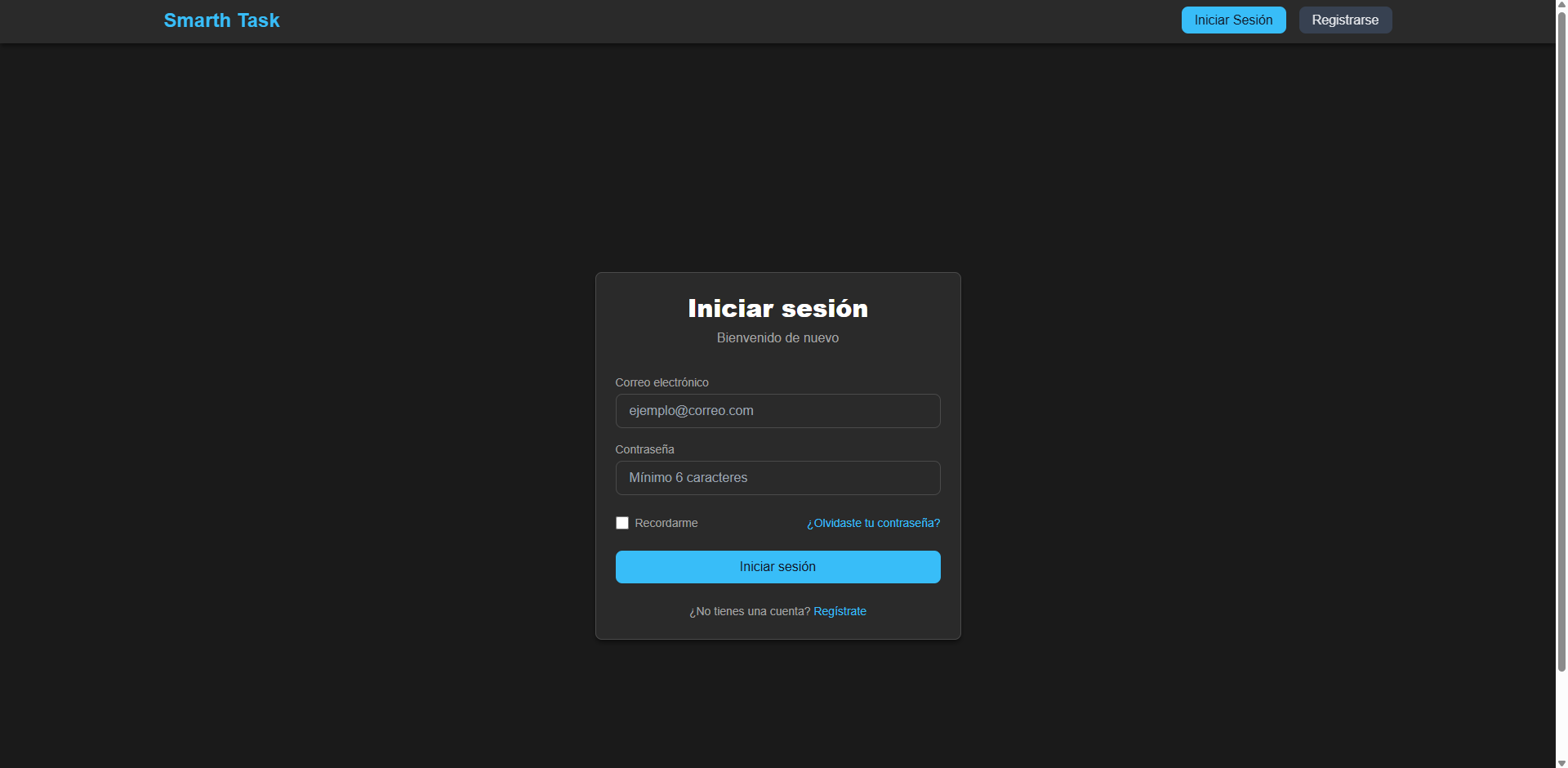
* Crear, editar, eliminar y visualizar tareas.
* Subtareas y seguimiento de progreso.
* Vista Kanban para organización visual.



* 1. Roles y Permisos
* Asignación de roles a usuarios.
* Gestión de permisos para tareas específicas.  
  
  1. Notificaciones
* Recibe notificaciones sobre cambios en tareas, mensajes, etc.



* Visualización de notificaciones en la interfaz.  
  
  1. Chat
* Conversaciones en tiempo real entre usuarios.
* Historial de mensajes.
  1. Autenticación
* Registro y login de usuarios.



* Gestión de sesiones y cookies.

1. **Pruebas y Testing**
   1. Pruebas Unitarias - Integración (Backend)

* Desde la carpeta backend:
* npm test
  1. Pruebas Unitarias - Integración (Frontend)
* Desde la carpeta frontend:
* npm test
  1. Pruebas End-to-End (Cypress)
* Desde la raíz del proyecto o carpeta cypress:
* npm run cypress:open

1. **Preguntas Frecuentes**

* ¿No puedo iniciar sesión?

Verifica que el backend esté corriendo y que la base de datos esté correctamente configurada.

* ¿No se ven las tareas o notificaciones?

Asegúrate de que tanto el frontend como el backend estén activos y que no haya errores en la consola.