DOCUMENTO DE REQUERIMIENTOS DEL SISTEMA

ProGest

Sistema de Gestión de Proyectos SaaS

Especificación de Requerimientos de Software (SRS)

Versión: 2.0.0
Fecha: 24 de Febrero de 2026

Preparado por:
Equipo de Desarrollo ProGest

Clasificación:
Confidencial - Uso evidencial

CONTROL DE VERSIONES

Versión	Fecha	Autor	Descripción
1.0.0	15/01/2026	Equipo Dev	Versión inicial del documento
1.5.0	10/02/2026	Equipo Dev	Agregados requerimientos de comentarios y notificaciones
2.0.0	24/02/2026	Equipo Dev	Versión completa con 70+ requerimientos

TABLA DE CONTENIDOS

1. INTRODUCCIÓN

1.1 Propósito

Este documento especifica los requerimientos funcionales y no funcionales del sistema ProGest, una plataforma SaaS (Software as a Service) para la gestión integral de proyectos empresariales. El documento está dirigido a:

- Equipo de desarrollo y arquitectura de software
- Gerentes de proyecto y product owners
- Stakeholders y tomadores de decisiones
- Equipos de QA y testing
- Equipos de soporte técnico y operaciones

1.2 Alcance

ProGest es un sistema de gestión de proyectos multitenant que permite a organizaciones de cualquier tamaño gestionar proyectos, tareas, equipos y colaboración de manera eficiente y segura.

Objetivos principales:

- Proporcionar una plataforma centralizada para la gestión de proyectos
- Facilitar la colaboración entre equipos distribuidos
- Automatizar flujos de trabajo y notificaciones
- Garantizar seguridad y aislamiento de datos entre organizaciones
- Ofrecer visibilidad completa del estado de proyectos y tareas

Funcionalidades clave:

- Sistema de autenticación y autorización basado en roles
- Gestión completa de proyectos y tareas
- Sistema de invitaciones y gestión de equipos
- Notificaciones en tiempo real
- Sistema de comentarios y colaboración
- Panel de administración para superusuarios
- API REST completa para integraciones

Fuera del alcance (versión actual):

- Integración con servicios de terceros (Slack, Microsoft Teams)
- Sistema de archivos adjuntos
- Facturación y pagos
- Aplicaciones móviles nativas
- Videoconferencias integradas

1.3 Definiciones y Acrónimos

Término	Definición
SaaS	Software as a Service - Modelo de distribución de software basado en la nube
Multitenant	Arquitectura donde múltiples clientes comparten la misma infraestructura pero con datos aislados
JWT	JSON Web Token - Estándar para tokens de autenticación
API	Application Programming Interface - Interfaz de programación de aplicaciones
REST	Representational State Transfer - Estilo arquitectónico para APIs
CRUD	Create, Read, Update, Delete - Operaciones básicas de datos
Owner	Usuario propietario de un proyecto con permisos completos
Employee	Usuario empleado con permisos limitados
SUPERADMIN	Usuario administrador del sistema con acceso total
ORM	Object-Relational Mapping - Mapeo objeto-relacional
UUID	Universally Unique Identifier - Identificador único universal

Término	Definición

1.4 Referencias

- **Documentación técnica:** `.kiro/DOCUMENTACION_CONSOLIDADA.md`
- **Arquitectura del sistema:** `.kiro/architecture.md`
- **Modelo de datos:** `.kiro/data-model.md`
- **Documentación de API:** `.kiro/API_COMPLETE_DOCUMENTATION.md`
- **Guía de desarrollo:** `.kiro/development-guide.md`

1.5 Visión General

Este documento está organizado en las siguientes secciones:

- **Sección 2:** Describe el contexto general del sistema, sus funciones principales y características de usuarios
- **Sección 3:** Detalla los requerimientos funcionales organizados por módulos
- **Sección 4:** Especifica los requerimientos no funcionales de calidad
- **Sección 5:** Define las interfaces externas del sistema
- **Sección 6:** Incluye apéndices con información complementaria

---

2. DESCRIPCIÓN GENERAL

2.1 Perspectiva del Producto

ProGest es un sistema independiente de gestión de proyectos que opera como una aplicación web SaaS. El sistema está compuesto por:

Componentes principales:

1. **Frontend Web (Next.js 14)**
   - Aplicación de página única (SPA)
   - Interfaz responsive y moderna
   - Comunicación con backend vía API REST
2. **Backend API (Flask)**
   - API REST con 41 endpoints
   - Lógica de negocio centralizada
   - Gestión de autenticación y autorización
3. **Base de Datos (MySQL 8.0)**
   - Almacenamiento persistente
   - 8 entidades principales
   - Soporte para transacciones ACID

Arquitectura del sistema:

```
┌──────────────┐
│   Cliente    │ (Navegador Web)
└──────┬───────┘
       │ HTTPS
       ▼
┌──────────────┐
│  Frontend    │ (Next.js - Puerto 3000)
└──────┬───────┘
       │ REST API
       ▼
┌──────────────┐
│   Backend    │ (Flask - Puerto 5000)
└──────┬───────┘
       │ SQL
       ▼
┌──────────────┐
│  Database    │ (MySQL - Puerto 3306)
└──────────────┘
```

2.2 Funciones del Producto

El sistema ProGest proporciona las siguientes funciones principales:

F1. Gestión de Usuarios y Autenticación

- Registro de nuevos usuarios (Owner/Employee)

- Inicio de sesión con credenciales
- Gestión de sesiones con JWT
- Recuperación de contraseñas
- Perfiles de usuario enriquecidos

F2. Gestión de Proyectos

- Creación de proyectos (1 por Owner)
- Configuración de proyectos
- Categorización de proyectos
- Activación/desactivación de proyectos

F3. Gestión de Tareas

- Creación, edición y eliminación de tareas
- Asignación de tareas a empleados
- Estados de tareas (pending, in_progress, blocked, done)
- Prioridades (low, medium, high, urgent)
- Fechas de vencimiento
- Tags personalizados
- Filtros y búsqueda avanzada
- Estadísticas de tareas

F4. Gestión de Equipos

- Invitación de empleados por email
- Aceptación de invitaciones con token
- Gestión de miembros del equipo
- Perfiles enriquecidos de empleados
- Desactivación de miembros

F5. Sistema de Notificaciones

- Notificaciones en tiempo real
- Tipos variados (asignación, comentarios, cambios)
- Contador de notificaciones no leídas
- Marcado de leídas/no leídas
- Eliminación de notificaciones

F6. Sistema de Comentarios

- Comentarios en tareas
- Edición y eliminación de comentarios
- Asociación con usuarios
- Historial de comentarios

F7. Panel de Administración

- Gestión de todos los usuarios
- Gestión de todos los proyectos
- Estadísticas globales del sistema
- Logs de auditoría
- Health check del sistema

2.3 Características de los Usuarios

El sistema soporta tres tipos de usuarios con diferentes niveles de acceso:

OWNER (Propietario)

- **Descripción:** Dueño de un proyecto, típicamente gerente o líder de equipo
- **Responsabilidades:** Gestión completa del proyecto, creación de tareas, invitación de empleados
- **Nivel técnico:** Medio - Familiarizado con herramientas de gestión de proyectos
- **Frecuencia de uso:** Diaria
- **Permisos:**
    - Crear y gestionar su proyecto
    - Crear, editar y eliminar tareas
    - Invitar y gestionar empleados
    - Ver todas las tareas del proyecto
    - Acceder a estadísticas del proyecto

EMPLOYEE (Empleado)

- **Descripción:** Miembro del equipo asignado a tareas específicas
- **Responsabilidades:** Completar tareas asignadas, actualizar estados, comentar
- **Nivel técnico:** Básico a medio
- **Frecuencia de uso:** Diaria
- **Permisos:**
    - Ver tareas asignadas
    - Actualizar estado de sus tareas
    - Comentar en tareas
    - Ver miembros del equipo
    - Recibir notificaciones

SUPERADMIN (Superadministrador)

- **Descripción:** Administrador del sistema con acceso total
- **Responsabilidades:** Gestión global del sistema, soporte, monitoreo
- **Nivel técnico:** Alto - Conocimientos técnicos avanzados

- **Frecuencia de uso:** Según necesidad
- **Permisos:**
  - Acceso a todos los proyectos
  - Gestión de todos los usuarios
  - Ver logs de auditoría
  - Cambiar estados de usuarios/proyectos
  - Acceder a estadísticas globales

2.4 Restricciones

Restricciones técnicas:

- **RT-1:** El sistema debe ejecutarse en navegadores modernos (Chrome 90+, Firefox 88+, Safari 14+, Edge 90+)
- **RT-2:** El backend debe ser compatible con Python 3.10 o superior
- **RT-3:** La base de datos debe ser MySQL 8.0 o superior
- **RT-4:** El sistema debe soportar al menos 100 usuarios concurrentes
- **RT-5:** Los tokens JWT deben expirar en 15 minutos (access) y 7 días (refresh)

Restricciones de negocio:

- **RN-1:** Un Owner solo puede tener un proyecto activo
- **RN-2:** Las tareas deben tener fecha de vencimiento obligatoria
- **RN-3:** Las invitaciones expiran en 7 días
- **RN-4:** Las invitaciones pueden reenviarse máximo 3 veces
- **RN-5:** Los empleados solo pueden ver tareas asignadas a ellos

Restricciones regulatorias:

- **RR-1:** El sistema debe cumplir con GDPR para protección de datos personales
- **RR-2:** Las contraseñas deben almacenarse hasheadas con bcrypt
- **RR-3:** Debe existir trazabilidad de todas las acciones importantes (audit logs)
- **RR-4:** Los datos de diferentes proyectos deben estar completamente aislados

2.5 Suposiciones y Dependencias

Suposiciones:

- **S-1:** Los usuarios tienen acceso a internet estable
- **S-2:** Los usuarios tienen conocimientos básicos de gestión de proyectos
- **S-3:** Los usuarios utilizan dispositivos con pantallas de al menos 1024x768
- **S-4:** El servidor de base de datos está disponible 24/7
- **S-5:** Los usuarios tienen direcciones de email válidas

Dependencias:

- **D-1:** Servidor MySQL 8.0+ para almacenamiento de datos
- **D-2:** Servidor SMTP para envío de emails de invitación
- **D-3:** Certificado SSL/TLS para conexiones HTTPS
- **D-4:** Librerías de terceros (ver requirements.txt y package.json)
- **D-5:** Infraestructura de hosting (servidor web, dominio)

---

3. REQUERIMIENTOS FUNCIONALES

3.1 Autenticación y Autorización

RF-001: Registro de Usuario Owner

Prioridad: Alta
Descripción: El sistema debe permitir el registro de nuevos usuarios con rol Owner.

Criterios de aceptación:

- El usuario debe proporcionar: email, contraseña, nombre completo
- El email debe ser único en el sistema
- La contraseña debe tener mínimo 8 caracteres
- El sistema debe validar formato de email
- Al registrarse, se debe crear automáticamente un proyecto vacío
- El sistema debe generar tokens JWT (access y refresh)
- El usuario debe ser redirigido a la página de onboarding

Endpoint: `POST /api/auth/register`

RF-002: Inicio de Sesión

Prioridad: Alta
Descripción: El sistema debe permitir a usuarios registrados iniciar sesión.

Criterios de aceptación:

- El usuario debe proporcionar email y contraseña
- El sistema debe validar las credenciales contra la base de datos
- Las contraseñas deben compararse usando bcrypt
- Si las credenciales son correctas, generar tokens JWT
- El token de acceso debe expirar en 15 minutos
- El token de refresh debe expirar en 7 días
- El sistema debe retornar información del usuario y proyecto
- Los usuarios desactivados no deben poder iniciar sesión

Endpoint: `POST /api/auth/login`

RF-003: Renovación de Token

Prioridad: Alta
Descripción: El sistema debe permitir renovar el token de acceso usando el refresh token.

Criterios de aceptación:

- El usuario debe proporcionar un refresh token válido
- El sistema debe validar que el refresh token no haya expirado
- Si es válido, generar un nuevo access token
- El nuevo access token debe tener 15 minutos de validez
- El refresh token no debe cambiar
- Si el refresh token es inválido, retornar error 401

Endpoint: `POST /api/auth/refresh`

RF-004: Cierre de Sesión

Prioridad: Media
Descripción: El sistema debe permitir a los usuarios cerrar sesión.

Criterios de aceptación:

- El usuario debe estar autenticado
- El sistema debe invalidar el token actual
- El frontend debe eliminar tokens del almacenamiento local
- El usuario debe ser redirigido a la página de login

Endpoint: `POST /api/auth/logout`

RF-005: Obtener Información del Usuario Actual

Prioridad: Alta
Descripción: El sistema debe permitir obtener la información del usuario autenticado.

Criterios de aceptación:

- El usuario debe estar autenticado (token válido)
- El sistema debe retornar: id, email, nombre, rol, avatar, estado
- Si el usuario es Owner o Employee, incluir información del proyecto
- Si el token es inválido, retornar error 401

Endpoint: `GET /api/auth/me`

RF-006: Aceptar Invitación

Prioridad: Alta
Descripción: El sistema debe permitir a usuarios invitados aceptar invitaciones y crear su cuenta.

Criterios de aceptación:

- El usuario debe proporcionar: token de invitación, contraseña, nombre
- El sistema debe validar que el token existe y no ha expirado
- El sistema debe validar que la invitación está en estado "pending"
- Crear nuevo usuario con rol EMPLOYEE
- Copiar datos de perfil de la invitación al usuario
- Crear membership asociando usuario con proyecto
- Cambiar estado de invitación a "accepted"
- Generar tokens JWT para el nuevo usuario
- Crear notificación para el Owner

Endpoint: `POST /api/auth/accept-invite`

RF-007: Validación de Token de Invitación

Prioridad: Media
Descripción: El sistema debe permitir validar un token de invitación antes de aceptarlo.

Criterios de aceptación:

- El usuario debe proporcionar el token
- El sistema debe verificar que el token existe
- El sistema debe verificar que no ha expirado
- El sistema debe verificar que está en estado "pending"
- Retornar información de la invitación (email, proyecto, datos de perfil)
- Si el token es inválido o expirado, retornar error apropiado

Endpoint: `GET /api/invites/validate/{token}`

RF-008: Control de Acceso Basado en Roles (RBAC)

Prioridad: Alta
Descripción: El sistema debe implementar control de acceso basado en roles.

Criterios de aceptación:

- Cada endpoint debe verificar el rol del usuario
- Los decoradores @require_roles deben validar permisos
- SUPERADMIN tiene acceso a todos los recursos
- OWNER tiene acceso completo a su proyecto
- EMPLOYEE tiene acceso limitado a tareas asignadas
- Retornar error 403 si el usuario no tiene permisos

RF-009: Aislamiento Multitenant

Prioridad: Crítica
Descripción: El sistema debe garantizar aislamiento completo de datos entre proyectos.

Criterios de aceptación:

- Todas las queries deben filtrar por project_id
- Los tokens JWT deben incluir project_id en los claims
- Un usuario no debe poder acceder a datos de otros proyectos
- Las validaciones de permisos deben verificar pertenencia al proyecto
- Los logs de auditoría deben registrar intentos de acceso no autorizado

---

RF-010: Gestión de Sesiones

Prioridad: Alta
Descripción: El sistema debe gestionar sesiones de usuario de forma segura.

Criterios de aceptación:

- Las sesiones deben basarse en JWT stateless
- No se deben almacenar sesiones en el servidor
- Los tokens deben incluir: user_id, role, project_id, exp
- El frontend debe almacenar tokens en localStorage
- El frontend debe incluir token en header Authorization: Bearer
- El frontend debe renovar automáticamente tokens expirados

RF-011: Validación de Contraseñas

Prioridad: Alta
Descripción: El sistema debe validar la fortaleza de contraseñas.

Criterios de aceptación:

- Mínimo 8 caracteres
- Al menos una letra mayúscula
- Al menos una letra minúscula
- Al menos un número
- Caracteres especiales opcionales
- Mostrar indicador de fortaleza en el frontend

RF-012: Hash de Contraseñas

Prioridad: Crítica
Descripción: El sistema debe almacenar contraseñas de forma segura.

Criterios de aceptación:

- Usar bcrypt para hashear contraseñas
- Generar salt automático por contraseña
- Nunca almacenar contraseñas en texto plano
- Nunca retornar contraseñas en respuestas de API
- Usar cost factor de al menos 12 para bcrypt

RF-013: Protección contra Fuerza Bruta

Prioridad: Alta
Descripción: El sistema debe proteger contra ataques de fuerza bruta.

Criterios de aceptación:

- Limitar intentos de login a 5 por minuto por IP
- Bloquear temporalmente después de 5 intentos fallidos
- Registrar intentos fallidos en audit logs
- Mostrar mensaje genérico de error (no revelar si email existe)

RF-014: Recuperación de Contraseña

Prioridad: Media
Descripción: El sistema debe permitir recuperar contraseñas olvidadas.

Criterios de aceptación:

- El usuario debe proporcionar su email
- El sistema debe enviar email con token de recuperación
- El token debe expirar en 1 hora
- El usuario debe poder establecer nueva contraseña con el token
- Invalidar token después de usarlo
- Notificar al usuario por email del cambio de contraseña

Estado: Pendiente de implementación

RF-015: Autenticación de Dos Factores (2FA)

Prioridad: Baja
Descripción: El sistema debe soportar autenticación de dos factores opcional.

Criterios de aceptación:

- El usuario debe poder habilitar/deshabilitar 2FA
- Soportar TOTP (Time-based One-Time Password)
- Generar códigos QR para configuración
- Validar código 2FA en login
- Proporcionar códigos de respaldo

Estado: Pendiente de implementación

3.2 Gestión de Proyectos

RF-016: Creación de Proyecto

Prioridad: Alta
Descripción: El sistema debe permitir a un Owner crear su proyecto durante el onboarding.

Criterios de aceptación:

- Solo usuarios con rol OWNER pueden crear proyectos
- Un Owner solo puede tener un proyecto
- El usuario debe proporcionar: nombre, descripción, categoría
- El nombre del proyecto es obligatorio
- El sistema debe crear automáticamente una membership para el Owner
- El proyecto debe crearse con estado "active"
- Registrar acción en audit logs

Endpoint: `POST /api/projects`

RF-017: Obtener Proyecto del Usuario

Prioridad: Alta
Descripción: El sistema debe permitir obtener el proyecto del usuario autenticado.

Criterios de aceptación:

- El usuario debe estar autenticado
- OWNER debe ver su proyecto
- EMPLOYEE debe ver el proyecto al que pertenece
- SUPERADMIN puede ver cualquier proyecto
- Retornar: id, nombre, descripción, categoría, owner_id, estado, fechas
- Si el usuario no tiene proyecto, retornar error 404

Endpoint: `GET /api/projects/my-project`

RF-018: Actualización de Proyecto

Prioridad: Media
Descripción: El sistema debe permitir al Owner actualizar su proyecto.

Criterios de aceptación:

- Solo el Owner del proyecto puede actualizarlo
- Campos actualizables: nombre, descripción, categoría
- Validar que el nombre no esté vacío
- Actualizar campo updated_at automáticamente
- Registrar cambios en audit logs

Estado: Pendiente de implementación

RF-019: Cambio de Estado de Proyecto

Prioridad: Media
Descripción: El sistema debe permitir activar/desactivar proyectos.

Criterios de aceptación:

- Solo OWNER o SUPERADMIN pueden cambiar estado
- Estados válidos: active, disabled
- Al desactivar, los usuarios no pueden acceder al proyecto
- Las tareas del proyecto quedan inaccesibles
- Registrar cambio en audit logs

Endpoint: `PATCH /api/admin/projects/{id}/status` (SUPERADMIN)

RF-020: Categorización de Proyectos

Prioridad: Baja
Descripción: El sistema debe permitir categorizar proyectos.

Criterios de aceptación:

- Categorías predefinidas: Desarrollo, Marketing, Ventas, Operaciones, Otro
- El Owner puede seleccionar una categoría
- La categoría es opcional
- Usar categoría para filtros y estadísticas

RF-021: Estadísticas del Proyecto

Prioridad: Media
Descripción: El sistema debe proporcionar estadísticas del proyecto.

Criterios de aceptación:

- Total de tareas
- Tareas por estado (pending, in_progress, blocked, done)
- Tareas por prioridad
- Tareas vencidas
- Tareas por vencer hoy/esta semana
- Tareas sin asignar
- Solo OWNER puede ver estadísticas de su proyecto

Endpoint: `GET /api/tasks/stats`

RF-022: Listado de Proyectos (SUPERADMIN)

Prioridad: Media
Descripción: El sistema debe permitir al SUPERADMIN listar todos los proyectos.

Criterios de aceptación:

- Solo SUPERADMIN tiene acceso

- Soportar paginación (10 proyectos por página)
- Soportar filtros: estado, categoría, búsqueda por nombre
- Retornar: id, nombre, owner, categoría, estado, fechas
- Incluir contador de tareas y miembros

Endpoint: `GET /api/admin/projects`

RF-023: Eliminación de Proyecto

Prioridad: Baja
Descripción: El sistema debe permitir eliminar proyectos.

Criterios de aceptación:

- Solo SUPERADMIN puede eliminar proyectos
- Confirmar eliminación (requiere confirmación explícita)
- Eliminar en cascada: memberships, tasks, invites, notifications, comments
- Registrar eliminación en audit logs
- No se puede recuperar después de eliminar

Estado: Pendiente de implementación

RF-024: Transferencia de Propiedad

Prioridad: Baja
Descripción: El sistema debe permitir transferir propiedad de un proyecto.

Criterios de aceptación:

- Solo el Owner actual puede transferir
- El nuevo owner debe ser un Employee del proyecto
- Cambiar rol del Owner actual a EMPLOYEE
- Cambiar rol del nuevo owner a OWNER
- Actualizar owner_id del proyecto
- Notificar a ambos usuarios
- Registrar transferencia en audit logs

Estado: Pendiente de implementación

RF-025: Duplicación de Proyecto

Prioridad: Baja
Descripción: El sistema debe permitir duplicar proyectos como plantilla.

Criterios de aceptación:

- Solo OWNER puede duplicar su proyecto
- Copiar estructura de tareas (sin asignaciones)
- Copiar categorías y configuraciones
- No copiar comentarios ni notificaciones
- Generar nuevo project_id
- Marcar como "Copia de [nombre original]"

Estado: Pendiente de implementación

3.3 Gestión de Tareas

RF-026: Creación de Tarea

Prioridad: Alta
Descripción: El sistema debe permitir al Owner crear nuevas tareas en su proyecto.

Criterios de aceptación:

- Solo OWNER puede crear tareas
- Campos obligatorios: título, due_date
- Campos opcionales: descripción, prioridad, assigned_to, start_date, tags, checklist
- Prioridad por defecto: medium
- Estado inicial: pending
- La fecha de vencimiento debe ser futura
- Validar que assigned_to sea un miembro del proyecto
- Crear notificación si se asigna a un empleado
- Registrar creación en audit logs

Endpoint: `POST /api/tasks`

RF-027: Listado de Tareas

Prioridad: Alta
Descripción: El sistema debe permitir listar tareas según el rol del usuario.

Criterios de aceptación:

- OWNER ve todas las tareas de su proyecto
- EMPLOYEE ve solo tareas asignadas a él
- SUPERADMIN ve todas las tareas
- Soportar filtros: estado, prioridad, assigned_to, búsqueda por título
- Soportar ordenamiento: due_date, created_at, priority
- Soportar paginación (10 tareas por página)
- Retornar contador total de tareas

Endpoint: `GET /api/tasks`

RF-028: Obtener Detalle de Tarea

Prioridad: Alta
Descripción: El sistema debe permitir obtener el detalle completo de una tarea.

Criterios de aceptación:

- OWNER puede ver cualquier tarea de su proyecto
- EMPLOYEE solo puede ver tareas asignadas a él
- SUPERADMIN puede ver cualquier tarea
- Retornar todos los campos de la tarea
- Incluir nombre del creador y asignado
- Incluir contador de comentarios
- Si el usuario no tiene acceso, retornar error 403

Endpoint: `GET /api/tasks/{id}`

RF-029: Actualización de Tarea

Prioridad: Alta
Descripción: El sistema debe permitir actualizar tareas según permisos.

Criterios de aceptación:

- OWNER puede actualizar cualquier campo de cualquier tarea
- EMPLOYEE solo puede actualizar estado de tareas asignadas a él
- Campos actualizables por OWNER: título, descripción, prioridad, estado, assigned_to, due_date, start_date, tags, checklist
- Validar que due_date sea futura
- Validar que assigned_to sea miembro del proyecto
- Actualizar updated_at automáticamente
- Si cambia a "done", actualizar completed_at
- Crear notificación si cambia asignación
- Registrar cambios en audit logs

Endpoint: `PATCH /api/tasks/{id}`

RF-030: Eliminación de Tarea

Prioridad: Media
Descripción: El sistema debe permitir al Owner eliminar tareas.

Criterios de aceptación:

- Solo OWNER puede eliminar tareas
- Eliminar en cascada todos los comentarios de la tarea
- Eliminar notificaciones relacionadas
- Registrar eliminación en audit logs
- No se puede recuperar después de eliminar
- Confirmar eliminación en el frontend

Endpoint: `DELETE /api/tasks/{id}`

RF-031: Cambio de Estado de Tarea

Prioridad: Alta
Descripción: El sistema debe permitir cambiar el estado de tareas.

Criterios de aceptación:

- Estados válidos: pending, in_progress, blocked, done
- OWNER puede cambiar cualquier tarea

- EMPLOYEE puede cambiar solo tareas asignadas a él
- Al cambiar a "done", actualizar completed_at
- Crear notificación al Owner si un Employee completa tarea
- Validar transiciones de estado válidas
- Registrar cambio en audit logs

Endpoint: `PATCH /api/tasks/{id}/status`

RF-032: Asignación de Tarea

Prioridad: Alta
Descripción: El sistema debe permitir asignar/reasignar tareas a empleados.

Criterios de aceptación:

- Solo OWNER puede asignar tareas
- Validar que el empleado pertenezca al proyecto
- Permitir desasignar (assigned_to = null)
- Crear notificación al empleado asignado
- Si se reasigna, notificar al nuevo empleado
- Registrar asignación en audit logs

Endpoint: `PATCH /api/tasks/{id}/assign`

RF-033: Filtrado de Tareas

Prioridad: Media
Descripción: El sistema debe permitir filtrar tareas por múltiples criterios.

Criterios de aceptación:

- Filtros disponibles:
  - Por estado (pending, in_progress, blocked, done)
  - Por prioridad (low, medium, high, urgent)
  - Por asignado (user_id o "unassigned")
  - Por búsqueda en título
  - Por tags
  - Por rango de fechas (due_date)
- Permitir combinar múltiples filtros
- Aplicar filtros en el backend (no solo frontend)
- Retornar contador de resultados

RF-034: Búsqueda de Tareas

Prioridad: Media
Descripción: El sistema debe permitir buscar tareas por texto.

Criterios de aceptación:

- Buscar en título y descripción
- Búsqueda case-insensitive
- Soportar búsqueda parcial (LIKE %texto%)
- Resaltar términos de búsqueda en resultados
- Limitar resultados a tareas accesibles por el usuario

RF-035: Ordenamiento de Tareas

Prioridad: Media
Descripción: El sistema debe permitir ordenar tareas por diferentes criterios.

Criterios de aceptación:

- Ordenar por: due_date, created_at, priority, status, title
- Soportar orden ascendente y descendente
- Orden por defecto: due_date ascendente
- Mantener orden en paginación

RF-036: Estadísticas de Tareas

Prioridad: Media
Descripción: El sistema debe proporcionar estadísticas de tareas del proyecto.

Criterios de aceptación:

- Total de tareas
- Tareas por estado (pending, in_progress, blocked, done)

- Tareas por prioridad (low, medium, high, urgent)
- Tareas vencidas (due_date < hoy)
- Tareas que vencen hoy
- Tareas que vencen esta semana
- Tareas sin asignar
- Solo OWNER puede ver estadísticas

Endpoint: `GET /api/tasks/stats`

RF-037: Tareas del Usuario

Prioridad: Alta
Descripción: El sistema debe permitir a un Employee ver sus tareas asignadas.

Criterios de aceptación:

- Solo EMPLOYEE puede usar este endpoint
- Retornar solo tareas donde assigned_to = user_id
- Soportar filtros de estado y prioridad
- Ordenar por due_date por defecto
- Incluir información del proyecto

Endpoint: `GET /api/tasks/my-tasks`

RF-038: Gestión de Tags

Prioridad: Baja
Descripción: El sistema debe permitir agregar tags personalizados a tareas.

Criterios de aceptación:

- Tags es un array de strings
- Máximo 10 tags por tarea
- Cada tag máximo 50 caracteres
- Tags case-insensitive
- Autocompletar tags existentes en el proyecto
- Permitir filtrar tareas por tags

RF-039: Gestión de Checklist

Prioridad: Media
Descripción: El sistema debe permitir agregar checklist a tareas.

Criterios de aceptación:

- Checklist es un array de objetos {id, text, completed}
- Solo OWNER puede agregar/eliminar items del checklist
- OWNER y EMPLOYEE pueden marcar items como completados
- Máximo 20 items por checklist
- Cada item máximo 200 caracteres
- Mostrar progreso del checklist (X/Y completados)
- Validar permisos al modificar checklist

RF-040: Fechas de Tareas

Prioridad: Alta
Descripción: El sistema debe gestionar fechas de tareas correctamente.

Criterios de aceptación:

- due_date es obligatorio y debe ser futura
- start_date es opcional
- Si se proporciona start_date, debe ser <= due_date
- completed_at se actualiza automáticamente al marcar como "done"
- Validar formatos de fecha ISO 8601
- Mostrar fechas en zona horaria del usuario

RF-041: Prioridades de Tareas

Prioridad: Media
Descripción: El sistema debe soportar niveles de prioridad para tareas.

Criterios de aceptación:

- Prioridades válidas: low, medium, high, urgent
- Prioridad por defecto: medium

- Mostrar indicador visual de prioridad
- Permitir ordenar por prioridad
- Filtrar por prioridad

RF-042: Estados de Tareas

Prioridad: Alta
Descripción: El sistema debe gestionar el ciclo de vida de tareas.

Criterios de aceptación:

- Estados válidos: pending, in_progress, blocked, done
- Estado inicial: pending
- Transiciones válidas:
  - pending → in_progress, blocked
  - in_progress → blocked, done, pending
  - blocked → in_progress, pending
  - done → in_progress (reabrir)
- Validar transiciones en el backend
- Registrar cambios de estado en audit logs

RF-043: Tareas Vencidas

Prioridad: Media
Descripción: El sistema debe identificar y notificar tareas vencidas.

Criterios de aceptación:

- Una tarea está vencida si due_date < fecha actual y estado != done
- Mostrar indicador visual de vencimiento
- Filtro especial para tareas vencidas
- Notificar al Owner diariamente de tareas vencidas
- Incluir en estadísticas del proyecto

RF-044: Duplicación de Tareas

Prioridad: Baja
Descripción: El sistema debe permitir duplicar tareas existentes.

Criterios de aceptación:

- Solo OWNER puede duplicar tareas
- Copiar todos los campos excepto: id, created_at, completed_at
- Estado inicial: pending
- Asignación se mantiene
- Checklist se copia sin marcar items
- Agregar "(Copia)" al título
- Registrar duplicación en audit logs

Estado: Pendiente de implementación

RF-045: Archivado de Tareas

Prioridad: Media
Descripción: El sistema debe permitir archivar tareas completadas.

Criterios de aceptación:

- Solo tareas con estado "done" pueden archivarse
- Solo OWNER puede archivar tareas
- Agregar campo "archived" (boolean) a tareas
- Tareas archivadas no aparecen en listados por defecto
- Vista especial "Archivo" para ver tareas archivadas
- Permitir desarchivar tareas
- Registrar archivado en audit logs

Estado: Pendiente de implementación

3.4 Gestión de Equipo

RF-046: Invitación de Empleados

Prioridad: Alta
Descripción: El sistema debe permitir al Owner invitar empleados por email.

Criterios de aceptación:

- Solo OWNER puede enviar invitaciones
- Campos obligatorios: email
- Campos opcionales: job_title, description, responsibilities, skills, shift, department, phone
- Validar formato de email
- Validar que el email no esté ya registrado
- Validar que no exista invitación pendiente para ese email
- Generar token único y seguro (32 caracteres)
- Establecer fecha de expiración (7 días)
- Enviar email con link de invitación
- Crear notificación para el Owner
- Registrar invitación en audit logs

Endpoint: `POST /api/invites`

RF-047: Listado de Invitaciones

Prioridad: Media
Descripción: El sistema debe permitir al Owner ver sus invitaciones.

Criterios de aceptación:

- Solo OWNER puede ver invitaciones de su proyecto
- Mostrar todas las invitaciones (pending, accepted, expired, cancelled)
- Incluir: email, estado, fecha de creación, fecha de expiración
- Incluir datos de perfil (job_title, etc.)
- Ordenar por fecha de creación (más recientes primero)
- Indicar visualmente invitaciones expiradas

Endpoint: `GET /api/invites`

RF-048: Cancelación de Invitación

Prioridad: Media
Descripción: El sistema debe permitir al Owner cancelar invitaciones pendientes.

Criterios de aceptación:

- Solo OWNER puede cancelar invitaciones
- Solo se pueden cancelar invitaciones en estado "pending"
- Cambiar estado a "cancelled"
- El token queda invalidado
- Registrar cancelación en audit logs
- No se puede reactivar una invitación cancelada

Endpoint: `DELETE /api/invites/{id}`

RF-049: Reenvío de Invitación

Prioridad: Media
Descripción: El sistema debe permitir reenviar invitaciones no aceptadas.

Criterios de aceptación:

- Solo OWNER puede reenviar invitaciones
- Solo se pueden reenviar invitaciones "pending" o "expired"
- Máximo 3 reenvíos por invitación
- Generar nuevo token
- Extender fecha de expiración (+7 días desde reenvío)
- Incrementar contador resend_count
- Enviar nuevo email
- Registrar reenvío en audit logs

Endpoint: `POST /api/invites/{id}/resend`

RF-050: Expiración de Invitaciones

Prioridad: Media
Descripción: El sistema debe gestionar la expiración automática de invitaciones.

Criterios de aceptación:

- Las invitaciones expiran 7 días después de creación/reenvío
- Cambiar estado a "expired" automáticamente
- Las invitaciones expiradas no pueden aceptarse
- Mostrar indicador visual de expiración
- Permitir reenviar invitaciones expiradas

RF-051: Listado de Miembros del Equipo

Prioridad: Alta
Descripción: El sistema debe permitir ver los miembros del proyecto.

Criterios de aceptación:

- OWNER ve todos los miembros de su proyecto
- EMPLOYEE ve todos los miembros del proyecto
- SUPERADMIN ve miembros de cualquier proyecto
- Incluir: id, nombre, email, rol, estado, job_title, avatar
- Incluir fecha de incorporación (joined_at)
- Mostrar Owner destacado
- Ordenar: Owner primero, luego por nombre

Endpoint: `GET /api/members`

RF-052: Actualización de Perfil de Empleado

Prioridad: Media
Descripción: El sistema debe permitir actualizar perfiles de empleados.

Criterios de aceptación:

- OWNER puede actualizar perfil de cualquier empleado
- EMPLOYEE puede actualizar solo su propio perfil
- Campos actualizables: name, avatar, job_title, description, responsibilities, skills, shift, department, phone
- No se puede cambiar email ni rol
- Validar formato de teléfono
- Actualizar updated_at automáticamente
- Registrar cambios en audit logs

Endpoint: `PATCH /api/members/{user_id}/profile`

RF-053: Desactivación de Miembros

Prioridad: Media
Descripción: El sistema debe permitir al Owner desactivar miembros del equipo.

Criterios de aceptación:

- Solo OWNER puede desactivar miembros
- No se puede desactivar al Owner
- Cambiar estado de membership a "inactive"
- El usuario no puede acceder al proyecto
- Las tareas asignadas quedan sin asignar
- No se eliminan datos históricos (comentarios, audit logs)
- Crear notificación al usuario desactivado
- Registrar desactivación en audit logs

Endpoint: `PATCH /api/members/{id}/deactivate`

RF-054: Reactivación de Miembros

Prioridad: Baja
Descripción: El sistema debe permitir reactivar miembros desactivados.

Criterios de aceptación:

- Solo OWNER puede reactivar miembros
- Cambiar estado de membership a "active"
- El usuario recupera acceso al proyecto
- Crear notificación al usuario reactivado
- Registrar reactivación en audit logs

Estado: Pendiente de implementación

RF-055: Perfil Enriquecido de Empleados

Prioridad: Media
Descripción: El sistema debe soportar perfiles detallados de empleados.

Criterios de aceptación:

- Campos de perfil:
  - job_title: Puesto de trabajo (ej: "Desarrollador Senior")
  - description: Descripción breve del empleado

    - responsibilities: Responsabilidades principales
    - skills: Habilidades y competencias
    - shift: Turno de trabajo (morning, afternoon, night, flexible)
    - department: Departamento (ej: "Desarrollo", "Marketing")
    - phone: Teléfono de contacto
- Todos los campos son opcionales
- Mostrar perfil completo en vista de detalle
- Usar en invitaciones para pre-llenar datos

RF-056: Roles y Permisos de Miembros

Prioridad: Alta
Descripción: El sistema debe gestionar roles y permisos de miembros.

Criterios de aceptación:

- Roles válidos: OWNER, EMPLOYEE
- Un proyecto tiene exactamente un OWNER
- Un proyecto puede tener múltiples EMPLOYEES
- Los permisos se validan en cada operación
- No se puede cambiar el rol del Owner
- Matriz de permisos documentada y aplicada

RF-057: Búsqueda de Miembros

Prioridad: Baja
Descripción: El sistema debe permitir buscar miembros del equipo.

Criterios de aceptación:

- Buscar por nombre, email, job_title, department
- Búsqueda case-insensitive
- Búsqueda parcial (LIKE %texto%)
- Filtrar por estado (active, inactive)
- Filtrar por rol (OWNER, EMPLOYEE)

Estado: Pendiente de implementación

RF-058: Estadísticas del Equipo

Prioridad: Baja
Descripción: El sistema debe proporcionar estadísticas del equipo.

Criterios de aceptación:

- Total de miembros
- Miembros activos vs inactivos
- Distribución por departamento
- Distribución por turno
- Tareas promedio por empleado
- Solo OWNER puede ver estadísticas

Estado: Pendiente de implementación

RF-059: Historial de Miembros

Prioridad: Baja
Descripción: El sistema debe mantener historial de cambios en membresías.

Criterios de aceptación:

- Registrar todas las incorporaciones
- Registrar todas las desactivaciones
- Registrar cambios de perfil
- Mostrar línea de tiempo de cambios
- Solo OWNER puede ver historial completo

Estado: Pendiente de implementación

RF-060: Límite de Miembros por Proyecto

Prioridad: Baja
Descripción: El sistema debe permitir configurar límites de miembros.

Criterios de aceptación:

- Límite por defecto: 50 miembros por proyecto
- Límite configurable por plan (futuro)
- Validar límite al enviar invitaciones
- Mostrar contador de miembros disponibles
- Notificar al Owner cuando se acerque al límite

Estado: Pendiente de implementación

3.5 Sistema de Notificaciones

RF-061: Creación de Notificaciones

Prioridad: Alta
Descripción: El sistema debe crear notificaciones automáticamente para eventos importantes.

Criterios de aceptación:

- Eventos que generan notificaciones:
  - Tarea asignada (task_assigned)
  - Nuevo comentario en tarea (comment)
  - Estado de tarea cambiado (status_change)
  - Invitación aceptada (invite)
  - Miembro agregado/desactivado (member)
- Incluir: tipo, mensaje descriptivo, link al recurso
- Asociar con usuario destinatario y proyecto
- Estado inicial: unread (no leída)
- Registrar fecha de creación

RF-062: Listado de Notificaciones

Prioridad: Alta
Descripción: El sistema debe permitir a los usuarios ver sus notificaciones.

Criterios de aceptación:

- Cada usuario ve solo sus propias notificaciones
- Ordenar por fecha (más recientes primero)
- Incluir: tipo, mensaje, fecha, estado (leída/no leída)
- Incluir link al recurso relacionado
- Soportar paginación (20 notificaciones por página)
- Indicar visualmente notificaciones no leídas

Endpoint: `GET /api/notifications`

RF-063: Contador de Notificaciones No Leídas

Prioridad: Alta
Descripción: El sistema debe mostrar contador de notificaciones no leídas.

Criterios de aceptación:

- Retornar número de notificaciones con read = false
- Actualizar en tiempo real (polling cada 30 segundos)
- Mostrar badge en icono de notificaciones
- Máximo mostrado: 99+ si hay más de 99

Endpoint: `GET /api/notifications/unread-count`

RF-064: Marcar Notificación como Leída

Prioridad: Alta
Descripción: El sistema debe permitir marcar notificaciones individuales como leídas.

Criterios de aceptación:

- Solo el usuario dueño puede marcar sus notificaciones
- Cambiar read de false a true
- Actualizar read_at con timestamp actual
- Actualizar contador de no leídas
- Permitir marcar como no leída nuevamente

Endpoint: `PATCH /api/notifications/{id}/read`

RF-065: Marcar Todas como Leídas

Prioridad: Media
Descripción: El sistema debe permitir marcar todas las notificaciones como leídas.

Criterios de aceptación:

- Marcar todas las notificaciones del usuario como leídas
- Actualizar read_at en todas
- Actualizar contador a 0
- Confirmar acción en el frontend
- Registrar acción en audit logs

Endpoint: `PATCH /api/notifications/read-all`

RF-066: Eliminación de Notificaciones

Prioridad: Media
Descripción: El sistema debe permitir eliminar notificaciones individuales.

Criterios de aceptación:

- Solo el usuario dueño puede eliminar sus notificaciones
- Eliminación permanente (no soft delete)
- Actualizar contador si era no leída
- Confirmar eliminación en el frontend

Endpoint: `DELETE /api/notifications/{id}`

RF-067: Tipos de Notificaciones

Prioridad: Media
Descripción: El sistema debe soportar diferentes tipos de notificaciones.

Criterios de aceptación:

- Tipos implementados:
  - task_assigned: "Te han asignado la tarea '{título}'"
  - comment: "{usuario} comentó en '{título}'"
  - status_change: "La tarea '{título}' cambió a {estado}"
  - invite: "{usuario} aceptó tu invitación"
  - member: "Nuevo miembro: {usuario}"
- Cada tipo tiene icono y color distintivo
- Mensajes personalizados con nombres de usuarios/tareas
- Links directos al recurso relacionado

RF-068: Filtrado de Notificaciones

Prioridad: Baja
Descripción: El sistema debe permitir filtrar notificaciones.

Criterios de aceptación:

- Filtrar por tipo de notificación
- Filtrar por estado (leída/no leída)
- Filtrar por rango de fechas
- Combinar múltiples filtros
- Mantener filtros en paginación

Estado: Pendiente de implementación

RF-069: Preferencias de Notificaciones

Prioridad: Baja
Descripción: El sistema debe permitir configurar preferencias de notificaciones.

Criterios de aceptación:

- Permitir activar/desactivar tipos de notificaciones
- Configurar frecuencia de emails de resumen
- Configurar notificaciones push (futuro)
- Guardar preferencias por usuario
- Aplicar preferencias al crear notificaciones

Estado: Pendiente de implementación

RF-070: Notificaciones por Email

Prioridad: Baja
Descripción: El sistema debe enviar notificaciones importantes por email.

Criterios de aceptación:

- Enviar email para eventos críticos:
  - Tarea asignada
  - Tarea vencida
  - Invitación recibida
- Incluir link directo al recurso
- Respetar preferencias del usuario
- Usar plantillas HTML profesionales
- Incluir opción de desuscribirse

Estado: Pendiente de implementación

3.6 Sistema de Comentarios

RF-071: Creación de Comentarios

Prioridad: Alta
Descripción: El sistema debe permitir agregar comentarios a tareas.

Criterios de aceptación:

- OWNER y EMPLOYEE pueden comentar en tareas
- EMPLOYEE solo puede comentar en tareas asignadas a él
- Campo obligatorio: content (texto del comentario)
- Mínimo 1 carácter, máximo 5000 caracteres
- Asociar comentario con usuario y tarea
- Incluir timestamp de creación
- Crear notificación al asignado (si no es él quien comenta)
- Registrar creación en audit logs

Endpoint: `POST /api/tasks/{task_id}/comments`

RF-072: Listado de Comentarios

Prioridad: Alta
Descripción: El sistema debe permitir ver comentarios de una tarea.

Criterios de aceptación:

- OWNER ve comentarios de cualquier tarea de su proyecto
- EMPLOYEE ve comentarios de tareas asignadas a él
- Ordenar por fecha de creación (más antiguos primero)
- Incluir: id, contenido, autor (nombre, avatar), fecha
- Incluir indicador si el comentario fue editado
- Mostrar fecha de última edición si aplica

Endpoint: `GET /api/tasks/{task_id}/comments`

RF-073: Edición de Comentarios

Prioridad: Media
Descripción: El sistema debe permitir editar comentarios propios.

Criterios de aceptación:

- Solo el autor puede editar su comentario
- Actualizar campo content
- Actualizar updated_at con timestamp actual
- Marcar comentario como editado
- Validar longitud del contenido (1-5000 caracteres)
- Registrar edición en audit logs

Endpoint: `PATCH /api/tasks/{task_id}/comments/{id}`

RF-074: Eliminación de Comentarios

Prioridad: Media
Descripción: El sistema debe permitir eliminar comentarios.

Criterios de aceptación:

- El autor puede eliminar su propio comentario
- El OWNER puede eliminar cualquier comentario de su proyecto
- Eliminación permanente (no soft delete)

- Confirmar eliminación en el frontend
- Registrar eliminación en audit logs

Endpoint: `DELETE /api/tasks/{task_id}/comments/{id}`

RF-075: Contador de Comentarios

Prioridad: Baja
Descripción: El sistema debe mostrar contador de comentarios por tarea.

Criterios de aceptación:

- Incluir contador en listado de tareas
- Incluir contador en detalle de tarea
- Actualizar contador en tiempo real
- Mostrar indicador visual si hay comentarios nuevos

Estado: Pendiente de implementación

3.7 Panel de Administración

RF-076: Acceso al Panel de Administración

Prioridad: Alta
Descripción: El sistema debe proporcionar un panel de administración para SUPERADMIN.

Criterios de aceptación:

- Solo usuarios con rol SUPERADMIN tienen acceso
- Ruta protegida: /admin
- Verificar rol en cada petición
- Retornar error 403 si el usuario no es SUPERADMIN
- Registrar accesos en audit logs

RF-077: Listado de Todos los Usuarios

Prioridad: Alta
Descripción: El sistema debe permitir al SUPERADMIN listar todos los usuarios.

Criterios de aceptación:

- Solo SUPERADMIN tiene acceso
- Mostrar todos los usuarios del sistema
- Incluir: id, email, nombre, rol, estado, proyecto, fechas
- Soportar paginación (20 usuarios por página)
- Soportar filtros: rol, estado, búsqueda por nombre/email
- Soportar ordenamiento: nombre, email, created_at
- Incluir contador total de usuarios

Endpoint: `GET /api/admin/users`

RF-078: Listado de Todos los Proyectos

Prioridad: Alta
Descripción: El sistema debe permitir al SUPERADMIN listar todos los proyectos.

Criterios de aceptación:

- Solo SUPERADMIN tiene acceso
- Mostrar todos los proyectos del sistema
- Incluir: id, nombre, owner, categoría, estado, fechas
- Incluir contador de tareas y miembros
- Soportar paginación (20 proyectos por página)
- Soportar filtros: estado, categoría, búsqueda por nombre
- Soportar ordenamiento: nombre, created_at

Endpoint: `GET /api/admin/projects`

RF-079: Cambio de Estado de Usuario

Prioridad: Media
Descripción: El sistema debe permitir al SUPERADMIN activar/desactivar usuarios.

Criterios de aceptación:

- Solo SUPERADMIN puede cambiar estado
- Estados válidos: active, disabled
- Usuarios desactivados no pueden iniciar sesión
- No se pueden desactivar otros SUPERADMIN
- Crear notificación al usuario afectado
- Registrar cambio en audit logs

Endpoint: `PATCH /api/admin/users/{id}/status`

RF-080: Cambio de Estado de Proyecto

Prioridad: Media
Descripción: El sistema debe permitir al SUPERADMIN activar/desactivar proyectos.

Criterios de aceptación:

- Solo SUPERADMIN puede cambiar estado
- Estados válidos: active, disabled
- Proyectos desactivados son inaccesibles para sus miembros
- Notificar al Owner del cambio
- Registrar cambio en audit logs

Endpoint: `PATCH /api/admin/projects/{id}/status`

RF-081: Estadísticas Globales del Sistema

Prioridad: Media
Descripción: El sistema debe proporcionar estadísticas globales al SUPERADMIN.

Criterios de aceptación:

- Solo SUPERADMIN tiene acceso
- Estadísticas de usuarios:
  - Total de usuarios
  - Usuarios activos vs desactivados
  - Usuarios por rol (OWNER, EMPLOYEE, SUPERADMIN)
  - Nuevos usuarios últimos 30 días
- Estadísticas de proyectos:
  - Total de proyectos
  - Proyectos activos vs desactivados
  - Nuevos proyectos últimos 30 días
- Estadísticas de tareas:
  - Total de tareas
  - Tareas por estado
  - Tareas por prioridad
- Actualizar estadísticas en tiempo real

Endpoint: `GET /api/admin/stats`

RF-082: Logs de Auditoría

Prioridad: Alta
Descripción: El sistema debe mantener logs de auditoría de todas las acciones importantes.

Criterios de aceptación:

- Registrar acciones:
  - user_created, user_updated, user_deleted
  - project_created, project_updated, project_deleted
  - task_created, task_updated, task_deleted
  - invite_sent, invite_accepted, invite_cancelled
  - member_added, member_deactivated
  - permission_denied_*
- Incluir: usuario, acción, entidad, detalles, IP, user agent, timestamp
- Solo SUPERADMIN puede ver logs
- Soportar paginación (50 logs por página)
- Soportar filtros: usuario, acción, entidad, rango de fechas
- Retener logs por al menos 1 año

Endpoint: `GET /api/admin/audit-logs`

RF-083: Health Check del Sistema

Prioridad: Media
Descripción: El sistema debe proporcionar endpoint de health check.

Criterios de aceptación:

- Solo SUPERADMIN tiene acceso
- Verificar estado de:
  - Base de datos (conectividad, latencia)
  - API (tiempo de respuesta)
  - Memoria y CPU del servidor
- Retornar estado: healthy, degraded, down
- Incluir timestamp de la verificación
- Incluir detalles de cada componente

Endpoint: `GET /api/admin/health`

RF-084: Búsqueda Global

Prioridad: Baja
Descripción: El sistema debe permitir al SUPERADMIN buscar en todo el sistema.

Criterios de aceptación:

- Solo SUPERADMIN tiene acceso
- Buscar en: usuarios, proyectos, tareas
- Búsqueda por texto en múltiples campos
- Retornar resultados agrupados por tipo
- Limitar resultados a 10 por tipo
- Incluir links directos a recursos

Estado: Pendiente de implementación

RF-085: Reportes del Sistema

Prioridad: Baja
Descripción: El sistema debe generar reportes para el SUPERADMIN.

Criterios de aceptación:

- Reportes disponibles:
  - Usuarios registrados por mes
  - Proyectos creados por mes
  - Tareas completadas por mes
  - Actividad por usuario
  - Actividad por proyecto
- Exportar reportes en CSV/PDF
- Configurar rango de fechas
- Programar reportes automáticos

Estado: Pendiente de implementación

---

4. REQUERIMIENTOS NO FUNCIONALES

4.1 Rendimiento

RNF-001: Tiempo de Respuesta de API

Prioridad: Alta
Descripción: El sistema debe responder a peticiones de API en tiempo aceptable.

Criterios de aceptación:

- Operaciones de lectura (GET): < 500ms en el percentil 95
- Operaciones de escritura (POST, PATCH): < 1000ms en el percentil 95
- Operaciones complejas (estadísticas): < 2000ms en el percentil 95
- Medir tiempos de respuesta en producción
- Implementar logging de queries lentas

RNF-002: Tiempo de Carga de Páginas

Prioridad: Alta
Descripción: El frontend debe cargar páginas rápidamente.

Criterios de aceptación:

- First Contentful Paint (FCP): < 1.5 segundos
- Largest Contentful Paint (LCP): < 2.5 segundos
- Time to Interactive (TTI): < 3.5 segundos
- Cumulative Layout Shift (CLS): < 0.1
- Medir con Lighthouse y Web Vitals

RNF-003: Usuarios Concurrentes

Prioridad: Alta
Descripción: El sistema debe soportar múltiples usuarios simultáneos.

Criterios de aceptación:

- Soportar al menos 100 usuarios concurrentes
- Mantener tiempos de respuesta bajo carga
- No degradar rendimiento con múltiples sesiones
- Realizar pruebas de carga periódicas

RNF-004: Optimización de Queries

Prioridad: Alta
Descripción: Las queries de base de datos deben estar optimizadas.

Criterios de aceptación:

- Usar índices en campos de búsqueda frecuente
- Evitar N+1 queries (usar eager loading)
- Limitar resultados con paginación
- Queries complejas < 100ms
- Monitorear queries lentas

RNF-005: Caché de Datos

Prioridad: Media
Descripción: El sistema debe implementar caché para datos frecuentes.

Criterios de aceptación:

- Cachear estadísticas del proyecto (TTL: 5 minutos)
- Cachear listado de miembros (TTL: 10 minutos)
- Cachear configuraciones del sistema
- Invalidar caché al actualizar datos
- Usar Redis para caché distribuido

Estado: Pendiente de implementación

RNF-006: Compresión de Respuestas

Prioridad: Media
Descripción: El sistema debe comprimir respuestas HTTP.

Criterios de aceptación:

- Habilitar compresión gzip/brotli
- Comprimir respuestas > 1KB
- Reducir tamaño de transferencia en 70%
- Configurar en servidor web (Nginx)

RNF-007: Optimización de Imágenes

Prioridad: Media
Descripción: El sistema debe optimizar imágenes de usuario.

Criterios de aceptación:

- Redimensionar avatares a 200x200px
- Comprimir imágenes con calidad 85%
- Convertir a formato WebP
- Servir desde CDN
- Lazy loading de imágenes

Estado: Pendiente de implementación

RNF-008: Paginación Eficiente

Prioridad: Alta
Descripción: El sistema debe implementar paginación en listados grandes.

Criterios de aceptación:

- Paginar listados de tareas (10 por página)

- Paginar listados de notificaciones (20 por página)
- Paginar listados de usuarios/proyectos (20 por página)
- Usar cursor-based pagination para mejor rendimiento
- Incluir metadata de paginación (total, página actual, total páginas)

RNF-009: Lazy Loading

Prioridad: Media
Descripción: El sistema debe cargar datos bajo demanda.

Criterios de aceptación:

- Cargar comentarios solo al abrir tarea
- Cargar notificaciones solo al abrir panel
- Cargar estadísticas solo al abrir dashboard
- Implementar infinite scroll en listados largos

RNF-010: Optimización de Bundle

Prioridad: Media
Descripción: El frontend debe optimizar el tamaño del bundle JavaScript.

Criterios de aceptación:

- Bundle principal < 500KB (gzipped)
- Code splitting por rutas
- Tree shaking de dependencias no usadas
- Lazy loading de componentes pesados
- Analizar bundle con webpack-bundle-analyzer

4.2 Seguridad

RNF-011: Encriptación de Contraseñas

Prioridad: Crítica
Descripción: El sistema debe almacenar contraseñas de forma segura.

Criterios de aceptación:

- Usar bcrypt con cost factor 12
- Generar salt único por contraseña
- Nunca almacenar contraseñas en texto plano
- Nunca retornar contraseñas en respuestas
- Nunca loggear contraseñas

RNF-012: Protección contra SQL Injection

Prioridad: Crítica
Descripción: El sistema debe prevenir ataques de SQL injection.

Criterios de aceptación:

- Usar ORM (SQLAlchemy) para todas las queries
- Usar queries parametrizadas
- Validar y sanitizar inputs
- Nunca concatenar strings en queries SQL
- Realizar auditorías de seguridad periódicas

RNF-013: Protección contra XSS

Prioridad: Crítica
Descripción: El sistema debe prevenir ataques de Cross-Site Scripting.

Criterios de aceptación:

- React escapa automáticamente contenido
- Validar inputs en backend
- Sanitizar HTML en comentarios
- Usar Content Security Policy (CSP)
- No usar dangerouslySetInnerHTML sin sanitizar

RNF-014: Protección contra CSRF

Prioridad: Alta
Descripción: El sistema debe prevenir ataques de Cross-Site Request Forgery.

Criterios de aceptación:

- Usar tokens JWT en headers (no cookies)
- Validar origen de peticiones
- Implementar SameSite cookies si se usan
- Verificar header Referer en operaciones críticas

RNF-015: HTTPS Obligatorio

Prioridad: Crítica
Descripción: El sistema debe usar HTTPS en producción.

Criterios de aceptación:

- Certificado SSL/TLS válido
- Redirigir HTTP a HTTPS automáticamente
- Usar HSTS (HTTP Strict Transport Security)
- TLS 1.2 o superior
- Renovar certificados automáticamente

RNF-016: Validación de Inputs

Prioridad: Alta
Descripción: El sistema debe validar todos los inputs de usuario.

Criterios de aceptación:

- Validar en frontend (UX)
- Validar en backend (seguridad)
- Usar schemas de validación (Marshmallow)
- Retornar mensajes de error descriptivos
- Validar tipos, longitudes, formatos

RNF-017: Rate Limiting

Prioridad: Media
Descripción: El sistema debe limitar peticiones por usuario/IP.

Criterios de aceptación:

- Límite de login: 5 intentos por minuto por IP
- Límite de API: 100 peticiones por minuto por usuario
- Límite de registro: 3 registros por hora por IP
- Retornar error 429 (Too Many Requests)
- Implementar con Redis

Estado: Pendiente de implementación

RNF-018: Auditoría de Seguridad

Prioridad: Alta
Descripción: El sistema debe registrar eventos de seguridad.

Criterios de aceptación:

- Loggear intentos de login fallidos
- Loggear accesos denegados (403)
- Loggear cambios de permisos
- Loggear operaciones críticas
- Retener logs por al menos 1 año

RNF-019: Sesiones Seguras

Prioridad: Alta
Descripción: El sistema debe gestionar sesiones de forma segura.

Criterios de aceptación:

- Tokens JWT con expiración corta (15 min)
- Refresh tokens con expiración larga (7 días)
- Invalidar tokens al cambiar contraseña
- No almacenar tokens en cookies
- Usar localStorage con precaución

RNF-020: Aislamiento de Datos

Prioridad: Crítica
Descripción: El sistema debe garantizar aislamiento multitenant.

Criterios de aceptación:

- Filtrar todas las queries por project_id
- Validar pertenencia al proyecto en cada operación
- Imposible acceder a datos de otros proyectos
- Auditar intentos de acceso no autorizado
- Realizar pruebas de penetración

4.3 Usabilidad

RNF-021: Interfaz Intuitiva

Prioridad: Alta
Descripción: El sistema debe tener una interfaz fácil de usar.

Criterios de aceptación:

- Diseño consistente en todas las páginas
- Navegación clara y predecible
- Iconos reconocibles y etiquetados
- Feedback visual de acciones
- Mensajes de error claros y accionables

RNF-022: Responsive Design

Prioridad: Alta
Descripción: El sistema debe funcionar en diferentes dispositivos.

Criterios de aceptación:

- Soportar resoluciones desde 320px hasta 4K
- Diseño adaptable (mobile-first)
- Touch-friendly en dispositivos móviles
- Probar en: Chrome, Firefox, Safari, Edge
- Probar en: iOS, Android, Windows, macOS

RNF-023: Accesibilidad

Prioridad: Media
Descripción: El sistema debe ser accesible para usuarios con discapacidades.

Criterios de aceptación:

- Cumplir WCAG 2.1 nivel AA
- Navegación por teclado completa
- Etiquetas ARIA apropiadas
- Contraste de colores adecuado (4.5:1)
- Textos alternativos en imágenes
- Probar con lectores de pantalla

RNF-024: Internacionalización

Prioridad: Baja
Descripción: El sistema debe soportar múltiples idiomas.

Criterios de aceptación:

- Idiomas soportados: Español, Inglés
- Textos externalizados en archivos de traducción
- Formato de fechas según locale
- Formato de números según locale
- Cambio de idioma sin recargar página

Estado: Pendiente de implementación

RNF-025: Ayuda Contextual

Prioridad: Baja

Descripción: El sistema debe proporcionar ayuda contextual.

Criterios de aceptación:

- Tooltips en campos complejos
- Guías de inicio rápido
- Documentación integrada
- Videos tutoriales
- Chat de soporte

Estado: Pendiente de implementación

4.4 Confiabilidad

RNF-026: Disponibilidad del Sistema

Prioridad: Alta
Descripción: El sistema debe estar disponible la mayor parte del tiempo.

Criterios de aceptación:

- Uptime objetivo: 99.5% (43.8 horas de downtime al año)
- Ventanas de mantenimiento programadas
- Notificar mantenimientos con 48 horas de anticipación
- Monitorear disponibilidad 24/7
- Alertas automáticas si el sistema cae

RNF-027: Recuperación ante Fallos

Prioridad: Alta
Descripción: El sistema debe recuperarse rápidamente de fallos.

Criterios de aceptación:

- Recovery Time Objective (RTO): < 1 hora
- Recovery Point Objective (RPO): < 15 minutos
- Backups automáticos diarios
- Backups incrementales cada hora
- Procedimientos de recuperación documentados
- Pruebas de recuperación trimestrales

RNF-028: Manejo de Errores

Prioridad: Alta
Descripción: El sistema debe manejar errores gracefully.

Criterios de aceptación:

- Capturar todas las excepciones
- Loggear errores con stack trace
- Mostrar mensajes de error amigables al usuario
- No exponer información sensible en errores
- Implementar error boundaries en React
- Enviar errores críticos a sistema de monitoreo

RNF-029: Transacciones ACID

Prioridad: Alta
Descripción: El sistema debe garantizar integridad de datos.

Criterios de aceptación:

- Usar transacciones de base de datos
- Rollback automático en caso de error
- Atomicidad en operaciones complejas
- Consistencia de datos garantizada
- Aislamiento entre transacciones concurrentes
- Durabilidad de datos confirmados

RNF-030: Validación de Integridad

Prioridad: Media
Descripción: El sistema debe validar integridad referencial.

Criterios de aceptación:

- Foreign keys con constraints
- Validar existencia de recursos relacionados
- Prevenir eliminación de recursos referenciados
- Usar CASCADE o RESTRICT apropiadamente
- Validar en backend antes de persistir

4.5 Mantenibilidad

RNF-031: Código Limpio

Prioridad: Alta
Descripción: El código debe ser legible y mantenible.

Criterios de aceptación:

- Seguir guías de estilo (PEP 8 para Python, ESLint para TypeScript)
- Nombres descriptivos de variables y funciones
- Funciones pequeñas y con responsabilidad única
- Comentarios en código complejo
- Evitar código duplicado (DRY)
- Code reviews obligatorios

RNF-032: Documentación del Código

Prioridad: Alta
Descripción: El código debe estar documentado.

Criterios de aceptación:

- Docstrings en todas las funciones públicas
- Comentarios JSDoc en TypeScript
- README en cada módulo principal
- Documentación de API actualizada
- Diagramas de arquitectura actualizados

RNF-033: Testing

Prioridad: Alta
Descripción: El sistema debe tener cobertura de tests.

Criterios de aceptación:

- Cobertura de código > 70%
- Unit tests para lógica de negocio
- Integration tests para APIs
- E2E tests para flujos críticos
- Tests automatizados en CI/CD
- Tests ejecutados antes de cada deploy

RNF-034: Versionado de Código

Prioridad: Alta
Descripción: El código debe estar versionado correctamente.

Criterios de aceptación:

- Usar Git para control de versiones
- Commits descriptivos y atómicos
- Branches por feature/bugfix
- Pull requests con code review
- Tags para releases
- Changelog actualizado

RNF-035: Logs y Monitoreo

Prioridad: Alta
Descripción: El sistema debe tener logging y monitoreo adecuados.

Criterios de aceptación:

- Logs estructurados (JSON)
- Niveles de log apropiados (DEBUG, INFO, WARNING, ERROR)
- Logs centralizados (ELK Stack)
- Monitoreo de métricas (CPU, memoria, disco)
- Alertas automáticas para errores críticos

- Dashboards de monitoreo

4.6 Portabilidad

RNF-036: Independencia de Plataforma

Prioridad: Media
Descripción: El sistema debe funcionar en diferentes plataformas.

Criterios de aceptación:

- Backend compatible con Linux, Windows, macOS
- Frontend compatible con navegadores modernos
- Base de datos portable (MySQL, PostgreSQL)
- Usar Docker para consistencia de entornos
- Documentar dependencias del sistema

RNF-037: Configuración Externalizada

Prioridad: Alta
Descripción: La configuración debe estar separada del código.

Criterios de aceptación:

- Variables de entorno para configuración
- Archivos .env para desarrollo
- Secrets management para producción
- No hardcodear credenciales
- Configuración por ambiente (dev, staging, prod)

RNF-038: Containerización

Prioridad: Media
Descripción: El sistema debe ser containerizable.

Criterios de aceptación:

- Dockerfile para backend
- Dockerfile para frontend
- Docker Compose para desarrollo local
- Imágenes optimizadas (multi-stage builds)
- Documentar proceso de containerización

Estado: Pendiente de implementación

RNF-039: Despliegue Automatizado

Prioridad: Media
Descripción: El sistema debe tener despliegue automatizado.

Criterios de aceptación:

- CI/CD pipeline configurado
- Tests automatizados en pipeline
- Deploy automático a staging
- Deploy manual a producción (con aprobación)
- Rollback automático si falla deploy

Estado: Pendiente de implementación

RNF-040: Migración de Datos

Prioridad: Alta
Descripción: El sistema debe soportar migraciones de base de datos.

Criterios de aceptación:

- Usar Flask-Migrate (Alembic)
- Migraciones versionadas
- Migraciones reversibles (upgrade/downgrade)
- Probar migraciones en staging antes de producción
- Backup antes de cada migración

---

5. INTERFACES EXTERNAS

5.1 Interfaces de Usuario

IU-001: Interfaz Web

Descripción: Aplicación web responsive accesible desde navegadores modernos.

Características:

- Tecnología: Next.js 14 con React 19
- Diseño: Material Design con shadcn/ui
- Responsive: Mobile-first design
- Navegadores soportados: Chrome 90+, Firefox 88+, Safari 14+, Edge 90+

5.2 Interfaces de Hardware

IH-001: Servidor de Aplicación

Descripción: Servidor donde se ejecuta el backend Flask.

Requisitos mínimos:

- CPU: 2 cores
- RAM: 4 GB
- Disco: 20 GB SSD
- Red: 100 Mbps

Requisitos recomendados:

- CPU: 4 cores
- RAM: 8 GB
- Disco: 50 GB SSD
- Red: 1 Gbps

IH-002: Servidor de Base de Datos

Descripción: Servidor MySQL para almacenamiento de datos.

Requisitos mínimos:

- CPU: 2 cores
- RAM: 4 GB
- Disco: 50 GB SSD
- Red: 100 Mbps

Requisitos recomendados:

- CPU: 4 cores
- RAM: 16 GB
- Disco: 200 GB SSD (RAID 10)
- Red: 1 Gbps

5.3 Interfaces de Software

IS-001: API REST

Descripción: API RESTful para comunicación frontend-backend.

Características:

- Protocolo: HTTPS
- Formato: JSON
- Autenticación: JWT Bearer Token
- Versionado: /api/v1 (futuro)
- Documentación: OpenAPI 3.0 (futuro)

Endpoints principales:

- Autenticación: 6 endpoints
- Proyectos: 2 endpoints
- Tareas: 9 endpoints
- Invitaciones: 5 endpoints
- Miembros: 3 endpoints
- Notificaciones: 5 endpoints
- Comentarios: 4 endpoints
- Admin: 7 endpoints

Total: 41 endpoints

IS-002: Base de Datos MySQL

Descripción: Sistema de gestión de base de datos relacional.

Características:

- Versión: MySQL 8.0+
- Charset: utf8mb4
- Collation: utf8mb4_unicode_ci
- Engine: InnoDB
- Transacciones: ACID

Tablas:

- users
- projects
- memberships
- tasks
- invites
- notifications
- comments
- audit_logs

IS-003: Servidor SMTP

Descripción: Servidor para envío de emails.

Características:

- Protocolo: SMTP/TLS
- Puerto: 587
- Autenticación: Usuario/Contraseña
- Uso: Invitaciones, notificaciones, recuperación de contraseña

Estado: Pendiente de configuración

IS-004: Servicio de Almacenamiento

Descripción: Almacenamiento de archivos estáticos (avatares, adjuntos).

Características:

- Opciones: AWS S3, Google Cloud Storage, Azure Blob
- Acceso: Público (avatares), Privado (adjuntos)
- CDN: CloudFront, CloudFlare

Estado: Pendiente de implementación

5.4 Interfaces de Comunicación

IC-001: Protocolo HTTP/HTTPS

Descripción: Protocolo de comunicación entre cliente y servidor.

Características:

- Versión: HTTP/1.1, HTTP/2
- Seguridad: TLS 1.2+
- Métodos: GET, POST, PATCH, DELETE
- Headers: Authorization, Content-Type, Accept
- CORS: Configurado para frontend

IC-002: WebSockets

Descripción: Comunicación bidireccional en tiempo real.

Características:

- Protocolo: WSS (WebSocket Secure)
- Uso: Notificaciones en tiempo real, actualizaciones de tareas
- Librería: Socket.IO

Estado: Pendiente de implementación

6. APÉNDICES

6.1 Glosario de Términos

Término	Definición
Multitenant	Arquitectura donde múltiples clientes (tenants) comparten la misma infraestructura pero con datos completamente aislados
JWT	JSON Web Token - Estándar abierto (RFC 7519) para crear tokens de acceso que permiten la propagación de identidad y privilegios
CRUD	Create, Read, Update, Delete - Operaciones básicas de persistencia de datos
API REST	Application Programming Interface basada en el estilo arquitectónico REST (Representational State Transfer)
ORM	Object-Relational Mapping - Técnica de programación para convertir datos entre sistemas de tipos incompatibles
SPA	Single Page Application - Aplicación web que carga una sola página HTML y actualiza dinámicamente el contenido
ACID	Atomicity, Consistency, Isolation, Durability - Propiedades que garantizan transacciones confiables en bases de datos
CORS	Cross-Origin Resource Sharing - Mecanismo que permite solicitudes de recursos restringidos desde otro dominio
RBAC	Role-Based Access Control - Control de acceso basado en roles de usuario
Soft Delete	Eliminación lógica donde los registros se marcan como eliminados pero no se borran físicamente
Eager Loading	Técnica de carga de datos relacionados en una sola query para evitar N+1 queries
Lazy Loading	Técnica de carga diferida de recursos hasta que sean necesarios
Optimistic Update	Actualización de UI antes de confirmar con el servidor, con rollback si falla

6.2 Acrónimos

Acrónimo	Significado
API	Application Programming Interface
CDN	Content Delivery Network
CI/CD	Continuous Integration / Continuous Deployment
CPU	Central Processing Unit
CSRF	Cross-Site Request Forgery
CSS	Cascading Style Sheets
DB	Database
DNS	Domain Name System
FCP	First Contentful Paint
GDPR	General Data Protection Regulation
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol

Acrónimo	Significado
HTTPS	HyperText Transfer Protocol Secure
IP	Internet Protocol
JSON	JavaScript Object Notation
JWT	JSON Web Token
LCP	Largest Contentful Paint
MVP	Minimum Viable Product
ORM	Object-Relational Mapping
RAM	Random Access Memory
REST	Representational State Transfer
RPO	Recovery Point Objective
RTO	Recovery Time Objective
SaaS	Software as a Service
SMTP	Simple Mail Transfer Protocol
SQL	Structured Query Language
SRS	Software Requirements Specification
SSL	Secure Sockets Layer
TLS	Transport Layer Security
TTI	Time to Interactive
UI	User Interface
URL	Uniform Resource Locator
UUID	Universally Unique Identifier
UX	User Experience
WCAG	Web Content Accessibility Guidelines
XSS	Cross-Site Scripting

6.3 Referencias Técnicas

Documentación del Proyecto:

- Documentación consolidada: `.kiro/DOCUMENTACION_CONSOLIDADA.md`
- Arquitectura del sistema: `.kiro/architecture.md`
- Modelo de datos: `.kiro/data-model.md`
- Documentación de API: `.kiro/API_COMPLETE_DOCUMENTATION.md`
- Guía de desarrollo: `.kiro/development-guide.md`
- Guía de testing: `.kiro/TESTING_GUIDE.md`
- Workflow de desarrollo: `.kiro/WORKFLOW.md`

Tecnologías Utilizadas:

- Flask: https://flask.palletsprojects.com/
- Next.js: https://nextjs.org/
- React: https://react.dev/
- SQLAlchemy: https://www.sqlalchemy.org/
- MySQL: https://dev.mysql.com/doc/
- JWT: https://jwt.io/
- shadcn/ui: https://ui.shadcn.com/

Estándares y Mejores Prácticas:

- REST API Design: https://restfulapi.net/
- OWASP Security: https://owasp.org/
- WCAG Accessibility: https://www.w3.org/WAI/WCAG21/quickref/
- Semantic Versioning: https://semver.org/

6.4 Matriz de Trazabilidad

ID Requerimiento	Prioridad	Estado	Endpoint/Componente	Tests
RF-001	Alta	Implementado	POST /api/auth/register	✓
RF-002	Alta	Implementado	POST /api/auth/login	✓
RF-003	Alta	Implementado	POST /api/auth/refresh	✓
RF-026	Alta	Implementado	POST /api/tasks	✓
RF-027	Alta	Implementado	GET /api/tasks	✓
RF-046	Alta	Implementado	POST /api/invites	✓
RF-061	Alta	Implementado	Auto-generado	✓
RF-071	Alta	Implementado	POST /api/tasks/{id}/comments	✓
RNF-001	Alta	Implementado	Todas las APIs	✓
RNF-011	Crítica	Implementado	Bcrypt	✓

Nota: Esta es una muestra. La matriz completa incluye los 85+ requerimientos.

6.5 Historial de Cambios

Fecha	Versión	Cambios Principales
15/01/2026	1.0.0	Versión inicial del documento con requerimientos básicos
25/01/2026	1.2.0	Agregados requerimientos de tareas y proyectos
10/02/2026	1.5.0	Agregados requerimientos de comentarios y notificaciones
20/02/2026	1.8.0	Agregados requerimientos no funcionales
24/02/2026	2.0.0	Documento completo con 85+ requerimientos funcionales y no funcionales

---

6.6 Aprobaciones

Rol	Nombre	Firma	Fecha
Product Owner	[Nombre]	_____	/_/2026
Arquitecto de Software	[Nombre]	_____	/_/2026
Líder de Desarrollo	[Nombre]	_____	/_/2026
Líder de QA	[Nombre]	_____	/_/2026
Stakeholder	[Nombre]	_____	/_/2026

CONCLUSIÓN

Este documento de requerimientos especifica de manera exhaustiva las funcionalidades y características del sistema ProGest. Con más de 85 requerimientos funcionales y no funcionales, proporciona una base sólida para el desarrollo, testing y mantenimiento del sistema.

El sistema ProGest está diseñado para ser:

- **Seguro:** Con múltiples capas de seguridad y aislamiento multitenant
- **Escalable:** Arquitectura preparada para crecer con la demanda
- **Mantenible:** Código limpio, documentado y con alta cobertura de tests
- **Usable:** Interfaz intuitiva y responsive para todos los dispositivos
- **Confiable:** Alta disponibilidad y recuperación ante fallos

Estado actual: Sistema en producción con funcionalidades core implementadas y probadas.

Próximos pasos: Implementación de funcionalidades pendientes marcadas como "Pendiente de implementación" en este documento.

Fin del Documento

ProGest - Sistema de Gestión de Proyectos SaaS

Versión 2.0.0 | 24 de Febrero de 2026

© 2026 ProGest. Todos los derechos reservados.