PRD - Módulo Gestor de Tareas

Versión: 1.0
Estado: Draft para ingeniería/QA/UX
Owner: Producto
Audiencia: Ingeniería (FE/BE), QA, UX/UI, DevOps, Seguridad

> **Alcance del producto:** Este PRD especifica un **gestor de tareas completamente funcional** centrado en **gestión de tareas** y **múltiples formas de visualizarlas/ordenarlas** (vistas). **Se excluyen** módulos no relacionados directamente con tareas/vistas (p.ej., docs/whiteboard/chat/dashboards/automatizaciones/integraciones externas avanzadas/IA).

---

1) Resumen ejecutivo

Construiremos un gestor de tareas colaborativo y escalable con:

- **Jerarquía de organización** para contener tareas y delegar permisos.
- **Modelo de tarea completo** (campos, subtareas, checklist, comentarios, adjuntos, relaciones y dependencias).
- **Sistema de vistas** que permita observar y operar el mismo conjunto de tareas con distintos "lentes" (Lista, Tablero, Calendario, Gantt), con **agrupación, orden, filtros** y configuración persistente.
- **Barra de vistas** para gestionar vistas (crear, reordenar, fijar/pinear, privacidad, agrupar por tipo).

Principios

1. **Operación desde la vista:** el usuario resuelve la mayoría de acciones sin salir de la vista.
2. **Consistencia:** el mismo dato se refleja igual en todas las vistas.
3. **Configuración persistente:** vistas guardan filtros/orden/agrupación/columnas.
4. **Escalabilidad:** listas con miles de tareas deben ser fluidas.

---

2) Objetivos y métricas

Objetivos

- Proveer un flujo completo: **captura → organización → ejecución → planificación → seguimiento**.
- Permitir que equipos organicen trabajo por **estatus**, **responsables**, **fechas**, **prioridad**, **etiquetas**, **tipo** y **campos personalizados**.
- Garantizar permisos por contenedor y privacidad por vista.

Métricas de éxito (definición operativa)

- **TTFT (Time To First Task):** mediana de tiempo desde onboarding a primera tarea creada.
- **Triage Time:** tiempo para dejar una lista organizada (vistas guardadas con filtro+agrupación+orden).

- **Adopción de vistas:** % usuarios que usan ≥2 vistas por semana.
- **Eficiencia de operación:** % de ediciones realizadas “en vista” vs. en detalle.
- **Rendimiento:** p95 de render inicial de lista < 2s para 1.000 tareas; scroll virtualizado estable.

---

3) Alcance

Incluido

- Gestión de contenedores: **Workspace → Space → (Folder opcional) → List**
- Gestión de tareas: CRUD + campos + estados + subtareas + checklist + comentarios + adjuntos
- Relaciones y dependencias
- Vistas: **Lista, Tablero, Calendario, Gantt**
- Configuración de vistas: filtros, agrupación, orden, campos visibles
- Barra de vistas: crear, renombrar, reordenar, fijar, privacidad/compartir
- Permisos y colaboración mínima
- Búsqueda y acciones masivas
- Auditoría mínima de actividad por tarea

Excluido

- Documentos, chats, pizarras, dashboards, OKRs, time-tracking, automatizaciones, integraciones externas avanzadas, IA.

---

4) Usuarios, roles y permisos

Roles

- **Owner/Admin:** administra estructura y permisos.
- **Member:** trabaja en tareas según permisos del contenedor.
- **Guest:** acceso restringido a contenedores o vistas compartidas.

Modelo de permisos (mínimo viable)

Permisos se aplican por **contenedor** (Space/Folder/List) con herencia hacia abajo.

- **View container:** ver tareas y vistas.
- **Edit tasks:** editar tareas dentro del contenedor.
- **Manage views:** crear/editar vistas públicas del contenedor.
- **Manage structure:** crear/editar/eliminar contenedores.

Reglas clave

- Una **vista privada** solo la ve el creador y quienes se compartan explícitamente.
- Una **vista pública** la ve cualquiera con permiso de ver el contenedor.
- Acciones en una vista respetan permisos del contenedor (p.ej., no se puede editar tareas si solo se tiene lectura).

---

5) Conceptos y definiciones

- **Contenedor:** entidad que agrupa tareas y define permisos (Space/Folder/List).
- **Lista:** unidad mínima obligatoria donde viven las tareas.
- **Vista:** proyección configurable de tareas de un contenedor (o agregación de contenedores) con ajustes persistentes.
- **Grupo (Grouping):** partición del conjunto de tareas por un campo (p.ej., estatus, responsable).
- **Orden (Sorting):** criterio(s) para ordenar tareas dentro de un grupo o global.
- **Filtro:** predicado que incluye/excluye tareas.
- **Campos personalizados:** definiciones de campo por contenedor (o nivel superior), con valores por tarea.

---

6) Requerimientos funcionales

6.1 Jerarquía y navegación

Estructura

- **Workspace** contiene **Spaces**.
- **Space** contiene **Lists** y opcionalmente **Folders**.
- **Folder** contiene **Lists**.
- **List** contiene **Tasks**.

Requerimientos

- Crear/editar/eliminar Workspace/Space/Folder/List (según permisos).
- Navegación lateral por jerarquía.
- Cada List define:
- conjunto de **estatus**
- definiciones de **campos personalizados** (si se define a nivel List)

Reglas

- Una tarea **debe** pertenecer al menos a una List (**home_list**).
- (Opcional post-MVP) Una tarea puede pertenecer a múltiples Lists: **home_list** define estatus y campos personalizados “canónicos”.

6.2 Modelo de tarea (Task Core)

Campos estándar

- `id` (UUID)
- `title` (string, requerido)
- `description` (rich text o markdown)
- `status_id` (requerido, en el contexto de `home_list`)
- `priority` (enum: none/low/normal/high/urgent, configurable post-MVP)
- `start_at` (datetime opcional)
- `due_at` (datetime opcional)

- `assignees` (lista de user_ids; permitir 0..N)
- `watchers` (lista de user_ids)
- `tags` (0..N)
- `task_type` (enum mínimo: task; extensible)
- `created_by`, `created_at`, `updated_at`
- `closed_at` (si estatus final/cerrado)

Componentes

- **Subtareas:** tareas con `parent_task_id`.
- **Checklist:** lista(s) con ítems marcables.
- **Comentarios:** hilo por tarea; opcional asignación de comentario a usuario.
- **Adjuntos:** archivos asociados a la tarea.
- **Relaciones:** enlaces entre tareas (p.ej., “relacionada con”).
- **Dependencias:** restricciones de planificación (p.ej., A → B).

Reglas

- Cambiar `status_id` actualiza:
- visibilidad en vistas agrupadas
- `closed_at` si el estatus es terminal
- `due_at` no puede ser < `start_at` (si ambos existen) salvo configuración que lo permita.
- Subtarea hereda por defecto `home_list` del padre (configurable).

Operaciones requeridas

- Crear/editar/eliminar tarea.
- Cambiar estado, prioridad, asignados, fechas, etiquetas, tipo.
- Duplicar tarea (con opciones: incluir subtareas, checklist, adjuntos).
- Acciones masivas sobre selección (editar campos y mover de estatus).

6.3 Estados (Statuses)

Requerimientos

- Configuración de estatus por List:
- crear/editar/ordenar
- marcar estatus como **abierto** o **cerrado**
- definir color (cosmético)
- Mapeo: `status_id` pertenece a una List (o a un set heredado).

Reglas

- Estatus cerrados:
- tareas aparecen bajo filtros/segmentos de “cerrado”
- no se eliminan; se archivan lógicamente

6.4 Subtareas y jerarquía de tareas

Requerimientos

- Crear/editar subtareas desde:
- panel de subtareas en detalle
- vista de Lista (toggle: mostrar subtareas)
- Soportar **anidación** (mínimo 1 nivel; recomendado N niveles).

Reglas

- Una subtarea es una tarea con `parent_task_id`.
- Si el padre se elimina, comportamiento:
- (default) bloquear eliminación si tiene subtareas, o pedir confirmación para convertir subtareas en tareas raíz.

6.5 Checklist

Requerimientos

- Crear múltiples checklists por tarea.
- Ítems:
- texto
- asignación opcional
- estado completado
- orden

Reglas

- Completar checklist no cierra tarea automáticamente.

6.6 Comentarios y actividad

Comentarios

- CRUD de comentarios (con permisos).
- Menciones a usuarios (notificación básica).
- Adjuntar archivos en comentarios (opcional MVP).

Actividad (audit log mínimo por tarea)

Registrar eventos:

- creación
- cambio de estatus
- cambio de fechas
- cambios de asignados
- cambios de prioridad
- comentarios

Campos del evento: `event_id`, `task_id`, `actor_id`, `type`, `payload_json`, `created_at`.

6.7 Adjuntos

Requerimientos

- Adjuntar archivos a tareas.
- Lista de adjuntos con:
- nombre
- tamaño
- tipo
- autor
- fecha
- URL segura de descarga

Reglas

- Permisos de descarga iguales a permisos de ver tarea.
- Al borrar tarea, adjuntos pasan a un estado “huérfano” y se eliminan tras retención (configurable) o se eliminan inmediatamente (MVP: inmediato).

6.8 Relaciones y dependencias

Relaciones (link)

- Tipo: `relates_to` (MVP)
- Bidireccional.

Dependencias

- Tipo mínimo: **finish-to-start** (A termina → B puede empezar).
- CRUD desde:
- detalle de tarea
- vista Gantt (conectores)

Reglas de consistencia

- No permitir ciclos en dependencias.
- Si una tarea predecesora se mueve, puede disparar advertencia para sucesoras (MVP: advertencia; post-MVP: auto-shift opcional).

---

7) Sistema de Vistas (Views)

7.1 Modelo común de vista

Cada vista tiene:

- `view_id`

- `scope_type` (Space/Folder/List)
- `scope_id`
- `view_type` (list/board/calendar/gantt)
- `name`
- `is_pinned`
- `is_private`
- `shared_with` (lista de user_ids, solo si privada y compartida)
- `order_index`
- `settings_json` (columnas visibles, campos en tarjeta, etc.)
- `filters_json`
- `grouping_json`
- `sorting_json`

Comportamiento común

- **Persistencia:** cambios en filtros/orden/agrupación deben persistir (auto-save) o solicitar "Guardar" según configuración global de la organización.
- **Herencia:** vistas "por defecto" pueden existir por contenedor (al menos una lista y un tablero en cada lista).
- **Permisos:** crear/editar vista pública requiere `Manage views`.

7.2 Barra de Vistas (Views Bar)

Requerimientos

- Mostrar vistas del scope actual.
- Acciones:
- crear nueva vista
- renombrar
- duplicar
- eliminar
- reordenar (drag)
- pin/unpin
- cambiar privacidad
- agrupar por tipo (toggle)

Reglas

- Pinned se listan antes que unpinned.
- Reordenamiento persiste por usuario (si personal) o por contenedor (si pública) según política:
- MVP recomendado:
    - vistas públicas: orden global del contenedor
    - vistas privadas: orden por usuario

---

8) Requerimientos por tipo de vista

8.1 Vista Lista (List View)

Objetivo

Operación rápida, edición en línea, vista tabular configurable.

UI/Interacciones

- Tabla con filas (tareas) y columnas (campos).
- Expandir fila para ver:
- descripción breve
- subtareas (según configuración)
- checklist (opcional)

Configuración

- Columnas:
- agregar/quitar
- reordenar
- ancho
- Mostrar/ocultar subtareas.

Agrupación (Grouping)

Permitir agrupar por:

- estatus
- responsable
- prioridad
- etiqueta
- tipo
- fecha de vencimiento (bucket por rango: overdue/today/this week/no due date)
- campos personalizados soportados (ver sección 10)

Orden (Sorting)

- Multi-criterio (lista ordenada de reglas).
- Opción: ordenar dentro de cada grupo o global.

Filtros

- Builder de filtros con operadores por tipo:
- equals / not equals
- contains
- is empty / is not empty
- before/after/between para fechas
- in/not in para enums y multi-select

Regla crítica: herencia de contexto al crear

Cuando el usuario crea una tarea dentro de una vista con:

- **agrupación activa:** el valor del campo agrupado se asigna automáticamente.
- **filtros activos:** si el filtro fija un valor específico (p.ej., priority = high), la tarea nace con ese valor.

Acciones

- Crear tarea al final o dentro de un grupo.
- Edición en línea de campos.
- Selección múltiple y acciones masivas.
- Drag & drop:
- reordenar dentro de grupo
- mover entre grupos (si el grouping es por campo editable)

8.2 Vista Tablero (Board View)

Objetivo

Gestión visual de flujo por columnas, ideal para estados.

Modelo

- Columnas = valores del campo agrupado (default: estatus).
- Tarjetas = tareas.

Interacciones

- Drag & drop de tarjetas entre columnas:
- actualiza el campo agrupado.
- Reordenar tarjetas dentro de columna.
- Crear tarea dentro de columna:
- hereda el valor de la columna.

Configuración

- Campo de agrupación (default estatus; permitir otros campos enumerables).
- Subgrupos por segundo campo (filas):
- responsable/prioridad/tipo/etiqueta/campo personalizado enumerado.
- Campos visibles en tarjeta.

Acciones

- Abrir detalle desde tarjeta.
- Edición rápida mínima desde tarjeta (p.ej., asignado, fecha, prioridad) — configurable.

8.3 Vista Calendario (Calendar View)

Objetivo

Planificación por fechas; ver carga y mover tareas en el tiempo.

Requisitos

- Modos: día/semana/mes.
- Render de tareas según:
- `start_at` y/o `due_at`.
- Si solo existe `due_at`, mostrar como evento de un día (o “deadline marker”).
- Drag & drop:
- mover fecha ajusta `start_at` y/o `due_at` según regla.
- Resize (opcional MVP): ajustar duración.

Configuración

- Campo de fecha principal (due vs start; MVP: usar due como default).
- Campos visibles en tarjeta.
- Filtros (estatus, responsable, prioridad, etiquetas, tipo).

Reglas

- Si una tarea no tiene fechas, no aparece (o aparece en un bucket “sin fecha” lateral; MVP: no aparece).

8.4 Vista Gantt (Gantt View)

Objetivo

Planificación temporal con dependencias.

Requisitos

- Eje temporal con escalas: semana/mes (MVP), día (recomendado), trimestre (post-MVP).
- Barras de tareas por `start_at` → `due_at` (si falta start, inferir start = due - duración default).
- Sidebar con lista de tareas (jerarquía opcional: tarea/subtarea).
- Drag & drop de barra:
- mover ajusta fechas.
- Creación/edición de dependencias:
- conectar barras para crear dependencia.

Reglas

- Validación anti-ciclos.
- Dependencias violadas generan warning (p.ej., sucesora empieza antes de que predecesora termine).

---

9) Búsqueda, filtros globales y vistas personales

Búsqueda

- Búsqueda por texto sobre título y descripción.
- Filtros combinables con búsqueda.

“Modo personal” (My tasks)

- Toggle para mostrar tareas asignadas al usuario actual dentro del scope.
- Opción incluir/excluir subtareas.

---

10) Campos personalizados (Custom Fields)

Objetivo

Extender modelo sin tocar esquema cada vez.

Tipos soportados (MVP recomendado)

- `dropdown` (single-select)
- `multi_select`
- `checkbox`
- `date`
- `users` (multi-user)
- `number` (opcional MVP)
- `text` (opcional MVP)

Definición

- `field_id`, `scope_type`, `scope_id`, `name`, `type`, `options`, `is_required`, `default_value`, `order_index`.

Valores por tarea

- `task_id`, `field_id`, `value_json`.

Operaciones

- Crear/editar/eliminar campos (según permisos).
- Mostrar en:
- columnas de lista
- tarjetas de tablero
- filtros
- agrupación/orden (solo en tipos enumerables/fecha/users)

Reglas

- Eliminar un campo conserva historial (soft delete) o elimina valores (MVP: soft delete recomendado).

11) Notificaciones (mínimo viable)

- Notificar a:
- asignados cuando se asigna
- watchers cuando hay comentario o cambios relevantes
- menciones en comentarios

Canales (MVP):

- In-app (campana/inbox).
- Email (post-MVP).

---

12) Reglas de consistencia y sincronización

Fuente de verdad

- Backend es fuente de verdad.
- Cliente mantiene cache y aplica actualizaciones optimistas con rollback.

Concurrencia

- ETag/versionado por tarea.
- Conflictos:
- si dos usuarios editan el mismo campo, prevalece la última escritura (MVP) y se registra en actividad.

Reglas cross-view

- Cambios en una vista deben reflejarse en todas las vistas del mismo scope.
- Orden manual en una vista:
- se almacena por vista y por grupo (si aplica).

---

13) Requerimientos no funcionales

Performance

- Lista: virtualización de filas.
- Queries paginadas + filtros/orden en servidor.
- Índices en BD por: `home_list_id`, `status_id`, `assignees`, `due_at`, `updated_at`.

Disponibilidad

- SLA objetivo MVP: 99.5%.

Seguridad

- RBAC por contenedor.
- Validación server-side de permisos.

- URLs firmadas para adjuntos.

Observabilidad

- Logs estructurados.
- Métricas: latencia API, errores por endpoint, tiempo de render.

---

14) Diseño de datos (Especificación)

14.1 Entidades principales

Workspace

- `workspace_id`, `name`, `created_at`.

User

- `user_id`, `email`, `display_name`, `status`.

Space

- `space_id`, `workspace_id`, `name`, `visibility`, `created_at`.

Folder (opcional)

- `folder_id`, `space_id`, `name`.

List

- `list_id`, `parent_type` (space/folder), `parent_id`, `name`, `created_at`.

Status

- `status_id`, `list_id`, `name`, `category` (open/closed), `order_index`, `color`.

Task

- `task_id`, `home_list_id`, `parent_task_id` (nullable), `title`, `description`, `status_id`, `priority`, `start_at`, `due_at`, `task_type`, `created_by`, `created_at`, `updated_at`, `closed_at`.

TaskAssignee

- `task_id`, `user_id`.

TaskWatcher

- `task_id`, `user_id`.

Tag

- `tag_id`, `workspace_id`, `name`, `color`.

TaskTag

- `task_id`, `tag_id`.

Comment

- `comment_id`, `task_id`, `author_id`, `body`, `assigned_to` (nullable), `created_at`, `updated_at`.

Attachment

- `attachment_id`, `task_id`, `uploader_id`, `filename`, `mime_type`, `size_bytes`, `storage_key`, `created_at`.

Checklist

- `checklist_id`, `task_id`, `name`, `order_index`.

ChecklistItem

- `item_id`, `checklist_id`, `content`, `is_done`, `assignee_id` (nullable), `order_index`.

Relationship

- `relationship_id`, `task_id`, `other_task_id`, `type`.

Dependency

- `dependency_id`, `predecessor_task_id`, `successor_task_id`, `type`.

CustomFieldDefinition

- `field_id`, `scope_type`, `scope_id`, `name`, `type`, `options_json`, `is_required`, `default_value_json`, `order_index`, `is_deleted`.

CustomFieldValue

- `task_id`, `field_id`, `value_json`.

View

- `view_id`, `scope_type`, `scope_id`, `view_type`, `name`, `is_pinned`, `is_private`, `order_index`, `owner_id`, `settings_json`, `filters_json`, `grouping_json`, `sorting_json`.

ViewShare

- `view_id`, `user_id`.

TaskActivityEvent

- `event_id`, `task_id`, `actor_id`, `type`, `payload_json`, `created_at`.

14.2 Relaciones e integridad

- `Task.home_list_id` → `List.list_id`
- `Task.status_id` → `Status.status_id` (y status.list_id debe coincidir con home_list_id)
- `Task.parent_task_id` → `Task.task_id` (mismo home_list_id por defecto)
- Dependencias: no ciclos (validación al crear/actualizar)
- `View.scope_type/scope_id` define el conjunto consultable.

14.3 Consultas base (server)

- Listar tareas por scope con:
- filtros (AND/OR)
- orden multi-criterio
- paginación cursor-based
- Agregaciones por grouping (para encabezados y conteos)

---

15) API (contratos mínimos)

> Nota: no se prescribe REST vs GraphQL; se definen capacidades.

Tareas

- `CreateTask`
- `UpdateTask`
- `DeleteTask`
- `GetTask`
- `ListTasks(scope, filters, sorting, grouping, pagination)`

Estados

- `ListStatuses(list_id)`
- `CreateStatus`
- `UpdateStatus`
- `ReorderStatuses`

Vistas

- `CreateView`
- `UpdateView`
- `DeleteView`
- `ListViews(scope)`
- `ReorderViews(scope)`
- `ShareView`

Campos personalizados

- `CreateCustomField`
- `UpdateCustomField`
- `DeleteCustomField(soft)`
- `SetCustomFieldValue`

Comentarios / Adjuntos / Checklist

- CRUD básico.

Notificaciones

- `ListNotifications(user)`
- `MarkNotificationRead`

---

16) Reglas UX/UI (criterios de calidad)

- Acciones clave siempre accesibles en la vista:
- crear tarea
- cambiar estado
- asignar responsable
- ajustar fecha
- Feedback inmediato (optimista) con rollback si falla.
- Estados vacíos con CTA claras.
- Atajos de teclado (post-MVP recomendado).

---

17) QA — Criterios de aceptación por flujos

Flujo A: Captura y triage en Lista

1. Crear tarea en un grupo de estatus “Pendiente” → tarea nace con ese estatus.
2. Filtrar por prioridad alta y crear tarea → prioridad alta aplicada.
3. Editar asignado/fecha en línea → se refleja en Tablero y Calendario.

Flujo B: Kanban en Tablero

1. Crear tarea en columna “En progreso” → estatus correcto.
2. Arrastrar tarjeta a “Hecho” → tarea queda cerrada y aparece en “Cerrado”.

Flujo C: Planificación

1. Asignar fechas → aparece en Calendario.
2. Mover en Calendario → actualiza fechas.

Flujo D: Gantt y dependencias

1. Crear tareas con rangos de fecha → aparecen como barras.
2. Conectar dependencia A→B → se registra y valida.
3. Intentar crear ciclo → se bloquea.

---

18) Riesgos y decisiones

- Complejidad de Gantt: priorizar versión simplificada.

- Campos personalizados: limitar tipos en MVP.
- Multi-list membership: post-MVP para reducir complejidad.
- Concurrencia: política last-write-wins + auditoría.

---

19) Entregables

- Backend: modelo de datos + endpoints + permisos + consultas.
- Frontend: navegación + vistas + detalle de tarea + barra de vistas.
- QA: suites automatizadas (API + E2E) para flujos A–D.
- DevOps: CI/CD + observabilidad.

---

20) Anexos

A) Esquemas JSON sugeridos

filters_json (ejemplo)

```
{
  "op": "AND",
  "rules": [
    {"field": "priority", "operator": "eq", "value": "high"},
    {"field": "assignees", "operator": "contains", "value": "user_123"}
  ]
}
```

sorting_json (ejemplo)

```
[
  {"field": "due_at", "direction": "asc"},
  {"field": "priority", "direction": "desc"}
]
```

grouping_json (ejemplo)

```
{
  "field": "status_id",
  "mode": "buckets",
  "bucket_rules": null
}
```

settings_json (Lista — ejemplo)

```
{
  "columns": ["title", "status_id", "assignees", "due_at", "priority"],
```

```
  "show_subtasks": true,
  "row_density": "comfortable"
}
```

B) Convenciones de orden manual

- `order_index` por vista + por grupo.
- Para soportar drag&drop estable, mantener una clave ordenable (p.ej., rank lexicográfico o números con gaps).