Universidad Da Vinci De Guatemala

Facultad de Ingeniería

Carrera: Ingeniería en Sistemas

“Evaluación Final”

Ronald Isaias Godinez Hernández

202402155

Curso: Programación Web

Guatemala, diciembre de 2025

Índice

Introducción

La gestión eficiente de tareas académicas es un desafío recurrente para los estudiantes universitarios. La mayoría utiliza herramientas improvisadas como notas en el celular, recordatorios aislados o mensajes en aplicaciones de mensajería, lo que provoca desorganización, estrés y, en muchos casos, el olvido de fechas importantes. Frente a esta problemática, surge la necesidad de una solución centralizada, intuitiva y adaptada al contexto académico, que permita a los estudiantes planificar sus actividades de manera más clara y efectiva.

UniTask Planner es una propuesta de innovación web que busca abordar este desafío mediante una plataforma diseñada específicamente para estudiantes universitarios. Este proyecto integra un sistema de registro y visualización de tareas, asignación de prioridades, recordatorios automáticos y un dashboard inteligente que muestra información relevante derivada de consultas SQL. Su objetivo es mejorar la organización personal, optimizar la gestión del tiempo y ofrecer una herramienta moderna que acompañe al estudiante en su vida académica diaria.

Definición del problema y solución

El Problema

Los estudiantes universitarios suelen manejar sus tareas, proyectos, exámenes y recordatorios en muchos lugares diferentes: libretas, notas del celular, chats de WhatsApp, fotos del pizarrón, etc. Esto provoca que:

- Olviden fechas de entrega importantes.
- Se organicen de forma reactiva (“hago lo que me urge, no lo que más conviene”).
- No tengan claridad de la carga real de trabajo por semana.

El resultado es estrés, noches sin dormir y baja calidad en el trabajo entregado, no por falta de capacidad, sino por mala gestión del tiempo y las tareas.

La Solución: Aplicación Web “UniTask Planner”

UniTask Planner es una aplicación web diseñada para estudiantes universitarios que permite:

- Registrar tareas, proyectos, exámenes y actividades académicas.
- Asignar fecha de entrega, materia/curso y prioridad.
- Recibir recordatorios automáticos antes de la fecha límite.
- Ver una vista semanal y mensual de todas sus tareas.
- Mostrar recomendaciones de qué tarea es más conveniente hacer primero, según fecha de entrega y prioridad.

La funcionalidad principal del MVP será:

1. CRUD de tareas (crear, listar, actualizar, marcar como completada).
2. Recordatorios basados en fecha de entrega.
3. Dashboard con vista de tareas próximas por vencer.

Justificación: ¿Por qué es innovadora o necesaria?

Aunque existen apps genéricas de to-do o calendarios, pocas están enfocadas específicamente en el contexto universitario**,** donde:

- Las tareas están ligadas a cursos/materias.
- Hay periodos fuertes (parciales, exámenes finales, proyectos).
- Se requiere priorizar entre varias entregas a corto plazo.

UniTask Planner aporta valor porque:

- Traducirá la carga académica en una agenda clara y priorizada**.**
- Centraliza todas las tareas por curso en un solo lugar.
- Ofrece recomendaciones simples de **"**qué hacer hoy**"** basadas en urgencia e importancia.
- Puede integrarse a futuro con calendarios u otros servicios (visión a largo plazo).

Propuesta Técnica (Arquitectura)

Arquitectura General

El sistema seguirá una arquitectura clásica cliente–servidor**:**

- **Frontend (Cliente Web):** SPA (Single Page Application) desarrollada con React.
- **Backend (API REST):** Servidor en Node.js con Express.
- **Base de Datos:** PostgreSQL (SQL).
- **Prototipo visual:** Generado con v0 para mostrar la interfaz propuesta en el video y en el documento.

Frontend: Comparativa y elección

Tecnologías evaluadas:

- **React:**
  - Ventajas: gran comunidad, abundante documentación, componentes reutilizables, fácil integración con APIs REST.
  - Ideal para SPA con estados dinámicos (listas de tareas, filtros, etc.).
- **Vue:**
  - Ventajas: curva de aprendizaje suave, sintaxis muy amigable, buena para proyectos pequeños y medianos.

- **Angular:**
  - Ventajas: framework muy completo, incluye muchas herramientas integradas.
  - Desventajas: curva de aprendizaje más alta, más pesado para un proyecto pequeño.

Elección final: React

Se elige React porque:

- Permite crear interfaces reactivas y componentes reutilizables para tareas, tarjetas, dashboard, etc.
- Es ampliamente utilizado en la industria, por lo que da una visión más profesional.
- Se integra muy bien con APIs REST y herramientas como Vercel para despliegue.

Backend y Publicación

Lenguaje/Framework backend:

Node.js + Express

Ventajas:

- Muy usado para APIs REST.
- Ecosistema maduro de paquetes (JWT, validación, ORMs, etc.).
- JavaScript tanto en frontend como backend (facilita el desarrollo).

Publicación:

- **Frontend:** Vercel (ideal para proyectos React, despliegue sencillo directamente desde GitHub).
- **Backend:** Puede publicarse en servicios como Render / Railway / AWS (en el documento puedes mencionar uno concreto, por ejemplo *Render*).
- **Base de Datos:** PostgreSQL en el mismo proveedor del backend (por ejemplo, base de datos gestionada en Render).

Persistencia de Datos (Base de Datos)

Elección: SQL (PostgreSQL)

Justificación:

- Las tareas universitarias tienen estructura clara: título, descripción, fecha de entrega, curso, estado, prioridad.
- Las relaciones son bien definidas (un usuario tiene muchas tareas).
- Necesitamos consultas como: “tareas próximas a vencer”, “tareas por curso”, etc.
- Un modelo relacional es ideal para este tipo de datos.

Esquema de Base de Datos (Modelo Lógico)

Tabla: usuarios

- id_usuario (PK, serial)
- nombre
- correo (único)
- contraseña_hash
- fecha_registro

Tabla: cursos

- id_curso (PK, serial)
- id_usuario (FK → usuarios.id_usuario)
- nombre_curso
- codigo_curso (ej. BD101)
- color (para identificar visualmente)

Tabla: tareas

- id_tarea (PK, serial)
- id_usuario (FK → usuarios.id_usuario)
- id_curso (FK → cursos.id_curso, opcional)
- titulo
- descripcion
- fecha_entrega (date/time)
- prioridad (baja, media, alta)
- estado (pendiente, en_progreso, completada)
- fecha_creacion
- fecha_actualizacion

Tabla: recordatorios

- id_recordatorio (PK, serial)
- id_tarea (FK → tareas.id_tarea)
- fecha_recordatorio
- enviado (boolean)

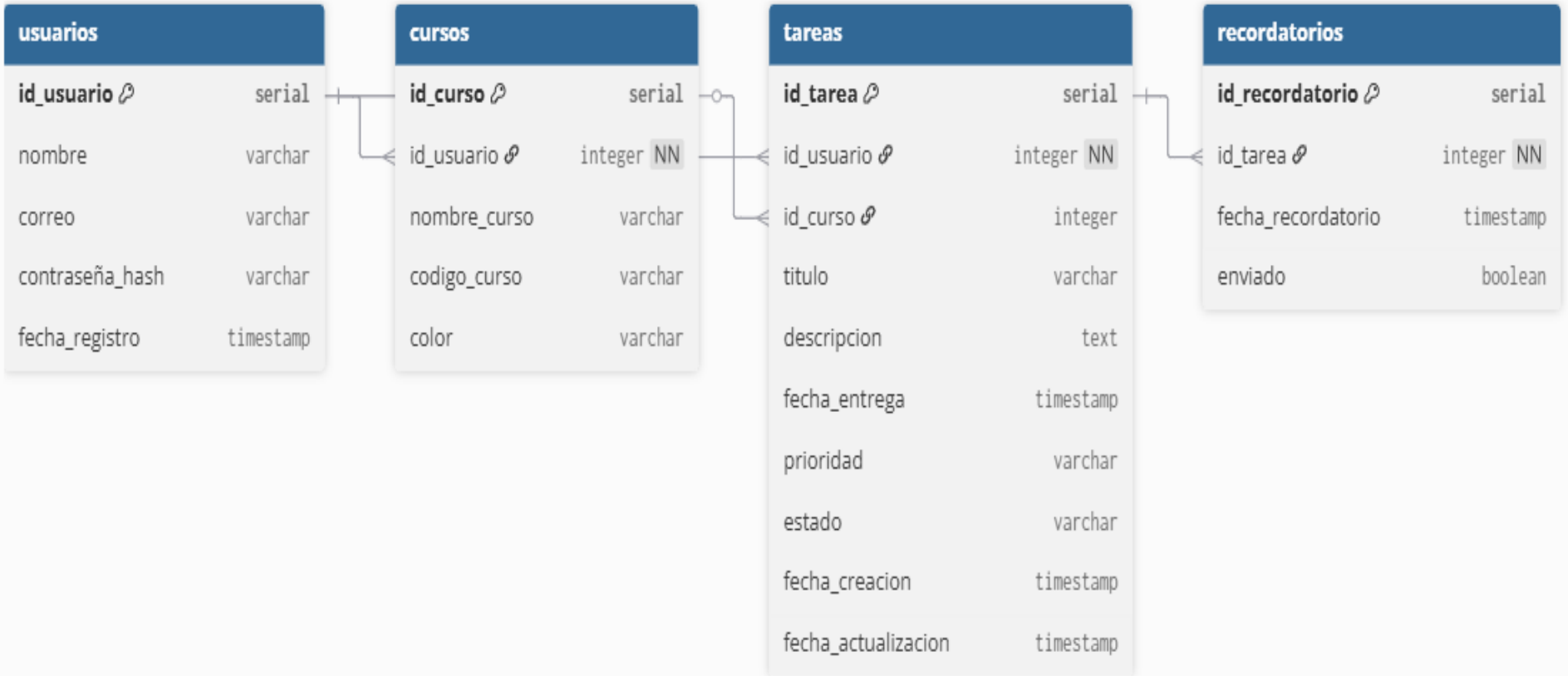
Este modelo permite:

- Ver tareas por usuario.
- Organizar por cursos.
- Programar y registrar recordatorios enviados.

FACULTAD DE INGENIERÍA

UNIVERSIDAD DA VINCI DE GUATEMALA

Diagrama EDR



Diseño de Interfaz de Programación (API)

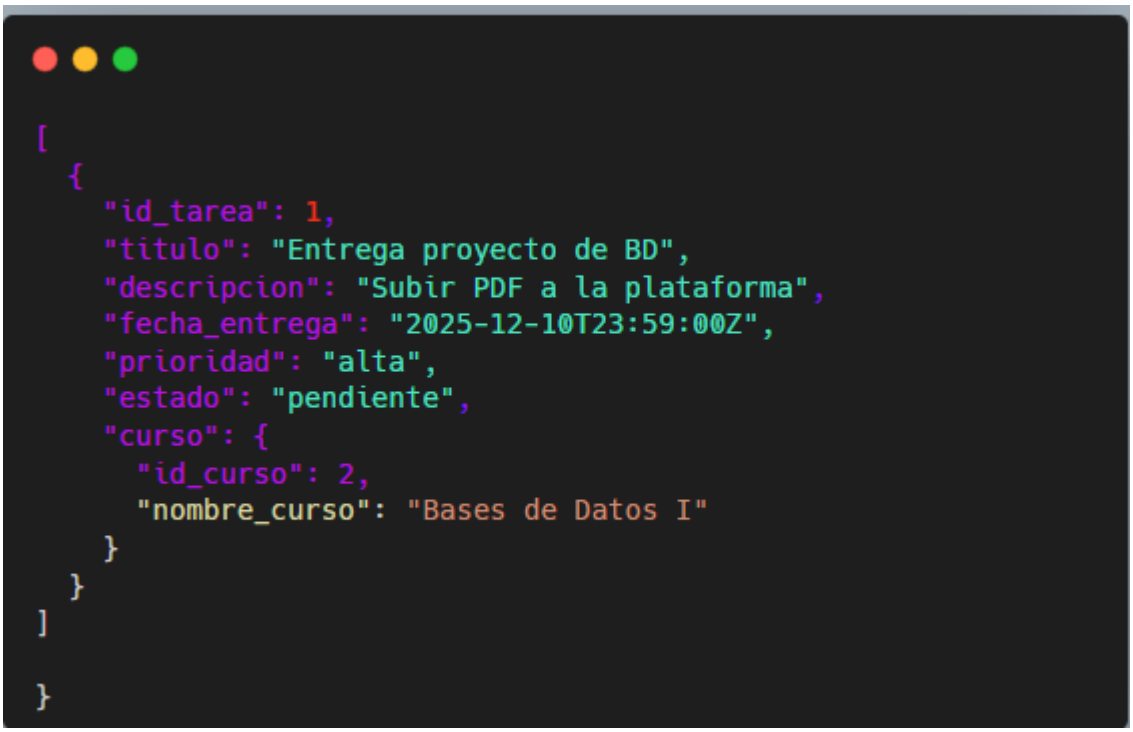
Endpoint 1: Obtener lista de tareas próximas

- **Método:** GET
- **Ruta:** /api/v1/tareas
- **Descripción:** Retorna todas las tareas del usuario autenticado, con opción de filtrar por estado o rango de fechas.

Parámetros de consulta (query):

- estado: pendiente, en progreso, completada.
- desde: fecha mínima de entrega.
- hasta: fecha máxima de entrega.

Respuesta Exitosa (200 OK):



```
[
  {
    "id_tarea": 1,
    "titulo": "Entrega proyecto de BD",
    "descripcion": "Subir PDF a la plataforma",
    "fecha_entrega": "2025-12-10T23:59:00Z",
    "prioridad": "alta",
    "estado": "pendiente",
    "curso": {
      "id_curso": 2,
      "nombre_curso": "Bases de Datos I"
    }
  }
]

}
```

Endpoint 2: Crear nueva tarea

- **Método:** POST
- **Ruta:** /api/v1/tareas
- **Descripción:** Registra una nueva tarea asociada al usuario.

Cuerpo de la petición (JSON):

```json
{
  "titulo": "Estudiar para examen parcial",
  "descripcion": "Temas: normalización y modelo relacional",
  "fecha_entrega": "2025-12-08T18:00:00Z",
  "prioridad": "alta",
  "id_curso": 2
}
```

Respuesta Exitosa (201 Created):

```json
{
  "id_tarea": 5,
  "mensaje": "Tarea creada exitosamente"
}
```

Endpoint 3: Actualizar estado de una tarea (marcar como completada)

- **Método:** PATCH
- **Ruta:** /api/v1/tareas/:id_tarea
- **Descripción:** Actualiza campos específicos de una tarea, por ejemplo, el estado.

Cuerpo de la petición (JSON):

```
{
  "estado": "completada"
}
```

Respuesta Exitosa (200 OK):

```
{
  "id_tarea": 5,
  "estado": "completada",
  "mensaje": "Tarea actualizada correctamente"
}
```

Planificación y Costos

Estimación de Esfuerzo

	Tarea	Horas
1	Análisis del problema y definición del MVP	**3**
2	Diseño de modelo de datos (ERD)	**4**
3	Diseño de arquitectura (frontend/backend)	**3**
4	Implementación backend (API básica)	**10**
5	Implementación frontend (pantallas clave)	**10**
6	Integración frontend–backend	**5**
7	Configuración de despliegue (Vercel, etc.)	**3**
8	Pruebas básicas y ajustes	**4**
9	Elaboración del documento técnico	**5**
10	Preparación del pitch y grabación de video	**3**

Total, estimado sin buffer:
3 + 4 + 3 + 10 + 10 + 5 + 3 + 4 + 5 + 3 = **50 horas**

20% de 50 horas = 10 horas extra
Total ajustado = 60 horas

Siguiendo la recomendación de agregar un **20% de colchón** por incertidumbre:

Como desarrollador junior, tiene un costo de **Q 40.00 por hora**.
El esfuerzo total del proyecto se estima en **60 horas**, incluyendo un 20% de colchón.
El costo total del desarrollo sería de aproximadamente **Q 2,400.00**.

Consultas SQL

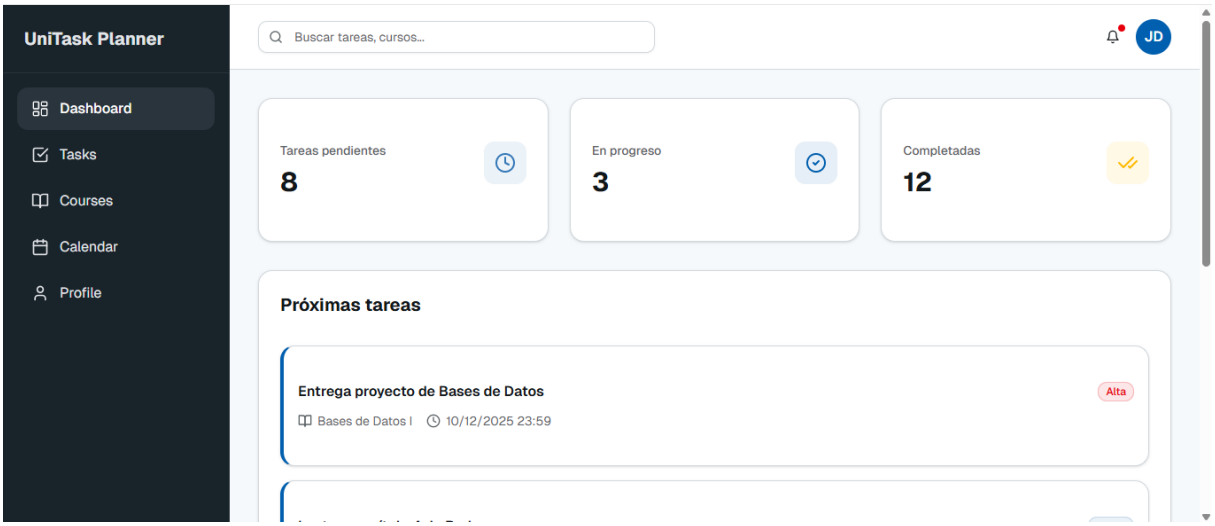
Consulta 1: Tareas próximas a vencer (para el Dashboard)

```sql
SELECT
    t.id_tarea,
    t.titulo,
    t.fecha_entrega,
    t.prioridad,
    t.estado,
    c.nombre_curso
FROM tareas t
LEFT JOIN cursos c ON t.id_curso = c.id_curso
WHERE
    t.id_usuario = :id_usuario
    AND t.estado <> 'completada'
    AND t.fecha_entrega >= NOW()
ORDER BY t.fecha_entrega ASC
LIMIT 5;
```

Resultado simulado – Consulta 1 (próximas tareas)

```json
[
  {
    "id_tarea": 10,
    "titulo": "Entrega proyecto de Bases de Datos",
    "fecha_entrega": "2025-12-10T23:59:00",
    "prioridad": "alta",
    "estado": "pendiente",
    "nombre_curso": "Bases de Datos I"
  },
  {
    "id_tarea": 12,
    "titulo": "Lectura capítulo 4 de Redes",
    "fecha_entrega": "2025-12-09T20:00:00",
    "prioridad": "media",
    "estado": "pendiente",
    "nombre_curso": "Redes de Computadoras"
  }
]
```

Interfaz en vO



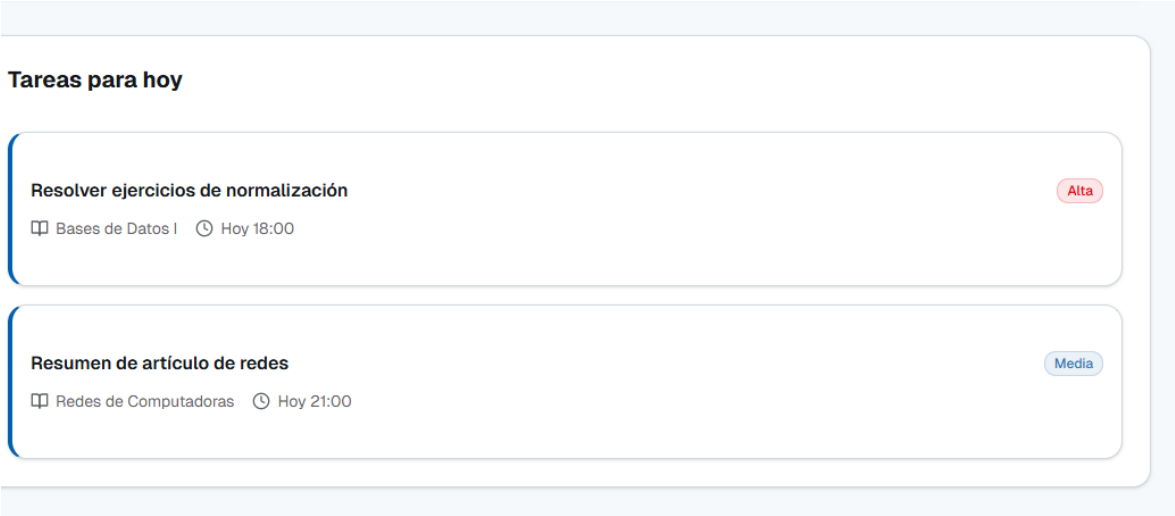
Consulta 2: Tareas para hoy

```sql
SELECT
    t.id_tarea,
    t.titulo,
    t.fecha_entrega,
    t.prioridad,
    c.nombre_curso
FROM tareas t
LEFT JOIN cursos c ON t.id_curso = c.id_curso
WHERE
    t.id_usuario = :id_usuario
    AND DATE(t.fecha_entrega) = CURRENT_DATE
    AND t.estado <> 'completada'
ORDER BY t.fecha_entrega ASC;
```

Resultado simulado – Consulta 2 (tareas para hoy)


```json
[
  {
    "id_tarea": 15,
    "titulo": "Resolver ejercicios de normalización",
    "fecha_entrega": "2025-12-04T18:00:00",
    "prioridad": "alta",
    "nombre_curso": "Bases de Datos I"
  },
  {
    "id_tarea": 16,
    "titulo": "Resumen de artículo de redes",
    "fecha_entrega": "2025-12-04T21:00:00",
    "prioridad": "media",
    "nombre_curso": "Redes de Computadoras"
  }
]
```

Interfaz en vO



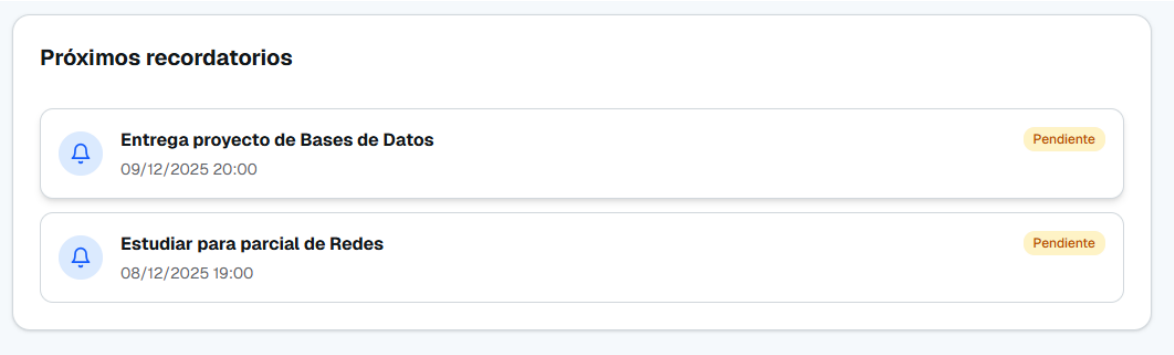
Consulta 3: Próximos recordatorios programados

```sql
SELECT
    r.id_recordatorio,
    t.titulo,
    r.fecha_recordatorio,
    r.enviado
FROM recordatorios r
INNER JOIN tareas t ON r.id_tarea = t.id_tarea
WHERE
    t.id_usuario = :id_usuario
    AND r.enviado = FALSE
    AND r.fecha_recordatorio >= NOW()
ORDER BY r.fecha_recordatorio ASC
LIMIT 5;
```

Resultado simulado – Consulta 3 (recordatorios)

```json
[
  {
    "id_recordatorio": 5,
    "titulo": "Entrega proyecto de Bases de Datos",
    "fecha_recordatorio": "2025-12-09T20:00:00",
    "enviado": false
  },
  {
    "id_recordatorio": 6,
    "titulo": "Estudiar para parcial de Redes",
    "fecha_recordatorio": "2025-12-08T19:00:00",
    "enviado": false
  }
]
```

Interfaz vO


Próximos recordatorios
Entrega proyecto de Bases de Datos
09/12/2025 20:00
Pendiente
Estudiar para parcial de Redes
08/12/2025 19:00
Pendiente

Conclusión

El desarrollo de UniTask Planner permitió demostrar cómo una arquitectura bien planteada, un modelo de datos sólido y un prototipo visual funcional pueden converger para resolver una necesidad real dentro del ámbito educativo. A través de consultas SQL, representaciones visuales en v0 y una propuesta técnica clara, se evidenció que es posible transformar información académica dispersa en un sistema estructurado y accesible que facilita la toma de decisiones del estudiante. La aplicación, aunque en fase prototipo, presenta una base firme para una implementación completa a futuro.

Finalmente, este proyecto destaca la importancia de la organización académica y el impacto que una herramienta tecnológica puede tener en el rendimiento y bienestar del estudiante. UniTask Planner no solo representa una solución innovadora, sino también un paso hacia una cultura de estudio más ordenada y consciente. Con un mayor desarrollo y ampliación de funcionalidades, esta aplicación tiene el potencial de convertirse en un aliado indispensable para la comunidad universitaria.