Gestor de Tareas

10/02/2026

—

DevSecOps Team

Stack: Vue 3 + Laravel 10 + PostgreSQL
Nivel: Junior

🎯 Objetivo

Construir una pequeña aplicación web para **gestionar tareas (To-Do)** con un frontend en Vue y una API REST en Laravel.

📌 Requerimientos funcionales

1️⃣ Backend – Laravel + PostgreSQL

📂 **Modelo:**

Requerimientos no funcionales

- Organización del código
- Separación de responsabilidades
- Diseño limpio
- Buenas prácticas REST + Vue
- **UX/UI:** Interfaz intuitiva y responsiva. Feedback visual claro (mensajes de éxito/error, indicadores de carga).

📂 Modelo:

Entidades

Entidad	Campo	Tipo	Notas
Task	`id`	`UUID/BigInt`	Clave primaria
	`title`	`String`	Título de la tarea principal
	`description`	`Text`	Descripción detallada
	`status`	`Enum`	`(pending, inProgress, done)`
	`createdAt`	`Timestamp`	Fecha de creación
Item	`id`	`UUID/BigInt`	Clave primaria
	`taskId`	`UUID/BigInt`	Clave foránea, relación con `Task`
	`title`	`String`	Nombre del sub-item

Entidad	Campo	Tipo	Notas
	`isCompleted`	`Boolean`	Estado de completado (true/false)
	`priority`	`Enum`	`(low, medium, high)`

Relación:

- Una `Task` tiene muchos `Items` (1:N).
- Un `Item` pertenece a una `Task` (N:1).

Reglas de negocio:

- Si todos los `Items` asociados a una `Task` están completos (`isCompleted = true`) → la `Task` puede marcarse como `"done"`.
- No se puede marcar una `Task` como `"done"` si tiene `Items` incompletos (`isCompleted = false`).
- **Validaciones:** Se deben implementar utilizando `Form Requests` en Laravel.
- **UX/UI:** El sistema debe proveer una **retroalimentación visual** inmediata al intentar cambiar el estado de una tarea, mostrando un mensaje de error claro y amigable si la regla de negocio (items incompletos) se incumple. Por ejemplo, deshabilitar el botón "Marcar como Terminada" si hay ítems pendientes y mostrar un *tooltip* explicativo.

🔧 Funcionalidades API

Crear una API REST con las siguientes rutas:

Método	Endpoint	Acción
GET	/api/tasks	Listar tareas
POST	/api/tasks	Crear tarea
PUT	/api/tasks/{id}	Actualizar tarea

DELETE	/api/tasks/{id}	Eliminar tarea

✔ Validar datos de entrada
✔ Usar Eloquent
✔ Respuestas en JSON
✔ Usar PostgreSQL como base de datos

2️⃣ Frontend – Vue 3

🖥️ Funcionalidades

Crear una interfaz simple que permita:

- Ver listado de tareas
- Crear una nueva tarea
- Marcar una tarea como completada
- Eliminar una tarea

🧩 Requisitos técnicos

- Vue 3 (Composition API **o** Options API)
- Uso de `axios` o `fetch`
- Manejo básico de estado
- Componentes reutilizables (mínimo 2)
- Mostrar mensajes simples de error o éxito

⭐ Extras (opcional, suma puntos)

(No obligatorios, pero bien valorados)

- Filtros: completadas / pendientes
- Uso de migraciones y seeders
- Uso de FormRequest para validaciones
- Diseño simple con CSS o framework ligero (Tailwind)
- README con instrucciones para correr el proyecto

📦 Entregables

- Repositorio Git (GitHub / GitLab)
- Instrucciones claras para:
    - Backend (`php artisan serve`, migraciones)
    - Frontend (`npm install`, `npm run dev`)
- Código funcional y entendible
- Docker compose para levantamiento del proyecto.
- Despliegue en prod del proyecto : Vercel , Laravel Cloud , Digitalocean , Aws , Gcp , Azure , etc.

🧠 Criterios de evaluación

- Comprensión de Vue y Laravel
- Organización del código
- Buenas prácticas básicas
- Correcto uso de PostgreSQL
- Claridad en commits y README
- Despliegue en producción
