**Prueba Técnica - Task Management System**

**📋 Descripción General**

**Desarrollar un Sistema de Gestión de Tareas basado en el diseño proporcionado, implementando un tablero Kanban con columnas de estado, gestión completa de tareas y un dashboard con estadísticas visuales.**

**Duración máxima: 3 días (72 horas desde el inicio)**

**🎨 Diseño de Referencia**

**Interfaz Principal - Task Board**

**La interfaz debe mostrar:**

* **Header:** 
  + **Título "Tasks List" a la izquierda**
  + **Avatares de miembros del equipo (centro-derecha)**
  + **Botón "Add New" (verde, esquina derecha)**
* **Tablero Kanban con 4 columnas:** 
  + **In Progress (indicador morado)**
  + **Reviews (indicador naranja)**
  + **Completed (indicador verde)**
  + **Done (indicador azul)**

**Cada columna muestra el contador de tareas (ej: "In Progress 4")**

**Card de Tarea - Especificaciones**

**Cada tarjeta debe mostrar:**

* **Título de la tarea (ej: "Test Planning", "User Acceptance Test")**
* **Tags/Categorías como badges coloreados (ej: UX/UI, Design, Development, Marketing)**
* **Footer con:** 
  + **ID de tarea (ej: 29, 125, 281)**
  + **Ícono y contador de comentarios**
  + **Ícono y contador de archivos adjuntos**
  + **Avatares de usuarios asignados (máximo 3 visibles)**

**🛠 Stack Tecnológico Requerido**

**Frontend**

* **Framework: Nuxt 3 o 4**
* **Estilos: TailwindCSS**
* **Componentes UI: shadcn/ui (OBLIGATORIO)**
* **Gestión de Estado: Pinia**
* **TypeScript: Obligatorio**
* **Iconos: Lucide Icons**

**Backend**

* **Framework: AdonisJS 6**
* **Base de Datos: PostgreSQL**
* **ORM: Lucid**
* **Validación: VineJS**

**📝 Requerimientos Funcionales**

**1. Modelo de Base de Datos**

**Tabla: users**

**- id (SERIAL PRIMARY KEY)**

**- email (VARCHAR UNIQUE NOT NULL)**

**- full\_name (VARCHAR NOT NULL)**

**- avatar\_url (VARCHAR NULL)**

**- created\_at (TIMESTAMP)**

**- updated\_at (TIMESTAMP)**

**Tabla: tasks**

**- id (SERIAL PRIMARY KEY)**

**- title (VARCHAR NOT NULL)**

**- status (VARCHAR NOT NULL) -- 'in\_progress' | 'reviews' | 'completed' | 'done'**

**- created\_by (INTEGER REFERENCES users(id))**

**- deleted\_at (TIMESTAMP NULL) -- Soft Delete obligatorio**

**- created\_at (TIMESTAMP)**

**- updated\_at (TIMESTAMP)**

**Tabla: task\_categories**

**- id (SERIAL PRIMARY KEY)**

**- name (VARCHAR NOT NULL) -- UX/UI, Design, Development, Marketing, etc.**

**- color (VARCHAR NOT NULL) -- Hex color**

**- created\_at (TIMESTAMP)**

**- updated\_at (TIMESTAMP)**

**Tabla: task\_category\_pivot**

**- task\_id (INTEGER REFERENCES tasks(id) ON DELETE CASCADE)**

**- category\_id (INTEGER REFERENCES task\_categories(id))**

**PRIMARY KEY (task\_id, category\_id)**

**Tabla: task\_assignees**

**- task\_id (INTEGER REFERENCES tasks(id) ON DELETE CASCADE)**

**- user\_id (INTEGER REFERENCES users(id))**

**- assigned\_at (TIMESTAMP DEFAULT NOW())**

**PRIMARY KEY (task\_id, user\_id)**

**Tabla: task\_comments**

**- id (SERIAL PRIMARY KEY)**

**- task\_id (INTEGER REFERENCES tasks(id) ON DELETE CASCADE)**

**- user\_id (INTEGER REFERENCES users(id))**

**- comment (TEXT NOT NULL)**

**- created\_at (TIMESTAMP)**

**- updated\_at (TIMESTAMP)**

**Tabla: task\_attachments**

**- id (SERIAL PRIMARY KEY)**

**- task\_id (INTEGER REFERENCES tasks(id) ON DELETE CASCADE)**

**- file\_name (VARCHAR NOT NULL)**

**- file\_path (VARCHAR NOT NULL)**

**- file\_size (INTEGER NOT NULL)**

**- uploaded\_by (INTEGER REFERENCES users(id))**

**- created\_at (TIMESTAMP)**

**2. Funcionalidades Core**

**A. Gestión de Tareas (CRUD)**

**Crear Tarea (Modal con botón "Add New"):**

* **Campo de título (requerido)**
* **Selector de estado inicial**
* **Multi-selector de categorías**
* **Multi-selector de usuarios asignados**
* **Validación en frontend y backend**

**Visualización de Tareas:**

* **Distribuidas en columnas según estado**
* **Mostrar toda la información especificada en el diseño**
* **Contadores actualizados en tiempo real**

**Editar Tarea (Click en card):**

* **Modal/Sheet con formulario de edición**
* **Cambiar título, estado, categorías y asignados**
* **Actualización inmediata en el board**

**Eliminar Tarea:**

* **Soft delete (no borrado físico)**
* **Opción de restaurar desde una vista de "Tareas Eliminadas"**
* **Confirmación antes de eliminar**

**Cambio de Estado:**

* **Click en botón dentro del card o mediante edición**
* **Actualización visual inmediata**
* **Persistencia en base de datos**

**B. Sistema de Comentarios y Archivos**

**Comentarios:**

* **Agregar comentario desde el detalle de tarea**
* **Listado cronológico con autor y timestamp**
* **Contador visible en el card**

**Archivos:**

* **Upload de archivos (máximo 5MB)**
* **Tipos permitidos: PDF, DOC, DOCX, PNG, JPG**
* **Descarga de archivos**
* **Contador visible en el card**

**3. Dashboard de Estadísticas**

**Crear una vista /dashboard con los siguientes componentes usando shadcn/ui:**

**Cards de Métricas**

**// Usar el componente Card de shadcn**

**- Total de Tareas Activas**

**- Tareas Completadas este Mes**

**- Tareas en Revisión**

**- Promedio de Finalización (días)**

**Gráfico de Barras - Tareas por Estado**

**// Usar BarChart de shadcn/ui**

**💻 Requerimientos de UI/UX**

**Componentes shadcn/ui Obligatorios:**

**// Layout**

**- Card // Cards de tareas y métricas**

**- Button // Acciones (Add New, cambiar estado)**

**- Badge // Tags de categorías**

**// Formularios**

**- Dialog/Sheet // Modales para crear/editar**

**- Form // Con react-hook-form y zod**

**- Input // Campos de texto**

**- Select // Estados**

**- MultiSelect // Categorías y usuarios**

***Utilizar Zod para schemas de validaciones en formularios***

**// Feedback**

**- Toast // Notificaciones**

**- AlertDialog // Confirmaciones**

**🎯 Criterios de Evaluación**

**1. Implementación Funcional**

* **CRUD completo funcionando**
* **Soft deletes implementado correctamente**
* **Dashboard con todos los gráficos**
* **Manejo de archivos y comentarios**

**2. Calidad del Código**

* **TypeScript: Interfaces y tipos bien definidos**
* **Organización: Estructura clara y modular**
* **Reutilización: Componentes y composables**
* **Nomenclatura: Consistente y descriptiva**

**3. Uso de shadcn/ui**

* **Implementación correcta de componentes**
* **Personalización coherente con el diseño**
* **Uso de variantes y props adecuados**

**4. Backend y Base de Datos**

* **Migraciones y modelos bien estructurados**
* **Validaciones con VineJS**
* **Queries optimizadas**
* **Manejo de errores**

**5. UI/UX**

* **Fidelidad al diseño propuesto**
* **Estados de carga y error**
* **Feedback visual (toasts, loading)**

**📦 Entregables**

1. **Repositorio Git con:**
   * **README.md con instrucciones de instalación**
   * **.env.example con variables necesarias**
   * **Seeders con datos de prueba**
   * **Commits descriptivos**
2. **Migraciones del backend de adonis :**
   * **Script de migraciones**
   * **Seeder con:** 
     + **4 categorías (UX/UI, Design, Development, Marketing)**
     + **15-20 tareas distribuidas en los estados**
3. **Documentación:**
   * **Colección de Postman**
   * **Comentarios en funciones complejas**
   * **Explicación de decisiones técnicas tomadas**

**⚠️ Puntos Importantes**

* **NO implementar: Drag & drop, filtros, ordenamiento, búsqueda**
* **Obligatorio: Soft deletes, TypeScript, shadcn/ui**
* **IDs secuenciales: Usar SERIAL en PostgreSQL**
* **Sin autenticación: Para simplificar, usar un usuario fijo en el backend**

**🚀 Puntos Extra (Opcionales)**

* **Tests funcionales en el backend**
* **Animaciones suaves en las transiciones**
* **Dark mode**

**Tiempo estimado por área:**

* **Día 1: Backend completo + Base de datos**
* **Día 2: Frontend con task board funcional**
* **Día 3: Dashboard + pulido final**

**¡Éxito! 🎯**