Prueba técnica: Catálogo de productos (Full Stack Medium)

Objetivo

Construir una aplicación web tipo **Catálogo** que permita administrar **categorías** y **productos**, incluyendo:

- CRUD completo de **Categorías**
- CRUD completo de **Productos**
- Listado de productos con **paginación y filtros desde el backend**, consultando a **SQL Server**
- Realizar un endpoint para Cargar masivamente los datos de productos mediante un xlsx o csv.
- Campos de auditoría en tablas: **FechaCreacion** y **FechaModificacion**

El candidato tiene libertad de definir **tipos de datos**, longitudes, índices y llaves, pero debe justificar decisiones importantes (por ejemplo: índices para filtros).

Stack permitido (obligatorio)

Backend (elige 1)

- **.NET (Web API)** o
- **NodeJS (Express/NestJS)**

Frontend

- **React**
- **Ant Design** (obligatorio para UI)
- **Redux Toolkit** (opcional)

Base de datos

- **SQL Server**

Reglas y alcance

1. **No usar paginación en frontend** (o sea, no traer todo y paginar en UI).
    - La tabla debe pedir datos al backend con page, pageSize, filters, sort.

2. Los filtros deben aplicarse **en el backend y preferiblemente en SQL Server** (query parametrizada).
3. Se debe manejar un flujo real: **crear / editar / eliminar / listar**.
4. Validaciones mínimas:
    - Categoría: nombre obligatorio y único
    - Producto: nombre obligatorio, precio > 0, categoría obligatoria
5. Manejo de errores: respuestas claras y consistentes.
6. Auditoría:
    - FechaCreacion se asigna al crear
    - FechaModificacion se actualiza al modificar
7. Entregar README con pasos de ejecución.

Modelo de datos (inventado, tipos libres)

Tabla: Categorias

Campos sugeridos (puedes agregar más si lo consideras):

- **IdCategoria**
- **Nombre**
- **Descripcion**
- **Activo** (para soft delete o estado)
- **FechaCreacion**
- **FechaModificacion**

Tabla: Productos

Campos sugeridos:

- **IdProducto**
- **IdCategoria** (FK)
- **Nombre**
- **Descripcion**
- **Sku** (opcional pero recomendado como identificador)
- **Precio**

- **Stock**
- **Activo**
- **FechaCreacion**
- **FechaModificacion**

El candidato decide tipos (int, uniqueidentifier, nvarchar, decimal, etc.), constraints, índices.

Funcionalidades requeridas

1) Módulo de Categorías

- Listar categorías (puede ser paginado o simple, tu decisión)
- Crear categoría
- Editar categoría
- Eliminar categoría (hard delete o soft delete, explicar decisión)
- Validar nombre único

2) Módulo de Productos

- Listado principal con tabla (Ant Design Table)
- Crear producto
- Editar producto
- Eliminar producto (hard o soft)
- Detalle opcional (modal o página)

Listado de productos (lo más importante)

Tabla con paginación backend

Debe incluir:

- Columnas: Nombre, Categoría, Precio, Stock, Activo, Fechas, Acciones (editar/eliminar)
- **Paginación**: page, pageSize
- **Ordenamiento**: por Nombre, Precio, FechaCreacion (mínimo 2 campos)
- **Filtros** (mínimo 3):
    - Por texto: Nombre (contains)

- Por categoría: IdCategoria
- Por rango de precio: PrecioMin y PrecioMax
- (extra opcional) Activo (true/false)

El backend debe responder con:

- items (lista paginada)
- total (total de registros que cumplen filtros)
- page, pageSize

Contrato de API

Categorías

- GET /api/categorias
- POST /api/categorias
- PUT /api/categorias/{id}
- DELETE /api/categorias/{id}

Productos

- GET /api/productos?page=1&pageSize=10&search=teclado&idCategoria=3&precioMin=100&precioMax=500&activo=true&sortBy=precio&sortDir=asc
- GET /api/productos/{id}
- POST /api/productos
- POST /api/productosMasivo
- PUT /api/productos/{id}
- DELETE /api/productos/{id}

Puedes implementar filtros con DTO/query params. Se evaluará que sea limpio y escalable.

Requisitos técnicos evaluables

Backend

- Arquitectura básica por capas (Controller → Service → Repository o similar)
- Consultas seguras:

- Query parametrizada / ORM bien usado
- Nada de concatenar strings para SQL

- Paginación/filtrado hecho en SQL Server
- Logging básico (opcional)
- Manejo correcto de status codes

Frontend

- React con Ant Design:
  - Table + Pagination + Filters
  - Forms (Form + validation)
  - Modals o páginas (libre)
- Manejo de estado:
  - local state o Redux Toolkit (opcional)
- UX mínima:
  - Loading states
  - Mensajes de éxito/error

BD (SQL Server)

- Script SQL de creación (DDL)
- Constraints (PK, FK, unique, etc.)
- Campos de auditoría
- (Deseable) índices para filtros frecuentes

Entregables

1. Repositorio con:
   - /backend
   - /frontend
   - /database (scripts)
2. **README** que incluya:
   - Requisitos

- Cómo correr backend
- Cómo correr frontend
- Cómo crear la base de datos
- Variables de entorno

3. (A elección, es requerido elegir uno) Postman collection o Swagger/OpenAPI

Bonus (sube nota)

- Soft delete + filtros por estado
- Seed de datos iniciales (categorías/productos)
- Swagger (si aplica)
- Tests básicos (unitarios o integración)
- Docker compose (API + SQL Server)
- Exportable de postman

Criterios de evaluación (rubrica)

- Correctitud funcional (CRUD completo)
- Paginación y filtros reales en backend/BD
- Calidad de código (nombres, estructura, legibilidad)
- Manejo de errores y validaciones
- Performance mínima (no traer todo)
- UI usable con Ant Design
- Documentación (README claro)