PRUEBA TÉCNICA FULLSTACK TD

Caso Práctico: Sistema de Gestión de Blog - Nestjs

BACKEND (3H):

Contexto:

Estás desarrollando un sistema de gestión de blog simple. Necesitas crear una API REST con **NestJS** para administrar artículos de blog.

Requisitos Funcionales:

1. **CRUD de Artículos:**
    - Crear un nuevo artículo (título, contenido, fecha de publicación, autor).
    - Obtener todos los artículos (opcional).
    - Obtener un artículo por ID.
    - Actualizar un artículo existente.
    - Eliminar un artículo.
2. **Búsqueda de Artículos:**
    - Permitir buscar artículos por título(opcional).

Restricciones:

- No es necesario implementar autenticación ni persistencia en base de datos. Puedes utilizar un array en memoria para almacenar los artículos.
- El enfoque debe estar en la estructura del proyecto NestJS, el diseño de los controladores y servicios, y el manejo de errores.

Tareas:

1. **Crear el Proyecto:** Iniciar un nuevo proyecto NestJS.
2. **Definir DTOs:** Crear DTOs (Data Transfer Objects) para las solicitudes y respuestas de la API (crear artículo, actualizar artículo, etc.).
3. **Crear el Controlador:** Implementar el controlador `ArticulosController` con los endpoints necesarios.
4. **Crear el Servicio:** Implementar el servicio `ArticulosService` con la lógica para manejar los artículos (almacenamiento en memoria, búsqueda, etc.).
5. **Manejo de Errores:** Implementar un manejo de errores básico para devolver respuestas HTTP adecuadas en caso de errores (por ejemplo, artículo no encontrado).
6. **Pruebas (Opcional):** Si queda tiempo, escribir algunas pruebas unitarias básicas para el controlador y el servicio.

FRONTEND (1H):

Requisitos funcionales:

1. La aplicación debe consumir la API creada con **NestJS** y mostrar la lista de elementos en una tabla.
2. Debe haber un formulario para agregar nuevos elementos a la lista.
3. Implementar un botón de "Eliminar" para quitar elementos de la lista.
4. Se tomará en consideración el manejo de estados.

Entrega:

- Se deberá enviar un link de un repositorio github con el back y front en el mismo repositorio para su respectiva revisión.
- Las buenas prácticas serán consideradas.