PROYECTO INTEGRADOR DEL SEGUNDO PARCIAL

Desarrolle una aplicación web dinámica utilizando ReactJS con TypeScript, orientada a optimizar los casos de estudio (consulte los adjuntos para más detalles). La aplicación debe cumplir con los siguientes requisitos:

1. **Menú de navegación**: Incluya un menú interactivo con las siguientes opciones:
    - Inicio
    - Entidad Padre 1
    - Entidad Padre 2
    - Entidad Intermedia
    - Acerca de (contendrá datos personales del desarrollador).
2. **Rutas y navegación**: Implemente el paquete react-router-dom para garantizar una navegación fluida y estructurada entre las diferentes vistas de la aplicación.
3. **Almacenamiento de datos**: Gestione los datos de la aplicación utilizando una API Rest (NodeJs + JavaScript + PostgreSQL).

Las funciones para implementar dependen del caso de estudio asignado, por ejemplo, para el sistema de **E-commerce: Productos y Órdenes se debe incluir:**

1. Menú de navegación

Debe incluir las siguientes opciones principales en un menú interactivo:

- **Inicio**: Página de bienvenida o resumen general del sistema.
- **Productos**: Vista para gestionar la información de los productos.
- **Órdenes**: Vista para gestionar las órdenes realizadas por los clientes.
- **Detalles de Órdenes**: Tabla intermedia que relaciona productos con órdenes y muestra detalles como cantidad y precio unitario.
- **Acerca de**: Página con información del desarrollador (nombre, correo electrónico, descripción breve).

2. Gestión de datos

La aplicación debe permitir realizar operaciones CRUD (Crear, Leer, Actualizar y Eliminar) para las entidades:

- **Productos**:

- Agregar un nuevo producto con los campos id_producto, nombre, precio y stock.
- Editar y eliminar productos existentes.
- Listar todos los productos en un formato tabular interactivo.

- **Órdenes**:
  - Crear nuevas órdenes, especificando el id_orden, fecha y cliente.
  - Visualizar el historial de órdenes realizadas.
- **Detalles de Órdenes** (Tabla intermedia):
  - Permitir asociar productos a una orden, especificando cantidad y precio_unitario.
  - Mostrar el desglose de cada orden con sus productos relacionados y el total calculado.

3. Almacenamiento

- Guardar los datos en la base de datos PostgreSQL.

4. Navegación y rutas

- Utilizar el paquete react-router-dom para gestionar las rutas de la aplicación.
- Cada opción del menú debe redirigir a su respectiva vista:
  - /inicio
  - /productos
  - /ordenes
  - /detalles-orden
  - /acerca-de

5. Validación de datos

- Validar los campos requeridos al agregar o editar productos, órdenes y detalles de órdenes:
  - Ejemplo: El precio y la cantidad deben ser números mayores que cero.
- Mostrar mensajes de error cuando los datos no cumplan con los requisitos.

6. Interfaz de usuario

- Diseñar una interfaz responsiva e intuitiva utilizando **CSS**.
- Utilizar tablas para listar los datos de productos, órdenes y detalles.

- Agregar botones para realizar acciones CRUD directamente desde las tablas

CASOS DE ESTUDIO

1. Sistema de Gestión de Cursos y Estudiantes

- **Descripción:** Un estudiante puede inscribirse en múltiples cursos, y un curso puede tener múltiples estudiantes.
- **Entidades:**
  - **Estudiantes:** id_estudiante, nombre, apellido, correo
  - **Cursos:** id_curso, nombre_curso, descripcion
- **Relación:**
  - Tabla intermedia: Inscripciones con los atributos id_estudiante y id_curso.