Ejercicios React con Backend Integrado - REPASO EXAMEN

Cada ejercicio cubre conceptos fundamentales en React, Context API, Custom Hooks, rutas protegidas, integración con backend, y mejores prácticas de desarrollo intentando abarcar todo lo que hemos realizado en clase a lo largo de este curso 2024/2025 .

---

1. Sistema de Autenticación Completo

📌 **Conceptos**: `useContext`, Custom Hooks, Protected Routes, JWT, localStorage, Validaciones

🎯 **Objetivo**: Implementar un sistema de autenticación seguro que permita a los usuarios registrarse, iniciar sesión y acceder a rutas protegidas.

🔷 **Tareas**:

1. **Crear un contexto global de autenticación (`AuthContext`)** que maneje el estado de autenticación.
2. **Implementar un custom hook (`useAuth`)** con:
   - Estado global de autenticación.
   - Funciones `login` y `register` que interactúen con el backend.
   - Manejo seguro del token JWT (`localStorage` + `Authorization` headers).
   - Función `logout` para eliminar el token y limpiar el estado.
3. **Diseñar formularios de login y registro** con validaciones (email, password, etc.).
4. **Implementar rutas protegidas** con React Router v7 (`<Navigate />` y `useNavigate`).
5. **Mostrar mensajes de error** en autenticación y manejar respuestas del backend.

💡 **Bonus**:

- **"Recordarme"**: Persistencia del usuario autenticado en localStorage.

- **Refresh Token**: Renovación automática del token.
- **Protección de rutas a nivel backend**.

---

2. Gestor de Productos con CRUD Completo

🔗 **Conceptos**: `useEffect`, Custom Hooks, API Fetching, MongoDB, React Query

🎯 **Objetivo**: Desarrollar un CRUD completo de productos, integrando un backend con MongoDB.

🔷 **Tareas**:

1. **Crear un custom hook (`useProducts`)** que maneje:
   - Obtener la lista de productos desde el backend.
   - Crear, actualizar y eliminar productos.
   - Manejo de estado de carga y errores.
2. **Diseñar un grid de productos** con Tailwind CSS.
3. **Crear formularios de agregar/editar productos** con validaciones.
4. **Implementar un sistema de confirmación** antes de eliminar productos.
5. **Proteger las operaciones CRUD** con autenticación mediante token JWT.

💡 **Bonus**:

- **Búsqueda y filtros** (nombre, categoría, precio, stock).
- **Paginación y ordenamiento**.
- **Carga optimizada con React Query o SWR**.

---

3. Dashboard de Estadísticas

🔗 **Conceptos**: `useEffect`, Custom Hooks, Protected Routes, Visualización de Datos

🎯 **Objetivo**: Implementar un dashboard visual que muestre estadísticas clave sobre los productos.

🔷 **Tareas**:

1. **Crear un custom hook (`useDashboardStats`)** para calcular:
   - Total de productos en la base de datos.

    - Valor total del inventario.
    - Productos con bajo stock.
2. **Implementar gráficos** con una librería como Recharts o Chart.js.
3. **Crear un diseño responsivo con Tailwind CSS**.
4. **Proteger la ruta del dashboard** para solo usuarios autenticados.
5. **Actualizar datos en tiempo real** con `useEffect` y WebSockets.

💡 **Bonus**:

- **Exportación de datos** a CSV o PDF.
- **Filtros por fecha o categoría**.
- **Panel interactivo con más visualizaciones**.

---

4. Sistema de Búsqueda Avanzada

🔑 **Conceptos**: Custom Hooks, `useEffect`, Debounce, Caché

🎯 **Objetivo**: Implementar un sistema de búsqueda eficiente para productos.

🔷 **Tareas**:

1. **Crear un custom hook (`useSearch`)** que incluya:
    - **Búsqueda con debounce** para evitar múltiples llamadas al backend.
    - **Historial de búsquedas recientes** almacenado en `localStorage`.
    - **Filtros avanzados** (categoría, precio, stock).
    - **Caché de resultados** para evitar llamadas redundantes.
2. **Diseñar la interfaz de búsqueda** con Tailwind.
3. **Actualizar los resultados en tiempo real**.
4. **Persistir las últimas búsquedas** en `localStorage`.

💡 **Bonus**:

- **Autocompletado** con sugerencias del backend.
- **Búsqueda por voz** usando Web Speech API.
- **Guardado de filtros favoritos**.

---

5. Sistema de Gestión de Inventario

🔗 **Conceptos**: `useContext`, Custom Hooks, Rutas Protegidas, CRUD

🎯 **Objetivo**: Implementar un sistema de gestión de inventario para actualizar stock y registrar movimientos.

🔷 **Tareas**:

1. **Crear un custom hook (`useInventory`)** para:
   - Actualizar el stock de productos.
   - Registrar movimientos (entradas/salidas).
   - Generar alertas para stock bajo.
   - Guardar el historial de cambios.
2. **Implementar un `InventoryContext`** para centralizar la gestión del estado.
3. **Crear formularios de movimientos de stock**.
4. **Proteger las rutas de gestión** con autenticación.

💡 **Bonus**:

- **Escaneo de códigos de barras** con la cámara.
- **Gestión de múltiples almacenes**.
- **Exportación de informes de inventario**.

---

6. Sistema de Notificaciones en Tiempo Real

🔗 **Conceptos**: `useEffect`, Custom Hooks, Context API, WebSockets

🎯 **Objetivo**: Implementar un sistema de notificaciones para cambios en productos.

🔷 **Tareas**:

1. **Crear un custom hook (`useNotifications`)** que:
   - Gestione notificaciones nuevas y leídas.
   - Agrupe por tipo (errores, alertas, stock bajo).
   - Persista el estado de notificaciones.
2. **Implementar un `NotificationContext`** para compartir datos globalmente.
3. **Diseñar la UI de notificaciones** con Tailwind.
4. **Recibir notificaciones en tiempo real** con WebSockets.

💡 **Bonus**:

- **Sonidos de notificación personalizados**.
- **Push notifications en el navegador**.
- **Centro de notificaciones avanzado**.

---

7. Formulario Multi-paso de Productos

🗝️ **Conceptos**: Custom Hooks, Form Management, Context API

🎯 **Objetivo**: Crear un formulario multi-paso para registrar productos.

🔷 **Tareas**:

1. **Crear un custom hook (`useMultiStepForm`)** que maneje:
   - Validación en cada paso.
   - Persistencia de datos parciales.
   - Navegación entre pasos.
2. **Diseñar un UI con pasos progresivos** en Tailwind.
3. **Implementar vista previa final** antes del envío.

💡 **Bonus**:

- **Guardado automático**.
- **Modo borrador**.
- **Wizard personalizable**.

---

8. Panel de Administración

🗝️ **Conceptos**: Rutas Protegidas, Custom Hooks, Gestión de Permisos

🎯 **Objetivo**: Crear un panel de administración seguro y funcional.

🔷 **Tareas**:

1. **Crear un custom hook (`useAdmin`)** que maneje:
   - Gestión de permisos y roles.
   - Auditoría de acciones.
   - Configuración global.
2. **Diseñar un layout administrativo** con navegación.
3. **Proteger rutas administrativas**.

4. **Implementar breadcrumbs y logs de actividad**.

💡 **Bonus**:

- **Temas personalizables**.
- **Gestión avanzada de permisos**.

Endpoits del backend disponibles:

Ruta	Método	Autenticación	Descripción	Datos Esperados	Respuesta
Autenticación					
`/api/auth/register`	POST	No	Registro de nuevo usuario	`{ name, email, password }`	`{ token, user }`
`/api/auth/login`	POST	No	Iniciar sesión	`{ email, password }`	`{ token, user }`
Productos					
`/api/products`	GET	No	Obtener todos los productos	-	`[{ id, name, ... }]`
`/api/products/:id`	GET	No	Obtener un producto	-	`{ id, name, ... }`
`/api/products`	POST	Sí	Crear nuevo producto	`{ name, price, ... }`	`{ id, name, ... }`
`/api/products/:id`	PUT	Sí	Actualizar producto	`{ name, price, ... }`	`{ id, name, ... }`
`/api/products/:id`	DELETE	Sí	Eliminar producto	-	`{ message: "..." }`

Notas:

- Para rutas autenticadas, enviar token en header: `Authorization: Bearer <token>`
- Los errores devuelven: `{ message: "Mensaje de error" }`
- Base URL: `http://localhost:3000`