Taller Práctico - Aplicación CRUD con React y React Router

Temática: Construcción de una SPA (Single Page Application) con operaciones CRUD (Create, Read, Update, Delete) y navegación.

Objetivo del Taller

Aplicar los conceptos de React, incluyendo el manejo de estado con hooks (useState), manejo de eventos, y navegación con React Router para construir una aplicación de gestión de usuarios completamente funcional.

Descripción del Reto

Desarrollar una aplicación web para gestionar un listado de usuarios. A diferencia de talleres anteriores, los datos no vendrán de una API externa, sino que se gestionarán localmente dentro del estado de la aplicación (CRUD con datos estáticos).

La aplicación debe permitir:

- **Leer (Read):** Mostrar una lista de usuarios existentes en la página principal.
- **Crear (Create):** Tener una página/vista separada con un formulario para agregar un nuevo usuario a la lista.
- **Eliminar (Delete):** Permitir eliminar un usuario directamente desde la lista.
- **Actualizar (Update):** Tener una página/vista de edición para modificar la información de un usuario existente.

Meta: Dominar el flujo de datos y eventos en React, y estructurar una aplicación multi-vista usando React Router.

Sugerencia de Interfaz (Mockup)

Vista Principal (Lista /)

Code snippet

```
+-----------------------------------------------------+
| Gestión de Usuarios          [ Crear Nuevo Usuario + ] |
+-----------------------------------------------------+
|                                                     |
|  • Ana García (ana@email.com)   [Editar] [Eliminar]    |
|  • Carlos López (carlos@email.com)[Editar] [Eliminar]    |
|  • María Silva (maria@email.com) [Editar] [Eliminar]    |
|                                                     |
+-----------------------------------------------------+
```

Vista de Formulario (/crear o /editar/:id)

Code snippet

```
+-----------------------------------------------------+
| Crear/Editar Usuario                    [ < Volver ] |
+-----------------------------------------------------+
|                                                     |
|  Nombre: [ Ana García          ]                    |
|                                                     |
|  Correo: [ ana@email.com       ]                    |
|                                                     |
|                          [ Guardar Cambios ]   |
|                                                     |
+-----------------------------------------------------+
```

Requisitos Mínimos

- **Configuración:** Proyecto creado con Vite (recomendado) o create-react-app.
- **Navegación:** Implementar react-router-dom para la navegación.
  - Se debe instalar (npm install react-router-dom).
  - Configurar <BrowserRouter>, <Routes>, y <Route>.
  - Definir al menos 3 rutas: / (Lista), /crear (Formulario de creación), y /editar/:id (Formulario de edición).
  - Usar <Link> o useNavigate para la navegación entre vistas.
- **Estado (State):**
  - Usar useState para almacenar el array de usuarios.
  - El estado debe inicializarse con 2 o 3 usuarios de ejemplo (datos estáticos).
- **Componentización:**
  - App.jsx: Componente principal que define las rutas y maneja el estado principal de los usuarios.
  - ListaUsuarios.jsx: (Vista READ) Muestra la lista, renderizada con .map().
  - FormularioUsuario.jsx: (Vistas CREATE/UPDATE) Un formulario *reutilizable* para crear y editar usuarios.
- **Manejo de Eventos:**
  - **onChange**: Para manejar los *inputs controlados* del formulario (nombre, correo).
  - **onSubmit**: Para manejar el envío del formulario (tanto para crear como para actualizar).
  - **onClick**: Para los botones de "Eliminar" y los enlaces de "Editar".

Universidad de los Andes
Colombia

Configuración Inicial del Proyecto (con Vite)

1. **Crear el Proyecto:** Abre tu terminal y ejecuta:
2. Bash
- npm create vite@latest mi-app-crud -- --template react
3.
4.
5. **Ingresar al Directorio:**
6. Bash
- cd mi-app-crud
7.
8.
9. **Instalar Dependencias:**
10. Bash
- npm install
11.
12.
13. **Instalar React Router:**
14. Bash
- npm install react-router-dom
15.
16.
17. **Limpiar Proyecto:**
    - Borra el contenido de App.css e index.css.
    - Reemplaza el contenido de App.jsx por un simple <h1>Hola Mundo</h1>.
18. **Ejecutar el Servidor:**
19. Bash
- npm run dev
20.
21.

Puntos a Desarrollar

Punto 1 | Configuración de Rutas y Estado

1. En App.jsx, importa useState de 'react' y BrowserRouter, Routes, y Route de react-router-dom.
2. Define tu estado principal (dentro de la función App):
3. JavaScript
- const [usuarios, setUsuarios] = useState([
- { id: 1, nombre: 'Ana García', correo: 'ana@email.com' },
- { id: 2, nombre: 'Carlos López', correo: 'carlos@email.com' }
-]);
4.

5.
6. Dentro del `return` de `App.jsx`, envuelve toda la lógica de tus vistas dentro del componente `<BrowserRouter>`.
7. Dentro de `<BrowserRouter>`, define tus `<Routes>` con las 3 rutas (`/`, `/crear`, `/editar/:id`).
8. Pasa `usuarios` y `setUsuarios` como props a los componentes de las rutas que lo necesiten.

Punto 2 | Operación READ y Evento (Delete)

1. Crea el componente `ListaUsuarios.jsx`. Recibirá `usuarios` y `setUsuarios` como props.
2. Renderiza la lista usando `.map()`. Cada item debe mostrar nombre, correo y tener su `key` única.
3. Agrega un `<Link>` de React Router para "Crear Nuevo Usuario" que apunte a `/crear`.
4. Agrega un `<Link>` "Editar" para cada usuario que apunte a `/editar/usuario.id`.
5. **Botón Eliminar (Evento `onClick`):**
    - Agrega un botón "Eliminar" a cada usuario.
    - Al hacer clic, implementa la función `eliminarUsuario(id)`.
    - Esta función debe pedir confirmación (`window.confirm('¿Estás seguro?')`).
    - Si se confirma, debe actualizar el estado: `setUsuarios(usuarios.filter(u => u.id !== id))`.

Punto 3 | Operación CREATE y Eventos (Formulario)

1. Crea el componente `FormularioUsuario.jsx`.
2. En la ruta `/crear` (dentro de `App.jsx`), renderiza este formulario.
3. Usa `useState` *dentro* de `FormularioUsuario.jsx` para manejar los inputs controlados (nombre y correo).
4. Implementa la lógica `agregarUsuario` en `App.jsx` y pásala como prop al formulario.
5. **Evento `onSubmit`:**
    - El formulario debe capturar el evento `onSubmit` (y prevenir el `event.preventDefault()`).
    - Debe crear un nuevo objeto de usuario, generando un ID único (p.ej., `id: Date.now()`).
    - Llamará a `setUsuarios([...usuarios, nuevoUsuario])`.
    - Usa el hook `useNavigate` para redirigir al usuario de vuelta a la lista (`/`).

Punto 4 | Operación UPDATE y Hooks de Ruta

1. Reutiliza `FormularioUsuario.jsx` para la ruta `/editar/:id`.
2. **`useParams`**: En el componente que renderiza el formulario (puedes crear un *wrapper* o hacerlo en `App.jsx`), usa el hook `useParams` para obtener el `:id` de la URL.
3. **Cargar Datos**: Usa el `id` para encontrar el usuario a editar (`usuarios.find(u => u.id === Number(id))`).

4. **Pre-llenar Formulario**: Pasa el `usuarioEncontrado` como prop a `FormularioUsuario.jsx`. Usa `useEffect` dentro del formulario para pre-llenar los campos (`setNombre(usuario.nombre)`, `setCorreo(usuario.correo)`) cuando la prop del usuario cambie.
5. **Evento `onSubmit` (Modo Edición):**
    - La lógica `onSubmit` ahora debe *actualizar* en lugar de crear.
    - Implementa `actualizarUsuario` en `App.jsx`.
    - Usará `setUsuarios(usuarios.map(u => (u.id === id ? usuarioActualizado : u)))`.
    - Navega de vuelta a `/` al terminar.

Buenas Prácticas

- **IDs Únicos:** Asegúrate de que cada usuario nuevo tenga un ID único. `Date.now()` es suficiente para este ejercicio.
- **Inputs Controlados:** Todos los campos del formulario deben usar `value` y `onChange`.
- **Validación:** (Opcional) Antes de enviar el formulario, verifica que los campos no estén vacíos.
- **Confirmación:** Siempre pide confirmación al eliminar.

Preguntas de Reflexión

- ¿Qué ventajas tiene practicar CRUD con datos estáticos antes de conectarlo a un backend real?
- ¿Por qué es necesario usar el *spread operator* (`...usuarios`) o `.map()`/`.filter()` para actualizar el estado, en lugar de modificar el array original?
- ¿Qué diferencia hay entre `useNavigate` y el componente `<Link>` de React Router?
- ¿Cómo cambia la lógica del componente `FormularioUsuario.jsx` para soportar tanto "Crear" como "Actualizar"?
-