React Nivel 10: Proyecto Final y Optimización



En este nivel final, consolidarás los conocimientos adquiridos para construir una aplicación completa y optimizada con React. Veremos cómo manejar estados globales, optimizar el rendimiento, implementar navegación, y utilizar funcionalidades avanzadas como lazy loading. Este nivel es el punto culminante de tu aprendizaje, preparándote para desarrollar aplicaciones robustas y escalables. 🛱

Temas que Cubriremos 譽

- 1. Introducción a la optimización en React.
- 2. Creación de un CRUD básico con localStorage.
- 3. Optimización con React.memo, useCallback y useMemo.
- 4. Implementación de lazy loading con React.lazy y Suspense.
- 5. Desarrollo de una aplicación completa: **Buscador de Películas**.

1. Introducción a la Optimización en React

React ofrece herramientas como React.memo, useCallback y useMemo para evitar renders innecesarios y mejorar el rendimiento de la aplicación. Estas optimizaciones son especialmente importantes en proyectos grandes con múltiples componentes que interactúan entre sí.

¿Por qué optimizar?

- Mejor experiencia de usuario: Minimizar el tiempo de respuesta y carga.
- Eficiencia: Reducir el trabajo del navegador.
- Escalabilidad: Permitir que la aplicación maneje más datos y funcionalidades sin problemas.

2. Creación de un CRUD Básico con localstorage



Un CRUD (Crear, Leer, Actualizar y Eliminar) es fundamental para manejar datos en aplicaciones. Utilizaremos localStorage para persistir los datos en el navegador.

Pasos Clave:

- 1. **Estado inicial:** Usa useState para manejar la lista de datos.
- 2. Persistencia: Almacena los datos en localStorage para que permanezcan después de recargar la página.
- 3. Funciones CRUD: Implementa funciones para crear, leer, editar y eliminar datos.

3. Optimización con React.memo, useCallback y useMemo 🚳 🗆

React.memo:

Evita renders innecesarios de componentes al memorizar su salida mientras las propiedades no cambien.

```
import React from 'react';

const MiComponente = React.memo(({ dato }) => {
  console.log("Renderizando...");
  return {dato};
});
```

useCallback:

Memoriza funciones para evitar que se vuelvan a crear en cada render.

```
const handleClick = useCallback(() => {
  console.log("Botón clickeado");
}, []);
```

useMemo:

Memoriza valores derivados para evitar cálculos costosos innecesarios.

```
const resultadoMemoizado = useMemo(() => {
  return datos.filter((item) => item.activo);
}, [datos]);
```

4. Implementación de Lazy Loading con React.lazy y Suspense

Lazy loading carga componentes de forma diferida, mejorando el tiempo de carga inicial de la aplicación.

Ejemplo de Lazy Loading:

Ventajas:

- Reduce el tiempo de carga inicial.
- Mejora el rendimiento en aplicaciones grandes.

5. Desarrollo de una Aplicación Completa: Buscador de Películas 22



Objetivo: Crear una aplicación que permita buscar películas utilizando la API de OMDb.

Pasos Clave:

1. Configuración inicial:

- o Crea un formulario para buscar películas.
- Consume la API de OMDb (https://www.omdbapi.com/) con fetch.

2. Manejo de estado:

- Usa useState para capturar el término de búsqueda y almacenar resultados.
- Maneja estados de carga y error.

3. Optimización:

- Usa React.memo para componentes individuales de la lista.
- Aplica useCallback para funciones de búsqueda.
- o Implementa lazy loading para cargar vistas diferidas.

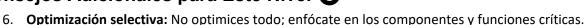
4. Navegación:

o Usa React Router para mostrar los detalles de una película en una nueva página.

5. Estado global:

Maneja la búsqueda y los resultados con Context API.

Consejos Adicionales para Este Nivel @



- 7. Divide y organiza: Separa tu aplicación en módulos o componentes reutilizables.
- 8. **Pruebas:** Testea cada funcionalidad para asegurar que todo funcione correctamente.
- 9. **Documenta:** Agrega comentarios en el código para explicar decisiones clave.

¡Felicidades por completar el **Nivel 10** y el curso completo! 🕭 Ahora tienes las habilidades necesarias para desarrollar aplicaciones modernas con React, optimizar su rendimiento y trabajar con datos dinámicos. 🔊