ChatGPT

Curso Práctico para Entender y Modificar Aplicaciones React Empresariales (SMRI)

Objetivo del curso

Aprender a leer, entender, modificar y escalar aplicaciones empresariales modernas construidas con React, React Router, Zustand, Context API y conexión a APIs reales.

Módulo 1: Fundamentos de React

¿Qué es React y cómo funciona?

React es una biblioteca de JavaScript para construir interfaces de usuario. Fue desarrollada por Facebook y se basa en una arquitectura de componentes reutilizables. React utiliza un concepto llamado "DOM virtual" que actualiza de forma eficiente solo los elementos que cambian en la interfaz.

Paso a paso para entenderlo: 1. React divide la interfaz en "componentes" reutilizables. 2. Cada componente puede tener su propio estado y propiedades. 3. React compara el DOM virtual con el DOM real para hacer actualizaciones eficientes.

JSX y componentes

JSX es una sintaxis similar a HTML que usamos en React para definir lo que se va a renderizar. Es una mezcla de JavaScript + XML.

Ejemplo de componente JSX:

```
function HelloWorld() {
  return <h1>Hola, mundo!</h1>;
}
```

Paso a paso: 1. Se crea una función que devuelve JSX. 2. Esa función se exporta para usarla en otros archivos. 3. Se importa y se usa como una etiqueta HTML.

Props y estado (`useState`)

Props son datos que se pasan desde un componente padre a un hijo. **useState** es un hook que permite manejar variables que cambian con el tiempo.

Ejemplo:

```
function Saludo({ nombre }) {
  return <h2>Hola, {nombre}</h2>;
}

function App() {
  const [nombre, setNombre] = useState("Juan");
  return <Saludo nombre={nombre} />;
}
```

Paso a paso: 1. `useState` inicializa una variable reactiva. 2. Se pasa esa variable como prop. 3. Al cambiarla con `setNombre`, React vuelve a renderizar el componente.

Eventos y renderizado condicional

React permite manejar eventos como clics, cambios de input, etc. También se puede mostrar contenido basado en condiciones.

Ejemplo:

```
function Boton() {
  const [activo, setActivo] = useState(true);

  return (
    <div>
      <button onClick={() => setActivo(!activo)}>
        {activo ? "Desactivar" : "Activar"}
      </button>
      {activo && <p>El sistema está activo.</p>}
    </div>
  );
}
```

Paso a paso: 1. Se define un estado con `useState`. 2. Se usa un evento `onClick` para cambiar ese estado. 3. Se usa `&&` o `? :` para mostrar u ocultar contenido.

Hook `useEffect` y su ciclo de vida

`useEffect` permite ejecutar código cuando el componente se monta, actualiza o desmonta (simula el ciclo de vida).

Ejemplo:

```
useEffect(() => {
  console.log("El componente se ha montado o actualizado");
  return () => {
    console.log("El componente se desmontó");
  };
}, [dependencia]);
```

Paso a paso: 1. Si el array de dependencias está vacío `[]`, solo se ejecuta al montar (como `componentDidMount`). 2. Si contiene variables, se ejecuta cada vez que esas variables cambian. 3. La función de retorno simula el desmontaje del componente (`componentWillUnmount`).

Ejercicios: - Componente `HelloWorld` - Contador básico - Input controlado - Formulario simple con validación básica

---

Módulo 2: React Router v6 Avanzado

- `createHashRouter`, `Route`, `Navigate`
- Anidamiento de rutas y rutas protegidas
- Uso de `<Suspense />` y `lazy()` para carga diferida

Ejercicio: - Crear rutas protegidas que redirigen si el usuario no está autenticado

---

Módulo 3: Context API y Estado Global

- Crear y consumir contextos (UserContext, SidebarContext)
- Proveedores y consumidores con `useContext`
- Desacoplar lógica de estado del árbol de componentes

Ejercicio: - Crear un contexto para `user` con nombre, edad e incrementador - Mostrar los datos desde varios componentes sin props

---

Módulo 4: Zustand para manejo avanzado del estado

- Crear un `store` global con Zustand
- `persist` para guardar sesión en `localStorage`
- Uso de `devtools`

Ejercicio: - Crear un `authStore` con funciones `loginUser` y `logoutUser` - Leer token y usuario desde componentes

---

Módulo 5: Autenticación real con API + Zustand + Rutas Protegidas

- Formulario de login real conectado a API (`login.services.js`)
- Adaptadores de respuesta con `loginReqAdapter`
- Redirección automática tras login
- Validación de sesión en `ProtectedRoutes`

Ejercicio: - Crear flujo de login completo: login → dashboard → logout

---

Módulo 6: Notificaciones con Sonner

- Usar la librería `sonner` para mostrar toasts
- Posicionar, cerrar y personalizar mensajes

Ejercicio: - Mostrar una notificación tras login exitoso

Módulo 7: Estructura de carpetas escalable

- Separación por dominios (`pages/`, `sections/`, `components/`, `context/`, `routers/`, `services/`)
- Uso de `lazy()` para modularidad

Ejercicio: - Reorganizar una nueva funcionalidad usando una carpeta `modules/`

Módulo Final: Recomendaciones para escalar o modernizar la app

Recomendaciones clave:

- Migrar a **TypeScript**: mejor tipado y validación
- Añadir **testing** con Vitest o React Testing Library
- Implementar **React Query** para manejo de APIs
- Extraer lógica en **hooks personalizados**
- CI/CD con GitHub Actions y despliegue en Vercel
- Accesibilidad y rendimiento con Lighthouse
- Uso de `env` para variables de entorno en producción

🎓 Conclusión

Este curso te brinda las bases necesarias para comprender, extender y escalar aplicaciones empresariales modernas con React. Está construido sobre ejemplos reales y estructurado para aplicar lo aprendido directamente en tu proyecto SMRI u otros proyectos similares.

Autor: Generado automáticamente a partir del análisis de tu código y estructura real de aplicación. Si deseas convertir esto en una plantilla para equipos, se puede personalizar aún más.