**Taller**

Jesús David Espitia Correa

Centro de Electricidad, Electrónica y Telecomunicaciones, SENA

2558108: Análisis y Desarrollo de Software

Isaura Maria Novoa

17 de noviembre 2023

**1.Flat list**

FlatList es un componente central de React Native para la representación eficiente de listas de temas. Está diseñado para manejar grandes conjuntos de datos y es ideal para listas muy largas o dinámicas. Básicamente, FlatList es un componente de interfaz de usuario que permite mostrar una lista de elementos en una aplicación móvil creada con React Native. Lo que hace especial a FlatList es su eficiencia: en lugar de renderizar todos los elementos de una lista a la vez, renderiza sólo los elementos que son visibles en la pantalla y reutiliza los elementos que no son visibles, optimizando el rendimiento y la memoria. Algunas de las características clave de FlatList incluyen:

**Rendimiento optimizado:** presente los elementos de la lista de manera eficiente, especialmente en listas largas, mostrando solo los elementos visibles en la pantalla. Admite listas dinámicas: puede manejar listas de diferentes longitudes y agregar o eliminar elementos dinámicamente.

**Personalización:** le permite personalizar la apariencia y el comportamiento de los elementos de la lista utilizando atributos como datos, renderItem, keyExtractor, etc.

**Manejo de eventos:** proporciona funciones como onEndReached para detectar cuándo el usuario llega al final de la lista y cargar más elementos si es necesario.

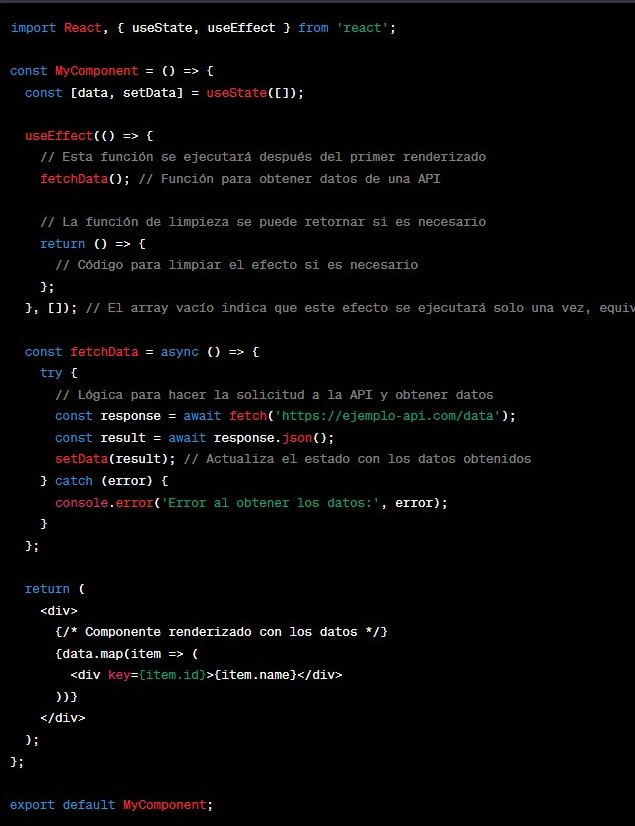
En resumen, FlatList es una herramienta esencial para mostrar listas de elementos de manera efectiva en aplicaciones móviles que utilizan React Native, ya que optimiza el rendimiento al representar solo elementos visibles, lo que resulta en una experiencia de usuario más fluida y rápida, especialmente para listas grandes.

**2.Use Effect**

**¿Qué hace useEffect?** Al usar este Hook, le estamos indicando a React que el componente tiene que hacer algo después de renderizarse. React recordará la función que le hemos pasado (nos referiremos a ella como nuestro “efecto”), y la llamará más tarde después de actualizar el DOM. En este efecto, actualizamos el título del documento, pero también podríamos hacer peticiones de datos o invocar alguna API imperativa.

**¿Por qué se llama a useEffect dentro del componente?** Poner useEffect dentro del componente nos permite acceder a la variable de estado count (o a cualquier prop) directamente desde el efecto. No necesitamos una API especial para acceder a ella, ya que se encuentra en el ámbito de la función. Los Hooks aprovechan los closures de JavaScript y evitan introducir APIs específicas de React donde JavaScript ya proporciona una solución.

**Ejemplo**

****