INVESTIGACIÓN SEGUNDA PARTE DE CITAS DE VETERINARIA

CIELO NICOLE MUNAR FINO

SERVICIO NACIONAL DE APRENDIZAJE (SENA)

ISAURA SUAREZ

2023

Investiga y explica el componente nativo FlatList que acabaste de utilizar.

FlatList es un componente específico de React Native diseñado para renderizar listas de datos de manera eficiente en aplicaciones móviles. Este componente es particularmente útil cuando se trabaja con conjuntos de datos grandes, ya que implementa técnicas de optimización para mejorar el rendimiento al desplazarse por la lista. FlatList es parte del conjunto de componentes básicos de React Native y proporciona una interfaz sencilla para trabajar con listas de datos dinámicas.

Este componente, tiene algunas propiedades clave, como las siguientes:

* data: Recibe un array de datos que se utilizará para renderizar la lista.
* renderItem: Una función que toma un elemento de datos y devuelve un componente que representa ese elemento. Esto permite personalizar la apariencia de cada elemento de la lista.
* keyExtractor: Una función que extrae una clave única para cada elemento de datos. Esta clave se utiliza internamente para optimizar la reutilización de componentes y mejorar el rendimiento.
* horizontal: Una propiedad booleana que determina si la lista se debe renderizar en orientación horizontal en lugar de vertical.
* numColumns: Especifica el número de columnas para la visualización en modo de cuadrícula. Puede usarse para crear una lista de cuadrícula en lugar de una lista lineal.
* onEndReached y onEndReachedThreshold: Propiedades que permiten manejar eventos cuando el usuario se acerca al final de la lista. Esto es útil para cargar más datos dinámicamente a medida que el usuario se desplaza.

Además de estas propiedades clave, tienen características , como:

* Eficiencia de rendimiento:

FlatList utiliza un enfoque de "renderizado bajo demanda" y reciclaje de elementos para optimizar el rendimiento al renderizar listas. Esto significa que solo los elementos visibles en la pantalla se renderizan, mejorando significativamente la eficiencia al trabajar con grandes conjuntos de datos.

* Reutilización de componentes:

Implementa el reciclaje de elementos, lo que significa que los componentes que salen de la vista se pueden reutilizar para representar nuevos elementos que entran en la vista. Esto reduce la carga en la memoria y mejora el rendimiento.

* Propiedad data:

Recibe un array de datos que se utilizarán para renderizar la lista.

* Propiedad renderItem:

Especifica una función que toma un elemento de datos y devuelve un componente que representa ese elemento. Esto proporciona flexibilidad para personalizar la apariencia de cada elemento de la lista.

* Propiedad keyExtractor:

Permite definir una función para extraer una clave única para cada elemento de datos. Esta clave se utiliza internamente para optimizar la reutilización de componentes y mejorar el rendimiento.

* Orientación horizontal y visualización en columnas:

FlatList admite tanto la orientación horizontal como la visualización en columnas, proporcionando flexibilidad en el diseño de la lista.

* Manejo de eventos de desplazamiento:

Proporciona eventos como onEndReached y onEndReachedThreshold para manejar acciones cuando el usuario se acerca al final de la lista. Esto es útil para cargar más datos dinámicamente.

* Manejo de actualizaciones de datos:

FlatList maneja automáticamente las actualizaciones en los datos y realiza un renderizado eficiente. Si la propiedad data se actualiza, FlatList compara los nuevos datos con los antiguos y actualiza solo los elementos que han cambiado.

Investiga que es useEffect. Explica a través de un ejemplo (con la aplicación móvil que se está desarrollando

Es un Hook de react que te permite [sincronizar un componente con un sistema externo.](https://es.react.dev/learn/synchronizing-with-effects)Este gancho se utiliza para realizar efectos secundarios en componentes funcionales. Los efectos secundarios pueden incluir cosas como llamadas a API, suscripciones a eventos, manipulación del DOM y otras operaciones que no deben realizarse directamente en el cuerpo del componente.

La función useEffect toma dos argumentos: una función y una matriz de dependencias opcional.

La función se ejecutará después de que el componente sea renderizado y después de cada actualización. La matriz de dependencias es opcional y se utiliza para indicar qué variables deben ser monitoreadas para que la función de efecto se vuelva a ejecutar. Si la matriz está vacía, el efecto se ejecutará solo después del montaje y antes de la desmontaje del componente.

**Configuración (función de efecto)**

La función que contiene la lógica de tu efecto. Esta función se ejecuta después del renderizado del componente y después de cada actualización del componente si se proporcionan dependencias.

Opcionalmente, esta función puede devolver otra función que actúa como una función de limpieza. Esta función de limpieza se ejecuta antes de que el componente se desmonte y antes de que se ejecute la próxima vez la función de configuración.

**Dependencias Opcionales**

Una matriz de dependencias que indica qué valores reactivos (props, estados, variables o funciones declaradas directamente dentro del cuerpo del componente) deben ser monitoreados.

Si se proporciona, React comparará cada valor de dependencia con su valor anterior utilizando el algoritmo de comparación Object.is.

Si no se especifican dependencias, el efecto se ejecutará después de cada renderizado del componente.