Trabajo Práctico: Investigación y Aplicación de Core Components en React Native

Objetivo:

Investigar y comprender el uso de los core components de React Native, explicando su

funcionalidad y mostrando ejemplos prácticos de uso.

Consigna:

Deberán seleccionar 7 componentes nativos (core components) de React Native (por

ejemplo: View, Text, Image, ScrollView, TextInput, Button, FlatList, entre otros).

Para cada componente elegido, deberán desarrollar:

1. Nombre del componente

2. ¿Para qué sirve?: Breve explicación de su función principal y casos de uso

comunes.

3. Ejemplo práctico: Fragmento de código que lo utilice correctamente dentro de una

pequeña interfaz o funcionalidad.

4. Explicación del ejemplo: ¿Qué hace el código? ¿Por qué y cuándo usarías ese

componente?

Requisitos:

● Los ejemplos deben estar escritos en JavaScript con JSX.

● Pueden usar la documentación oficial de React Native como fuente principal.

Formato de entrega:

● Documento de Google

● Se valorará la claridad en las explicaciones, la corrección del código y el esfuerzo en

los ejemplos.

## **1. View**

**¿Para qué sirve?** El componente View es uno de los más fundamentales de React Native. Sirve como contenedor visual que agrupa otros elementos. Se usa para estructurar la interfaz, distribuir contenido y aplicar estilos de maquetado como márgenes, paddings, alineaciones, fondos, etc.

**Ejemplo práctico:** (ver archivo ViewEjemplo.js)

**Explicación del ejemplo:** En el ejemplo se usa View para envolver un Text y aplicarle un estilo con fondo gris, márgenes y padding. Usaríamos View cada vez que queramos ordenar y agrupar elementos en pantalla o aplicarles estilos de contenedor.

## **2. Text**

**¿Para qué sirve?** Text se utiliza para mostrar texto en la pantalla. Es ideal para títulos, párrafos, etiquetas, mensajes o cualquier contenido textual que el usuario necesite leer.

**Ejemplo práctico:** (ver archivo TextEjemplo.js)

**Explicación del ejemplo:**Este ejemplo muestra un texto con tamaño de fuente grande y color azul. Se usa Text porque es el componente estándar para mostrar texto, y permite aplicar estilos como tipografía, color, tamaño y alineación.

## **3. Image**

**¿Para qué sirve?** Image se utiliza para mostrar imágenes desde una URL o desde los archivos locales del proyecto. Es útil para mostrar íconos, fotos de perfil, ilustraciones, etc.

**Ejemplo práctico:** (ver archivo ImageEjemplo.js)

**Explicación del ejemplo:** El ejemplo carga una imagen desde internet con un tamaño definido. Este componente es clave si queremos agregar elementos visuales como logos, fotos o gráficos a la app.

## **4. ScrollView**

**¿Para qué sirve?** ScrollView permite que una vista sea desplazable si su contenido excede el tamaño de la pantalla. Es útil para listas cortas, formularios o interfaces que ocupan mucho espacio vertical.

**Ejemplo práctico:** (ver archivo ScrollViewEjemplo.js)

**Explicación del ejemplo:** Se genera una lista de 20 textos que se pueden desplazar hacia abajo. Usamos ScrollView porque el contenido no cabe todo junto y queremos permitir al usuario hacer scroll para verlo.

## **5. TextInput**

**¿Para qué sirve?** TextInput permite que el usuario ingrese texto. Es esencial para formularios, búsquedas, inicios de sesión y cualquier interacción donde el usuario deba escribir.

**Ejemplo práctico:** (ver archivo TextInputEjemplo.js)

**Explicación del ejemplo:** Se muestra un campo de texto con borde gris donde el usuario puede escribir su nombre. Este componente se usa siempre que queramos recibir información escrita por parte del usuario.

## **6. Button**

**¿Para qué sirve?** Button permite agregar un botón interactivo que responde al toque del usuario. Es clave para confirmar acciones, enviar formularios o navegar por la app.

**Ejemplo práctico:** (ver archivo ButtonEjemplo.js)

**Explicación del ejemplo:** El botón muestra un texto y, al presionarlo, lanza una alerta. Usamos Button para permitir la interacción y ejecutar una función cuando el usuario hace clic.

## **7. FlatList**

**¿Para qué sirve?** FlatList sirve para mostrar listas largas de elementos de manera optimizada. Carga solo lo visible en pantalla, mejorando el rendimiento con grandes volúmenes de datos.

**Ejemplo práctico:** (ver archivo FlatListEjemplo.js)

**Explicación del ejemplo:** La lista muestra frutas usando FlatList, que recibe un arreglo de datos y una función para renderizar cada ítem. Es preferible a ScrollView para listas largas por su eficiencia.