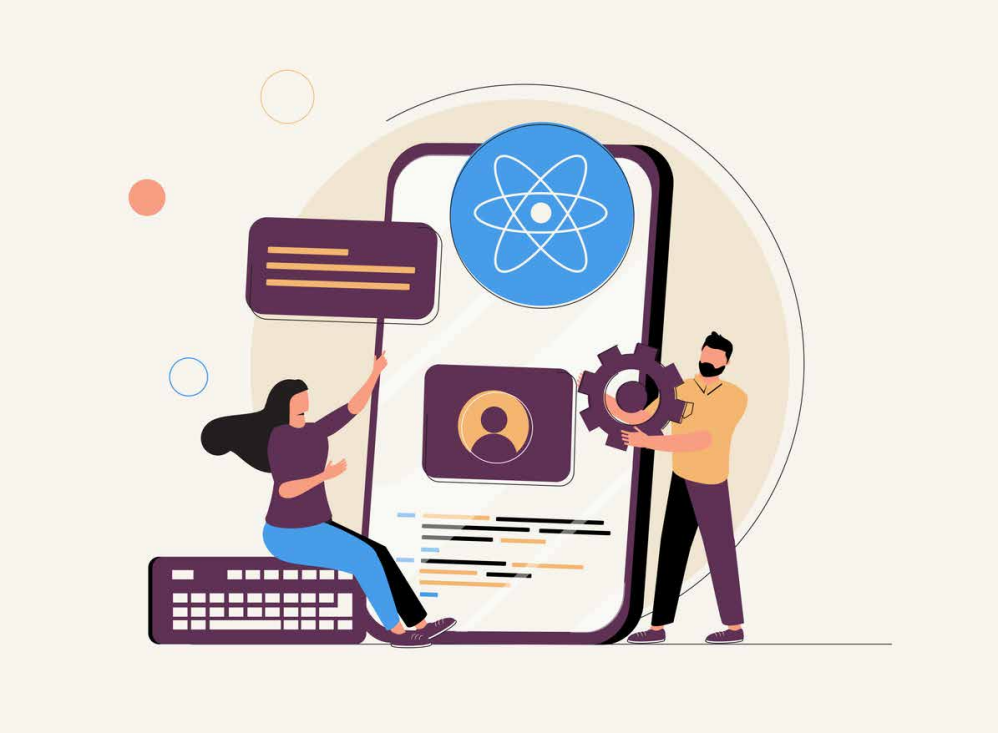
## React Nivel Avanzado

# Componentes Core y Nativos

# Introducción



Hay diferentes opciones para desarrollar aplicaciones que, regularmente, pretenden colaborar en el desarrollo tecnológico de las personas, para que manejen de forma correcta programas específicos. En este tema, se abordarán los componentes nativos y componentes Core útiles para diseñar Apps.

React Native facilita el desarrollo de la interfaz de usuario, al basarse en el uso de componentes, esto permite que, tras crear un proyecto, sea posible iniciar de forma inmediata con el diseño de la interfaz visual de la aplicación, tanto para iOS como para Android.

# Explicación

### Componentes Nativos y Componentes Core

De acuerdo con React Native (s.f.), los componentes nativos son aquellos que son diseñados para un lenguaje en específico, por ejemplo: una vista que se desarrolló exclusivamente para Android usando código como Kotlin o Java, o una vista que se diseñó para iOS mediante código Swift, estos componentes desarrollados, específicamente para determinadas plataformas, pueden ser invocados mediante JavaScript en React Native.

Los componentes Core vienen integrados de forma predeterminada en React Native, estos permiten hacer uso de los componentes nativos de cada plataforma, mediante el uso de la misma sintaxis para ambos.

En la siguiente tabla se muestran los componentes Core que son posibles de disponer en React Native y su similar en iOS, Android y web.

Tabla 1. Componentes Core de React Native

| Componente React Native | Componente React Native | iOS | Web | Descripción |
|---|---|---|---|---|
| <View> | <ViewGroup> | <UIView> |  (sin scroll) | Un contenedor con soporte para Flexbox, eventos táctiles y controles de accesibilidad. |
| <Text> | <TextView> | <UITextView> | <p> | Muestra textos y maneja eventos táctiles. |
| <Image> | <ImageView> | <UIImageView> | <img> | Muestra una imagen de cualquier tipo. |
| <ScrollView> | <ScrollView> | <UIScrollView> |  | Contenedor genérico que permite desplazamiento, puede contener múltiples componentes y vistas. |
| <TextInput> | <EditText> | <UITextField> | <input type=”text”> | Permite introducir texto. |

# Explicación

A continuación, se muestra el típico ejemplo de “Hola Mundo”:

```
import React from 'react';
import { Text, View } from 'react-native';

const HelloWorldApp = () => {
  return (
    <View
      style={{
        flex: 1,
        justifyContent: "center",
        alignItems: "center"
}}>
      <Text>Hello, world!</Text>
} ) </View>
```

El código anterior da como resultado lo siguiente:


Esta pantalla se obtuvo directamente del software que se está explicando en la computadora, para fines educativos.

# Explicación

En la última línea de código, se puede observar HelloWorldApp, lo cual está creando un nuevo componente. Cada que se compile una aplicación de React Native se crea uno nuevo, así como todo lo que se observe en la pantalla del dispositivo al momento que se ejecuta la app.

**Asignando propiedades a los componentes**

De acuerdo con Boduch, Derks, y Sakhniuk (2022), la mayoría de los componentes pueden ser personalizados cuando se crean con diferentes parámetros. A estos se les llama ***Props (propiedades)*** y con ellos se puede reusar un componente en diferentes lugares de la aplicación.

Se sugiere retomar el ejemplo de Hello Word para intentar que la aplicación salude a tres personas, se deben utilizar los Props para manejar el mismo componente, pero se le cambiará el parámetro del nombre.

En este caso se tendrá un componente **Greeting**, el cual mostrará el texto del saludo y recibirá un parámetro (propiedad) llamada **name**, cuyo valor se imprimirá dentro del componente **<Text>**, quedando de la siguiente forma.

```
const Greeting = (props) => {
  return (
    <View style={styles.center}>
      <Text>Hello {props.name}!</Text>
    }
);</View>
```

Ahora sólo se necesita llamar al componente **Greeting** en tres ocasiones y asignarle un valor diferente a la propiedad **name**, tal como se muestra a continuación.

```
const LotsOfGreetings = () => {
  return (
    <View style={[styles.center, {top: 50}]}>
      <Greeting name='Rexxar' />
      <Greeting name='Jaina' />
      <Greeting name='Valeera' />
} );</View>

export default LotsOfGreetings;
```

# Cierre

Ahora que se pueden identificar los componentes Core, será más sencillo desarrollar las interfaces necesarias, también se podrá identificar cómo y cuándo agregar propiedades a los componentes utilizados para reutilizarlos si es posible.

Los componentes Core permiten a los desarrolladores crear software de alta calidad y funcionalidad, como se pudo ver, proporcionan bases sólidas y confiables para poder construir. Los componentes Core también pueden reutilizarse en diferentes proyectos, de esta manera, se ahorra tiempo y reduce costos en el desarrollo de una App y facilita el mantenimiento de la aplicación a largo plazo.

# Referencias bibliográficas


- Boduch, A., Derks, R., y Sakhniuk, M. (2022). *React and React Native* (4a ed.). Inglaterra: Packt.
- React Native. (s.f.). *React Native Learn once, write anywhere*. Recuperado de https://reactnative.dev


# Para saber más

Para conocer más acerca de **Módulos nativos**, te sugerimos leer lo siguiente:

- React Native. (2023). *Core Components and APIs*. Recuperado de https://reactnative.dev/docs/components-and-apis
- React Native. (2023). *Native Modules NPM Package Setup*. Recuperado de https://reactnative.dev/docs/native-modules-setup

**Vídeo**

Para conocer más acerca de **Módulos nativos** te sugerimos revisar lo siguiente:

- Unsure Programmer. (2022, 25 de marzo). *Native iOS Module in React Native using Swift* [Archivo de video]. Recuperado de https://www.youtube.com/watch?v=DREQwNb99l0
- Unsure Programmer. (2022, 29 de marzo). *Native Android Module React Native | Java* [Archivo de video]. Recuperado de https://www.youtube.com/watch?v=wk_cR66AVXQ

## Checkpoint

Asegúrate de:

- Comprender las ventajas de los módulos nativos para el desarrollo de software.
- Comprender la sintaxis de los módulos nativos para el desarrollo de software.
- Comprender la transformación de los módulos nativos para el desarrollo de software.

## Requerimientos técnicos

- Android Studio.
- React.
- Xcode.

## Prework

- Deberás revisar los siguientes temas:
  - **Tema 1. Fundamentos de React.**
  - **Tema 2. Programando con React Native.**
  - **Tema 3. Componentes Core y Nativos**