React Nivel Avanzado

Primeros pasos con React Native

Introducción

Antes de iniciar desarrollando con React, hay que tener claros algunos conceptos básicos.

Componente
Es necesario pensar en los componentes como piezas de código que se pueden reutilizar tantas veces como sea necesario, la idea es reducir el código repetido y facilitar el desarrollo, incluso es posible compartir componentes entre aplicaciones y, de esta forma, se evita crear cosas que ya existen o usar elementos existentes como las librerías de interfaces de usuario o animaciones
(React Native, s.f.).

A su vez, cada componente puede ser hijo de otro componente y tener componentes dentro de sí mismo. Incluso la App que se creó, es un componente compuesto por otros componentes
(Boduch, Derks y Sakhniuk, 2022).

Ciclo de vida de los componentes
Como todo elemento dentro de nuestra aplicación, un componente tiene un ciclo de vida:

Componente	Características
Montaje	Abarca desde el momento en que se inicializa el componente y se asignan los valores iniciales a las variables (obtenidos mediante props), aquí se ejecuta un primer update, que es el que contendrá los valores iniciales.
Actualización	Se ejecuta cuando se actualiza un valor del componente, en esta etapa se evalúa si es necesario renderizar de nuevo el componente, según los valores modificados.
Limpieza	Permite ejecutar código una vez que el componente se ha actualizado o ha terminado de actualizarse el DOM, ejemplos de esto podrían ser: • Eliminar eventos asociados. • Eliminar suscripciones a servicios. • Realizar mutaciones.
Desmontaje	Ocurre cuando se va a eliminar el componente, permite eliminar ciertas llamadas o referencias a otros métodos.

Tabla 1. Ciclo de vida de los componentes

Introducción

Hook

Una característica muy útil en los componentes permite utilizar o reutilizar métodos para aumentar la capacidad del componente. En React, los hooks permiten agregar funcionalidades como manejo de estados, variables globales, entre otras.

Por convención se nombran con la palabra use en minúsculas y camel-case para las siguientes palabras.

Ejemplos:

- useState
- useRadioToggle

Render

Se refiere a pintar (renderizar) el código JSX de la aplicación en elementos visuales, dentro de la aplicación.

Cada vez que un componente cambia de estado o sufre alguna alteración importante, React invoca al método de renderizar en los componentes mediante métodos, se encuentra en el **return** del código.

Debug

Es el proceso mediante el cual se irá probando la app y analizar cierta línea de código, para determinar y si está realizando lo que se espera que realice.

JSX

Es una variación de JavaScript que permite escribir etiquetas dentro de los archivos JavaScript, facilita la inserción de variables, objetos y llamadas a métodos de forma más intuitiva en tiempo de desarrollo.

Estos conceptos servirán más adelante para facilitar la familiarización con React Native.

Explicación

Primeros pasos con React Native

React Native es un framework diseñado, específicamente, para desarrollar aplicaciones para Android, iOS y, en últimas versiones, web.

Una de las principales ventajas de este framework es que, debido a la popularidad de JavaScript, la curva de aprendizaje es mucho menor a otros lenguajes orientados a móvil, ya que no requiere que el desarrollador aprenda un nuevo lenguaje, sino que puede usar sus conocimientos en JavaScript para iniciar.

Como cualquier herramienta basada en Node, al crearse un proyecto desde cero, se pueden elegir diversos tipos de plantilla para iniciar. La plantilla, por defecto, incluye lo necesario para trabajar con una aplicación demo y contenido de prueba (React Native, s.f.).

Se puede crear una app con ayuda del framework Expo usando el siguiente comando:

```
npx create-expo-app demo
```

Creando componentes

Todos los componentes cuentan con un método **render**, el cual sirve para devolver un elemento de la interfaz de usuario, éste puede tener o no más hijos, pero siempre se conforma de, por lo menos, un elemento, vista o componente.

A continuación, se muestra un ejemplo de un componente que devuelve el texto **Hola Mundo**, dentro de un componente nativo **View** con un hijo **Text**, que también es un componente nativo de React Native.

```
function Section(){
  return (
    <View>
      <Text>Hola mundo</Text>
    </View>
  );
}
```

Explicación

Agregando variables

En una aplicación, normalmente, no se muestran datos estáticos, por lo mismo, los componentes pueden variar el contenido.

Para lograr esto se puede hacer uso de las variables, las variables se pueden usar dentro del render, declarándolas antes.

```
function Section(){
    const saludo = "Buen día."
  return (
    <View>
      <Text>Hola mundo. {saludo}</Text>
    </View>
  );
}
```

En este componente, el valor de **saludo** se declara después de iniciar el método y se imprime en el **render**, encerrando la variable dentro de llaves.

Pasando valores mediante propiedades del componente

En el componente actual, el valor que se pretende mostrar se declara dentro de la misma función del componente. Si lo que se desea es modificar el componente desde fuera del mismo, se puede utilizar el siguiente método.

```
function Section(){
    return (
        <View>
            <Text>Hola mundo. {saludo}</Text>
        </View>
    );
}
```

Al pasar parámetros de esta forma a la función, se le puede asignar un valor a los props con ese mismo nombre que se llama al componente, para renderizarlo.

```
function App () {
    return <View>
            <Section saludo="Buenos días" />
    </View>
}
```

Explicación

Probando nuestra la aplicación creada

Todo código desarrollado debe ser probado para garantizar que la aplicación funcione de la forma en que se espera.

A la acción de probar el código en la aplicación se le conoce como depuración, esto se logra con el comando:

npx expo start

Al momento de depurar la app, se puede probar el código de forma instantánea y asegurar de que funciona, en caso de haber algún error, se puede detectar y podremos solucionarlo en el mismo momento.

Cierre

El lenguaje de programación debe ser seleccionado, de acuerdo con las necesidades que tenga el proyecto que se va a implementar. El de React Native ofrece muchas ventajas, ya que su estructura, basada en JavaScript, permite que sea fácil y comprensible para cualquier programador.

Algunas de las ventajas que brindará este lenguaje son:

- Programación sencilla y rápida.
- Funcionalidad en cualquier sistema operativo.
- Lenguaje Open Source, por lo que no se tiene que pagar para utilizarlo.
- Puede ser puesto en práctica en cualquier aplicación.

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

Lecturas

Para conocer más acerca de **Programando con React Native, te sugerimos leer lo siguiente:**

- React Native. (2023). *Running On Device*. Recuperado de https://reactnative.dev/docs/running-on-device
- React Native. (2023). *Testing*. Recuperado de https://reactnative.dev/docs/testing-overview

Videos

Para conocer más acerca de **Programando con React Native,** revisa lo siguiente:

- CodeHunger Pvt Ltd. (2020, 10 de marzo). *React Native Tutorial -10: ScrollView & FlatList* [Archivo de video]. Recuperado de https://www.youtube.com/watch?v=onR2zFjMp-w
- ProgrammingKnowledge. (2021, 30 de septiembre). *React Native Tutorial 9 - Lists & ScrollView* [Archivo de video]. Recuperado de https://www.youtube.com/watch?v=tUsvvTZhvtM

Checkpoint

Asegúrate de:

- Comprender las ventajas del uso de React Native para el desarrollo de software.
- Conocer la sintaxis de React Native para el desarrollo de software.
- Diferenciar cada uno de los elementos que se encuentran en React Native, para el desarrollo de software.

Requerimientos técnicos

- Android Studio.
- React.

Prework

Deberás revisar los siguientes temas:

- **Tema 1. Fundamentos de React.**
- **Tema 2. Programando con React Native.**