React Nivel Avanzado

APIs

Introducción

El desarrollo de un software puede llegar a ser bastante complicado y algunas funcionalidades demasiado complejas de programar. Debido a esto, surgió el concepto de API, la cual te permitirá conectar tu desarrollo con otro sistema con el objetivo de que dichas funcionalidades no tengan que volver a ser programadas y puedan ser utilizadas por tu desarrollo.

Dentro de React Native puedes utilizar diferentes APIs nativas que te permitirán reutilizar código existente, simplificando tu desarrollo y agilizándolo de igual forma.

Explicación

Android APIs

Dentro de Android existen diferentes APIs que puedes utilizar con React Native (2022), estas son:

1. **BackHandler**: te ayudará a detectar cuando el usuario haga clic en el botón hacia atrás, podrás registrar el evento *listener* y controlar cuál va a ser la respuesta que dará tu aplicación.

   A continuación, se muestra un ejemplo:

```
import React, { useEffect } from "react";
import { Text, View, StyleSheet, BackHandler, Alert } from "react-native";

const App = () => {
  useEffect(() => {
    const backAction = () => {
      Alert.alert("Hold on!", "Are you sure you want to go back?", [
        {
          text: "Cancel",
          onPress: () => null,
          style: "cancel"
        },
        { text: "YES", onPress: () => BackHandler.exitApp() }
      ]);
      return true;
    };

    const backHandler = BackHandler.addEventListener(
      "hardwareBackPress",
      backAction
    );

    return () => backHandler.remove();
  }, []);

  return (
    <View style={styles.container}>
      <Text style={styles.text}>Click Back button!</Text>
    </View>
  );
};
```

Explicación

```
const styles = StyleSheet.create({
  container: {
    flex: 1,
    alignItems: "center",
    justifyContent: "center"
  },
  text: {
    fontSize: 18,
    fontWeight: "bold"
  }
});

export default App;
```

El resultado sería el siguiente:

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

Explicación

2. La siguiente API recibe el nombre de **Permisos de Android** (PermissionsAndroid). De acuerdo con React native (2022a), esta te brinda acceso a los permisos que se encuentran en el archivo AndroidManifest.xml.

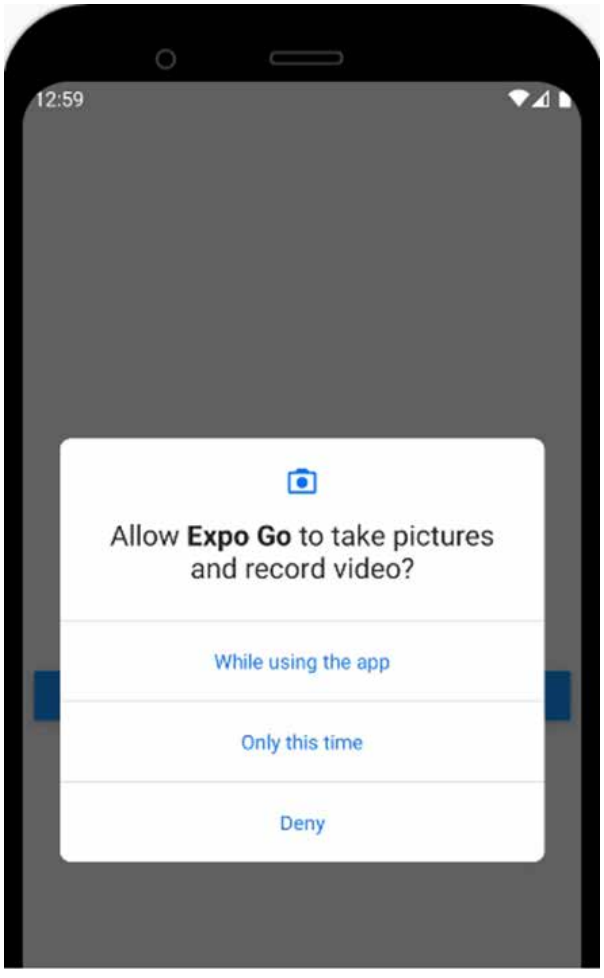
   Ejemplo:

```
import React from "react";
import { Button, PermissionsAndroid, SafeAreaView, StatusBar, StyleSheet,
Text, View } from "react-native";

const requestCameraPermission = async () => {
  try {
    const granted = await PermissionsAndroid.request(
      PermissionsAndroid.PERMISSIONS.CAMERA,
      {
        title: "Cool Photo App Camera Permission",
        message:
          "Cool Photo App needs access to your camera " +
          "so you can take awesome pictures.",
        buttonNeutral: "Ask Me Later",
        buttonNegative: "Cancel",
        buttonPositive: "OK"
      }
    );
    if (granted === PermissionsAndroid.RESULTS.GRANTED) {
      console.log("You can use the camera");
    } else {
      console.log("Camera permission denied");
    }
  } catch (err) {
    console.warn(err);
  }
};

const App = () => (
  <View style={styles.container}>
    <Text style={styles.item}>Try permissions</Text>
    <Button title="request permissions" onPress={requestCameraPermission} />
  </View>
);

const styles = StyleSheet.create({
  container: {
    flex: 1,
    justifyContent: "center",
    paddingTop: StatusBar.currentHeight,
    backgroundColor: "#ecf0f1",
```

Explicación

```
    padding: 8
  },
  item: {
    margin: 24,
    fontSize: 18,
    fontWeight: "bold",
    textAlign: "center"
  }
});

export default App;
```

El resultado sería el siguiente:



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

Explicación

Estos son los permisos que necesitas que el usuario conceda:

- READ_CALENDAR: para leer calendario.
- WRITE_CALENDAR: para escribir en calendario.
- CAMERA: consentimiento para abrir la cámara.
- READ_CONTACTS: para leer contactos.
- WRITE_CONTACTS: para escribir en contactos.
- GET_ACCOUNTS: para obtener tu cuenta.
- ACCESS_FINE_LOCATION: para encontrar tu ubicación mediante GPS.
- ACCESS_COARSE_LOCATION: para encontrar la ubicación del dispositivo usando internet.
- ACCESS_BACKGROUND_LOCATION: para encontrar la ubicación usando el archivo Android manifest.
- RECORD_AUDIO: para grabar audio.
- READ_PHONE_STATE: para acceder al estado del teléfono.
- CALL_PHONE: para hacer llamadas.
- READ_CALL_LOG: para leer el log de llamadas.
- WRITE_CALL_LOG: para escribir en el log de llamadas.
- ADD_VOICEMAIL: para añadir buzón de voz.
- PROCESS_OUTGOING_CALLS: para saber el proceso de una llamada saliente.
- BODY_SENSORS: para usar los sensores de movimiento.
- SEND_SMS: para mandar un SMS.
- RECEIVE_SMS: para recibir un SMS.
- READ_SMS: para leer un SMS.
- RECEIVE_MMS: para recibir un MMS.
- READ_EXTERNAL_STORAGE: para leer un almacenamiento externo.
- WRITE_EXTERNAL_STORAGE: para escribir en un almacenamiento externo.
- BLUETOOTH_CONNECT: para conectarse por bluetooth.
- Entre otras.

Las respuestas que el usuario podrá darle a tu aplicación son:

- GRANTED: 'otorgado'
- DENIED: 'denegado'
- NEVER_ASK_AGAIN: 'nunca preguntes de nuevo'

iOS APIs

De igual forma, existen diferentes APIs para iOS:

1. De acuerdo con React Native (2022b), una de las APIs recibe el nombre de **ActionSheetiOS**, que te muestra el componente de un Actionsheet nativo de iOS. Esto es una hoja de acción, en la cual el programador coloca el código para que si se hace clic en un botón pase algo.

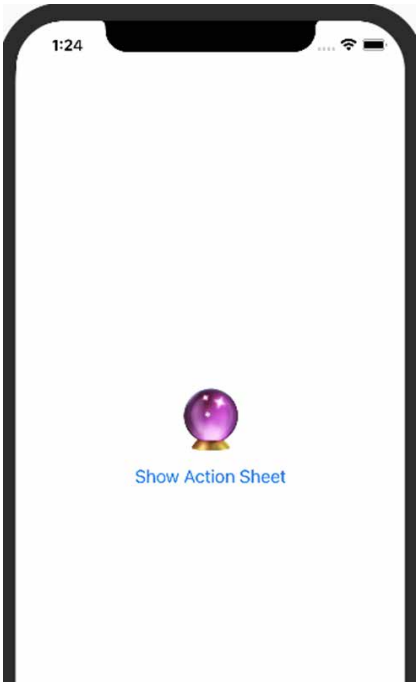
A continuación, se muestra un ejemplo en el cual el usuario deberá hacer clic en un botón para obtener un número al azar entre 1 y 100:

Explicación

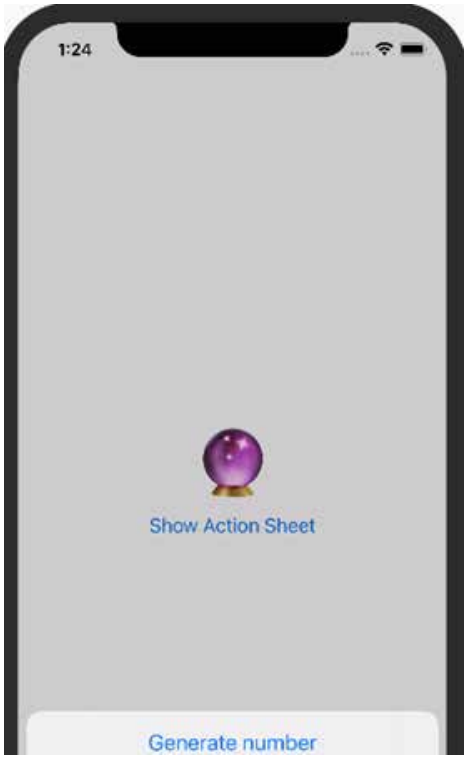
```
import React, { useState } from "react";
import { ActionSheetIOS, Button, StyleSheet, Text, View } from "react-native";

const App = () => {
  const [result, setResult] = useState("□");

  const onPress = () =>
    ActionSheetIOS.showActionSheetWithOptions(
      {
        options: ["Cancel", "Generate number", "Reset"],
        destructiveButtonIndex: 2,
        cancelButtonIndex: 0,
        userInterfaceStyle: 'dark'
      },
      buttonIndex => {
        if (buttonIndex === 0) {
          // cancel action
        } else if (buttonIndex === 1) {
          setResult(Math.floor(Math.random() * 100) + 1);
        } else if (buttonIndex === 2) {
          setResult("□");
        }
      }
    );
```

Explicación

Este sería el resultado:



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



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

Explicación

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

2. La segunda API recibe el nombre de **DynamicColorIOS**, el cual te ayudará a dar un color en específico a tu plataforma. De acuerdo con React Native (2022c), funciona de esta forma:

   Se toma un solo argumento como un objeto con dos claves obligatorias: darky light, y dos claves opcionales highContrastLighty highContrastDark. Estos corresponden a los colores que se usarán para el "modo claro" y el "modo oscuro" en iOS, y cuando el modo de accesibilidad de "alto contraste" está habilitado, se activará en modo obscuro y claro.

   En el tiempo de ejecución, el sistema elegirá cuál de los colores mostrar según la apariencia actual del sistema y la configuración de accesibilidad. Los colores dinámicos son útiles para la marca y otras especificaciones de la aplicación, que siguen respondiendo automáticamente a los cambios de configuración del sistema.

Explicación

El código para ejecutar dicha API es el siguiente:

```
import {DynamicColorIOS} from 'react-native';

const customDynamicTextColor = DynamicColorIOS({
  dark: 'lightskyblue',
  light: 'midnightblue',
});

const customContrastDynamicTextColor = DynamicColorIOS({
  dark: 'darkgray',
  light: 'lightgray',
  highContrastDark: 'black',
  highContrastLight: 'white',
});
```

Cierre

Las API son utilizadas para que una aplicación se integre con otras aplicaciones o sistemas que ya existen, de tal forma que el programador no tenga la necesidad de codificar dicha aplicación o sistema, lo cual te ayudará a aumentar la velocidad en tu desarrollo, ya que estarás reutilizando código existente y no tendrás que hacerlo desde cero.

Referencias Bibliográficas

- React Native. (2022). *BackHandler*. Recuperado de https://reactnative.dev/docs/backhandler
- React Native. (2022a). *PermissionsAndroid*. Recuperado de https://reactnative.dev/docs/permissionsandroid
- React Native. (2022b). *ActionSheetIOS*. Recuperado de https://reactnative.dev/docs/actionsheetios
- React Native. (2022c). *DynamicColorIOS*. Recuperado de https://reactnative.dev/docs/dynamiccolorios

Para saber más

Lecturas

Para conocer más acerca de **APIs**, te sugerimos leer:

- React Native. (2022). *AccesibilityInfo*. Recuperado de https://reactnative.dev/docs/accessibilityinfo
- React Native. (2022a). *Keyboard*. Recuperado de https://reactnative.dev/docs/keyboard

Videos

Para conocer más acerca de **APIs** te sugerimos revisar:

- Juan Zuluaga. (2022, 28 de enero). *Tutorial 64 – React / React-Native – 75 – Configurar el API de Google Maps Parte I* [Archivo de video]. Recuperado de https://www.youtube.com/watch?v=3Do1TnYmBi0
- MissCoding. (2022, 29 de julio). *Fetching an API (Simple GET Request) with Loading Spinner in React Native App* [Archivo de video]. Recuperado de https://www.youtube.com/watch?v=_VZfBrmunD4

Checkpoint

Asegúrate de:

- Comprender la importancia de las API para el desarrollo de las aplicaciones.
- Reconocer la sintaxis de las API para lograr un mejor desarrollo.
- Entender el uso de las API para lograr un mejor desarrollo.

Requerimientos técnicos

- Android Studio.
- React.
- Xcode.

Prework

- Deberás revisar los temas:
  - **Tema 1. Fundamentos de React.**
  - **Tema 2. Programando con React Native.**
  - **Tema 3. Componentes Core y Nativos.**
  - **Tema 4. Maquetando la app.**
  - **Tema 5. Eventos, estados y contexto.**
  - **Tema 6. Consumiendo servicios web.**
  - **Tema 7. Preparando la app para publicación.**
  - **Tema 8. API.**