Universidad Tecnológica de Tijuana

First Portfolio Advance

Degree: TSU in Information Technology in the area of Software Development Multiplatform.

Teacher: Ray Brunett Parra Galaviz

Student: Martinez Valenzuela Mariana Lizbeth

Group: 4B

Delivery date: 04/02/2025

Index

Introduction **3**
Partial 1 **3**
Project Creation 3
Practice 1 . Hello World. 4
Practice 2. ¡ Click me ! 6
Practice 3. Static and URL images 8
Practice 4. Message with alert. 10

Introduction

This portfolio presents the development of various practices in React Native using Expo, aiming to reinforce fundamental concepts in cross-platform mobile application development. Throughout this document, the procedures and lessons learned in each practice are detailed, exploring key aspects such as project creation, state management, user interaction, and image integration.

The included practices range from implementing a simple "Hello World" to integrating advanced components such as interactive forms and alerts. Each exercise contributes to understanding the essential tools of React Native and its ecosystem, enabling the development of functional and visually appealing applications. This portfolio not only documents the technical progress achieved but also reflects the learning process and the application of best practices in mobile software development.

Partial 1

Project Creation

To start a new project in Expo, we run the following commands in the terminal:

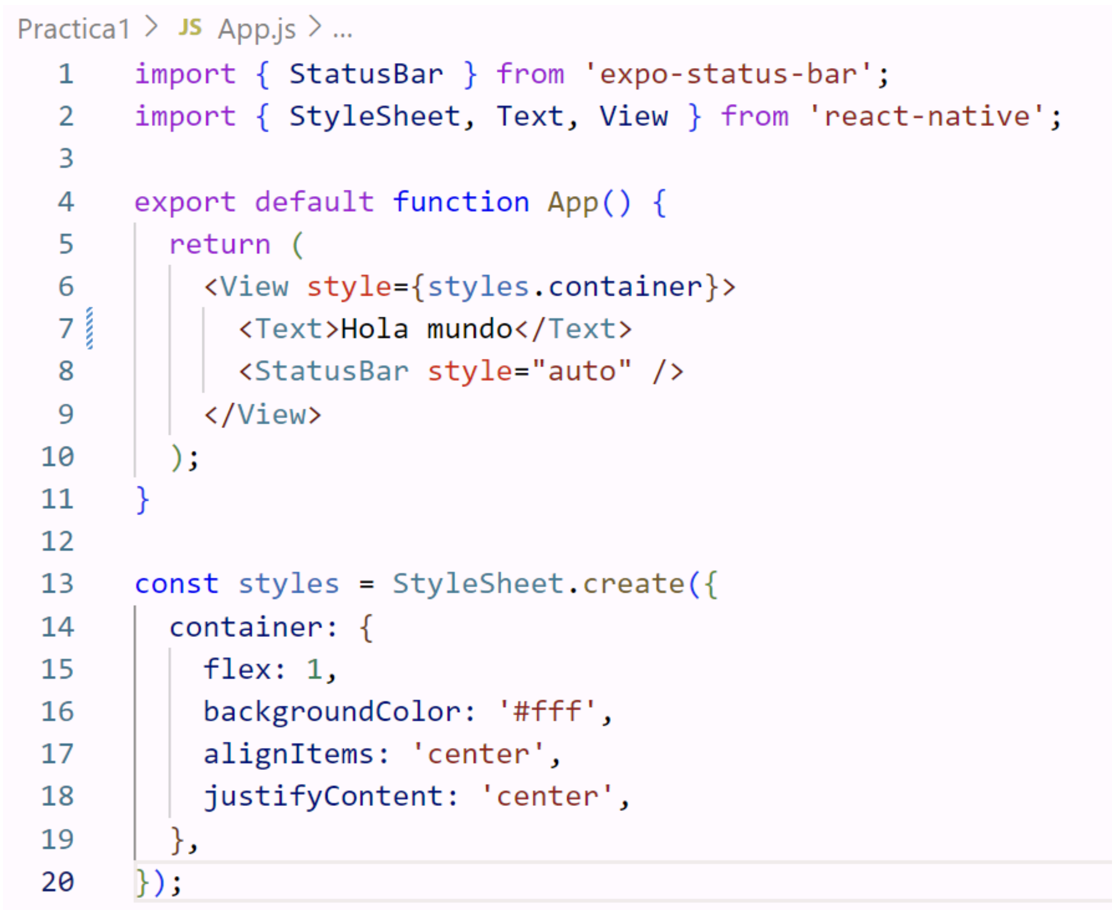
```
npx create-expo-app nombre --template
(Elegir la opción Blank)
cd nombre
npx expo install react-dom react-native-web
@expo/metro-runtime
npx expo start
```

These commands allow:

- Create a new project with Expo.
- Enter the project directory.
- Install additional dependencies required to run the application on the web.
- Start the Expo development server.

Practice 1 . Hello World.

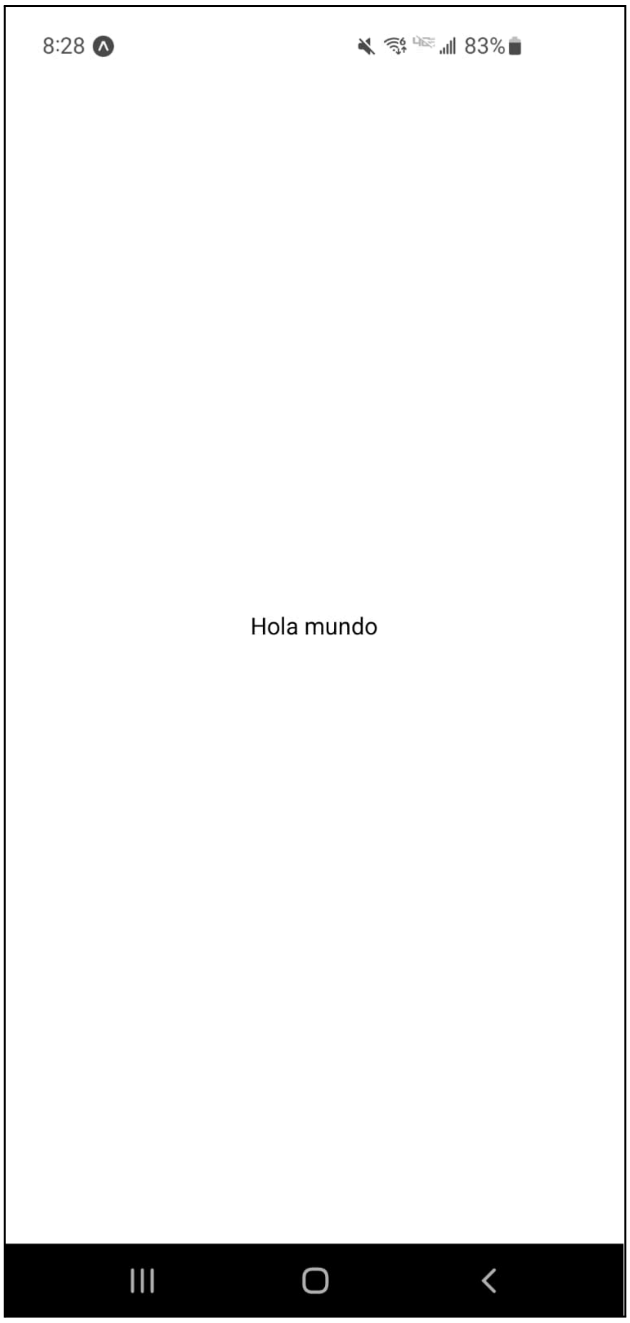
The following code represents our first React Native app using Expo. It is a simple app that displays a message on the screen:



Practica1 > JS App.js > ...

```javascript
import { StatusBar } from 'expo-status-bar';
import { StyleSheet, Text, View } from 'react-native';

export default function App() {
  return (
    <View style={styles.container}>
      <Text>Hola mundo</Text>
      <StatusBar style="auto" />
    </View>
  );
}

const styles = StyleSheet.create({
  container: {
    flex: 1,
    backgroundColor: '#fff',
    alignItems: 'center',
    justifyContent: 'center',
  },
});
```

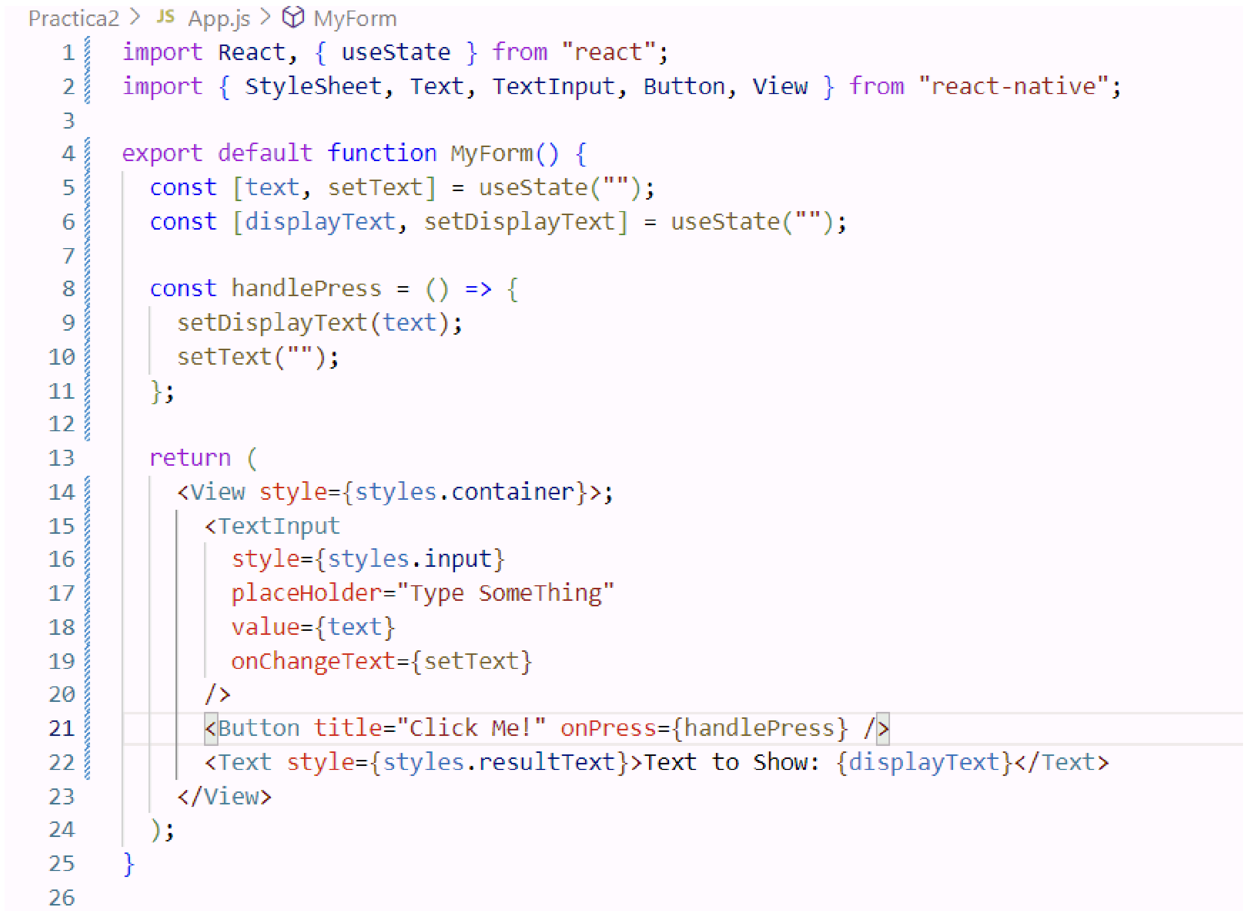
The components StatusBar, StyleSheet, Text, and View are imported from React Native. The function App represents the main component of the application, uses a View container to organize the content, displays the text "Hello world" using Text, adds the status bar with StatusBar, and defines a styles object with CSS properties to center the content on the screen.



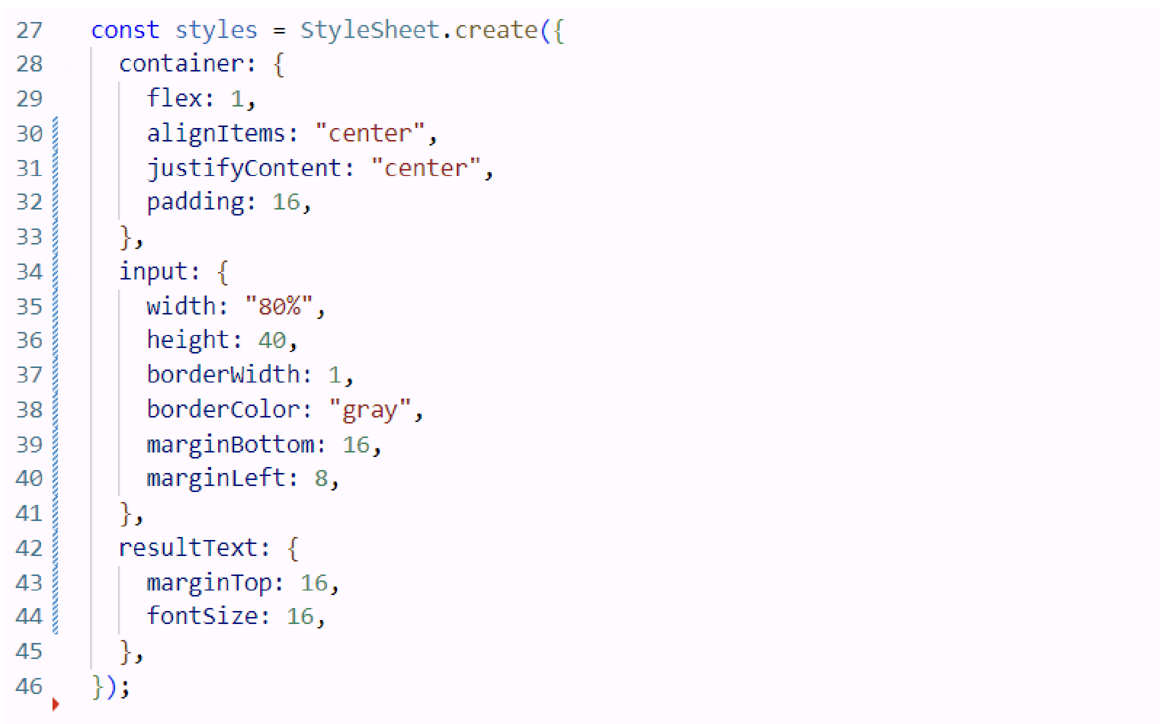
This first practice allowed us to familiarize ourselves with the basic structure of a React Native application and with the initial setup of the development environment using Expo.

Practice 2. ¡ Click me !

In this practice, we explored the use of state in React Native through the useState hook. We created an application that allows the user to enter text in an input field and display it on the screen when a button is pressed.

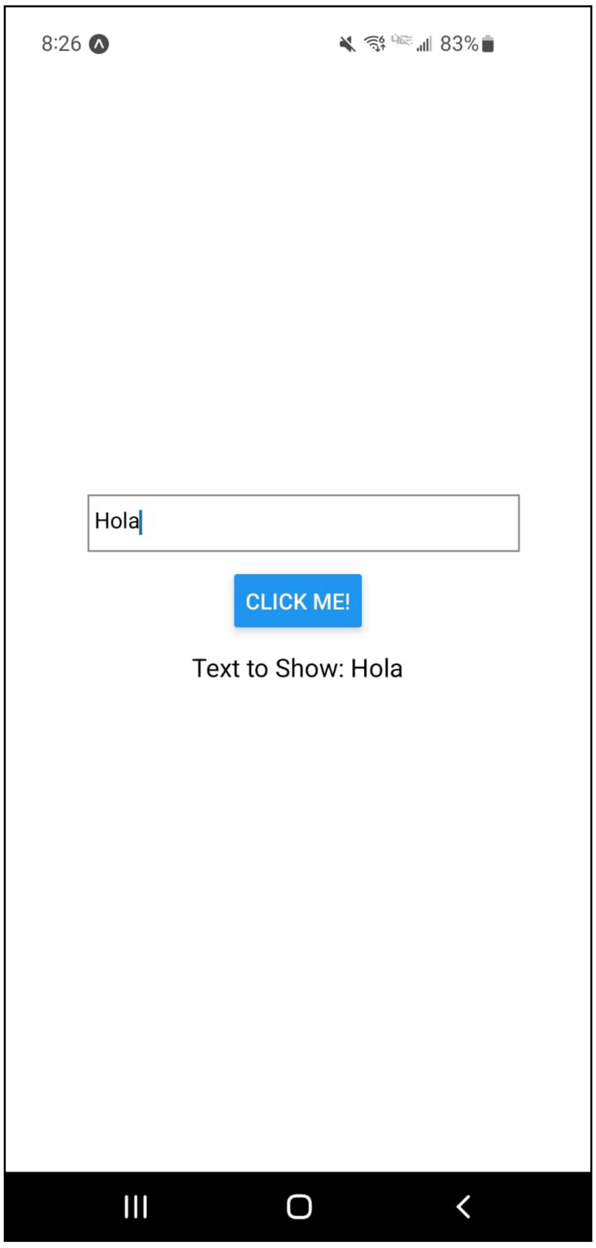


```javascript
import React, { useState } from "react";
import { StyleSheet, Text, TextInput, Button, View } from "react-native";

export default function MyForm() {
  const [text, setText] = useState("");
  const [displayText, setDisplayText] = useState("");

  const handlePress = () => {
    setDisplayText(text);
    setText("");
  };

  return (
    <View style={styles.container}>;
      <TextInput
        style={styles.input}
        placeHolder="Type SomeThing"
        value={text}
        onChangeText={setText}
      />
      <Button title="Click Me!" onPress={handlePress} />
      <Text style={styles.resultText}>Text to Show: {displayText}</Text>
    </View>
  );
}

```



```javascript
const styles = StyleSheet.create({
  container: {
    flex: 1,
    alignItems: "center",
    justifyContent: "center",
    padding: 16,
  },
  input: {
    width: "80%",
    height: 40,
    borderWidth: 1,
    borderColor: "gray",
    marginBottom: 16,
    marginLeft: 8,
  },
  resultText: {
    marginTop: 16,
    fontSize: 16,
  },
});
```

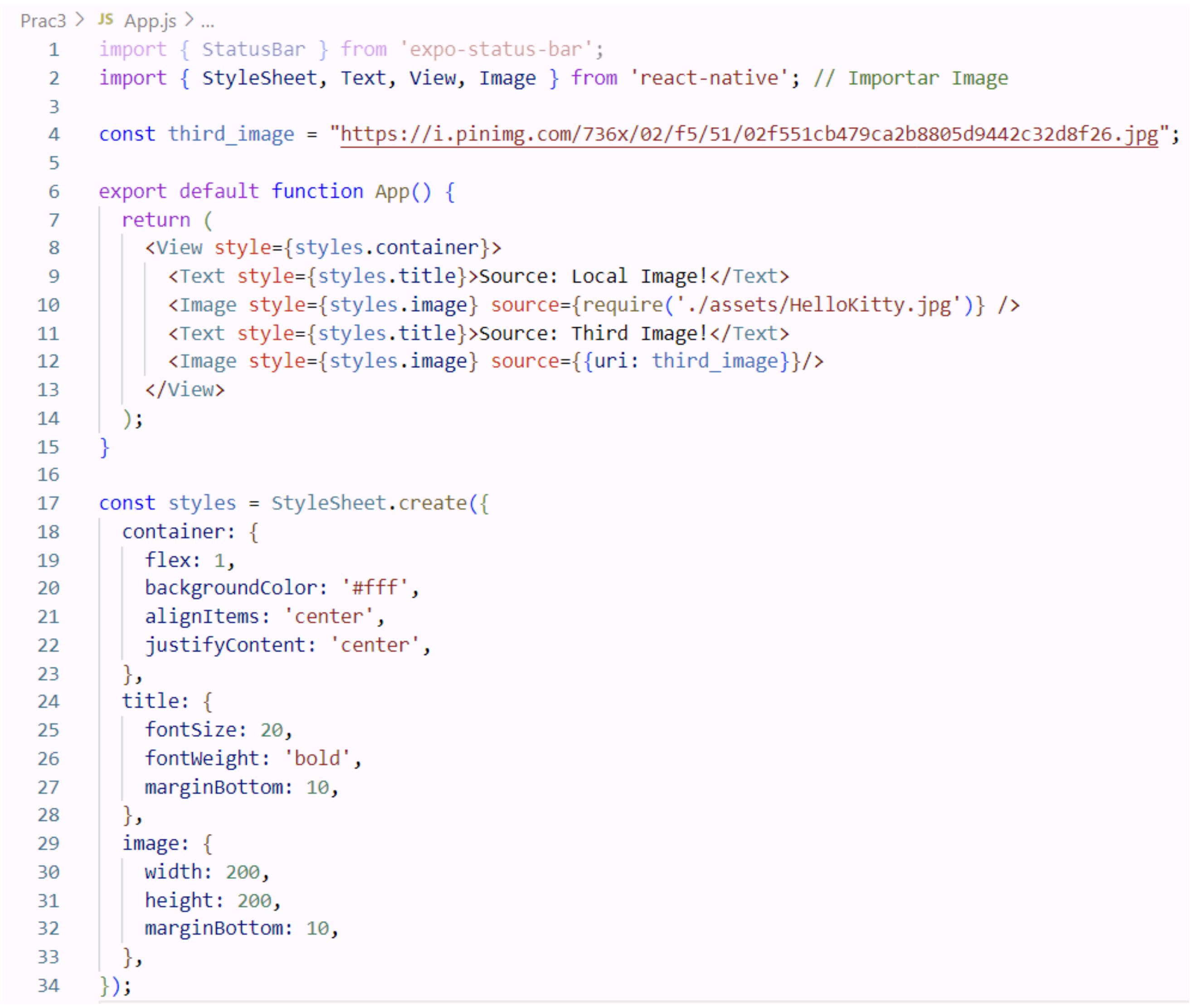
In this practice, the essential React Native components and the "useState" hook are imported to manage the application's state. The "text" variable stores the value entered by the user in the "TextInput", while "displayText" contains the text that will be shown on the screen. The "handlePress" function assigns the content of "text" to "displayText" and then clears the input field, allowing the user to enter a new value. The "TextInput" captures the user's input, the button triggers the "handlePress" function, and the "Text" displays the updated result. This practice helped us understand how to manage states in React Native and how to create dynamic interaction with the user through text inputs and buttons.



Text to Show: Hola

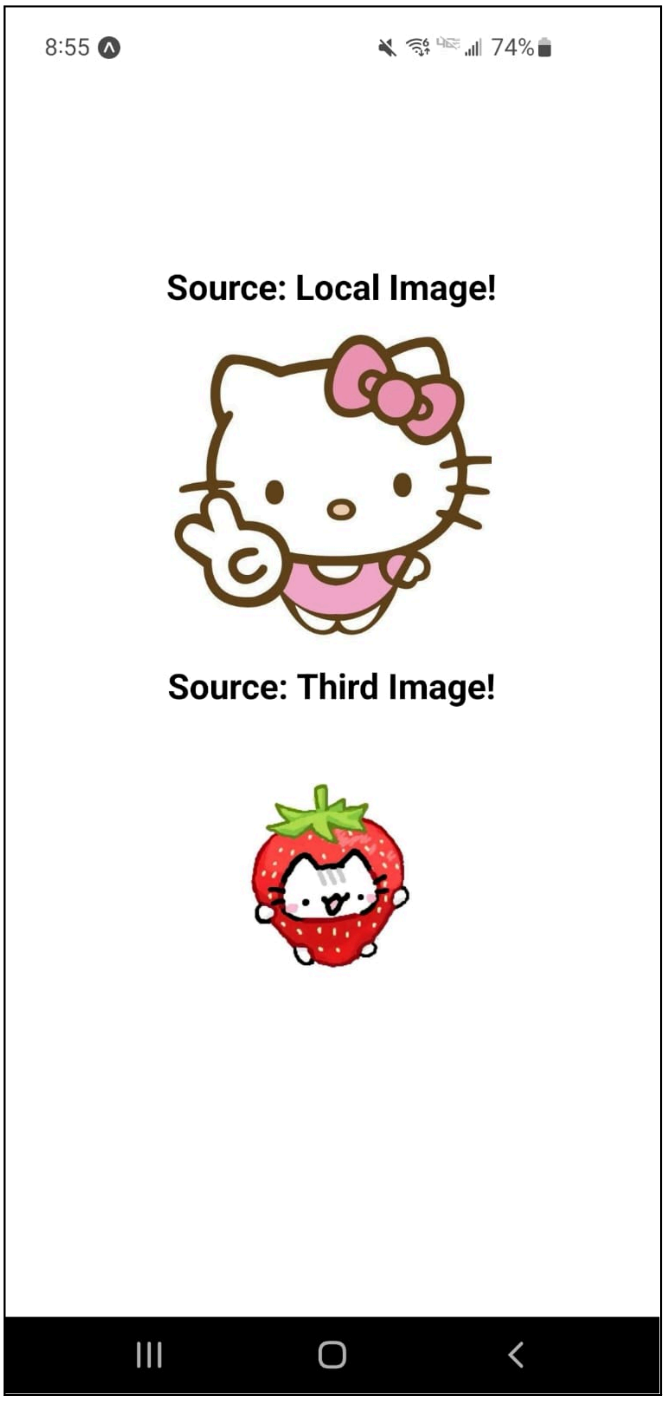
Practice 3. Static and URL images

In this practice, we learned how to display images in React Native using the "Image" component, loading them both from local files and from an external URL.

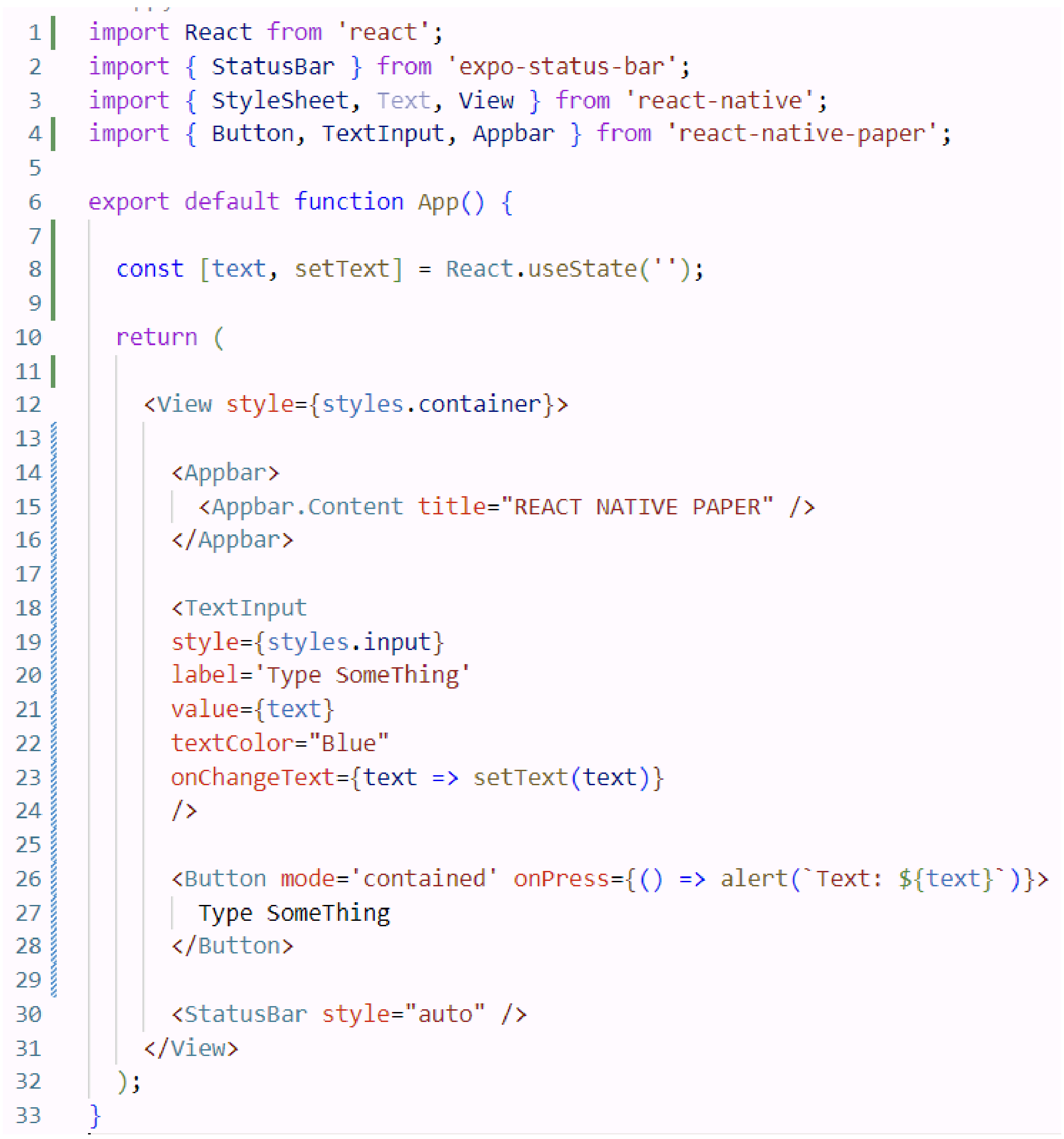


```javascript
import { StatusBar } from 'expo-status-bar';
import { StyleSheet, Text, View, Image } from 'react-native'; // Importar Image

const third_image = "https://i.pinimg.com/736x/02/f5/51/02f551cb479ca2b8805d9442c32d8f26.jpg";

export default function App() {
  return (
    <View style={styles.container}>
      <Text style={styles.title}>Source: Local Image!</Text>
      <Image style={styles.image} source={require('./assets/HelloKitty.jpg')} />
      <Text style={styles.title}>Source: Third Image!</Text>
      <Image style={styles.image} source={{uri: third_image}}/>
    </View>
  );
}

const styles = StyleSheet.create({
  container: {
    flex: 1,
    backgroundColor: '#fff',
    alignItems: 'center',
    justifyContent: 'center',
  },
  title: {
    fontSize: 20,
    fontWeight: 'bold',
    marginBottom: 10,
  },
  image: {
    width: 200,
    height: 200,
    marginBottom: 10,
  },
});
```

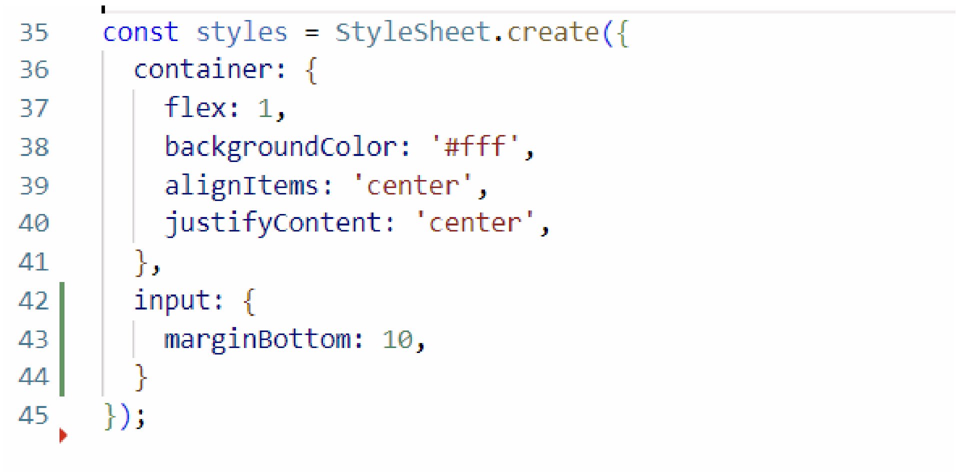
In this practice, we explored the use of the "Image" component in React Native to display both local and external images. The essential components, including "Image," are imported, and a variable "third_image" is defined to store the URL of an external image. Within the "App" component, the elements are organized in a centralized "View," where a title "Text" is shown first, indicating that the image comes from a local resource, followed by the image loaded using "require('./assets/HelloKitty.jpg')." Then, another title is added to indicate the origin of an external image, and it is displayed using "{uri: third_image}," allowing it to be loaded from the web. The applied styles ensure that the elements are centered, with a white background and a uniform size of 200x200 pixels for both images, as well as margins to improve presentation. This practice helped us understand how React Native handles images, learning to integrate them from local files and dynamically load them from external sources, which is essential for creating visually appealing and dynamic applications.



Practice 4. Message with alert.

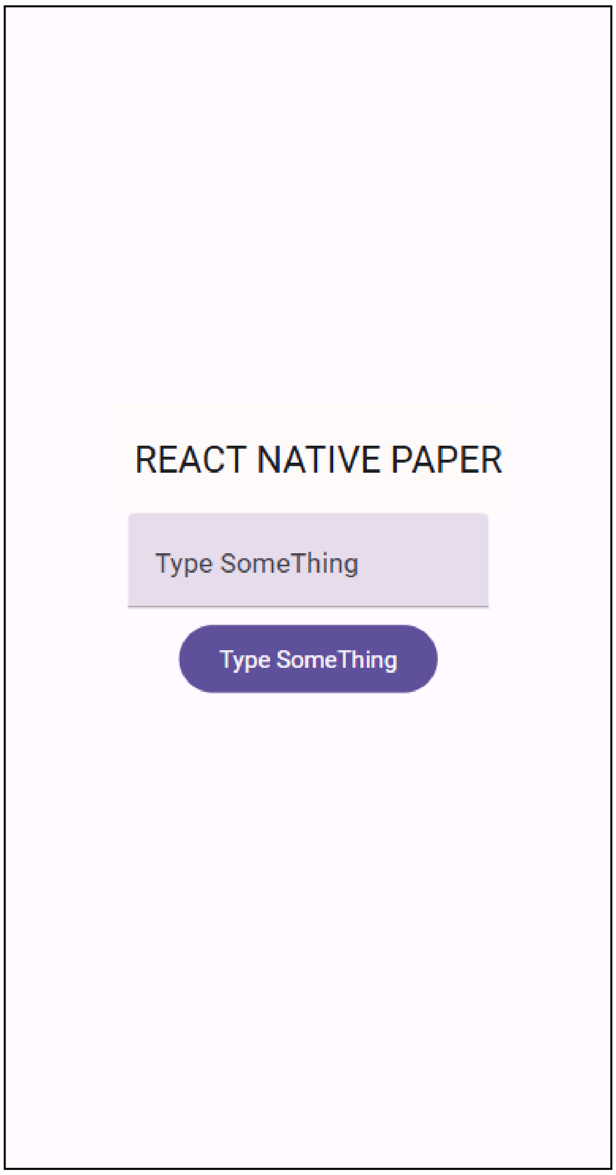


```
import React from 'react';
import { StatusBar } from 'expo-status-bar';
import { StyleSheet, Text, View } from 'react-native';
import { Button, TextInput, Appbar } from 'react-native-paper';

export default function App() {

  const [text, setText] = React.useState('');

  return (

    <View style={styles.container}>

      <Appbar>
        <Appbar.Content title="REACT NATIVE PAPER" />
      </Appbar>

      <TextInput
      style={styles.input}
      label='Type SomeThing'
      value={text}
      textColor="Blue"
      onChangeText={text => setText(text)}
      />

      <Button mode='contained' onPress={() => alert(`Text: ${text}`)}>
        Type SomeThing
      </Button>

      <StatusBar style="auto" />
    </View>
  );
}
```



```
const styles = StyleSheet.create({
  container: {
    flex: 1,
    backgroundColor: '#fff',
    alignItems: 'center',
    justifyContent: 'center',
  },
  input: {
    marginBottom: 10,
  }
});
```

In this practice, we explored the use of the "React Native Paper" library to enhance the appearance and functionality of components in our application. "TextInput," "Button," and "Appbar" are used to create a more stylish and interactive interface. The "Appbar" component adds a header bar with a title, while "TextInput" allows the user to enter text with a modern design. When the button is pressed, an alert is shown with the entered text, demonstrating how to handle events in React Native. This practice helped us understand how to use third-party components to improve the user experience in our applications.


REACT NATIVE PAPER
Type SomeThing
Type SomeThing