ACTIVIDADES PRÁCTICAS:

Título Eventos onPress y onChangeText en React Native

Objetivos

Aprender a implementar eventos onPress y onChangeText en React Native.

Temporalización

Dedicación estimada: 2 horas.

Actividades:

En este bloque de actividades prácticas aprenderemos a implementar los eventos onPress y onChangeInput.

Primeramente trabajaremos con el evento onPress en componentes 'core' como Text. Para ello actualizaremos los distintos estados necesarios para realizar cada ejercicio mediante useState.

Ejercicio 1.

Para este ejercicio trabajarás con el siguiente código:

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


export default function App() {


    return (
        <View style={styles.container}>
            <Text style={styles.text}>
                Texto
            </Text>
            <Text style={styles.text}>
                Texto
            </Text>
            <Text style={styles.text}>
                Texto
            </Text>
            <Text style={styles.text}>
                Texto
            </Text>
        </View>
    );
}

const styles = StyleSheet.create({
    container: {
        flex: 1,
        backgroundColor: 'white',
        alignItems: 'center',
        justifyContent: 'center',
    },
    text: {
        fontSize: 30,
        marginBottom: 20,
    },
});
```

Sin emplear arrays en los estados y utilizando una única función para manejar los cuatro eventos, implementa las instrucciones necesarias para que cuando el usuario pulse sobre cualquier componente Text, se cambie el texto contenido en ese componente. Si vuelve a pulsar sobre el mismo componente, volverá al valor inicial. Si vuelve a pulsar, se volverá a cambiar. Y así sucesivamente.

Ejercicio 2.

Para este ejercicio trabajarás con el siguiente código:

```
import { Text, Pressable, Image, View, StyleSheet } from 'react-native';


export default function App() {
  return (
    <View style={styles.containerRow}>
      <Pressable>
        <Image style={styles.image} source={} />
        <Text>Texto 1</Text>
      </Pressable>
      <Pressable>
        <Image style={styles.image} source={} />
        <Text>Texto 2</Text>
      </Pressable>
    </View>
  );
}

const styles = StyleSheet.create({
  containerRow: {
    flex: 1,
    backgroundColor: 'white',
    alignItems: 'center',
    justifyContent: 'center',
    flexDirection:'row',
  },
  image: {
    width: 100,
    height: 100
  },
});
```

Sin emplear arrays en los estados y utilizando una única función para manejar los dos eventos, implementa las instrucciones necesarias para que cuando el usuario pulse sobre el componente Image, se cambie la imagen de ese componente y el texto del componente Text ubicado debajo. Si vuelve a pulsar sobre el mismo componente, ambos componentes –Image y Text-, volverán al valor inicial. Si vuelve a pulsar, se volverán a cambiar. Y así sucesivamente. NOTA: implementa a cada componente Image una foto de tu elección.

Ejercicio 3.

Para este ejercicio trabajarás con el siguiente código:

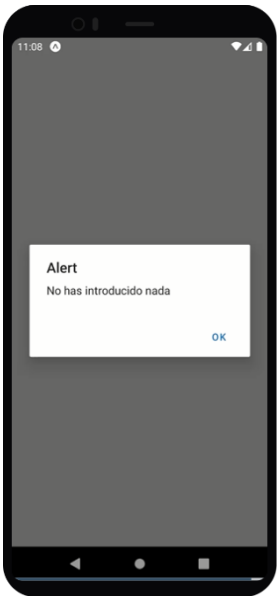
```
import { Pressable, View, StyleSheet } from 'react-native';


export default function App() {

  return (
    <View style={styles.container}>
      <Pressable style={styles.square}/>
      <Pressable style={styles.square}/>
    </View>
  );
}

const styles = StyleSheet.create({
  container: {
    flex:  1,
    backgroundColor: 'white',
    alignItems: 'center',
    justifyContent: 'center',
  },
  square: {
    marginTop: -6,
    width: 200,
    height: 200,
    backgroundColor: 'yellow',
  },
});
```

Sin emplear arrays en los estados y utilizando una única función para manejar los dos eventos, implementa las instrucciones necesarias para que cuando el usuario pulse sobre el cuadrado superior, los dos cuadrados sean de color verde y se aumente su tamaño –de ambos-. Si el usuario pulsa sobre el cuadrado superior, ambos serán de color amarillo y disminuirán su tamaño.

En las siguientes actividades prácticas aprenderemos a implementar el evento onChangeText en el componente ‘core’ TextInput. Para ello actualizaremos los distintos estados necesarios para realizar cada ejercicio mediante useState. También utilizaremos el componente ‘core’ Pressable y su evento asociado onPress.

Ejercicio 4.

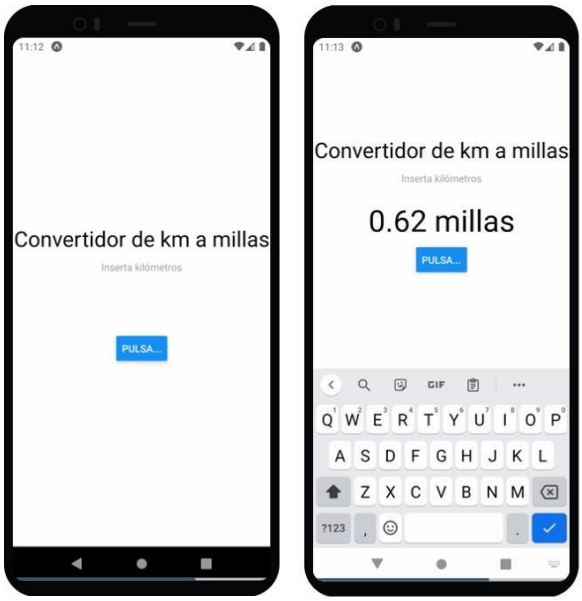
Implementa las instrucciones necesarias para renderizar en la interfaz gráfica un componente TextInput que recoja el contenido que introduzca el usuario, debajo, un componente Pressable. Implementa la lógica necesaria para que cuando el usuario pulse el botón de Pressable, salte una alerta indicando los siguientes casos: si se ha introducido texto, si se ha introducido un número o si no se ha introducido nada.



Ejercicio 5.

Implementa las instrucciones necesarias para renderizar en la interfaz gráfica un convertidor de kilómetros a millas. Para ello, renderiza un componente TextInput que recoja el contenido que introduzca el usuario, y, debajo, un componente Pressable. Implementa la lógica necesaria para que cuando el usuario pulse el botón de Pressable, se renderice debajo del componente TextInput la cantidad de kilómetros introducida convertida a millas. Si el usuario no introduce ningún dato, saltará una alerta indicándolo y se borrará el contenido introducido en TextInput. Si introduce texto, se le indicará que ha introducido texto y se borrará el contenido introducido en TextInput.

Nota: 1 kilómetro son 0.62 millas.



Ejercicio 6.

Siguiendo las instrucciones del ejercicio anterior, implementa ahora un convertidor de euros a dólares.

Ejercicio 7.

Siguiendo las instrucciones del ejercicio anterior, implementa ahora con la ayuda de los componentes TextInput y Pressable un validador de DNI. Nota: ten en cuenta que el DNI tendrá que estar formado por 8 números y una letra, cualquier otra combinación, no será válida y se indicará mediante una alerta al usuario.