



Universidad Politecnica de Queretaro
Tecnologias de la Informacion e Innovacion Digital
Fecha de entrega:28/10/25 Grupo:TIID211



Materia:Desarrollo Apps Moviles

Profesor: Ivan Isay Guerra Lopez

Alumno:

Irineo Atanacio Diego

Fecha de entrega:28/10/25



Activity Screen

Principalmente como en todos los códigos empezamos con los imports requeridos para la práctica, aquí `useState` nos va a ayudar a la variable `Cargando` y `ActivityIndicator` nos muestra el icono de carga con animación

```
JS MenuScreen.js M JS ActivityScreen.js U X JS ImagenScreen.js JS TextScreen.js JS BotonesScreen.js
INTRO > screens > JS ActivityScreen.js > ...
1 import { useState } from 'react';
2 import { Text, View, Button, StyleSheet, ActivityIndicator, Platform } from 'react-native';
3
```

El componente `ActivityScreen` dentro de el se declara el estado `cargando` y la función `carga()` que activa y desactiva la simulación

```
JS MenuScreen.js M JS ListScreen.js U JS ActivityScreen.js U X JS ImagenScreen.js
INTRO > screens > JS ActivityScreen.js > ...
4 |
5 export default function ActivityScreen()
6 {
7   const [cargando, setCargando] = useState(false);
8   const carga = () => {
9     setCargando(true);
10    setTimeout(() => {
11      setCargando(false);
12      if (Platform.OS === 'web') {
13        window.alert('Carga completa');
14      } else {
15        window.alert('Carga completa');
16      }
17    }, 3000);
18  };
19 }
```



Dentro del return se define la vista principal del View con un texto y un botón, el botón inicia la simulación de carga y cuando el estado cargando es verdadero

```
JS MenuScreen.js M   JS ListScreen.js U   JS ActivityScreen.js U X   JS ImagenScreen.js   JS TextScreen.js   JS BotonesScreen.js   JS RepasoScreen.js   JS ScrollView.js  
INTRO > screens > JS ActivityScreen.js > ...  
5   export default function ActivityScreen()  
20   return (  
21     <View style={styles.container}>  
22       <Text style={styles.texto}>Presione para iniciar la simulación de carga</Text>  
23       <View style={styles.botonesContainer}>  
24         {cargando ? (<ActivityIndicator size="large" color="red" />):(<Button color="brown" title="Presione para iniciar" onPress={carga} />)}  
25       </View>  
26     </View>  
27   );  
28 }
```

Aquí los estilos

```
JS MenuScreen.js M   JS ListScreen.js U   JS ActivityScreen.js U X   JS ImagenScreen.js  
INTRO > screens > JS ActivityScreen.js > ...  
29   const styles = StyleSheet.create({  
30     container: {  
31       flex: 1,  
32       backgroundColor: '■ #ffffff',  
33       alignItems: 'center',  
34       justifyContent: 'center',  
35     },  
36     texto: {  
37       color: '□ #000000ff',  
38       fontSize: 30,  
39       fontFamily: 'Times New Roman',  
40       fontWeight: 'bold',  
41       fontStyle: 'italic',  
42     },  
43     botonesContainer: {  
44       marginTop: 20,  
45       gap: 20,  
46     },  
47   },  
48   });
```



Conclusion

Aquí se demuestra como implementar una simulacion de carga funcional utilizando el ActivityIndicator y el manejo de estados con useState



ListScreen

Principalmente como en todos los códigos empezamos con los imports requeridos para la práctica, FlatList sirve para mostrar listas planas y simples, el SectionList se usa para mostrar listas con secciones o categorías

```
JS MenuScreen.js M   JS ListScreen.js U X   JS ActivityScreen.js U   JS ImagenScreen.js   JS TextScreen.js  
INTRO > screens > JS ListScreen.js > ListScreen  
1   import { Text, StyleSheet, View, FlatList, SectionList } from 'react-native'  
2  
3
```

Aquí el ListScreen contiene la estructura visual de ambas listas

```
JS MenuScreen.js M   JS ListScreen.js U X   JS ActivityScreen.js U   JS ImagenScreen.js   JS TextScreen.js   JS BotonesScreen.js  
INTRO > screens > JS ListScreen.js > ListScreen  
3  
4   export default function ListScreen() {  
5     const ejercicios = [  
6  
7  
8       {id: '1', nombre: 'Sentadillas', descripcion: 'Ejercicio para piernas y gluteos'},  
9       {id: '2', nombre: 'Sentadillas', descripcion: 'Ejercicio para piernas y gluteos'},  
10     {id: '3', nombre: 'Sentadillas', descripcion: 'Ejercicio para piernas y gluteos'},  
11     {id: '4', nombre: 'Sentadillas', descripcion: 'Ejercicio para piernas y gluteos'},  
12     {id: '5', nombre: 'Sentadillas', descripcion: 'Ejercicio para piernas y gluteos'},  
13     {id: '6', nombre: 'Sentadillas', descripcion: 'Ejercicio para piernas y gluteos'},  
14     {id: '7', nombre: 'Sentadillas', descripcion: 'Ejercicio para piernas y gluteos'},  
15     {id: '8', nombre: 'Sentadillas', descripcion: 'Ejercicio para piernas y gluteos'},  
16     {id: '9', nombre: 'Sentadillas', descripcion: 'Ejercicio para piernas y gluteos'},  
17     {id: '10', nombre: 'Sentadillas', descripcion: 'Ejercicio para piernas y gluteos'},  
18   ];  
19  
20   // SectionList  
21   const contactos = [  
22     {titulo: 'A', data: ['Alejandra', 'Ana la de las tortillas', 'Alicia']},  
23     {titulo: 'M', data: ['Mecanico', 'Mi Ex', 'Mama de mi ex']},  
24     {titulo: 'T', data: ['TheWillyrex', 'Tulio', 'Trevor']},  
25   ]  
26
```



FlatList se usa para mostrar elementos simples y repetitivos

```
JS MenuScreen.js M JS ListScreen.js U X JS ActivityScreen.js U JS ImagenScreen.js JS TextScreen.js
INTRO > screens > JS ListScreen.js > ListScreen
 4 export default function ListScreen() {
26
27   return (
28     <View style={styles.container}>
29       <View style={styles.listContainer}>
30         <Text style={styles.titulo}>Lista de Ejercicios</Text>
31         <FlatList
32           data={ejercicios}
33           renderItem={({ item }) => (
34             <View style={styles.item}>
35               <Text style={styles.nombre}>{item.nombre}</Text>
36               <Text style={styles.descripcion}>{item.descripcion}</Text>
37             </View>
38           )}
39           keyExtractor={item => item.id}
40         />
41       </View>
42     </View>
  )
}
```

SectionList agrupa los elementos por secciones, en este caso por letra inicial del nombre

```
JS MenuScreen.js M JS ListScreen.js U X JS ActivityScreen.js U JS ImagenScreen.js JS TextScreen.js
INTRO > screens > JS ListScreen.js > ListScreen
 4 export default function ListScreen() {
43
44   <View style={styles.listContainer}>
45     <Text style={styles.titulo}>Lista de Contactos</Text>
46     <SectionList
47       sections={contactos}
48       renderItem={({ item }) => (
49         <Text style={styles.item}>{item}</Text>
50       )}
51       renderSectionHeader={({ section }) => (
52         <Text style={styles.header}>{section.titulo}</Text>
53       )}
54     />
55   </View>
56
57 </View>
58
59
60 );
61 }
```



Estos son los estilos utilizados

```
JS MenuScreen.js M   JS ListScreen.js U X   JS ActivityScreen.js U   JS ImagenS
INTRO > screens > JS ListScreen.js > ListScreen
62  const styles = StyleSheet.create({
63    container:{
64      flex:1,
65      padding: 10,
66    },
67    listContainer:{
68      flex:1,
69      marginBottom: 32,
70    },
71    titulo:{
72      fontSize: 20,
73      fontWeight: 'bold',
74      textAlign: 'center',
75      margin: 10,
76    },
77    item:{
78      padding: 10,
79      backgroundColor: '#7c80f1ff',
80      marginHorizontal: 10,
81      marginVertical: 10,
82      borderRadius: 5,
83    },
84    header:{
85      fontSize: 18,
86      backgroundColor: '#ffffff',
87      padding: 10,
88      marginTop: 10,
89    },
90    nombre:{
91      fontSize: 16,
92      fontWeight: 'bold',
93      marginBottom: 5,
94    },
95    descripcion:{
96      fontSize: 14,
97      color: '#ffffff',
98    },

```



Universidad Politecnica de Queretaro
Tecnologias de la Informacion e Innovacion Digital
Fecha de entrega:28/10/25 Grupo:TIID211

Conclusion

En esta practica demostramos el uso de FlatList y SectionList para mostrar informacion de manera estructurada, eficiente y organizada