UNIVERSIDAD NACIONAL Y AUTÓNOMA DE MÉXICO

FACULTAD DE CONTADURÍA Y ADMINISTRACIÓN

“FlatList”

:: UNIDAD 4: Programación para Apple iOS

:: ACTIVIDAD COMPLEMENTARIA 2

MATERIA: **.:** PROGRAMACIÓN DE DISPOSITIVOS MÓVILES **:.**

CLAVE: 1668

LICENCIATURA EN INFORMÁTICA

PLAN 2012

REALIZO: Emmanuel Alejandro Pérez Hernández

No. 423142118

Grupo: 8691

ASESOR: MARTINEZ FERNANDEZ JUAN MANUEL

martes, 4 de noviembre de 2025

UNIDAD 4. Actividad Complementaria 2

a) Investiga para qué sirve el componente FlatList de Expo.

1. Renderizado eficiente de listas grandes

- Renderiza solo los elementos visibles en pantalla (virtualización)
- Optimiza el rendimiento con listas de miles de elementos
- No carga todos los elementos en memoria a la vez

2. Características principales

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

<FlatList
  data={arrayDeDatos}
  renderItem={({ item }) => <Elemento item={item} />}
  keyExtractor={item => item.id}
/>
```

3. Propiedades esenciales

- data: Array de elementos a mostrar
- renderItem: Función que renderiza cada elemento
- keyExtractor: Extrae keys únicas para cada elemento
- horizontal: Lista horizontal si es true

4. Funcionalidades avanzadas

```
<FlatList
  data={data}
  renderItem={renderItem}
  keyExtractor={item => item.id}

  // Scroll y paginación
  onEndReached={cargarMasDatos}
  onRefresh={refrescarDatos}
  refreshing={cargando}

  // Personalización
```

```
  ListHeaderComponent={Header}
  ListFooterComponent={Footer}
  ItemSeparatorComponent={Separador}
  ListEmptyComponent={ListaVacia}

  // Performance
  initialNumToRender={10}
  maxToRenderPerBatch={10}
  windowSize={5}
/>
```

5. Ventajas sobre ScrollView

FLATLIST	SCROLLVIEW
VIRTUALIZACIÓN	Renderiza todo
EFICIENTE CON MUCHOS DATOS	Lento con muchos datos
LAZY LOADING	Carga completa
OPTIMIZADO PARA LISTAS	Para contenido estático

6. Casos de uso comunes

- Listas de contactos
- Feeds de noticias/social media
- Productos en e-commerce
- Mensajes en chat
- Historial de transacciones

7. Mejores prácticas

- Keys únicas y estables
- Usar initialNumToRender para controlar elementos iniciales
- Implementar onEndReached para paginación infinita
- Usar ListEmptyComponent para estados vacíos
- Optimizar renderItem con React.memo si es necesario

b) Modifica tu app móvil para incluir un FlatList con datos simulados.

Codigo de la App en Snack.Expo con FlatList

```
import React from 'react';
import {
  StyleSheet,
  Text,
  View,
  FlatList,
  SafeAreaView,
  StatusBar,
  TouchableOpacity
} from 'react-native';

// Datos simulados para la FlatList
const datosTareas = [
  {
    id: '1',
    titulo: 'Estudiar Componentes iOS',
    descripcion: 'Aprender sobre FlatList, ScrollView, etc.',
    prioridad: 'Alta',
    completada: false
  },
  {
    id: '2',
    titulo: 'Practicar con Expo',
    descripcion: 'Crear aplicaciones móviles multiplataforma',
    prioridad: 'Media',
    completada: true
  },
  {
    id: '3',
    titulo: 'Diseñar Interfaces',
    descripcion: 'Implementar UI/UX para apps móviles',
    prioridad: 'Alta',
    completada: false
  },
  {
    id: '4',
    titulo: 'Integrar APIs',
    descripcion: 'Conectar app móvil con backend Node.js',
    prioridad: 'Media',
    completada: false
  },
  {
    id: '5',
    titulo: 'Probar en Dispositivos',
    descripcion: 'Testing en iOS simulator y Android',
    prioridad: 'Baja',
```

```
    completada: true
  },
  {
    id: '6',
    titulo: 'Publicar en App Store',
    descripcion: 'Proceso de subir app a tiendas',
    prioridad: 'Media',
    completada: false
  }
];

// Componente para cada item de la lista
const ItemTarea = ({ titulo, descripcion, prioridad, completada }) => (
  <TouchableOpacity style={[
    styles.item,
    completada ? styles.completada : styles.pendiente
  ]}>
    <View style={styles.contenidoItem}>
      <Text style={styles.titulo}>{titulo}</Text>
      <Text style={styles.descripcion}>{descripcion}</Text>
      <View style={styles.footerItem}>
        <Text style={[
          styles.prioridad,
          prioridad === 'Alta' ? styles.prioridadAlta :
          prioridad === 'Media' ? styles.prioridadMedia :
          styles.prioridadBaja
        ]}>
          {prioridad}
        </Text>
        <Text style={styles.estado}>
          {completada ? '✅ Completada' : '⬜ Pendiente'}
        </Text>
      </View>
    </View>
  </TouchableOpacity>
);

// Componente principal
export default function App() {
  const renderItem = ({ item }) => (
    <ItemTarea
      titulo={item.titulo}
      descripcion={item.descripcion}
      prioridad={item.prioridad}
      completada={item.completada}
    />
  );

  return (
    <SafeAreaView style={styles.container}>
```

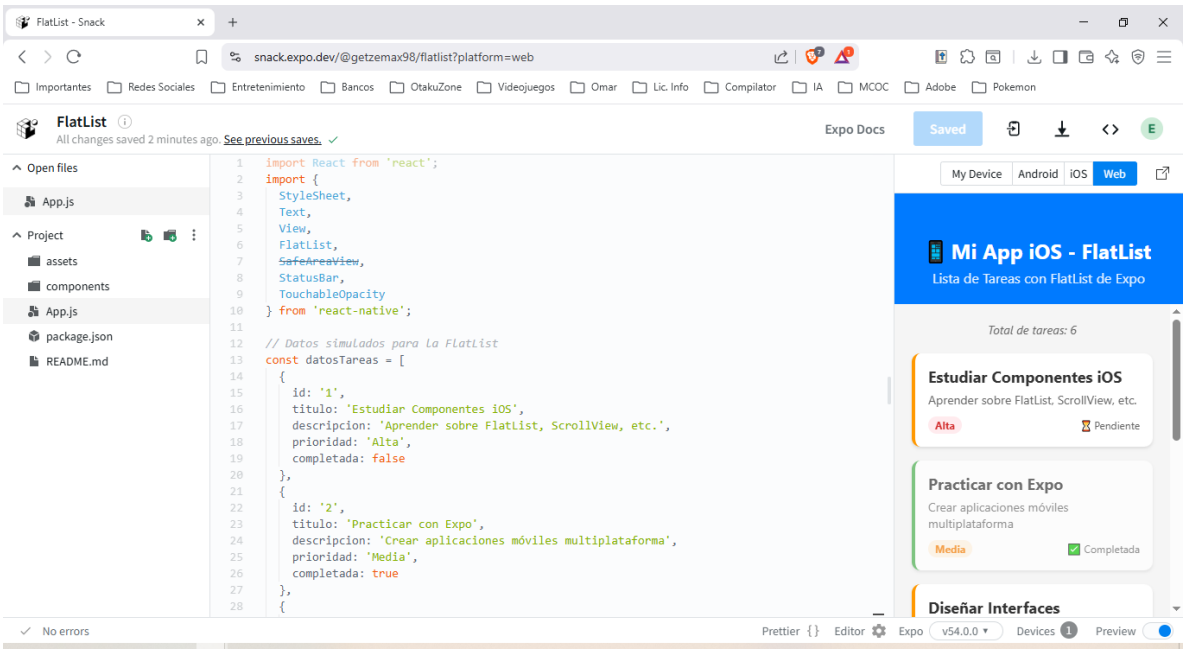
```
      <StatusBar barStyle="dark-content" />

      <View style={styles.header}>
        <Text style={styles.tituloApp}>📱 Mi App iOS - FlatList</Text>
        <Text style={styles.subtitulo}>
          Lista de Tareas con FlatList de Expo
        </Text>
      </View>

      <FlatList
        data={datosTareas}
        renderItem={renderItem}
        keyExtractor={item => item.id}
        style={styles.lista}
        showsVerticalScrollIndicator={true}
        ListEmptyComponent={
          <Text style={styles.listaVacia}>No hay tareas disponibles</Text>
        }
        ListHeaderComponent={
          <Text style={styles.contador}>
            Total de tareas: {datosTareas.length}
          </Text>
        }
      />
    </SafeAreaView>
  );
}

// Estilos
const styles = StyleSheet.create({
  container: {
    flex: 1,
    backgroundColor: '#f5f5f5',
  },
  header: {
    backgroundColor: '#007AFF',
    padding: 20,
    paddingTop: 50,
  },
  tituloApp: {
    fontSize: 24,
    fontWeight: 'bold',
    color: 'white',
    textAlign: 'center',
  },
  subtitulo: {
    fontSize: 16,
    color: 'white',
    textAlign: 'center',
    marginTop: 5,
```

```
  opacity: 0.9,
},
lista: {
  flex: 1,
  padding: 10,
},
item: {
  backgroundColor: 'white',
  padding: 15,
  marginVertical: 8,
  marginHorizontal: 10,
  borderRadius: 10,
  shadowColor: '#000',
  shadowOffset: {
    width: 0,
    height: 2,
  },
  shadowOpacity: 0.1,
  shadowRadius: 3,
  elevation: 3,
},
completada: {
  opacity: 0.7,
  borderLeftWidth: 4,
  borderLeftColor: '#4CAF50',
},
pendiente: {
  borderLeftWidth: 4,
  borderLeftColor: '#FF9800',
},
contenidoItem: {
  flex: 1,
},
titulo: {
  fontSize: 18,
  fontWeight: 'bold',
  color: '#333',
  marginBottom: 5,
},
descripcion: {
  fontSize: 14,
  color: '#666',
  marginBottom: 10,
},
footerItem: {
  flexDirection: 'row',
  justifyContent: 'space-between',
  alignItems: 'center',
},
prioridad: {
```

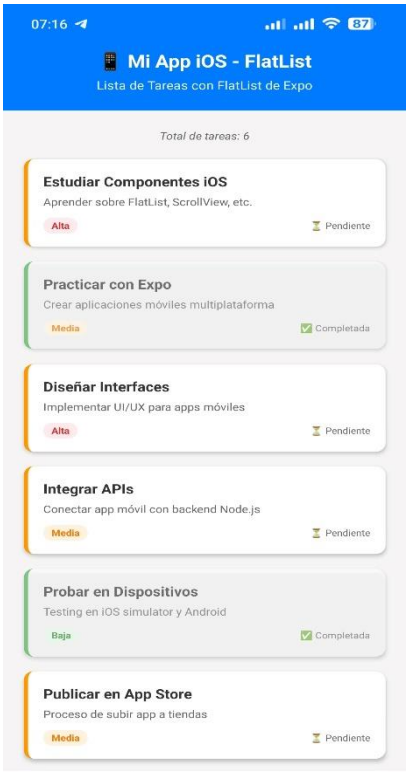
```
    fontSize: 12,
    fontWeight: 'bold',
    paddingHorizontal: 8,
    paddingVertical: 2,
    borderRadius: 10,
  },
  prioridadAlta: {
    backgroundColor: '#FFEBEE',
    color: '#D32F2F',
  },
  prioridadMedia: {
    backgroundColor: '#FFF3E0',  // ✓ COMILLA CORREGIDA AQUÍ
    color: '#F57C00',
  },
  prioridadBaja: {
    backgroundColor: '#E8F5E8',
    color: '#388E3C',
  },
  estado: {
    fontSize: 12,
    color: '#666',
  },
  contador: {
    textAlign: 'center',
    fontSize: 14,
    color: '#666',
    marginVertical: 10,
    fontStyle: 'italic',
  },
  listaVacia: {
    textAlign: 'center',
    fontSize: 16,
    color: '#999',
    marginTop: 50,
  },
});
```

c) **Agrega capturas de pantalla, explicaciones y la liga de tu Snack en un PDF y súbelo.**

https://snack.expo.dev/@getzemax98/flatlist



Código de la app móvil modificado para incluir un FlatList con datos simulados



I. REFERENCIAS BIBLIOGRÁFICAS

crowdbotics. (2021, October 6). *How to Add a Search Bar in a FlatList in React Native Apps — Crowdbotics*. Crowdbotics. https://crowdbotics.com/posts/blog/add-search-bar-flatlist-react-native-apps/