Módulo 4: Elementos Visuales y Flujo de Interacción entre Pantallas

Lección: Selección Binaria y Opciones Múltiples en Flutter

Objetivos de la Lección

- Comprender el uso del widget `Checkbox` para selección múltiple de opciones binarias.
- Aprender a implementar el widget `RadioButton` para selección de una sola opción dentro de un grupo de opciones excluyentes.
- Familiarizarse con las propiedades principales y el manejo de eventos en ambos widgets.
- Implementar ejemplos básicos en Flutter para aplicar `Checkbox` y `RadioButton`.

Introducción de la Lección

En Flutter, los widgets `Checkbox` y `RadioButton` son fundamentales para crear interfaces de usuario que permitan seleccionar opciones. El widget `Checkbox` facilita la selección múltiple, permitiendo al usuario activar o desactivar varias opciones binarias. Por otro lado, el widget `RadioButton` permite seleccionar solo una opción de un conjunto, ideal para preguntas de opción única. En esta lección, explicaremos cómo usar estos widgets, sus propiedades y cómo manejar sus eventos de selección.

Checkbox: Selección Binaria y Opciones Múltiples

El widget Checkbox permite al usuario realizar selecciones múltiples, activando o desactivando varias opciones de manera independiente. Este widget es útil para configuraciones o preferencias donde el usuario puede elegir múltiples opciones.

Propiedades Principales de Checkbox:

- **value**: Determina si la casilla de verificación está activada (`true`) o desactivada (`false`).
- **onChanged**: Callback que se ejecuta cada vez que cambia el estado de la casilla.
- **activeColor**: Define el color de la casilla cuando está seleccionada.
- **checkColor**: Configura el color de la marca de verificación.

Ejemplo Simple de Checkbox:

En este ejemplo, configuramos tres casillas de verificación que representan diferentes preferencias de usuario. Cada vez que el usuario selecciona o deselecciona una opción, el estado de la casilla cambia y el valor se imprime en la consola.

```
import 'package:flutter/material.dart';

void main() {
  runApp(MyApp());
}

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
```

```
      home: CheckboxExample(),
    );
  }
}

class CheckboxExample extends StatefulWidget {
  @override
  _CheckboxExampleState createState() =>
_CheckboxExampleState();
}

class _CheckboxExampleState extends State<CheckboxExample> {
  bool isOption1Selected = false;
  bool isOption2Selected = false;
  bool isOption3Selected = false;

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Ejemplo de Checkbox'),
      ),
      body: Column(
        children: [
          CheckboxListTile(
            title: Text('Opción 1'),
            value: isOption1Selected,
            onChanged: (bool? value) {
              setState(() {
                isOption1Selected = value ?? false;
```

```
            });
            print('Opción 1: $isOption1Selected');
          },
        ),
        CheckboxListTile(
          title: Text('Opción 2'),
          value: isOption2Selected,
          onChanged: (bool? value) {
            setState(() {
              isOption2Selected = value ?? false;
            });
            print('Opción 2: $isOption2Selected');
          },
        ),
        CheckboxListTile(
          title: Text('Opción 3'),
          value: isOption3Selected,
          onChanged: (bool? value) {
            setState(() {
              isOption3Selected = value ?? false;
            });
            print('Opción 3: $isOption3Selected');
          },
        ),
      ],
    ),
  );
 }
}
```

En este ejemplo:

- Usamos `CheckboxListTile`, una combinación de `Checkbox` y `ListTile`, que facilita la disposición de etiquetas y casillas.
- La propiedad `value` define si la casilla está activada.
- `onChanged` actualiza el estado de cada casilla y muestra el resultado en la consola.

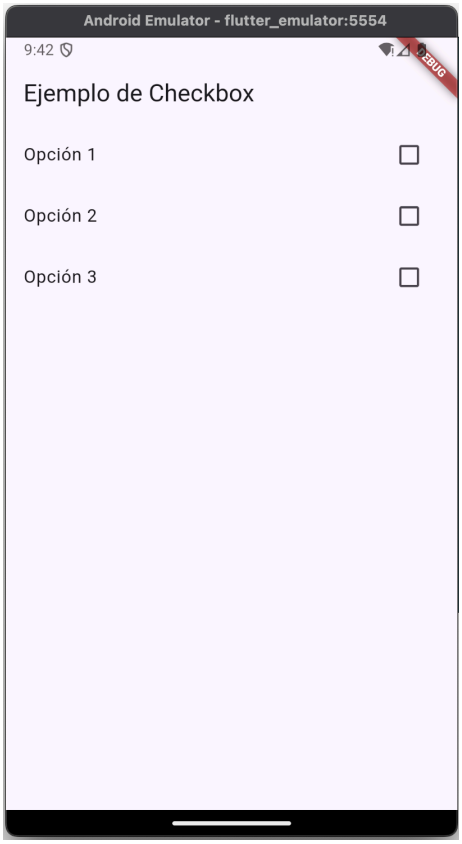

Imagen de la pantalla que incluye el Widget `Checkbox`.

Los CheckBox son listas que permiten seleccionar varias de las opciones presentadas.

Creado por Javier A. Dastas (2024)

`RadioButton`: Selección de Opción Única

El widget `RadioButton` permite al usuario seleccionar solo una opción de un conjunto de opciones. Esto es ideal para preguntas de opción única, donde solo se puede elegir una respuesta entre varias opciones.

Propiedades Principales de `RadioButton`:

- **value**: Identifica el valor de cada opción del grupo.
- **groupValue**: Determina el valor del botón de radio actualmente seleccionado en el grupo.
- **onChanged**: Callback que se ejecuta cuando se selecciona una opción.

Ejemplo Simple de `RadioButton`:

En este ejemplo, configuramos un grupo de botones de radio que permite al usuario seleccionar solo un tipo de suscripción.

```
import 'package:flutter/material.dart';

void main() {
  runApp(MyApp());
}

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: RadioButtonExample(),
    );
  }
}
```

```
class RadioButtonExample extends StatefulWidget {
  @override
  _RadioButtonExampleState createState() =>
_RadioButtonExampleState();
}

class _RadioButtonExampleState extends State<RadioButtonExample>
{
  String? selectedSubscription = 'Básica';

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Ejemplo de RadioButton'),
      ),
      body: Column(
        children: [
          RadioListTile<String>(
            title: Text('Suscripción Básica'),
            value: 'Básica',
            groupValue: selectedSubscription,
            onChanged: (String? value) {
              setState(() {
                selectedSubscription = value;
              });
              print('Suscripción seleccionada:
$selectedSubscription');
            },
```

```
          ),
          RadioListTile<String>(
            title: Text('Suscripción Estándar'),
            value: 'Estándar',
            groupValue: selectedSubscription,
            onChanged: (String? value) {
              setState(() {
                selectedSubscription = value;
              });
              print('Suscripción seleccionada:
$selectedSubscription');
            },
          ),
          RadioListTile<String>(
            title: Text('Suscripción Premium'),
            value: 'Premium',
            groupValue: selectedSubscription,
            onChanged: (String? value) {
              setState(() {
                selectedSubscription = value;
              });
              print('Suscripción seleccionada:
$selectedSubscription');
            },
          ),
        ],
      ),
    );
  }
}
```

En este ejemplo:

- Usamos `RadioListTile`, que permite combinar un `RadioButton` con una etiqueta y simplifica la disposición.
- `value` asigna un valor único a cada opción.
- `groupValue` controla cuál opción está seleccionada en el grupo.
- `onChanged` actualiza el valor seleccionado en el grupo, permitiendo solo una selección a la vez.

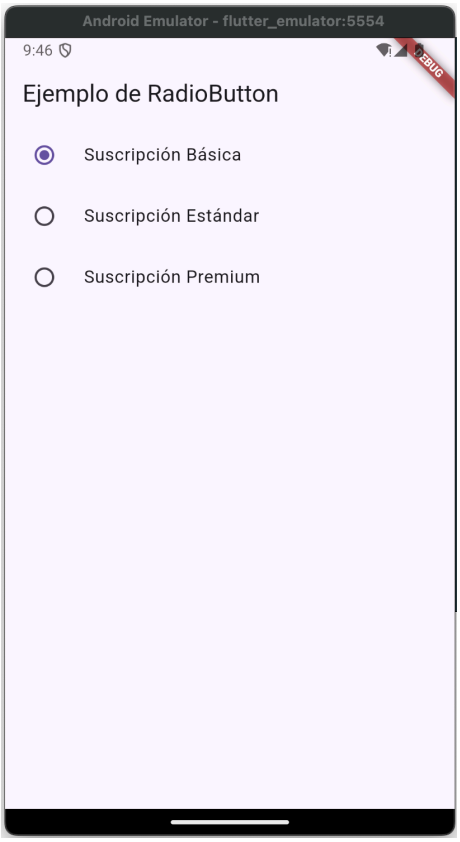

Imagen de la pantalla que incluye el Widget `Checkbox`.

Los RadioButton solo permiten seleccionar una de las opciones presentadas.

Creado por Javier A. Dastas (2024)

Relación con Otros Conceptos

El uso de `Checkbox` y `RadioButton` es fundamental en el diseño de formularios en Flutter, permitiendo a los desarrolladores implementar controles de selección múltiples o únicos. Estos widgets son útiles en la recolección de datos y preferencias del usuario, y pueden integrarse con la gestión de estado y la validación de formularios en Flutter, temas importantes en el diseño de aplicaciones centradas en la experiencia de usuario (UX).

Resumen de la Lección

En esta lección, aprendimos a utilizar los widgets `Checkbox` y `RadioButton` en Flutter. Vimos cómo `Checkbox` permite la selección múltiple en opciones binarias y cómo `RadioButton` facilita la selección única dentro de un grupo. Ambos widgets son altamente personalizables y esenciales para construir interfaces de usuario interactivas y efectivas.

Actividad de la Lección

En esta actividad, aplicaremos lo aprendido sobre los widgets `Checkbox` y `RadioButton` en Flutter para construir una aplicación que permita al usuario realizar selecciones múltiples y únicas, ofreciendo una experiencia de selección personalizada. Con esta actividad, los estudiantes podrán comprender cómo configurar preferencias de usuario y gestionar opciones exclusivas mediante la combinación de estos dos widgets.

La actividad tiene dos partes:

1. **Selección Múltiple con `Checkbox`**: Crear una lista de preferencias donde el usuario pueda seleccionar varias opciones, como deportes, música y películas. Al final, imprime en la consola las opciones seleccionadas por el usuario. Esta parte ayudará a reforzar el uso de `Checkbox` para permitir la selección de múltiples opciones en un entorno interactivo.
2. **Selección Única con `RadioButton`**: Configurar un grupo de opciones de planes de membresía (por ejemplo, Básico, Estándar y Premium) donde el usuario solo pueda seleccionar una opción. Esta parte permite a los estudiantes trabajar con el concepto de selección única en listas de opciones excluyentes, utilizando el widget `RadioButton`.

Entrega de la Actividad:

- Desarrolla un documento en formato PDF que incluya el código desarrollado para esta actividad, capturas de pantalla de la aplicación en ejecución, y una breve descripción de cómo cada widget ayuda a mejorar la experiencia de usuario en la selección de opciones.

Con esta actividad, se busca consolidar el conocimiento en la creación de interfaces interactivas y flexibles, donde el usuario pueda hacer selecciones múltiples o únicas de manera intuitiva y sencilla.