Módulo 4: Elementos Visuales y Flujo de Interacción entre Pantallas

Lección: Configuración de Preferencias y Alternativas en Flutter: Uso de Switch y DropdownButton

Objetivos de la Lección

- Comprender el uso del widget `Switch` para configuraciones de encendido/apagado en preferencias de usuario.
- Aprender a implementar el widget `DropdownButton` para la selección de una opción de una lista desplegable.
- Conocer las propiedades principales y personalización de cada widget.
- Aplicar ejemplos prácticos y básicos en Flutter para cada uno de los widgets.

Introducción de la Lección

En Flutter, los widgets `Switch` y `DropdownButton` son esenciales para la configuración de preferencias y la selección de opciones. El widget `Switch` se utiliza para alternar entre estados (encendido/apagado), ideal para configuraciones rápidas, como activar o desactivar notificaciones. Por otro lado, el `DropdownButton` permite seleccionar una opción de una lista desplegable, siendo útil para configuraciones con múltiples alternativas. En esta lección, explicaremos cómo usar estos widgets, sus propiedades y funcionalidades, además de ejemplos de código sencillos.

Switch: Configuración de Encendido/Apagado

El widget Switch permite al usuario alternar entre dos estados: encendido y apagado. Es comúnmente usado en aplicaciones móviles para activar o desactivar preferencias como notificaciones, modo oscuro o sonido.

Propiedades Principales de Switch:

- **value**: Controla el estado actual del interruptor (encendido `true` o apagado `false`).
- **onChanged**: Callback que se ejecuta cuando el usuario cambia el estado del Switch.
- **activeColor**: Define el color del interruptor cuando está en la posición de encendido.
- **inactiveThumbColor** y **inactiveTrackColor**: Configuran el color de la perilla y la pista cuando el Switch está apagado.

Ejemplo Simple de Switch:

En este ejemplo, configuramos un Switch que permite al usuario activar o desactivar las notificaciones. Cada vez que el usuario cambia el estado, 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: SwitchExample(),
    );
  }
}

class SwitchExample extends StatefulWidget {
  @override
  _SwitchExampleState createState() => _SwitchExampleState();
}

class _SwitchExampleState extends State<SwitchExample> {
  bool isNotificationEnabled = false;

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Ejemplo de Switch'),
      ),
      body: Center(
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: [
            Text('Notificaciones'),
            Switch(
              value: isNotificationEnabled,
              onChanged: (bool value) {
                setState(() {
```

```
                  isNotificationEnabled = value;
                });
                print("Notificaciones: ${isNotificationEnabled ?
'Activadas' : 'Desactivadas'}");
              },
              activeColor: Colors.blue,
            ),
          ],
        ),
      ),
    );
  }
}
```

En este ejemplo:

- **value**: Controla si el interruptor está activado o desactivado.
- **onChanged**: Actualiza el estado del `Switch` y muestra el valor en la consola.
- **activeColor**: Configura el color del `Switch` cuando está activado.

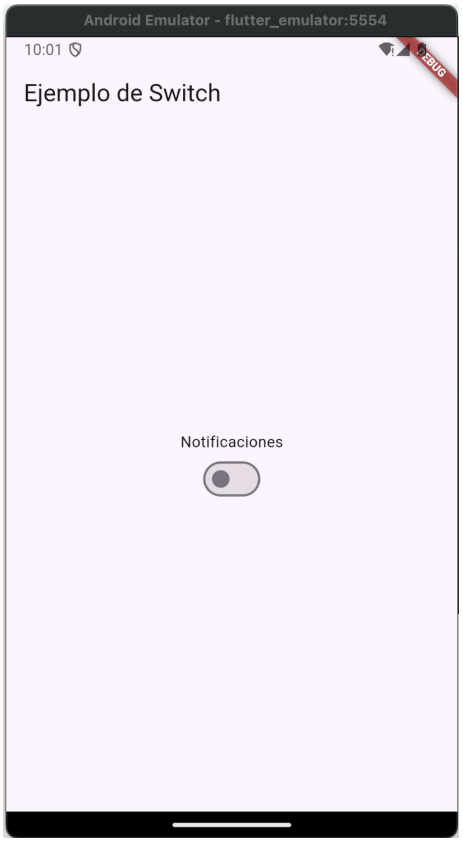


Imagen de la pantalla que incluye el Widget `Switch`.

Creado por Javier A. Dastas (2024)

`DropdownButton`: Selección en Lista Desplegable

El widget `DropdownButton` permite al usuario seleccionar una opción de una lista desplegable, ideal para configuraciones donde solo una alternativa puede estar activa (por ejemplo, seleccionar un idioma o una categoría).

Propiedades Principales de `DropdownButton`:

- **value**: Controla la opción seleccionada actual en el menú desplegable.
- **onChanged**: Callback que se ejecuta cuando el usuario selecciona una nueva opción.
- **items**: Lista de elementos que el `DropdownButton` muestra como opciones.
- **hint**: Texto que aparece cuando no hay ninguna opción seleccionada.

Ejemplo Simple de `DropdownButton`:

En este ejemplo, configuramos un `DropdownButton` que permite seleccionar un tema de color para la aplicación. Cada vez que el usuario selecciona una opción, el valor se imprime en la consola.

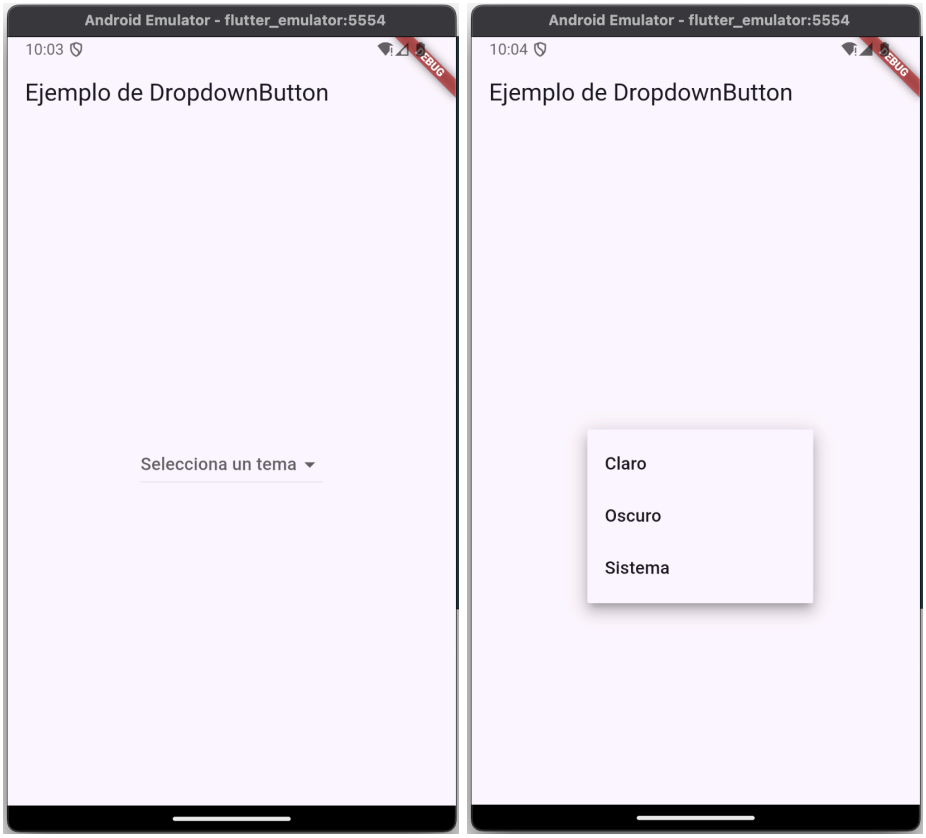
```
import 'package:flutter/material.dart';

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

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

class DropdownExample extends StatefulWidget {
  @override
  _DropdownExampleState createState() =>
_DropdownExampleState();
}

class _DropdownExampleState extends State<DropdownExample> {
  String? selectedTheme;
  final List<String> themes = ['Claro', 'Oscuro', 'Sistema'];
```

```
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Ejemplo de DropdownButton'),
      ),
      body: Center(
        child: DropdownButton<String>(
          value: selectedTheme,
          hint: Text('Selecciona un tema'),
          items: themes.map((String theme) {
            return DropdownMenuItem<String>(
              value: theme,
              child: Text(theme),
            );
          }).toList(),
          onChanged: (String? newValue) {
            setState(() {
              selectedTheme = newValue;
            });
            print("Tema seleccionado: $selectedTheme");
          },
        ),
      ),
    );
  }
}
```

En este ejemplo:

- **items**: Define la lista de opciones disponibles en el `DropdownButton`.

- **value**: Almacena la opción seleccionada actualmente.
- **onChanged**: Cambia el valor del tema y muestra la selección en la consola.
- **hint**: Texto que aparece si ninguna opción ha sido seleccionada aún.



Imágenes de la pantalla que incluye el Widget `DropdownButton`.

Creado por Javier A. Dastas (2024)

Relación con Otros Conceptos

Los widgets `Switch` y `DropdownButton` son fundamentales en la creación de interfaces de usuario que permiten a los usuarios personalizar sus preferencias y opciones en aplicaciones móviles. Estos widgets se integran comúnmente en formularios y configuraciones, proporcionando una experiencia interactiva y personalizada. En un flujo de trabajo de Ingeniería de Software, su implementación es clave para mejorar la experiencia de usuario (UX) al facilitar la personalización de la aplicación según las preferencias individuales.

Resumen de la Lección

En esta lección, aprendimos a utilizar los widgets `Switch` y `DropdownButton` en Flutter. Vimos cómo `Switch` permite alternar entre estados binarios, útil para configuraciones rápidas, y cómo `DropdownButton` facilita la selección de una opción de una lista desplegable. Ambos widgets son esenciales en la creación de aplicaciones interactivas y centradas en el usuario.

Actividad de la Lección

En esta actividad, aplicaremos lo aprendido sobre los widgets `Switch` y `DropdownButton` en Flutter para crear una aplicación que permita al usuario configurar preferencias de manera intuitiva y seleccionar opciones de listas desplegables. Esta actividad ayuda a comprender cómo configurar elementos de personalización en una interfaz de usuario, ofreciendo controles de encendido/apagado y selección de opciones.

La actividad tiene dos partes:

1. **Configuración de Preferencias de Usuario con `Switch`**:
    - Crear un `Switch` que permita activar o desactivar el modo oscuro en una aplicación.
    - Muestra el estado actual del `Switch` (activado o desactivado) en la pantalla, actualizándose cada vez que el usuario interactúa con el widget.
    - Esta parte ayuda a reforzar el uso de `Switch` en la configuración de opciones binarias dentro de una aplicación.
2. **Selección de Opciones con `DropdownButton`**:
    - Implementa un `DropdownButton` que permita al usuario seleccionar un idioma entre varias opciones (por ejemplo, Inglés, Español, Francés).
    - Muestra el idioma seleccionado en pantalla, permitiendo que el usuario vea su selección reflejada en la interfaz.
    - Esta parte permite trabajar con listas desplegables y comprender cómo manejar el cambio de estado en una lista de opciones únicas.

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 `Switch` y `DropdownButton` facilitan la configuración de preferencias y opciones en aplicaciones Flutter.

Con esta actividad, se busca consolidar el conocimiento en la creación de interfaces interactivas que combinan configuraciones rápidas y selecciones de opciones para mejorar la experiencia de usuario.