Lección: Uso del Widget Divider en Flutter

Objetivos de la Lección

- Comprender el propósito y funcionamiento del widget **Divider** en Flutter.
- Identificar las propiedades principales del widget **Divider** y cómo se pueden personalizar.
- Implementar ejemplos prácticos para aprender a usar **Divider** y mejorar la organización visual de la interfaz de usuario en aplicaciones.

¿Qué es el Widget Divider?

El widget **Divider** en Flutter se utiliza para crear una **línea horizontal o vertical** que ayuda a separar o dividir visualmente elementos en la interfaz de usuario. Es un elemento muy útil cuando se necesita organizar contenido en secciones distintas, como en una lista de configuración, menú, o cualquier otra pantalla donde se desee agrupar elementos visualmente.

El widget **Divider** es altamente personalizable y puede ajustarse en términos de grosor, color y longitud para adaptarse al diseño de la aplicación.

Propiedades Principales del Widget Divider

El widget **Divider** ofrece varias propiedades que permiten personalizar su apariencia. Las propiedades más comunes incluyen:

1. **color**: Permite cambiar el color de la línea de separación.
2. **thickness**: Controla el grosor del divider.
3. **indent**: Agrega un espacio en píxeles al inicio de la línea para crear una separación inicial.
4. **endIndent**: Agrega un espacio en píxeles al final de la línea para crear una separación final.

Estas propiedades permiten ajustar el **Divider** para que se integre de manera uniforme en el diseño de la aplicación.

Ejemplo Básico de Divider en una Column

El siguiente ejemplo muestra cómo usar el widget **Divider** para separar elementos de texto en una **Column**.

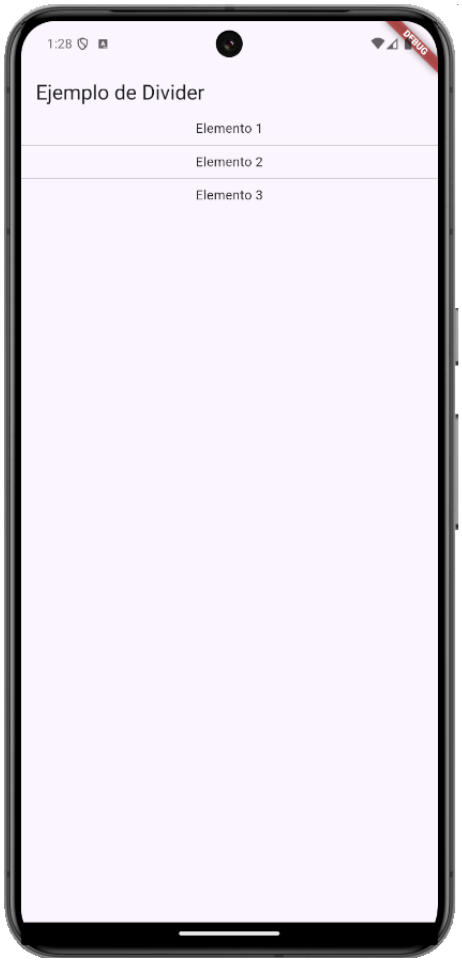
```
import 'package:flutter/material.dart';

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

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: Scaffold(
        appBar: AppBar(
          title: Text('Ejemplo de Divider'),
        ),
        body: Column(
          children: <Widget>[
            Text('Elemento 1'),
            Divider(),
            Text('Elemento 2'),
            Divider(),
            Text('Elemento 3'),
```

```
            ],
          ),
        ),
      );
    }
}
```

Explicación del Código:

- **Divider**: Se coloca entre los widgets `Text` para separarlos visualmente.
- **Column**: Organiza los elementos en una disposición vertical.

Este código crea tres elementos de texto, separados por un `Divider` simple. Esto permite visualizar cada elemento de forma más organizada.



Captura de la pantalla del ejemplo presentado con el Widget Divider.
Creado por Javier A. Dastas (2024)

Personalización de Divider

1. Cambiar el Color del Divider

Podemos cambiar el color del **Divider** usando la propiedad `color`. Esto es útil para adaptar el divider al esquema de colores de la aplicación.

```
Divider(
  color: Colors.blue,
)
```

Ejemplo Completo con Color Personalizado:

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

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

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: Scaffold(
        appBar: AppBar(
          title: Text('Divider con Color'),
        ),
        body: Column(
          children: <Widget>[
            Text('Elemento 1'),
            Divider(color: Colors.blue),
            Text('Elemento 2'),
            Divider(color: Colors.blue),
            Text('Elemento 3'),
          ],
        ),
      ),
    );
  }
```

```
}
```

Explicación del Código:

- **color: Colors.blue**: Cambia el color del divider a azul para darle un estilo distintivo.

2. Cambiar el Grosor del Divider

La propiedad `thickness` permite ajustar el grosor del divider, lo cual es útil si se necesita una línea más gruesa o delgada.

```
Divider(
  color: Colors.red,
  thickness: 2.0,
)
```

Ejemplo Completo con Grosor Personalizado:

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

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

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: Scaffold(
        appBar: AppBar(
          title: Text('Divider con Grosor Personalizado'),
```

```
        ),
        body: Column(
          children: <Widget>[
            Text('Elemento 1'),
            Divider(color: Colors.red, thickness: 2.0),
            Text('Elemento 2'),
            Divider(color: Colors.red, thickness: 2.0),
            Text('Elemento 3'),
          ],
        ),
      ),
    );
  }
}
```

Explicación del Código:

- **thickness: 2.0**: Cambia el grosor del divider a 2.0 píxeles, haciéndolo más visible.

3. Espaciado Inicial y Final con indent y endIndent

Las propiedades `indent` y `endIndent` permiten agregar espacio al inicio y al final del **Divider**. Esto es útil cuando se desea un divider más corto que el ancho total de la pantalla.

```
Divider(
  color: Colors.green,
  thickness: 1.5,
  indent: 20.0,
  endIndent: 20.0,
```

```
)
```

Ejemplo Completo con Espaciado Inicial y Final:

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

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

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: Scaffold(
        appBar: AppBar(
          title: Text('Divider con Espaciado'),
        ),
        body: Column(
          children: <Widget>[
            Text('Elemento 1'),
            Divider(
              color: Colors.green,
              thickness: 1.5,
              indent: 20.0,
              endIndent: 20.0,
            ),
            Text('Elemento 2'),
            Divider(
```

```
              color: Colors.green,
              thickness: 1.5,
              indent: 20.0,
              endIndent: 20.0,
            ),
            Text('Elemento 3'),
          ],
        ),
      ),
    );
  }
}
```

Explicación del Código:

- **indent: 20.0** y **endIndent: 20.0**: Crea espacio al inicio y al final del divider, haciendo que la línea no ocupe todo el ancho de la pantalla.

Ejemplo Completo Combinando Propiedades

En este ejemplo, combinamos todas las propiedades del **Divider** para personalizar su apariencia en una lista de texto.

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

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

class MyApp extends StatelessWidget {
```

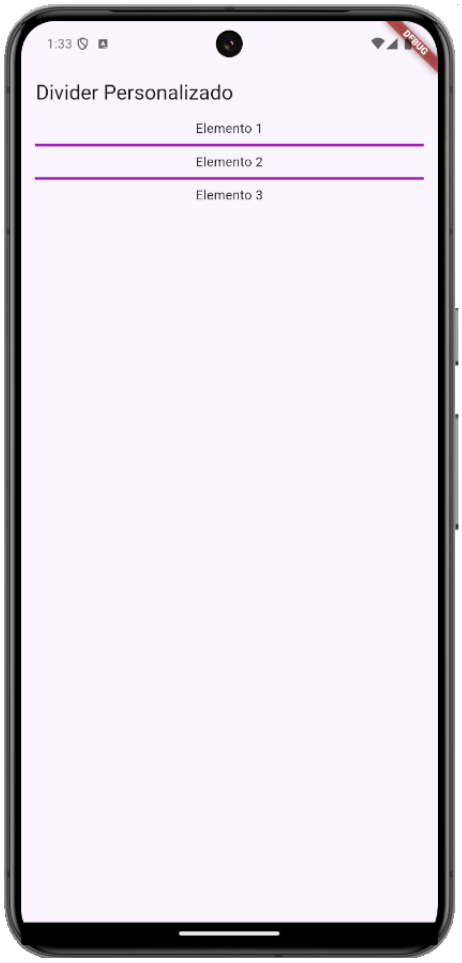
```
@override
Widget build(BuildContext context) {
  return MaterialApp(
    home: Scaffold(
      appBar: AppBar(
        title: Text('Divider Personalizado'),
      ),
      body: Column(
        children: <Widget>[
          Text('Elemento 1'),
          Divider(
            color: Colors.purple,
            thickness: 3.0,
            indent: 15.0,
            endIndent: 15.0,
          ),
          Text('Elemento 2'),
          Divider(
            color: Colors.purple,
            thickness: 3.0,
            indent: 15.0,
            endIndent: 15.0,
          ),
          Text('Elemento 3'),
        ],
      ),
    ),
```

```
      );
    }
}
```

Explicación del Código:

- **color: Colors.purple**: Cambia el color del divider a morado.
- **thickness: 3.0**: Cambia el grosor del divider a 3 píxeles.
- **indent** y **endIndent**: Agrega espacio a ambos lados del divider, dándole una apariencia más estilizada.

Este ejemplo muestra cómo se pueden combinar las propiedades para lograr un divider que se adapte al estilo y diseño de la aplicación.



Captura de la pantalla del ejemplo presentado combinando múltiples propiedades del Widget Divider. Creado por Javier A. Dastas (2024)

Usando Divider en un Row para Separación Vertical

Aunque el widget **Divider** está diseñado para dividir horizontalmente, se puede simular una línea vertical usándolo en una Row y ajustando su altura.

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


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

```
class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: Scaffold(
        appBar: AppBar(
          title: Text('Divider Vertical'),
        ),
        body: Row(
          mainAxisAlignment: MainAxisAlignment.center,
          children: <Widget>[
            Text('Elemento A'),
            VerticalDivider(
              color: Colors.black,
              thickness: 1.5,
              width: 20, // Espacio alrededor del Divider
            ),
            Text('Elemento B'),
          ],
        ),
      ),
    );
  }
}
```

Explicación del Código:

- **VerticalDivider**: Crea una línea vertical para dividir los elementos en una disposición horizontal (`Row`).
- **width**: Controla el espacio alrededor del divider vertical.

Captura de la pantalla del ejemplo presentado combinando el Widget Row con el Widget Divider para crear una división vertical.
Creado por Javier A. Dastas (2024)

Resumen de la Lección

En esta lección, hemos aprendido a utilizar el widget **Divider** en Flutter para dividir y organizar visualmente elementos en la interfaz de usuario. Exploramos cómo personalizar el **Divider** utilizando propiedades como `color`, `thickness`, `indent`, y

`endIndent` para ajustar su apariencia. También vimos cómo crear un **VerticalDivider** en una `Row` para dividir elementos de forma vertical.

El widget **Divider** es una herramienta sencilla pero muy útil ya que permite mejorar la claridad visual en aplicaciones, separando elementos de manera efectiva para una mejor organización de la interfaz.

Actividad de la Lección

Esta actividad te ayudará a aplicar los conceptos aprendidos sobre el uso de **Divider** y **VerticalDivider**, mejorando sus habilidades para organizar la interfaz en aplicaciones Flutter.

Instrucciones:

1. Crea una aplicación en Flutter que muestre cinco elementos de texto en una `Column`, separados por un `Divider` de color azul, con un grosor de 2 píxeles y un espaciado inicial y final de 30 píxeles.
2. Implementa una `Row` que muestre tres textos con un `VerticalDivider` negro, y ajusta el grosor y el espacio de los dividers para que queden centrados en la pantalla.
3. Entrega un documento en formato PDF con copia de tu código y copia de imágenes o capturas de pantalla demostrando que tu código funciona.