Lección: Comprendiendo el Widget Text en Flutter

Objetivos de la Lección

- Comprender el propósito y uso del widget **Text** en Flutter.
- Reconocer las propiedades principales del widget **Text** y cómo personalizarlas.
- Implementar ejemplos básicos de widgets **Text**.
- Aplicar estilos y alineaciones al widget **Text** para crear interfaces de usuario claras y atractivas.

Introducción al Widget Text en Flutter

En Flutter, el widget **Text** es uno de los widgets más básicos y utilizados, ya que es el principal responsable de mostrar **texto** en la interfaz de usuario. Este widget permite mostrar una cadena de texto en la pantalla, como títulos, párrafos o etiquetas, y ofrece una gran cantidad de **propiedades personalizables** para ajustar la apariencia y comportamiento del texto.

El widget **Text** es fundamental en casi cualquier aplicación, ya que la mayoría de las interfaces requieren algún tipo de texto. Flutter, además, ofrece control sobre aspectos como el **estilo del texto**, **alineación**, **tamaño**, **color**, y más.

Definición Básica del Widget Text

El widget **Text** se utiliza para mostrar texto simple en la pantalla. El texto es proporcionado como un parámetro y puede ser personalizado usando propiedades como `style`, `textAlign`, y `overflow`, entre otras.

Ejemplo Básico de un Widget Text:

```
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('Lección sobre el Widget Text'),
        ),
        body: Center(
          child: Text(
            '¡Hola, mundo!',
            style: TextStyle(fontSize: 24),
          ),
        ),
      ),
    );
  }
}
```

Explicación del Código:

1. **MaterialApp**: El widget `MaterialApp` envuelve toda la aplicación y proporciona temas y navegación predeterminados.
2. **Scaffold**: El widget `Scaffold` es un contenedor que estructura la pantalla, proporcionando la base para una **barra de aplicación (AppBar)** y el cuerpo de la página.
3. **AppBar**: Muestra un título utilizando un widget `Text` dentro de la barra de la aplicación.
4. **Text**: En el cuerpo de la aplicación, se utiliza el widget `Text` para mostrar el texto "`¡Hola, mundo!`". El estilo del texto se personaliza con un tamaño de fuente de `24`.

Propiedades Clave del Widget Text

El widget **Text** ofrece una serie de propiedades que permiten personalizar cómo se presenta el texto. A continuación, se describen las propiedades más utilizadas junto con ejemplos para ilustrar su uso.

1. TextStyle: Cambiar el Estilo del Texto

La propiedad `style` permite aplicar un **TextStyle** al widget `Text`, lo que permite personalizar el tamaño de la fuente, el color, el peso (negrita), la decoración (subrayado), y más.

Ejemplo de Texto Estilizado:

```
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('Texto Estilizado'),
        ),
        body: Center(
          child: Text(
            'Flutter es increíble!',
            style: TextStyle(
              fontSize: 30,
              fontWeight: FontWeight.bold,
              color: Colors.blue,
              decoration: TextDecoration.underline,
            ),
          ),
        ),
      ),
    );
  }
}
```

Explicación del Código:

- **fontSize**: Cambia el tamaño del texto (en este caso, 30).

- **fontWeight**: Aplica un peso de fuente en negrita (`FontWeight.bold`).
- **color**: Cambia el color del texto a azul (`Colors.blue`).
- **decoration**: Añade una subrayado al texto (`TextDecoration.underline`).

Captura de la pantalla del ejemplo presentado con Texto Estilizado.
Creado por Javier A. Dastas (2024)

2. textAlign: Alinear el Texto

La propiedad `textAlign` permite alinear el texto dentro de su contenedor. Las opciones más comunes incluyen:

- **TextAlign.left**: Alinea el texto a la izquierda (por defecto).
- **TextAlign.center**: Centra el texto.

- **TextAlign.right**: Alinea el texto a la derecha.
- **TextAlign.justify**: Justifica el texto para que ocupe todo el ancho disponible.

Ejemplo de Alineación 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('Alineación de Texto'),
        ),
        body: Center(
          child: Text(
            'Este texto está centrado.',
            textAlign: TextAlign.center,
            style: TextStyle(fontSize: 20),
          ),
        ),
      ),
    );
  }
```

```
}
```

Explicación del Código:

- **textAlign**: El texto está centrado dentro de su contenedor utilizando `TextAlign.center`.

Captura de la pantalla del ejemplo presentado con Alineación de Texto.
Creado por Javier A. Dastas (2024)

3. overflow: Control del Desbordamiento de Texto

La propiedad `overflow` controla cómo se maneja el texto cuando excede el espacio disponible. Algunas de las opciones más comunes incluyen:

- **TextOverflow.clip**: Corta el texto que no cabe sin mostrar ningún indicador.
- **TextOverflow.ellipsis**: Muestra puntos suspensivos (`...`) cuando el texto es demasiado largo.
- **TextOverflow.fade**: Desvanece el texto que no cabe en el contenedor.

Ejemplo de Manejo de Desbordamiento 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('Desbordamiento de Texto'),
        ),
        body: Center(
          child: Text(
            'Este es un texto muy largo que no cabrá en una sola
línea. Mira cómo se muestra...',
            overflow: TextOverflow.ellipsis,
            style: TextStyle(fontSize: 18),
          ),
        ),
```

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

Explicación del Código:

- **overflow**: Se utiliza `TextOverflow.ellipsis` para mostrar puntos suspensivos si el texto es demasiado largo y no cabe en una sola línea.

Personalización Avanzada con Text

El widget **Text** también permite realizar configuraciones avanzadas, como cambiar el **espaciado entre letras**(`letterSpacing`), el **espaciado entre líneas** (`height`), y otros detalles tipográficos.

Ejemplo con Espaciado 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('Espaciado Personalizado'),
        ),
        body: Center(
          child: Text(
            'Texto con espaciado personalizado.',
            style: TextStyle(
              fontSize: 20,
              letterSpacing: 2.0,  // Espaciado entre letras
              height: 1.5,          // Espaciado entre líneas
            ),
          ),
        ),
      ),
    );
  }
}
```

Explicación del Código:

- **letterSpacing**: Aumenta el espacio entre letras (`2.0`).
- **height**: Ajusta el espacio entre líneas (`1.5`).

Captura de la pantalla del ejemplo presentado con Espaciado Personalizado.
Creado por Javier A. Dastas (2024)

Resumen de la Lección

En esta lección, hemos aprendido a utilizar el widget **Text** en Flutter, un componente fundamental para mostrar texto en aplicaciones. Vimos cómo utilizar las propiedades clave de **Text** para personalizar la presentación del texto, como `TextStyle`, `textAlign`, y `overflow`. También aprendimos cómo aplicar **MaterialDesign** y **Scaffold** para estructurar y organizar la interfaz de usuario.

Con el control que proporciona el widget **Text**, es posible crear **interfaces de usuario ricas en contenido textual**, con estilos y alineaciones que se ajusten a las necesidades de la aplicación.

Actividad de la Lección

Estas actividades te ayudarán a aplicar los conceptos aprendidos sobre el widget **Text** y explorar más opciones de personalización y control del texto en Flutter.

Instrucciones:

1. Crea una aplicación simple en Flutter que muestre tres líneas de texto estilizado. Cada línea debe tener un tamaño, color y alineación diferentes.
2. Investiga cómo usar **RichText** en Flutter para combinar diferentes estilos en una sola línea de texto. Implementa un ejemplo en la misma aplicación anterior que combine texto en negrita, cursiva y subrayado en una misma oración.
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.