Lección: Cambiar el Tipo de Letra en una Pantalla en Flutter

Objetivos de la Lección

- Comprender cómo cambiar el tipo de letra en una pantalla en Flutter.
- Conocer las propiedades de estilo de texto en Flutter, como `fontSize`, `fontWeight`, `fontStyle`, y `fontFamily`.
- Aplicar fuentes personalizadas y propiedades de estilo de texto en una aplicación utilizando el widget **Text**.

Introducción

El widget **Text** en Flutter permite personalizar la apariencia del texto utilizando la propiedad `style`, que acepta un objeto `TextStyle`. Con `TextStyle`, podemos modificar aspectos como el tipo de letra, tamaño, grosor, estilo, y familia de fuentes del texto en pantalla.

Propiedades de TextStyle

El objeto `TextStyle` ofrece varias propiedades que permiten cambiar el estilo de texto:

1. **fontSize**: Controla el tamaño de la letra.
2. **fontWeight**: Controla el grosor de la letra (negrita).
3. **fontStyle**: Define el estilo de la fuente, como normal o cursiva.
4. **fontFamily**: Cambia la familia de fuentes del texto (útil para fuentes personalizadas).

Ejemplo Básico: Cambiar el Tamaño y el Grosor de la Letra

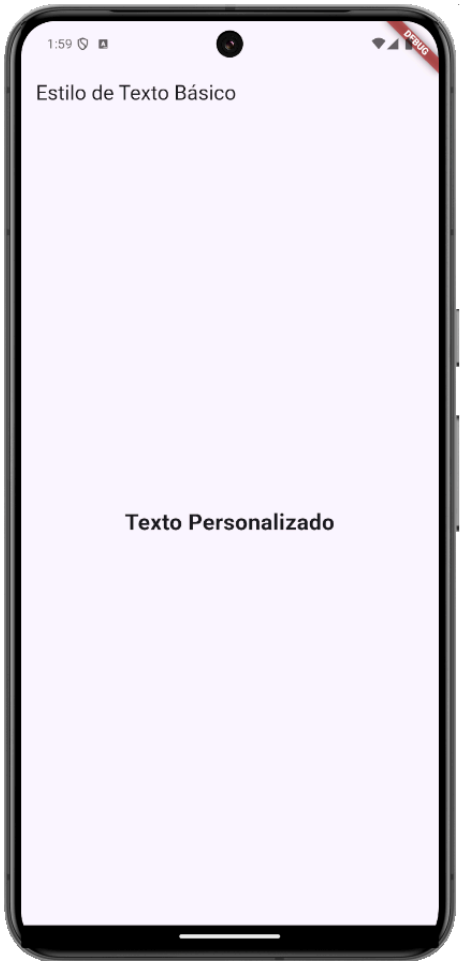
En este ejemplo, usaremos `fontSize` y `fontWeight` para cambiar el tamaño y el grosor de la letra en un texto básico.

```
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('Estilo de Texto Básico'),
        ),
        body: Center(
          child: Text(
            'Texto Personalizado',
            style: TextStyle(
              fontSize: 24, // Tamaño de la fuente
              fontWeight: FontWeight.bold, // Grosor de la
fuente
            ),
          ),
```

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

Explicación del Código:

1. **fontSize: 24**: Cambia el tamaño de la fuente a 24 puntos.
2. **fontWeight: FontWeight.bold**: Establece el texto en negrita, aumentando el grosor de las letras.



Captura de la pantalla del ejemplo presentando el cambio de tamaño y grosor del texto en Flutter. Creado por Javier A. Dastas (2024)

Cambiar el Estilo de la Letra a Cursiva

La propiedad `fontStyle` permite cambiar el estilo de la letra a cursiva o normal.

```
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 en Cursiva'),
        ),
        body: Center(
          child: Text(
            'Texto en cursiva',
            style: TextStyle(
              fontSize: 20,
              fontStyle: FontStyle.italic, // Establece el texto
en cursiva
            ),
          ),
        ),
```

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

Explicación del Código:

- **fontStyle: FontStyle.italic**: Cambia el texto a cursiva, dándole un estilo de letra inclinado.

Cambiar el Tipo de Letra Usando una Familia de Fuentes

Flutter permite usar una familia de fuentes específica mediante la propiedad `fontFamily`. Las fuentes personalizadas pueden definirse en el archivo `pubspec.yaml` y luego usarse en el texto.

Paso 1: Agregar una Fuente Personalizada en `pubspec.yaml`

Agrega la fuente a la carpeta `assets/fonts` (por ejemplo, `assets/fonts/Roboto-Regular.ttf`) y declárala en el archivo `pubspec.yaml`:

```
flutter:
  fonts:
    - family: Roboto
      fonts:
        - asset: assets/fonts/Roboto-Regular.ttf
```

Paso 2: Usar la Fuente Personalizada en el Código

Después de agregar y declarar la fuente, se puede aplicar usando `fontFamily`.

```
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('Fuente Personalizada'),
        ),
        body: Center(
          child: Text(
            'Texto con Fuente Personalizada',
            style: TextStyle(
              fontSize: 22,
              fontFamily: 'Roboto', // Aplica la fuente
personalizada
            ),
          ),
        ),
```

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

Explicación del Código:

- **fontFamily: 'Roboto'**: Cambia el tipo de letra a la fuente personalizada **Roboto** definida en `pubspec.yaml`.

Usando Fuentes Directamente de Google Fonts

Para utilizar fuentes de Google Fonts directamente en Flutter, puedes emplear la librería google_fonts, que facilita la integración de estas fuentes en tu proyecto. Aquí te explico los pasos:

Paso 1: Agregar la Dependencia

En tu archivo `pubspec.yaml`, agrega la dependencia `google_fonts`:

```
dependencies:
  flutter:
    sdk: flutter
  google_fonts: ^5.0.0
```

Luego, ejecuta `flutter pub get` para instalar la librería.

Paso 2: Importar la Librería

En tu archivo de código Dart, importa la librería:

```
import 'package:google_fonts/google_fonts.dart';
```

Paso 3: Usar la Fuente de Google Fonts

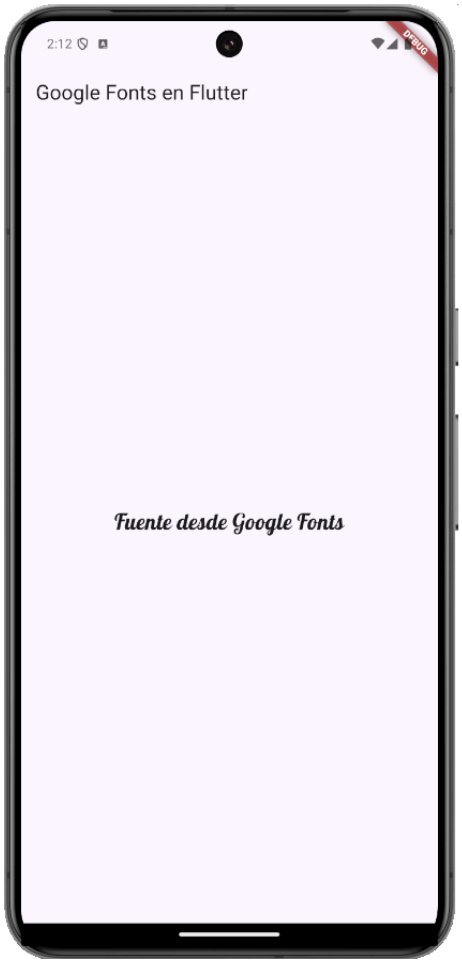
Ahora puedes usar cualquier fuente de Google Fonts en el widget **Text** llamando a `GoogleFonts.<nombreDeLaFuente>()` dentro de la propiedad `style`:

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

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

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: Scaffold(
        appBar: AppBar(
          title: Text('Google Fonts en Flutter'),
        ),
        body: Center(
          child: Text(
            'Fuente desde Google Fonts',
            style: GoogleFonts.lobster(
              textStyle: TextStyle(fontSize: 24),
            ),
          ),
        ),
```

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

Explicación del Código:

- En este ejemplo, hemos aplicado la fuente `Lobster` desde Google Fonts. Puedes personalizar más la apariencia del texto agregando propiedades como `fontSize`, `fontWeight`, y `color` dentro del `TextStyle`.



Captura de la pantalla del ejemplo utilizando la librería `google_fonts` para cambio de la fuente de texto en Flutter. Creado por Javier A. Dastas (2024)

Beneficios

La librería `google_fonts` descarga las fuentes automáticamente, por lo que no es necesario añadir los archivos de las fuentes al proyecto, simplificando así la integración y ahorrando espacio.

Combinación de Propiedades de Texto

Podemos combinar `fontSize`, `fontWeight`, `fontStyle`, y `fontFamily` para crear un estilo de texto completo y personalizado.

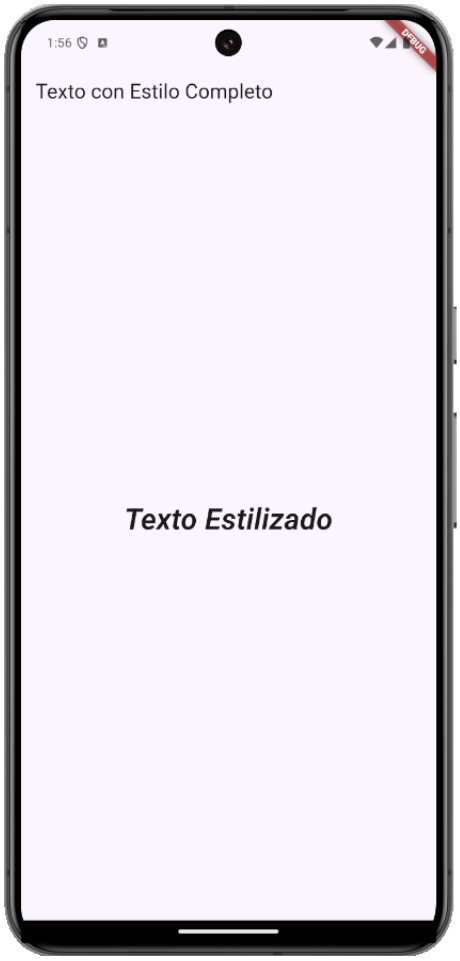
Ejemplo Completo con Varias Propiedades de Estilo de Texto

```
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('Texto con Estilo Completo'),
        ),
        body: Center(
          child: Text(
            'Texto Estilizado',
```

```
            style: TextStyle(
              fontSize: 32,
              fontWeight: FontWeight.w600, // Un grosor
intermedio
              fontStyle: FontStyle.italic,
              fontFamily: 'Roboto', // Utiliza la fuente
personalizada
            ),
          ),
        ),
      ),
    );
  }
}
```

Explicación del Código:

- **fontSize: 32**: Establece el tamaño de la fuente en 26 puntos.
- **fontWeight: FontWeight.w600**: Define un grosor intermedio.
- **fontStyle: FontStyle.italic**: Cambia el texto a cursiva.
- **fontFamily: 'Roboto'**: Usa la fuente personalizada **Roboto**.



Captura de la pantalla del ejemplo presentando todas las opciones para cambio de texto en Flutter. Creado por Javier A. Dastas (2024)

Usar Colores en el Texto

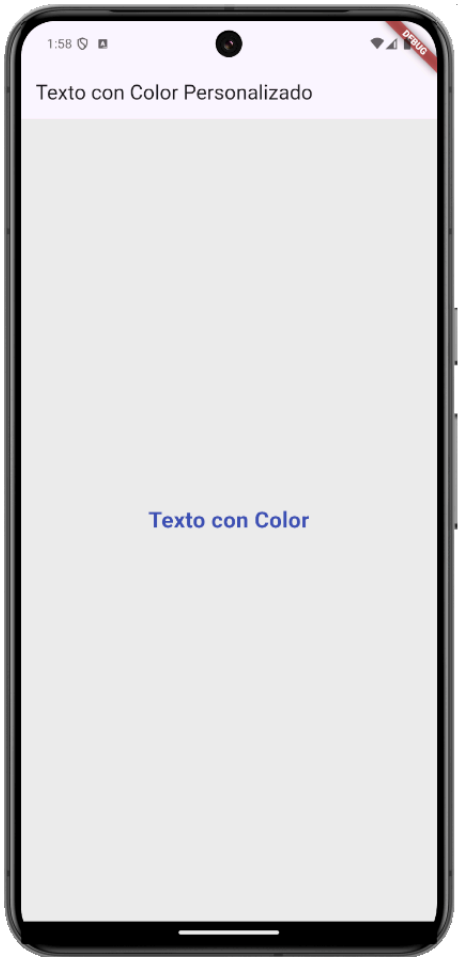
El estilo de texto también admite la propiedad `color` para personalizar el color del texto y mejorar el contraste con el fondo.

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


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

```
class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: Scaffold(
        backgroundColor: Colors.grey[200], // Fondo claro para
contraste
        appBar: AppBar(
          title: Text('Texto con Color Personalizado'),
        ),
        body: Center(
          child: Text(
            'Texto con Color',
            style: TextStyle(
              fontSize: 24,
              fontWeight: FontWeight.bold,
              fontFamily: 'Roboto',
              color: Colors.indigo, // Cambia el color del texto
a índigo
            ),
          ),
        ),
      ),
    );
  }
}
```

Explicación del Código:

- **color: Colors.indigo**: Cambia el color del texto a índigo, haciéndolo destacar sobre el fondo claro.



Captura de la pantalla del ejemplo presentando todas las opciones para cambio de texto y el color en Flutter. Creado por Javier A. Dastas (2024)

Resumen de la Lección

En esta lección, hemos aprendido cómo cambiar el tipo de letra en una pantalla en Flutter utilizando el widget **Text** y la propiedad `TextStyle`. Exploramos varias propiedades de `TextStyle`:

- **fontSize** para cambiar el tamaño del texto.
- **fontWeight** para cambiar el grosor de la letra.
- **fontStyle** para cambiar el estilo (normal o cursiva).
- **fontFamily** para aplicar una familia de fuentes personalizada.

Estas propiedades permiten personalizar completamente el estilo del texto y adaptar la interfaz de usuario al diseño deseado.

Actividad de la Lección

Esta actividad te ayudará a aplicar y experimentar con las propiedades de estilo de texto en Flutter, creando interfaces visualmente atractivas y personalizadas.

Instrucciones:

1. Crea una aplicación en Flutter que muestre una pantalla con tres textos en diferentes estilos:
    - El primer texto debe estar en negrita y tamaño 28.
    - El segundo texto debe estar en cursiva y tamaño 20.
    - El tercer texto debe utilizar una fuente personalizada y color rojo oscuro.
2. Usa el archivo `pubspec.yaml` para declarar una fuente personalizada y aplícala a uno de los textos de 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.