Lección: Uso de los Widgets Column y Row en Flutter

Objetivos de la Lección

- Comprender el propósito y uso de los widgets **Column** y **Row** para organizar elementos en Flutter.
- Reconocer la estructura básica de los widgets **Column** y **Row** y sus propiedades principales.
- Implementar ejemplos básicos que utilicen **Column** y **Row** para alinear y organizar texto dentro de una aplicación.

Introducción a los Widgets Column y Row en Flutter

Los widgets **Column** y **Row** son esenciales en Flutter para organizar la interfaz de usuario. **Column** organiza sus hijos en una **disposición vertical** (uno debajo de otro), mientras que **Row** organiza sus hijos en una **disposición horizontal** (uno al lado del otro).

Estos widgets son fundamentales para construir interfaces estructuradas, ya que permiten alinear y distribuir otros widgets (en este caso, solo usaremos `Text`) en la pantalla de forma ordenada. Ambos widgets también cuentan con propiedades que permiten ajustar la alineación y distribución de sus hijos para un mejor control de la interfaz.

Widget Column

El widget **Column** organiza sus hijos en una disposición vertical. Esto significa que cada elemento dentro de una `Column`se mostrará uno debajo de otro.

Propiedades Principales de Column

1. **mainAxisAlignment**: Controla cómo se distribuyen los widgets en el eje principal (vertical) de la columna.
2. **crossAxisAlignment**: Controla cómo se alinean los widgets en el eje cruzado (horizontal) de la columna.

Ejemplo Básico de Column

En este ejemplo, organizaremos tres widgets `Text` en una `Column`, centrados en la pantalla y distribuidos verticalmente.

```
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 Column'),
        ),
        body: Center(
          child: Column(
            mainAxisAlignment: MainAxisAlignment.center, //
Centra en el eje vertical
```

```
            crossAxisAlignment: CrossAxisAlignment.center, //
Centra en el eje horizontal
            children: <Widget>[
              Text('Texto 1'),
              Text('Texto 2'),
              Text('Texto 3'),
            ],
          ),
        ),
      ),
    );
  }
}
```

Explicación del Código:

1. **Column**: Organiza los widgets `Text` en una disposición vertical.
2. **mainAxisAlignment**: Centra los elementos verticalmente en el contenedor (en este caso, la pantalla).
3. **crossAxisAlignment**: Centra los elementos horizontalmente dentro de la columna.
4. **children**: Lista de widgets `Text` que se mostrarán en la `Column`.

Captura de la pantalla del ejemplo presentado con las propiedades principales del Widget Column. Creado por Javier A. Dastas (2024)

Propiedades de Alineación en Column

El widget **Column** ofrece varios valores para `mainAxisAlignment` y `crossAxisAlignment` que permiten controlar la posición de los elementos dentro de la columna.

1. **mainAxisAlignment**:
    - **MainAxisAlignment.start**: Alinea los elementos en la parte superior.
    - **MainAxisAlignment.center**: Centra los elementos verticalmente.

- **MainAxisAlignment.end**: Alinea los elementos en la parte inferior.
- **MainAxisAlignment.spaceAround**: Distribuye el espacio alrededor de los elementos.
- **MainAxisAlignment.spaceBetween**: Coloca espacio entre los elementos.
- **MainAxisAlignment.spaceEvenly**: Distribuye el espacio de manera uniforme.

2. **crossAxisAlignment**:
   - **CrossAxisAlignment.start**: Alinea los elementos a la izquierda de la columna.
   - **CrossAxisAlignment.center**: Centra los elementos horizontalmente.
   - **CrossAxisAlignment.end**: Alinea los elementos a la derecha de la columna.

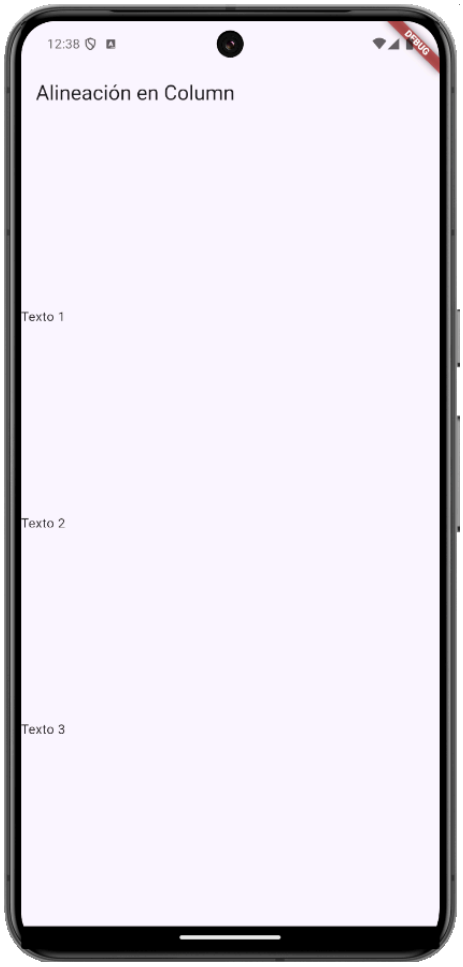
Ejemplo con Diferentes Valores de Alineación en Column

```
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('Alineación en Column'),
        ),
        body: Column(
          mainAxisAlignment: MainAxisAlignment.spaceEvenly,
          crossAxisAlignment: CrossAxisAlignment.start,
          children: <Widget>[
            Text('Texto 1'),
            Text('Texto 2'),
            Text('Texto 3'),
          ],
        ),
      ),
    );
  }
}
```

Explicación del Código:

- **mainAxisAlignment: MainAxisAlignment.spaceEvenly**: Distribuye el espacio de manera uniforme entre los elementos.
- **crossAxisAlignment: CrossAxisAlignment.start**: Alinea los elementos al inicio del eje cruzado (a la izquierda).



Captura de la pantalla del ejemplo presentado con diferentes propiedades de alineación del Widget Column. Creado por Javier A. Dastas (2024)

Widget Row

El widget **Row** organiza sus hijos en una disposición horizontal, lo que significa que cada elemento dentro de una Row se mostrará uno al lado del otro. **Row** es útil para colocar widgets en una línea horizontal.

Propiedades Principales de Row

1. **mainAxisAlignment**: Controla cómo se distribuyen los widgets en el eje principal (horizontal) de la fila.

2. **crossAxisAlignment**: Controla cómo se alinean los widgets en el eje cruzado (vertical) de la fila.

Ejemplo Básico de Row

En este ejemplo, organizaremos tres widgets `Text` en una `Row`, centrados en la pantalla y distribuidos horizontalmente.

```
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 Row'),
        ),
        body: Center(
          child: Row(
            mainAxisAlignment: MainAxisAlignment.center, //
Centra en el eje horizontal
            crossAxisAlignment: CrossAxisAlignment.center, //
Centra en el eje vertical
            children: <Widget>[
```

```
              Text('Texto 1'),
              Text('Texto 2'),
              Text('Texto 3'),
            ],
          ),
        ),
      ),
    );
  }
}
```

Explicación del Código:

1. **Row**: Organiza los widgets `Text` en una disposición horizontal.
2. **mainAxisAlignment**: Centra los elementos horizontalmente dentro del contenedor (en este caso, la pantalla).
3. **crossAxisAlignment**: Centra los elementos verticalmente dentro de la fila.
4. **children**: Lista de widgets `Text` que se mostrarán en la `Row`.

Captura de la pantalla del ejemplo presentado con las propiedades principales del Widget Row. Creado por Javier A. Dastas (2024)

Propiedades de Alineación en Row

Al igual que **Column**, el widget **Row** ofrece varias opciones para `mainAxisAlignment` y `crossAxisAlignment` para controlar la posición de los elementos dentro de la fila.

1. **mainAxisAlignment**:
    - **MainAxisAlignment.start**: Alinea los elementos al inicio de la fila.
    - **MainAxisAlignment.center**: Centra los elementos horizontalmente.
    - **MainAxisAlignment.end**: Alinea los elementos al final de la fila.

- **MainAxisAlignment.spaceAround**: Distribuye el espacio alrededor de los elementos.
- **MainAxisAlignment.spaceBetween**: Coloca espacio entre los elementos.
- **MainAxisAlignment.spaceEvenly**: Distribuye el espacio de manera uniforme.

2. **crossAxisAlignment**:
   - **CrossAxisAlignment.start**: Alinea los elementos en la parte superior de la fila.
   - **CrossAxisAlignment.center**: Centra los elementos verticalmente.
   - **CrossAxisAlignment.end**: Alinea los elementos en la parte inferior de la fila.

Ejemplo con Diferentes Valores de Alineación en Row

```
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 en Row'),
        ),
        body: Center(
          child: Row(
            mainAxisAlignment: MainAxisAlignment.spaceAround,
            crossAxisAlignment: CrossAxisAlignment.start,
            children: <Widget>[
              Text('Texto 1'),
              Text('Texto 2'),
              Text('Texto 3'),
            ],
          ),
        ),
      ),
    );
  }
}
```

Explicación del Código:

- **mainAxisAlignment: MainAxisAlignment.spaceAround**: Distribuye el espacio alrededor de los elementos de forma equilibrada.
- **crossAxisAlignment: CrossAxisAlignment.start**: Alinea los elementos al inicio del eje cruzado (arriba).

Captura de la pantalla del ejemplo presentado con algunas propiedades de alineación del Widget Row. Creado por Javier A. Dastas (2024)

Resumen de la Lección

En esta lección, hemos aprendido que los widgets **Column** y **Row** son esenciales para organizar el contenido en una interfaz de usuario de Flutter. **Column** organiza los widgets en una disposición vertical, mientras que **Row** los organiza en una disposición horizontal. Ambos widgets permiten controlar la posición y alineación de los elementos con propiedades como `mainAxisAlignment` y `crossAxisAlignment`.

Estos widgets son útiles para crear interfaces de usuario estructuradas y flexibles, permitiendo una disposición ordenada del contenido.

Actividad de la Lección

Esta actividad te ayudará a aplicar los conceptos aprendidos y a practicar la organización de elementos en Flutter utilizando los widgets **Column** y **Row**.

Instrucciones:

1. Crea una aplicación en Flutter que use una `Column` para mostrar tres textos alineados en el centro de la pantalla. Cada texto debe estar alineado a la izquierda dentro de la columna.
2. Implementa otra `Row` que contenga tres textos distribuidos de manera uniforme a lo largo de la pantalla y alineados en la parte inferior.
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.