Benemérita Universidad Autónoma de Puebla

Facultad de Ciencias de la Computación

Desarrollo de Aplicaciones Móviles

Profesor: Reynold Osuna González

Alumno:

David Carrillo Castillo

Layouts en Flutter

Primavera 2024

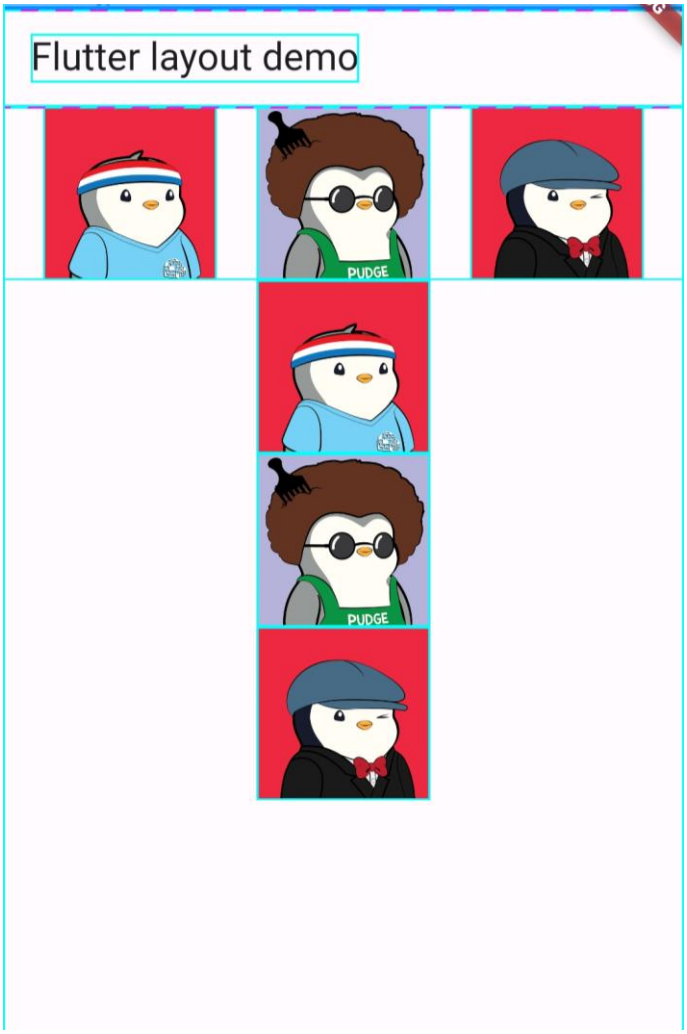
Leer y seguir las instrucciones del documento, entregar evidencia con capturas de pantalla de la implementación del código.

1) Primero, cuando se añaden imágenes a un proyecto, necesitas actualizar el archivo pubspec.yaml para acceder a ellas; en este ejemplo se utiliza Image.asset para mostrar las imágenes. No necesitas hacer esto si estás haciendo referencia a imágenes en línea mediante Image.network.

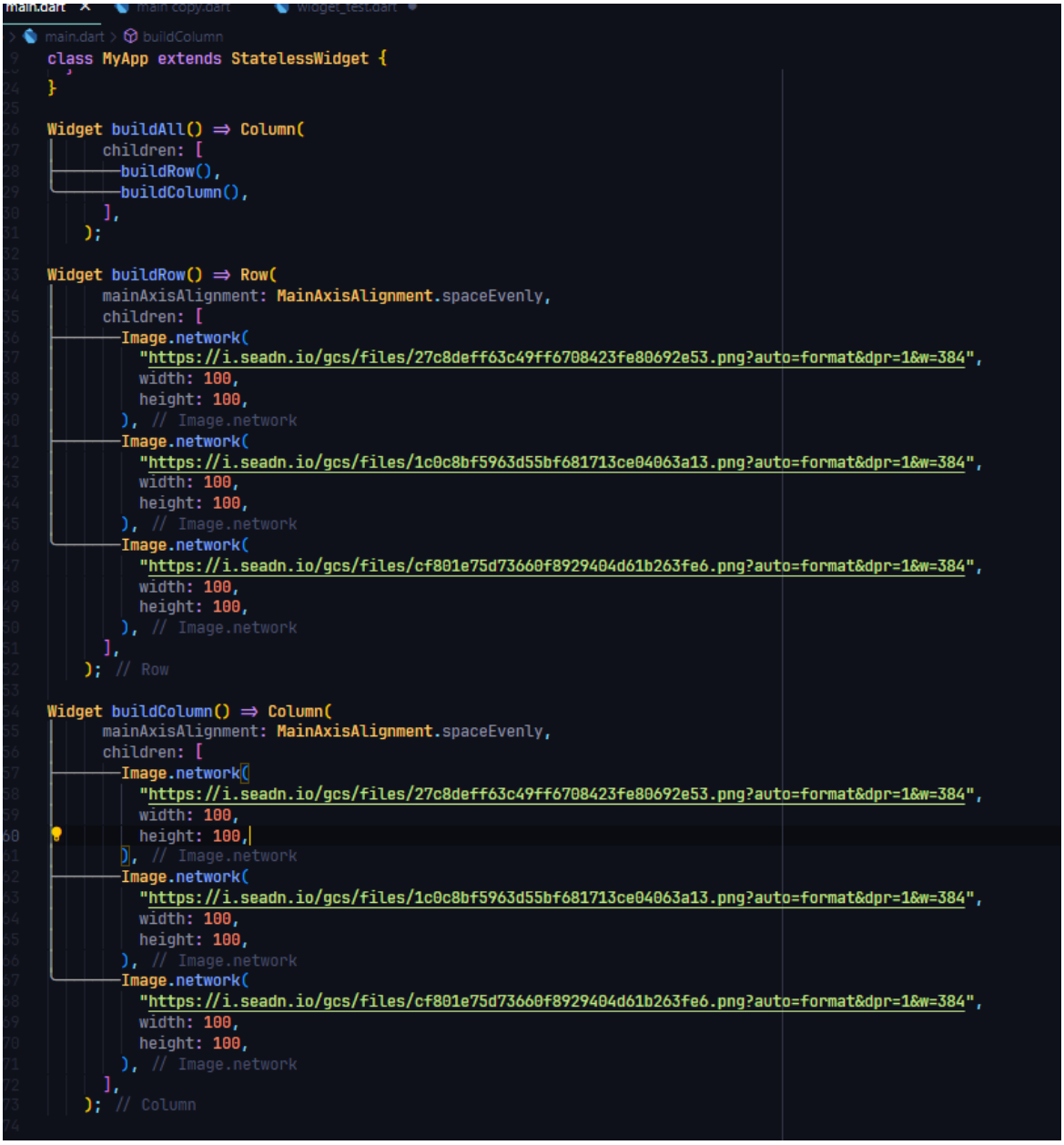
En el siguiente ejemplo, cada una de las 3 imágenes tiene un ancho de 100 píxeles. El cuadro de representación (en este caso, toda la pantalla) tiene un ancho de más de 300 píxeles, por lo que al establecer la alineación del eje principal en "spaceEvenly", se divide de manera uniforme el espacio horizontal libre entre, antes y después de cada imagen.

Las columnas funcionan de la misma manera que las filas. El siguiente ejemplo muestra una columna de 3 imágenes, cada una con 100 píxeles de altura. La altura del cuadro de representación (en este caso, toda la pantalla) es de más de 300 píxeles, por lo que al establecer la alineación del eje principal en "spaceEvenly", se divide de manera uniforme el espacio vertical libre entre, arriba y abajo de cada imagen.

Nuestra interfaz quedaría de la siguiente manera:



Con el siguiente código:



```dart
class MyApp extends StatelessWidget {
}

Widget buildAll() => Column(
      children: [
        buildRow(),
        buildColumn(),
      ],
    );

Widget buildRow() => Row(
      mainAxisAlignment: MainAxisAlignment.spaceEvenly,
      children: [
        Image.network(
          "https://i.seadn.io/gcs/files/27c8deff63c49ff6708423fe80692e53.png?auto=format&dpr=1&w=384",
          width: 100,
          height: 100,
        ), // Image.network
        Image.network(
          "https://i.seadn.io/gcs/files/1c0c8bf5963d55bf681713ce04063a13.png?auto=format&dpr=1&w=384",
          width: 100,
          height: 100,
        ), // Image.network
        Image.network(
          "https://i.seadn.io/gcs/files/cf801e75d73660f8929404d61b263fe6.png?auto=format&dpr=1&w=384",
          width: 100,
          height: 100,
        ), // Image.network
      ],
    ); // Row

Widget buildColumn() => Column(
      mainAxisAlignment: MainAxisAlignment.spaceEvenly,
      children: [
        Image.network(
          "https://i.seadn.io/gcs/files/27c8deff63c49ff6708423fe80692e53.png?auto=format&dpr=1&w=384",
          width: 100,
          height: 100,
        ), // Image.network
        Image.network(
          "https://i.seadn.io/gcs/files/1c0c8bf5963d55bf681713ce04063a13.png?auto=format&dpr=1&w=384",
          width: 100,
          height: 100,
        ), // Image.network
        Image.network(
          "https://i.seadn.io/gcs/files/cf801e75d73660f8929404d61b263fe6.png?auto=format&dpr=1&w=384",
          width: 100,
          height: 100,
        ), // Image.network
      ],
    ); // Column
```

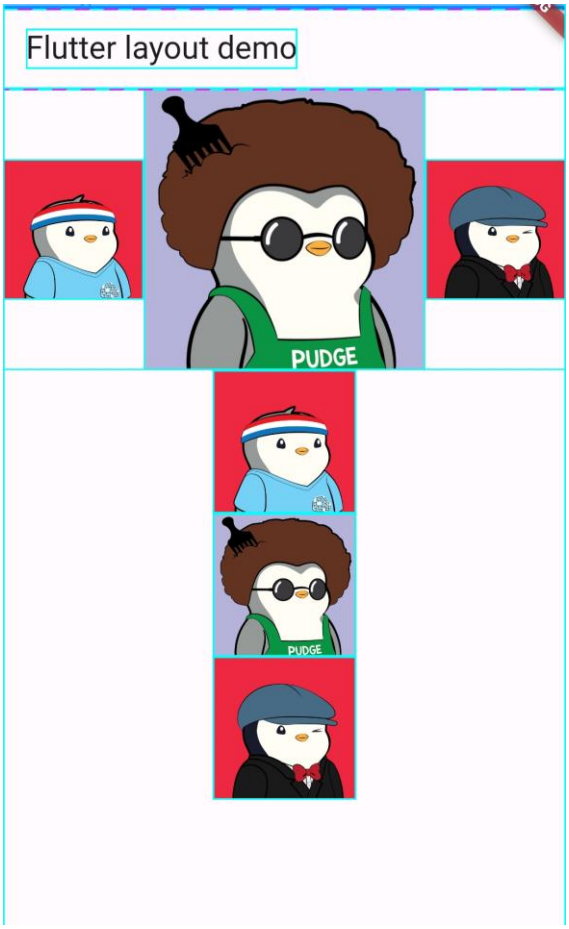
2) Dimensionamiento de widgets

Cuando un diseño es demasiado grande para caber en un dispositivo, aparece un patrón a rayas amarillas y negras a lo largo del borde afectado. Aquí tienes un ejemplo de una fila que es demasiado ancha:

Los widgets pueden redimensionarse para que encajen dentro de una fila o columna utilizando el widget "Expanded". Para solucionar el ejemplo anterior en el que la fila de imágenes es demasiado ancha para su cuadro de representación, envuelve cada imagen con un widget "Expanded".

Si quieres que un widget ocupe el doble de espacio que sus elementos hermanos, puedes utilizar la propiedad "flex" del widget "Expanded", un valor entero que determina el factor de flexibilidad de un widget. El valor predeterminado del factor de flexibilidad es 1. El siguiente código establece el factor de flexibilidad de la imagen del medio en 2:

Nuestra interfaz de usuario quedaria:



Con el siguiente código:

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

void main(List<String> args) {
  debugPaintSizeEnabled = true;
  runApp(const MyApp());
}

class MyApp extends StatelessWidget {
  const MyApp({super.key});

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Flutter layout demo',
      home: Scaffold(
        appBar: AppBar(
          title: const Text('Flutter layout demo'),
        ),
        body: Center(child: buildAll()),
      ),
    );
  }
}

Widget buildAll() => Column(
      children: [
```

```
        buildRow(),
        buildColumn(),
      ],
    );

Widget buildRow() => Row(
      mainAxisAlignment: MainAxisAlignment.spaceEvenly,
      children: [
        Expanded(
          child: Image.network(

"https://i.seadn.io/gcs/files/27c8deff63c49ff6708423fe80692e53.png?auto=format&dpr=1&w=384
",
          ),
        ),
        Expanded(
          flex: 2,
          child: Image.network(

"https://i.seadn.io/gcs/files/1c0c8bf5963d55bf681713ce04063a13.png?auto=format&dpr=1&w=384
",
          ),
        ),
        Expanded(
          child: Image.network(

"https://i.seadn.io/gcs/files/cf801e75d73660f8929404d61b263fe6.png?auto=format&dpr=1&w=384
",
          ),
        ),
      ],
    );

Widget buildColumn() => Column(
      mainAxisAlignment: MainAxisAlignment.spaceEvenly,
      children: [
        Image.network(

"https://i.seadn.io/gcs/files/27c8deff63c49ff6708423fe80692e53.png?auto=format&dpr=1&w=384
",
          width: 100,
          height: 100,
        ),
        Image.network(

"https://i.seadn.io/gcs/files/1c0c8bf5963d55bf681713ce04063a13.png?auto=format&dpr=1&w=384
",
          width: 100,
          height: 100,
        ),
        Image.network(

"https://i.seadn.io/gcs/files/cf801e75d73660f8929404d61b263fe6.png?auto=format&dpr=1&w=384
",
```

```
          width: 100,
          height: 100,
        ),
      ],
    );
```

3) Packing widgets

Por defecto, una fila o columna ocupa tanto espacio a lo largo de su eje principal como sea posible, pero si deseas agrupar los hijos de cerca, configura su propiedad "mainAxisSize" en "MainAxisSize.min". El siguiente ejemplo utiliza esta propiedad para agrupar los iconos de estrella juntos.

El layout framework permite anidar filas y columnas dentro de filas y columnas, tan profundamente como sea necesario:

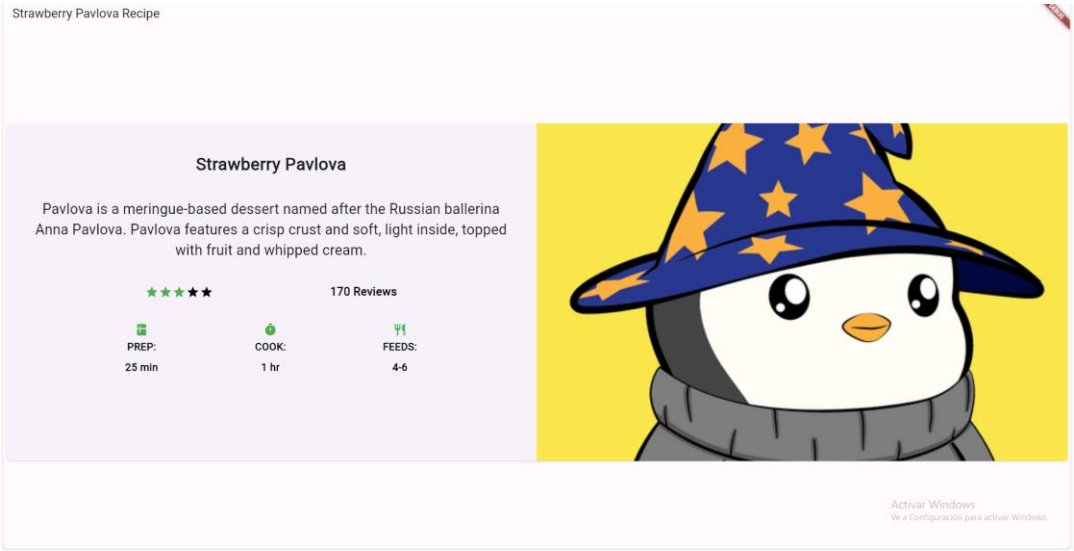
La sección resaltada se implementa como dos filas. La fila de calificaciones contiene cinco estrellas y el número de reseñas. La fila de iconos contiene tres columnas de iconos y texto. El árbol de widgets para la fila de calificaciones:

La variable ratings crea un row que contiene una row más pequeña que contiene 5 íconos de estrella y texto:

La columna izquierda se coloca en un SizedBox para limitar su ancho. Finalmente, la interfaz de usuario se construye con toda la fila (que contiene la columna izquierda y la imagen) dentro de una tarjeta (Card).

La imagen de Pavlova proviene de Pixabay. Puedes incrustar una imagen desde la web utilizando Image.network(), pero, para este ejemplo, la imagen se guarda en un directorio de imágenes en el proyecto, se agrega al archivo pubspec y se accede utilizando Images.asset

Nuestra interfaz de usuario quedaría:



Con el siguiente código:

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

import 'package:flutter/rendering.dart' show debugPaintSizeEnabled;

void main() {
  debugPaintSizeEnabled = false; // Set to true for visual layout
  runApp(const MyApp());
}
```

```dart
class MyApp extends StatelessWidget {
  const MyApp({Key? key});

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Flutter layout demo',
      home: buildHomePage('Strawberry Pavlova Recipe'),
    );
  }

  Widget buildHomePage(String title) {
    const titleText = Padding(
      padding: EdgeInsets.all(20),
      child: Text(
        'Strawberry Pavlova',
        style: TextStyle(
          fontWeight: FontWeight.w800,
          letterSpacing: 0.5,
          fontSize: 30,
        ),
      ),
    );

    const subTitle = Padding(
      padding: EdgeInsets.all(20),
      child: Text(
        'Pavlova is a meringue-based dessert named after the Russian ballerina '
        'Anna Pavlova. Pavlova features a crisp crust and soft, light inside, '
        'topped with fruit and whipped cream.',
        textAlign: TextAlign.center,
        style: TextStyle(
          fontFamily: 'Georgia',
          fontSize: 25,
        ),
      ),
    );

    var stars = Row(
      mainAxisSize: MainAxisSize.min,
      children: [
        Icon(Icons.star, color: Colors.green[500]),
        Icon(Icons.star, color: Colors.green[500]),
        Icon(Icons.star, color: Colors.green[500]),
        const Icon(Icons.star, color: Colors.black),
        const Icon(Icons.star, color: Colors.black),
      ],
    );

    final ratings = Container(
      padding: EdgeInsets.all(20),
      child: Row(
        mainAxisAlignment: MainAxisAlignment.spaceEvenly,
        children: [
```

```
        stars,
        const Text(
          '170 Reviews',
          style: TextStyle(
            color: Colors.black,
            fontWeight: FontWeight.w800,
            fontFamily: 'Roboto',
            letterSpacing: 0.5,
            fontSize: 20,
          ),
        ),
      ],
    ),
  );

  const descTextStyle = TextStyle(
    color: Colors.black,
    fontWeight: FontWeight.w800,
    fontFamily: 'Roboto',
    letterSpacing: 0.5,
    fontSize: 18,
    height: 2,
  );

  final iconList = DefaultTextStyle.merge(
    style: descTextStyle,
    child: Container(
      padding: const EdgeInsets.all(20),
      child: Row(
        mainAxisAlignment: MainAxisAlignment.spaceEvenly,
        children: [
          Column(
            children: [
              Icon(Icons.kitchen, color: Colors.green[500]),
              const Text('PREP:'),
              const Text('25 min'),
            ],
          ),
          Column(
            children: [
              Icon(Icons.timer, color: Colors.green[500]),
              const Text('COOK:'),
              const Text('1 hr'),
            ],
          ),
          Column(
            children: [
              Icon(Icons.restaurant, color: Colors.green[500]),
              const Text('FEEDS:'),
              const Text('4-6'),
            ],
          ),
        ],
      ),
```

```
      ),
    );

    final leftColumn = Container(
      padding: const EdgeInsets.fromLTRB(20, 30, 20, 20),
      child: Column(
        children: [
          titleText,
          subTitle,
          ratings,
          iconList,
        ],
      ),
    );

    final mainImage = Expanded(
      child: Image.network(

'https://i.seadn.io/gcs/files/9dce2aeb43838f1570b630a84981f79c.png?auto=format&dpr=1&w=384
',
        fit: BoxFit.cover,
      ),
    );

    return Scaffold(
      appBar: AppBar(
        title: Text(title),
      ),
      body: Center(
        child: Container(
          margin: const EdgeInsets.fromLTRB(0, 40, 0, 30),
          height: 600,
          child: Card(
            child: Row(
              crossAxisAlignment: CrossAxisAlignment.start,
              children: [
                Expanded(
                  child: leftColumn,
                ),
                mainImage,
              ],
            ),
          ),
        ),
      ),
    );
  }
}
```

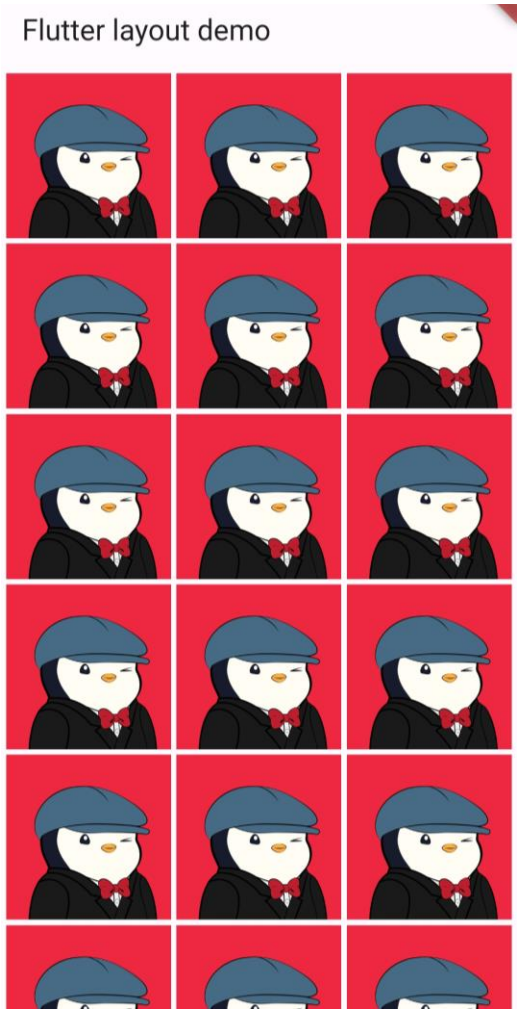
4) Widgets comunes para layout

Flutter cuenta con una amplia biblioteca de widgets de diseño. Aquí se presentan algunos de los más comúnmente utilizados. Para obtener información sobre otros widgets disponibles, siempre es posible consultar el catálogo de widgets o utilizar el cuadro de búsqueda en la documentación de referencia de la

API. Además, las páginas de widgets en la documentación de la API a menudo hacen sugerencias sobre widgets similares que podrían adaptarse mejor a necesidades particulares.

Los siguientes widgets se dividen en dos categorías: widgets estándar de la biblioteca de widgets y widgets especializados de la biblioteca Material. Cualquier aplicación puede utilizar la biblioteca de widgets, pero solo las aplicaciones Material pueden utilizar la biblioteca de Componentes Material

La vista quedaría de la siguiente manera:



Con el siguiente código:

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

import 'package:flutter/rendering.dart' show debugPaintSizeEnabled;

void main() {
  debugPaintSizeEnabled = false; // Set to true for visual layout

  runApp(const MyApp());
}

class MyApp extends StatelessWidget {
  const MyApp({super.key});

  static const showGrid = true; // Set to false to show ListView

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
```

```
      title: 'Flutter layout demo',
      home: Scaffold(
        appBar: AppBar(
          title: const Text('Flutter layout demo'),
        ),
        body: Center(child: showGrid ? _buildGrid() : _buildList()),
      ),
    );
  }

// #docregion grid

  Widget _buildGrid() => GridView.extent(
      maxCrossAxisExtent: 150,
      padding: const EdgeInsets.all(4),
      mainAxisSpacing: 4,
      crossAxisSpacing: 4,
      children: _buildGridTileList(30));

// The images are saved with names pic0.jpg, pic1.jpg...pic29.jpg.

// The List.generate() constructor allows an easy way to create

// a list when objects have a predictable naming pattern.

  List<Container> _buildGridTileList(int count) => List.generate(
      count,
      (i) => Container(
          child: Image.network(

'https://i.seadn.io/gcs/files/cf801e75d73660f8929404d61b263fe6.png?auto=format&dpr=1&w=384
')));

// #enddocregion grid

// #docregion list

  Widget _buildList() {
    return ListView(
      children: [
        _tile('CineArts at the Empire', '85 W Portal Ave', Icons.theaters),
        _tile('The Castro Theater', '429 Castro St', Icons.theaters),
        _tile('Alamo Drafthouse Cinema', '2550 Mission St', Icons.theaters),
        _tile('Roxie Theater', '3117 16th St', Icons.theaters),
        _tile('United Artists Stonestown Twin', '501 Buckingham Way',
            Icons.theaters),
        _tile('AMC Metreon 16', '135 4th St #3000', Icons.theaters),
        const Divider(),
        _tile('K\'s Kitchen', '757 Monterey Blvd', Icons.restaurant),
        _tile('Emmy\'s Restaurant', '1923 Ocean Ave', Icons.restaurant),
        _tile('Chaiya Thai Restaurant', '272 Claremont Blvd', Icons.restaurant),
        _tile('La Ciccia', '291 30th St', Icons.restaurant),
      ],
    );
```

```
  }

  ListTile _tile(String title, String subtitle, IconData icon) {
    return ListTile(
      title: Text(title,
          style: const TextStyle(
            fontWeight: FontWeight.w500,
            fontSize: 20,
          )),
      subtitle: Text(subtitle),
      leading: Icon(
        icon,
        color: Colors.blue[500],
      ),
    );
  }

// #enddocregion list
}
```

5) Card

Un Card, de la biblioteca Material, contiene información relacionada y puede estar compuesta por casi cualquier widget, pero a menudo se utiliza con ListTile. Card tiene un único hijo, pero su hijo puede ser una columna, fila, lista, cuadrícula u otro widget que admite varios hijos. De forma predeterminada, un Card reduce su tamaño a 0 por 0 píxeles. Puedes utilizar SizedBox para limitar el tamaño de Card.

En Flutter, un CardTarjeta presenta esquinas ligeramente redondeadas y una sombra que le da un efecto tridimensional. Cambiar la propiedad de elevación de Card te permite controlar el efecto de la sombra. Al establecer la elevación en 24, por ejemplo, levanta visualmente la Tarjeta más alejada de la superficie y hace que la sombra se dispersa más. Para ver una lista de valores admitidos de elevación, consulta "Elevation" en las directrices de Material. Especificar un valor no admitido deshabilita por completo la sombra.

Card:

Implementa una tarjeta Material.

Utilizada para presentar información relacionada.

Acepta un único hijo, pero ese hijo puede ser una fila, columna u otro widget que contenga una lista de hijos.

Se muestra con esquinas redondeadas y una sombra.

El contenido de Card no se puede desplazar.

Pertenece a la biblioteca Material.

La siguiente imagen corresponde a un Card que contiene 3 ListTiles y se dimensiona envolviéndola con un SizedBox. Un Separador (Divider) separa el primer y segundo ListTile

ListTile

ListTile, un widget de fila especializado de la biblioteca Material, nos permite crear una fila que contiene hasta 3 líneas de texto e iconos opcionales en la parte frontal y trasera. ListTile se utiliza con mayor frecuencia en Card o ListView, pero también puede utilizarse en otros lugares.

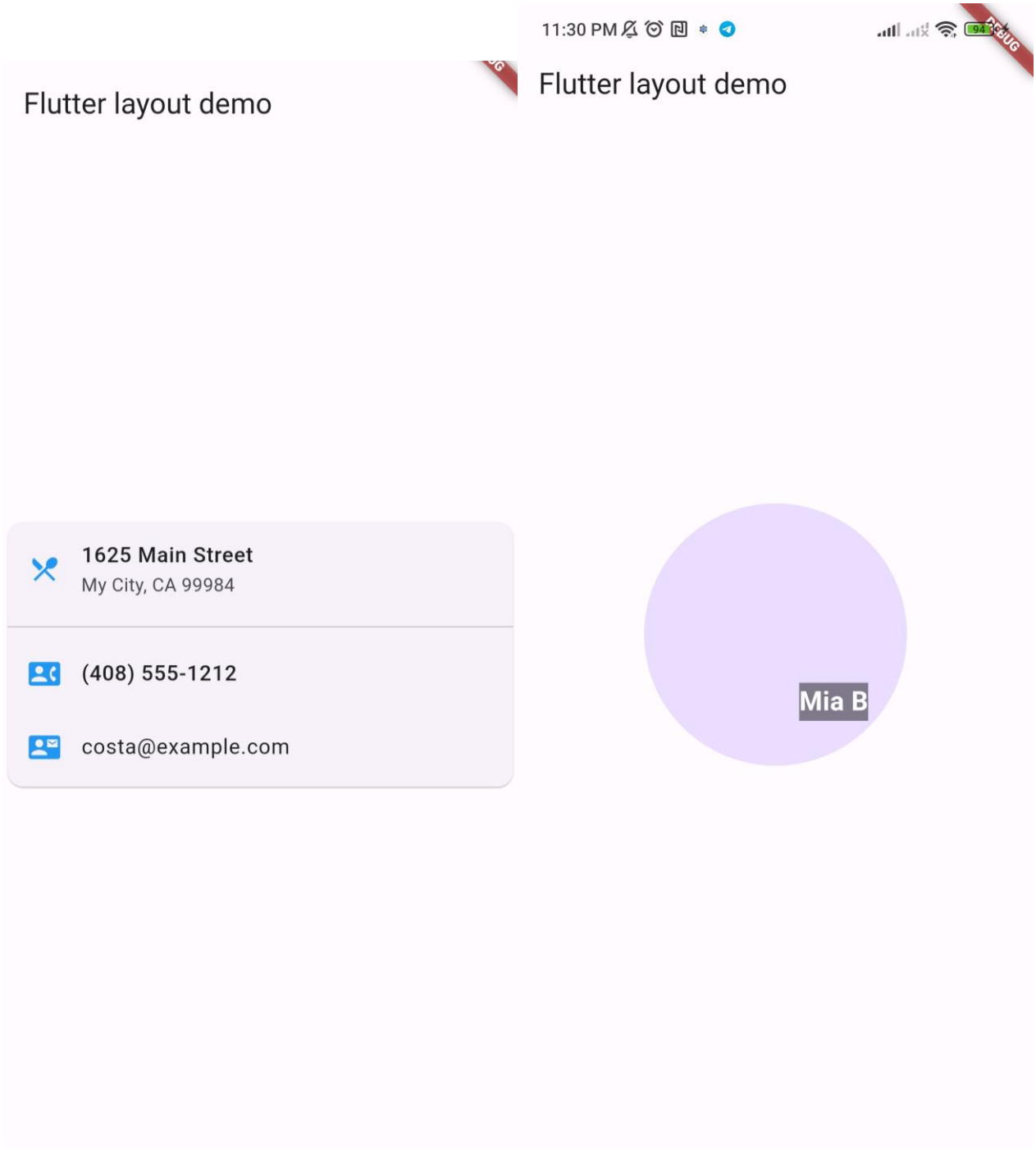
ListTile:

Es una Row especializada que contiene hasta 3 líneas de texto e iconos opcionales.

Es menos configurable que Row, pero más fácil de usar.

Pertenece a la biblioteca Materia

Por lo que tenemos la siguiente vista:



Siendo el código el siguiente:

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

import 'package:flutter/rendering.dart' show debugPaintSizeEnabled;

void main() {
  debugPaintSizeEnabled = false; // Set to true for visual layout

  runApp(const MyApp());
}
```

```dart
class MyApp extends StatelessWidget {
  const MyApp({super.key});

  static const showCard = true; // Set to false to show Stack

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Flutter layout demo',
      home: Scaffold(
        appBar: AppBar(
          title: const Text('Flutter layout demo'),
        ),
        body: Center(child: false ? _buildCard() : _buildStack()),
      ),
    );
  }

// #docregion Card

  Widget _buildCard() {
    return SizedBox(
      height: 210,
      child: Card(
        child: Column(
          children: [
            ListTile(
              title: const Text(
                '1625 Main Street',
                style: TextStyle(fontWeight: FontWeight.w500),
              ),
              subtitle: const Text('My City, CA 99984'),
              leading: Icon(
                Icons.restaurant_menu,
                color: Colors.blue[500],
              ),
            ),
            const Divider(),
            ListTile(
              title: const Text(
                '(408) 555-1212',
                style: TextStyle(fontWeight: FontWeight.w500),
              ),
              leading: Icon(
                Icons.contact_phone,
                color: Colors.blue[500],
              ),
            ),
            ListTile(
              title: const Text('costa@example.com'),
              leading: Icon(
                Icons.contact_mail,
                color: Colors.blue[500],
              ),
```

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

// #enddocregion Card

// #docregion Stack

  Widget _buildStack() {
    return Stack(
      alignment: const Alignment(0.6, 0.6),
      children: [
        const CircleAvatar(
          backgroundImage: AssetImage('images/pic.jpg'),
          radius: 100,
        ),
        Container(
          decoration: const BoxDecoration(
            color: Colors.black45,
          ),
          child: const Text(
            'Mia B',
            style: TextStyle(
              fontSize: 20,
              fontWeight: FontWeight.bold,
              color: Colors.white,
            ),
          ),
        ),
      ],
    );
  }

// #enddocregion Stack
}
```