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.

```
Row(
  mainAxisAlignment: MainAxisAlignment.spaceEvenly,
  children: [
    Image.asset('images/pic1.jpg'),
    Image.asset('images/pic2.jpg'),
    Image.asset('images/pic3.jpg'),
  ],
);
```

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.

```
Column(
  mainAxisAlignment: MainAxisAlignment.spaceEvenly,
  children: [
    Image.asset('images/pic1.jpg'),
    Image.asset('images/pic2.jpg'),
    Image.asset('images/pic3.jpg'),
  ],
);
```

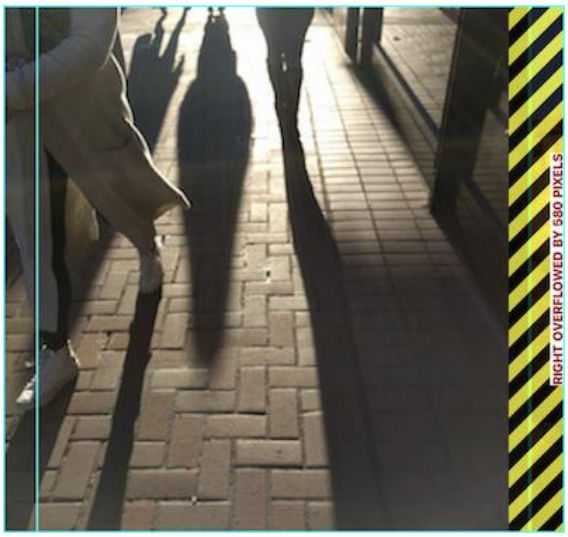
//

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

void main() {
  debugPaintSizeEnabled = true; // Remove to suppress visual layout
  runApp(MyApp());
}

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Flutter layout demo',
      home: Scaffold(
        appBar: AppBar(
          title: Text('Flutter layout demo'),
        ),
        // Change to buildColumn() for the other column example
        body: Center(child: buildRow()),
        // body: Center(child: buildColumn()),
      ),
    );
  }

  Widget buildRow() =>
      // #docregion Row
      Row(
        mainAxisAlignment: MainAxisAlignment.spaceEvenly,
        children: [
          Image.asset('images/pic1.jpg'),
          Image.asset('images/pic2.jpg'),
          Image.asset('images/pic3.jpg'),
        ],
      );
  // #enddocregion Row

    Widget buildColumn() =>
      // #docregion Column
      Column(
        mainAxisAlignment: MainAxisAlignment.spaceEvenly,
        children: [
          Image.asset('images/pic1.jpg'),
          Image.asset('images/pic2.jpg'),
          Image.asset('images/pic3.jpg'),
        ],
      );
```

//

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".

```
Row(
  crossAxisAlignment: CrossAxisAlignment.center,
  children: [
    Expanded(
      child: Image.asset('images/pic1.jpg'),
    ),
    Expanded(
      child: Image.asset('images/pic2.jpg'),
    ),
    Expanded(
      child: Image.asset('images/pic3.jpg'),
    ),
  ],
);
```

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:

```
Row(
  crossAxisAlignment: CrossAxisAlignment.center,
  children: [
```

```
    Expanded(
      child: Image.asset('images/pic1.jpg'),
    ),
    Expanded(
      flex: 2,
      child: Image.asset('images/pic2.jpg'),
    ),
    Expanded(
      child: Image.asset('images/pic3.jpg'),
    ),
  ],
);
```

//

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

void main() {
  debugPaintSizeEnabled = true; // Remove to suppress visual layout
  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'),
        ),
        // Change to buildFoo() for the other examples
        body: Center(child: buildExpandedImages()),
      ),
    );
  }

  Widget buildOverflowingRow() =>
```

```
    // #docregion overflowing-row
    Row(
      mainAxisAlignment: MainAxisAlignment.spaceEvenly,
      children: [
        Image.asset('images/pic1.jpg'),
        Image.asset('images/pic2.jpg'),
        Image.asset('images/pic3.jpg'),
      ],
    );
  // #enddocregion overflowing-row

  Widget buildExpandedImages() =>
    // #docregion expanded-images
    Row(
      crossAxisAlignment: CrossAxisAlignment.center,
      children: [
        Expanded(
          child: Image.asset('images/pic1.jpg'),
        ),
        Expanded(
          child: Image.asset('images/pic2.jpg'),
        ),
        Expanded(
          child: Image.asset('images/pic3.jpg'),
        ),
      ],
    );
  // #enddocregion expanded-images

  Widget buildExpandedImagesWithFlex() =>
    // #docregion expanded-images-with-flex
    Row(
      crossAxisAlignment: CrossAxisAlignment.center,
      children: [
        Expanded(
          child: Image.asset('images/pic1.jpg'),
        ),
        Expanded(
          flex: 2,
          child: Image.asset('images/pic2.jpg'),
        ),
        Expanded(
          child: Image.asset('images/pic3.jpg'),
        ),
      ],
    );
  // #enddocregion expanded-images-with-flex
}
```

//

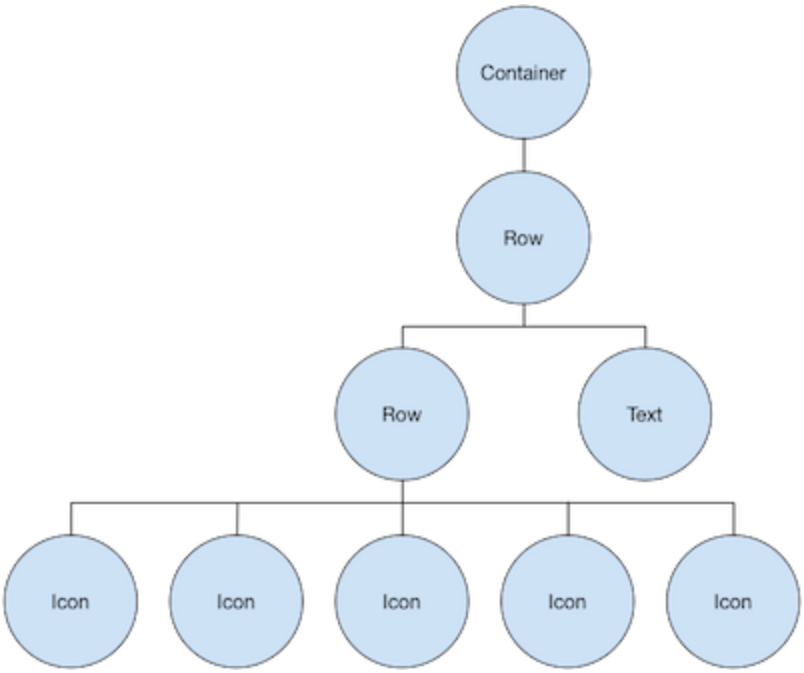
Packing widgets

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.

```
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),
  ],
)
```

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

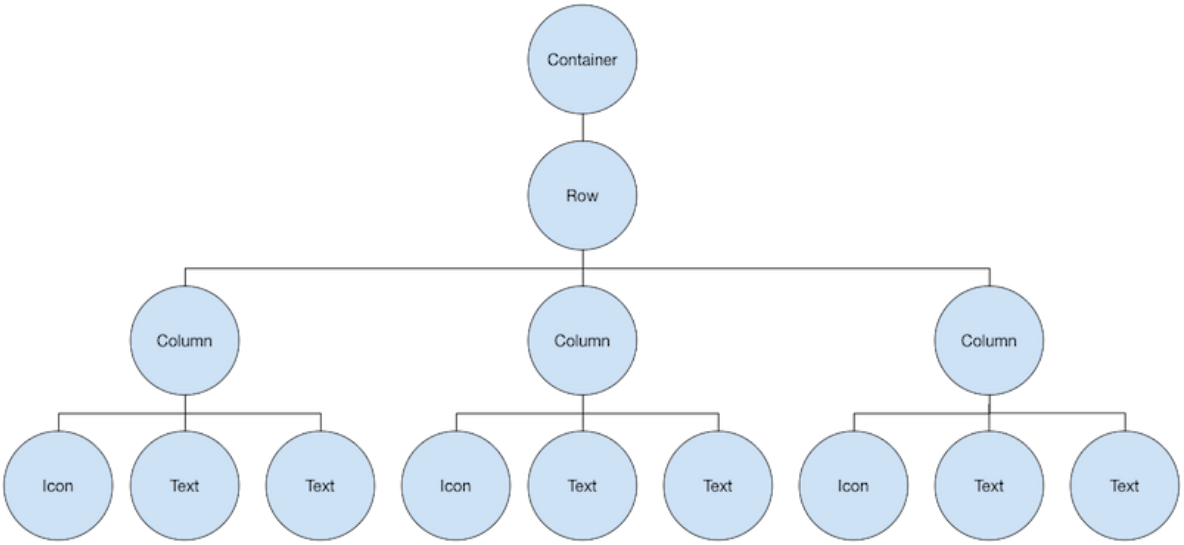
La sección resaltada se implementa como dos filas. La fila de calificaciones contiene cinco estrellas y el número de reseñas. La fila de iconos contiene tres columnas de iconos y texto. El árbol de widgets para la fila de calificaciones:



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

```
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: const 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,
        ),
      ),
    ],
  ),
);
```

La fila de iconos, debajo de la fila de calificaciones, contiene 3 columnas; cada columna contiene un icono y dos líneas de texto, como se puede ver en su árbol de widgets:



La variable iconList define la fila (row) de íconos:

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

// DefaultTextStyle.merge() allows you to create a default text
// style that is inherited by its child and all subsequent children.
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'),
          ],
        ),
      ],
    ),
  ),
);
```

La variable leftColumn contiene las calificaciones y la fila de íconos, así como el título y texto que describe la Pavlova:

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

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().

```
body: Center(
  child: Container(
    margin: const EdgeInsets.fromLTRB(0, 40, 0, 30),
    height: 600,
    child: Card(
      child: Row(
        crossAxisAlignment: CrossAxisAlignment.start,
        children: [
          SizedBox(
            width: 440,
            child: leftColumn,
          ),
          mainImage,
        ],
      ),
```

```
    ),
  ),
),
```

```
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
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});

  @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 = 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,
      ),
    );

    // #docregion ratings, stars
    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),
    ],
  );
  // #enddocregion stars

  final ratings = Container(
    padding: const 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,
          ),
        ),
      ],
    ),
  );
  // #enddocregion ratings

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

  // DefaultTextStyle.merge() allows you to create a default text
  // style that is inherited by its child and all subsequent children.
  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'),
            ],
          ),
        ],
      ),
    ),
  );
  // #enddocregion iconList

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

  final mainImage = Image.asset(
    'images/pavlova.jpg',
    fit: BoxFit.cover,
  );

  return Scaffold(
    appBar: AppBar(
      title: Text(title),
```

```
      ),
      // #docregion body
      body: Center(
        child: Container(
          margin: const EdgeInsets.fromLTRB(0, 40, 0, 30),
          height: 600,
          child: Card(
            child: Row(
              crossAxisAlignment: CrossAxisAlignment.start,
              children: [
                SizedBox(
                  width: 440,
                  child: leftColumn,
                ),
                mainImage,
              ],
            ),
          ),
        ),
      ),
      // #enddocregion body
    );
  }
}
```

//

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.

Widgets estándar

- Container: Agrega relleno, márgenes, bordes, color de fondo u otras decoraciones a un widget.
- GridView: Distribuye widgets en una cuadrícula desplazable.

- ListView: Organiza widgets en una lista desplazable.
- Stack: Superpone un widget sobre otro.

Widgets de Material

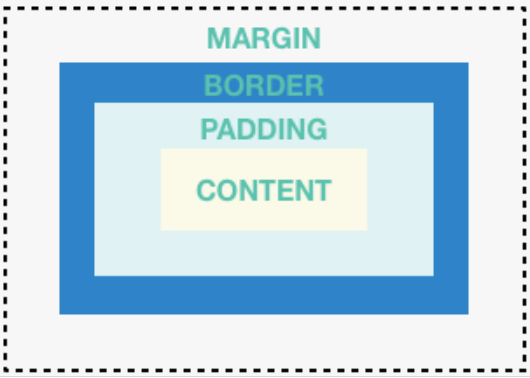
- Card: Organiza información relacionada en un cuadro con esquinas redondeadas y una sombra.
- ListTile: Organiza hasta 3 líneas de texto y, opcionalmente, iconos principales y secundarios en una fila.

Contenedor

Muchos diseños hacen un uso extensivo de Container's para separar widgets mediante relleno (padding) o para agregar bordes o márgenes. Puedes cambiar el fondo del dispositivo colocando todo el diseño en un Contenedor y cambiando su color de fondo o imagen.

Resumen (Container)

- Agrega relleno, márgenes, bordes.
- Cambia el color de fondo o la imagen.
- Contiene un único widget hijo, pero ese hijo puede ser un Row, Column o incluso la raíz de un árbol de widgets.



Ejemplos:

Este diseño consta de una columna con dos filas, cada una con 2 imágenes. Se utiliza un Contenedor para cambiar el color de fondo de la columna a un gris más claro.

```
Widget _buildImageColumn() {
  return Container(
    decoration: const BoxDecoration(
      color: Colors.black26,
    ),
    child: Column(
      children: [
        _buildImageRow(1),
```

```
        _buildImageRow(3),
      ],
    ),
  );
}
```

Un Container también es utilizado para agregar bordes redondeados y márgenes para cada imagen:

```
Widget _buildDecoratedImage(int imageIndex) => Expanded(
      child: Container(
        decoration: BoxDecoration(
          border: Border.all(width: 10, color: Colors.black38),
          borderRadius: const BorderRadius.all(Radius.circular(8)),
        ),
        margin: const EdgeInsets.all(4),
        child: Image.asset('images/pic$imageIndex.jpg'),
      ),
    );

Widget _buildImageRow(int imageIndex) => Row(
      children: [
        _buildDecoratedImage(imageIndex),
        _buildDecoratedImage(imageIndex + 1),
      ],
    );
```

Vista de cuadrícula (Grid View)

Utiliza GridView para organizar widgets en forma de una lista bidimensional. GridView proporciona dos listas preconfiguradas, o puedes crear tu propia cuadrícula personalizada. Cuando un GridView detecta que su contenido es demasiado largo para ajustarse a la caja de renderización, realiza un desplazamiento automático.

GridView:

Organiza widgets en una cuadrícula.

Detecta cuando el contenido de la columna excede la caja de renderización y proporciona desplazamiento automático.

Crea tu propia cuadrícula personalizada o utiliza una de las cuadrículas proporcionadas:

- GridView.count te permite especificar el número de columnas.

- GridView.extent te permite especificar el ancho máximo en píxeles de una casilla.

Usa GridView.extent para crear una cuadrícula con casillas de un máximo de 150 píxeles de ancho.

Usa GridView.count para crear una cuadrícula que tenga 2 casillas de ancho en modo retrato y 3 casillas de ancho en modo apaisado. Los títulos se crean configurando la propiedad footer para cada GridTile.

```dart
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.asset('images/pic$i.jpg')));
```

ListView

ListView, un widget similar a una columna, proporciona automáticamente la capacidad de desplazamiento cuando su contenido es demasiado extenso para su caja de representación.

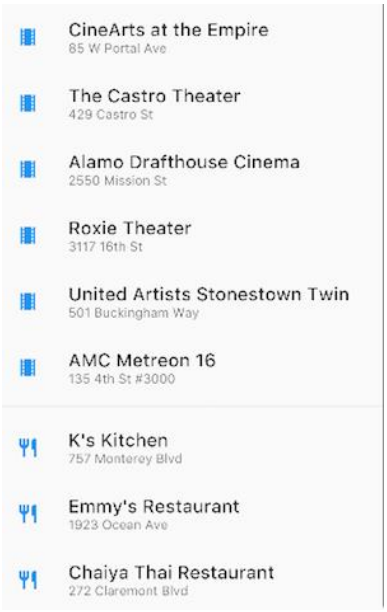
ListView:

Es un Column especializado para organizar una lista de elementos.

Puede ser dispuesto horizontal o verticalmente.

Detecta cuando su contenido no cabe y proporciona desplazamiento.

Menos configurable que Column, pero más fácil de usar y admite desplazamiento.



La imagen previa utiliza ListView para mostrar una lista de negocios utilizando ListTiles. Un separador (Divider) separa los teatros de los restaurantes.

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

```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.asset('images/pic$i.jpg')));
// #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
}
```

Stack

Se utiliza Stack para organizar widgets encima de un widget base, que a menudo es una imagen. Los widgets pueden superponerse completamente o parcialmente al widget base.

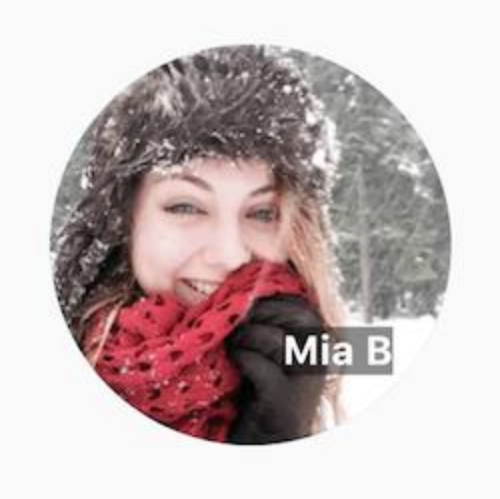
Stack:

Se utiliza para widgets que se superponen a otro widget.

El primer widget en la lista de hijos es el widget base; los hijos subsiguientes se superponen encima de ese widget base.

El contenido de un Stack no se puede desplazar.

Puedes optar por recortar los hijos que exceden la caja de representación.



En la imagen previa se utiliza Stack para superponer un Contenedor (que muestra su texto en un fondo negro translúcido) sobre un CircleAvatar. El Stack ajusta la posición del texto mediante la propiedad de alineación y Alinamientos (Alignments)

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

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.

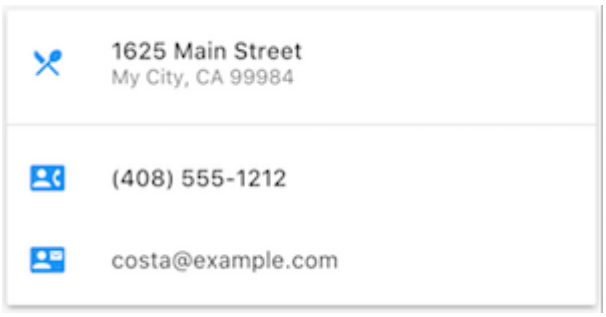
Acepta un único hijo, pero ese hijo puede ser una fila, columna u otro widget que contenga una lista de hijos.

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

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

```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 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: showCard ? _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
}
```