LABORATORIO 01	01 – introducción a flutter
OBJETIVO GENERAL	Conocer como funciona la aplicación
OBJETIVO ESPECÍFICO	Identificar que función tienen los widgets en flutter
INSTRUCCIONES	Vista la página Widgets Básicos de Flutter y describe para qué sirve cada uno, además de colocar la captura del ejemplo que la página oficial ofrece.
NOMBRE ESTUDIANTE	Itsai Zempoaltecatl Mejia
FECHA	21 febrero 2022
GRUPO	8SA

AppBar

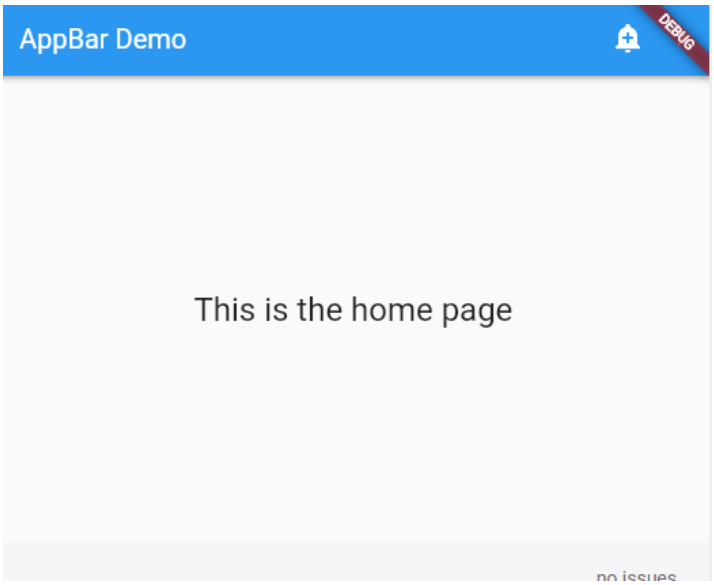
Un AppBar es una barra de herramientas y otros widgets. Las AppBars suelen tener una o mas acciones con IconButtons, que opcionalmente van seguidas de un PopupMenuButton para operaciones menos comunes.

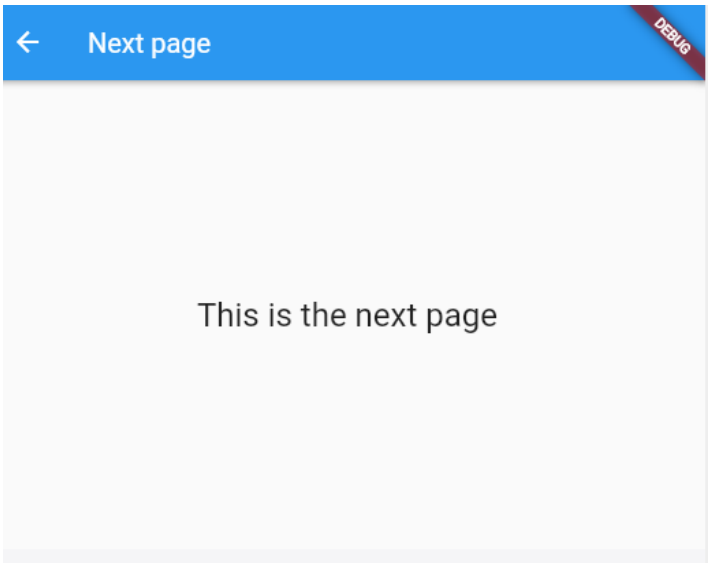
AppBar inserta su contenido en función del relleno de MediaQuery ambiental, para evitar intrusiones en la interfaz de usuario del sistema. Scaffold se encarga de ello cuando se usa en la propiedad Scaffold.appBar. Al animar una barra de aplicaciones, los cambios inesperados de MediaQuery (como es común en las animaciones de Hero) pueden hacer que el contenido salte repentinamente. Envuelva AppBar en un widget de MediaQuery y ajuste su relleno para que la animación sea fluida.

Este ejemplo muestra una AppBar con dos acciones simples. La primera acción abre un SnackBar, mientras que la segunda acción navega a una nueva página.

```dart
class MyStatelessWidget extends StatelessWidget {
  const MyStatelessWidget({Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text('AppBar Demo'),
        actions: <Widget>[
          IconButton(
            icon: const Icon(Icons.add_alert),
            tooltip: 'Show Snackbar',
            onPressed: () {
              ScaffoldMessenger.of(context).showSnackBar(
                  const SnackBar(content: Text('This is a snackbar')));
            },
          ),
          IconButton(
            icon: const Icon(Icons.navigate_next),
            tooltip: 'Go to the next page',
            onPressed: () {
              Navigator.push(context, MaterialPageRoute<void>(
                builder: (BuildContext context) {
                  return Scaffold(
                    appBar: AppBar(
                      title: const Text('Next page'),
                    ),
                    body: const Center(
                      child: Text(
                        'This is the next page',
                        style: TextStyle(fontSize: 24),
                      ),
                    ),
                  );
                },
              ));
            },
          ),
        ],
      ),
      body: const Center(
        child: Text(
          'This is the home page',
          style: TextStyle(fontSize: 24),
        ),
      ),
    );
  }
}
```





SnackBar

Column

Es un widget que muestra sus elementos hijos en una matriz vertical.

Este ejemplo usa una columna para organizar tres widgets verticalmente, el último hecho para llenar todo el espacio restante.

```
Column(
  children: const <Widget>[
    Text('Deliver features faster'),
    Text('Craft beautiful UIs'),
    Expanded(
      child: FittedBox(
        fit: BoxFit.contain, // otherwise the logo will be tiny
        child: FlutterLogo(),
      ),
    ),
  ],
)
```

En el ejemplo anterior, el texto y el logotipo están centrados en cada línea. En el siguiente ejemplo, crossAxisAlignment se establece en CrossAxisAlignment.start, de modo que los elementos secundarios estén alineados a la izquierda. mainAxisSize se establece en MainAxisSize.min, de modo que la columna se reduce para adaptarse a los elementos secundarios.

```
Column(
  crossAxisAlignment: CrossAxisAlignment.start,
  mainAxisSize: MainAxisSize.min,
  children: <Widget>[
    const Text('We move under cover and we move as one'),
    const Text('Through the night, we have one shot to live another day'),
    const Text('We cannot let a stray gunshot give us away'),
    const Text('We will fight up close, seize the moment and stay in it'),
    const Text('It's either that or meet the business end of a bayonet'),
    const Text('The code word is 'Rochambeau,' dig me?'),
    Text('Rochambeau!', style: DefaultTextStyle.of(context).style.apply(fontSizeFactor: 2.0)),
  ],
)
```

We move under cover and we move as one
Through the night, we have one shot to live another day
We cannot let a stray gunshot give us away
We will fight up close, seize the moment and stay in it
It's either that or meet the business end of a bayonet
The code word is 'Rochambeau,' dig me?
Rochambeau!

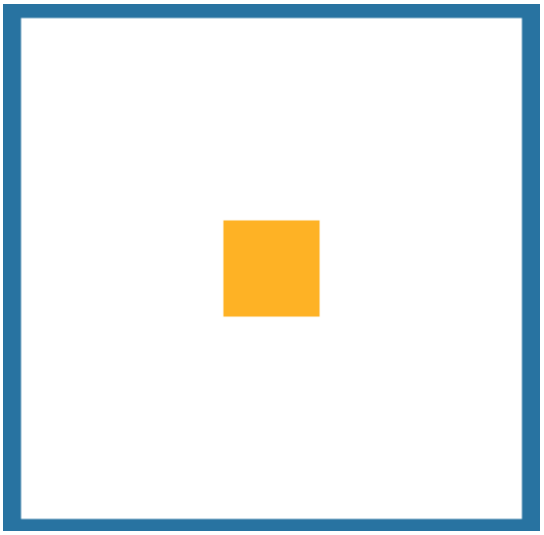
Container

Un conveniente widget que combina widgets comunes de pintura, posicionamiento y tamaño.

Un contenedor primero rodea al elemento secundario con relleno (inflado por cualquier borde presente en la decoración) y luego aplica restricciones adicionales a la extensión del relleno (incorporando el ancho y la altura como restricciones, si alguno no es nulo). Luego, el contenedor se rodea con un espacio vacío adicional descrito desde el margen.

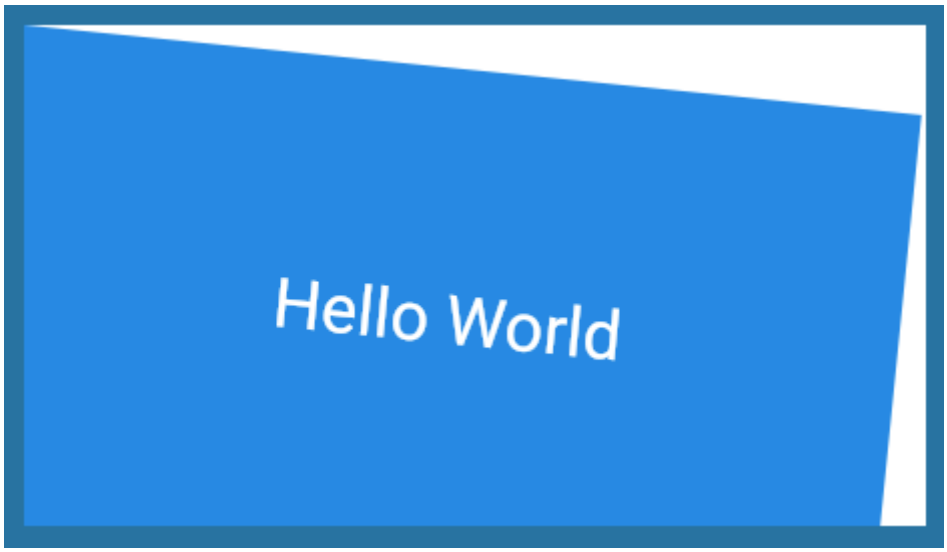
Este ejemplo muestra un cuadrado ámbar de 48x48 (colocado dentro de un widget central en caso de que el widget principal tenga sus propias opiniones sobre el tamaño que debe tomar el contenedor), con un margen para que se mantenga alejado de los widgets vecinos

```
Center(
  child: Container(
    margin: const EdgeInsets.all(10.0),
    color: Colors.amber[600],
    width: 48.0,
    height: 48.0,
  ),
)
```



Este ejemplo muestra cómo usar muchas de las funciones de Container a la vez. Las restricciones se establecen para ajustarse al tamaño de la fuente más un amplio espacio verticalmente, mientras se expanden horizontalmente para adaptarse al padre. El relleno se utiliza para asegurarse de que haya espacio entre el contenido y el texto. El color hace que la caja sea azul. La alineación hace que el niño quede centrado en el cuadro. Finalmente, la transformación aplica una ligera rotación a todo el artilugio para completar el efecto.

```
Container(
  constraints: BoxConstraints.expand(
    height: Theme.of(context).textTheme.headline4!.fontSize! * 1.1 + 200.0,
  ),
  padding: const EdgeInsets.all(8.0),
  color: Colors.blue[600],
  alignment: Alignment.center,
  child: Text('Hello World',
    style: Theme.of(context)
        .textTheme
        .headline4!
        .copyWith(color: Colors.white)),
  transform: Matrix4.rotationZ(0.1),
)
```



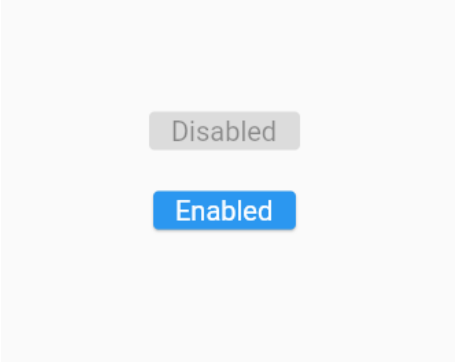
ElevatedButton

Un ElevatedButton es una etiqueta secundaria que se muestra en un widget Material cuya Material.elevation aumenta cuando se presiona el botón. Los widgets de texto e icono de la etiqueta se muestran en ButtonStyle.foregroundColor del estilo y el fondo relleno del botón es ButtonStyle.backgroundColor. El estilo predeterminado del botón elevado está definido por defaultStyleOf. El estilo de este botón elevado se puede anular con su parámetro de estilo. El estilo de todos los botones elevados en un subárbol se puede anular con ElevatedButtonTheme, y el estilo de todos los botones elevados en una aplicación se puede anular con la propiedad ThemeData.elevatedButtonTheme del tema.

Si las devoluciones de llamada onPressed y onLongPress son nulas, el botón se desactivará

```
final ButtonStyle style =
    ElevatedButton.styleFrom(textStyle: const TextStyle(fontSize: 20));

return Center(
  child: Column(
    mainAxisSize: MainAxisSize.min,
    children: <Widget>[
      ElevatedButton(
        style: style,
        onPressed: null,
        child: const Text('Disabled'),
      ),
      const SizedBox(height: 30),
      ElevatedButton(
        style: style,
        onPressed: () {},
        child: const Text('Enabled'),
      ),
    ],
```



FlutterLogo

El logo de Flutter, en forma de widget. Este widget respeta el IconTheme.

```
const FlutterLogo({
  Key? key,
  this.size,
  this.textColor = const Color(0xFF757575),
  this.style = FlutterLogoStyle.markOnly,
  this.duration = const Duration(milliseconds: 750),
  this.curve = Curves.fastOutSlowIn,
}) : assert(textColor != null),
     assert(style != null),
     assert(duration != null),
     assert(curve != null),
     super(key: key);
```

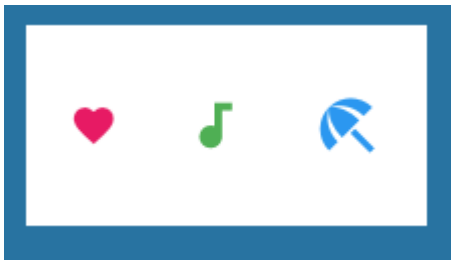
Icon

Un widget de ícono gráfico dibujado con un glifo de una fuente descrita en un IconData, como los IconData predefinidos del material en los íconos.

Los iconos no son interactivos. Para un ícono interactivo, considere el IconButton del material.

Este ejemplo muestra cómo crear una fila de iconos en diferentes colores y tamaños. El primer ícono usa una etiqueta semántica para anunciar en modos de accesibilidad como TalkBack y VoiceOver.

```
Row(
  mainAxisAlignment: MainAxisAlignment.spaceAround,
  children: const <Widget>[
    Icon(
      Icons.favorite,
      color: Colors.pink,
      size: 24.0,
      semanticLabel: 'Text to announce in accessibility modes',
    ),
    Icon(
      Icons.audiotrack,
      color: Colors.green,
      size: 30.0,
    ),
    Icon(
      Icons.beach_access,
      color: Colors.blue,
      size: 36.0,
    ),
  ],
)
```



Image

Un widget que muestra una imagen.

El constructor predeterminado se puede usar con cualquier ImageProvider, como NetworkImage, para mostrar una imagen de Internet.

```
const Image(
  image: NetworkImage('https://flutter.github.io/assets-for-api-docs/assets/widgets/owl.jpg'),
)
```

Image Widget también proporciona varios constructores para mostrar diferentes tipos de imágenes para mayor comodidad. En este ejemplo, use el constructor Image.network para mostrar una imagen de Internet.

```
Image.network('https://flutter.github.io/assets-for-api-docs/assets/widgets/owl-2.jpg')
```

PlaceHolder

Un widget que dibuja un cuadro que representa dónde se agregarán otros widgets algún día.

Este widget es útil durante el desarrollo para indicar que la interfaz aún no está completa.

De forma predeterminada, el tamaño del marcador de posición se adapta a su contenedor. Si el marcador de posición se encuentra en un espacio ilimitado, se ajustará a sí mismo de acuerdo con el FallbackWidth y FallbackHeight dados.

```
const Placeholder({
  Key? key,
  this.color = const Color(0xFF455A64), // Blue Grey 700
  this.strokeWidth = 2.0,
  this.fallbackWidth = 400.0,
  this.fallbackHeight = 400.0,
}) : super(key: key);
```

Row

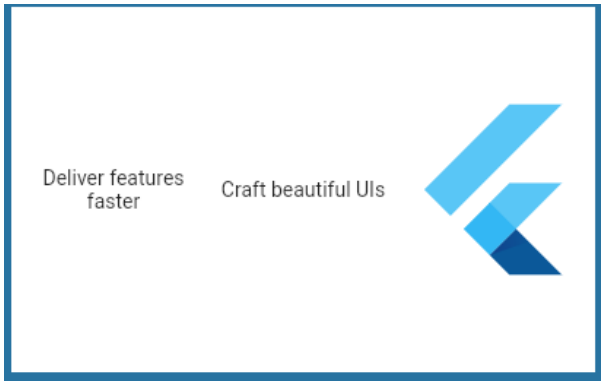
Un widget que muestra a sus elementos secundarios en una matriz horizontal.

Para hacer que un elemento secundario se expanda para llenar el espacio horizontal disponible, envuelva al elemento secundario en un widget Expandido.

El widget de Fila no se desplaza.

Este ejemplo divide el espacio disponible en tres (horizontalmente) y coloca el texto centrado en las dos primeras celdas y el logotipo de Flutter centrado en la tercera.

```
Row(
  children: const <Widget>[
    Expanded(
      child: Text('Deliver features faster', textAlign: TextAlign.center),
    ),
    Expanded(
      child: Text('Craft beautiful UIs', textAlign: TextAlign.center),
    ),
    Expanded(
      child: FittedBox(
        fit: BoxFit.contain, // otherwise the logo will be tiny
        child: FlutterLogo(),
      ),
    ),
  ],
)
```



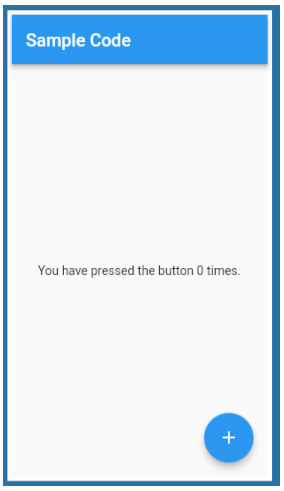
Scaffold

Implementa la estructura de disposición visual básica del diseño de materiales.

Esta clase proporciona API para mostrar cajones y hojas inferiores.

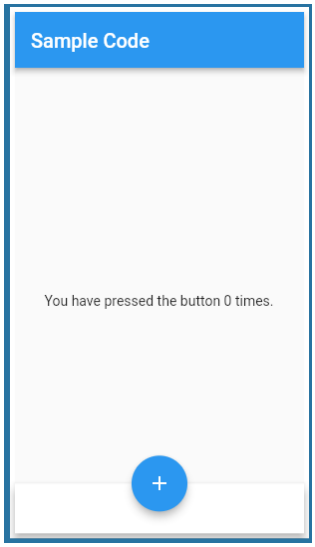
Para mostrar una hoja inferior persistente, obtenga el ScaffoldState para el BuildContext actual a través de Scaffold.of y use la función ScaffoldState.showBottomSheet.

Este ejemplo muestra un Scaffold con un cuerpo y FloatingActionButton. El cuerpo es un Texto colocado en un Centro para centrar el texto dentro del Andamio. El FloatingActionButton está conectado a una devolución de llamada que aumenta un contador.




```
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      title: const Text('Sample Code'),
    ),
    body: Center(child: Text('You have pressed the button $_count times.')),
    floatingActionButton: FloatingActionButton(
      onPressed: () => setState(() => _count++),
      tooltip: 'Increment Counter',
      child: const Icon(Icons.add),
    ),
  );
}
```

Este ejemplo muestra un Scaffold con un AppBar, un BottomAppBar y un FloatingActionButton. El cuerpo es un Texto colocado en un Centro para centrar el texto dentro del Andamio. El FloatingActionButton se centra y acopla dentro de BottomAppBar usando FloatingActionButtonLocation.centerDocked. El FloatingActionButton está conectado a una devolución de llamada que incrementa un contador.




```
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      title: const Text('Sample Code'),
    ),
    body: Center(
      child: Text('You have pressed the button $_count times.'),
    ),
    bottomNavigationBar: BottomAppBar(
      shape: const CircularNotchedRectangle(),
      child: Container(height: 50.0),
    ),
    floatingActionButton: FloatingActionButton(
      onPressed: () => setState(() {
        _count++;
      }),
      tooltip: 'Increment Counter',
      child: const Icon(Icons.add),
    ),
    floatingActionButtonLocation: FloatingActionButtonLocation.centerDocked,
  );
```

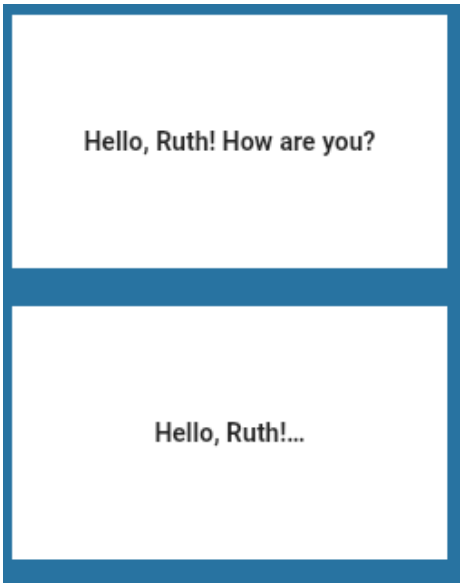
Text

El widget de texto muestra una cadena de texto con un solo estilo. La cadena puede dividirse en varias líneas o puede mostrarse en la misma línea según las restricciones de diseño.

El argumento de estilo es opcional. Cuando se omite, el texto usará el estilo del DefaultTextStyle más cercano. Si la propiedad TextStyle.inherit del estilo dado es verdadera (el valor predeterminado), el estilo dado se fusionará con el DefaultTextStyle más cercano. Este comportamiento de combinación es útil, por ejemplo, para poner el texto en negrita mientras se usa la familia de fuentes y el tamaño predeterminados.

Este ejemplo muestra cómo mostrar texto usando el widget Texto con el desbordamiento establecido en TextOverflow.ellipsis.

```
Text(
  'Hello, $_name! How are you?',
  textAlign: TextAlign.center,
  overflow: TextOverflow.ellipsis,
  style: const TextStyle(fontWeight: FontWeight.bold),
)
```



Con el constructor Text.rich, el widget Text puede mostrar un párrafo con TextSpans con estilos diferentes. El ejemplo que sigue muestra "Hola mundo hermoso" con diferentes estilos para cada palabra.

```
const Text.rich(
  TextSpan(
    text: 'Hello', // default text style
    children: <TextSpan>[
      TextSpan(text: ' beautiful ', style: TextStyle(fontStyle: FontStyle.italic)),
      TextSpan(text: 'world', style: TextStyle(fontWeight: FontWeight.bold)),
    ],
  ),
)
```