Ir al archivo **lib/main.dart** y borrar todo

Crear el siguiente código:

lib > main.dart > main

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

Run | Debug | Profile
void main() {
  runApp(MyApp());
}

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: Center(child: Text('Hola Mundo')),
    ); // MaterialApp
  }
}
```

Crear un scaffold: https://api.flutter.dev/flutter/material/Scaffold-class.html
Realizar lo siguiente:

```dart
class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: Scaffold(
        body: Center(child: Text('Hola Mundo')),
      ) // Scaffold
    ); // MaterialApp
  }
}
```

Crear una llave e identificar cada widget
Dar clic **Ctrl + .** para corregir el warning de **MyApp**

Agregar el performance:
agregar **const** para indicar que nunca va a cambiar

```
10    @override
11    Widget build(BuildContext context) {
12      return const MaterialApp(
13        home: Scaffold(
14          body: Center(child: Text('Hola Mundo')),
15        ) // Scaffold
16      ); // MaterialApp
17    }
18  }
```

Type: String

Quitar el tag debug

```
return const MaterialApp(
  debugShowCheckedModeBanner: false,
```

ESTRUCTURA DE DIRECTORIOS

Crear una carpeta en **/lib/presentation**: Cualquier cosa visual debe de ir aqui
Crear una carpeta llamada **/lib/presentation/screens**: Widget que cubre toda la pantalla
Crear un archivo llamado **counter_screen.dart**

Dentro del archivo agregar material tecleando **impm**
Agregar el stalessWidget tecleando **styles**
cambiar el nombre del Widget a **CounterScreen**

Modificar el archivo de la siguiente manera:

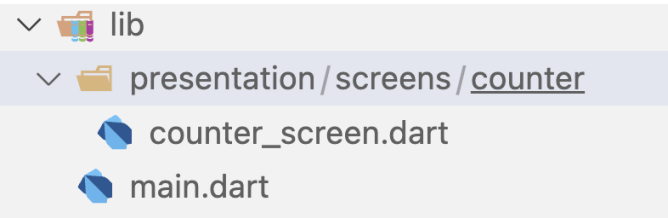
```
1  import 'package:flutter/material.dart';
2
3  class CounterScreen extends StatelessWidget {
4    const CounterScreen({super.key});
5
6    @override
7    Widget build(BuildContext context) {
8      return const Scaffold(
9        body: Center(child: Text('Counter Screen')),
10     ); // Scaffold
11   }
12 }
```

En el archivo **main.dart** modificarlo de la siguiente manera:

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

  @override
  Widget build(BuildContext context) {
    return const MaterialApp(
      debugShowCheckedModeBanner: false,
      home: CounterScreen()
    ); // MaterialApp
```

Crear un directorio en **screens/counter** y mover el archivo **counter_screen.dart**

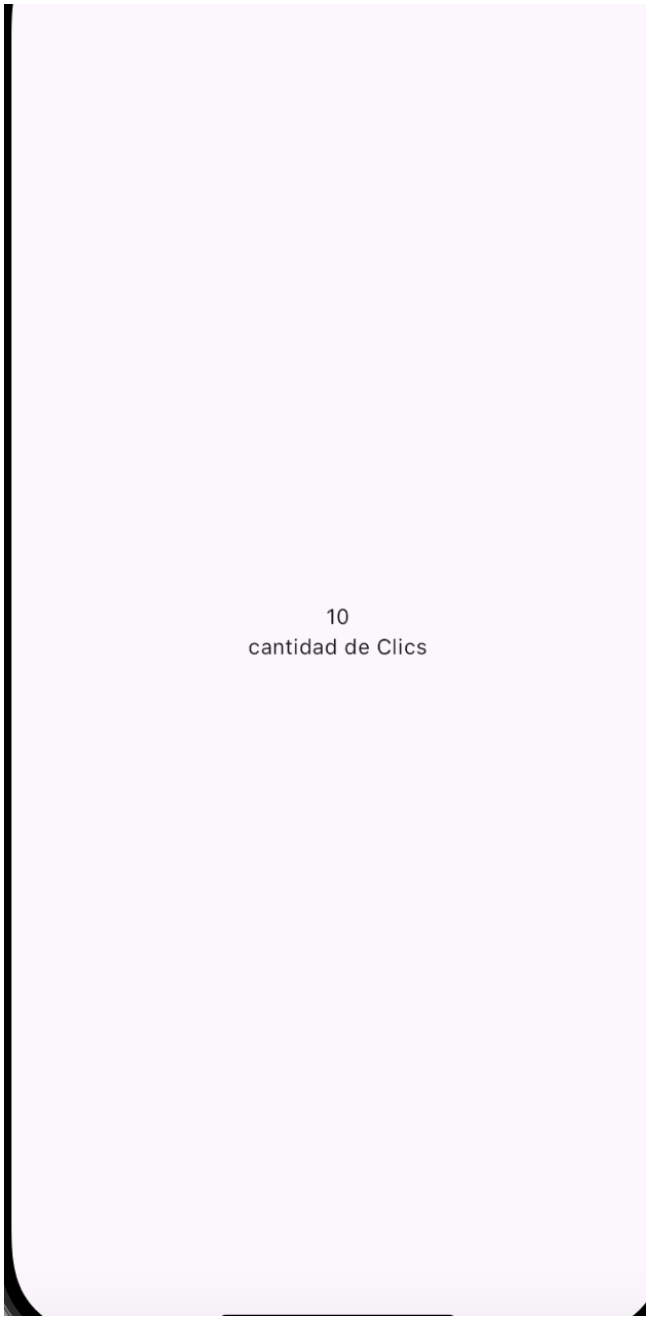


DISEÑO DE PANTALLA

Agregar lo siguiente

```dart
@override
Widget build(BuildContext context) {
  return const Scaffold(
    body: Center(
      child: Column(
        mainAxisAlignment: MainAxisAlignment.center,
        children: [
          Text('10'),
          Text('cantidad de Clics')
        ],
      ) // Column
    ) // Center
  ); // Scaffold
```

La app se debe ver de la siguiente manera:



Agregar el FloatingActionButton

```
7    Widget build(BuildContext context) {
8      return Scaffold(
9        body: const Center( // Center ⋯
18       floatingActionButton: FloatingActionButton(
19         onPressed: () {
20
21         },
22         child: const Icon( Icons.plus_one ),
23       ), // FloatingActionButton
24     ); // Scaffold
25   }
26 }
```

Modificar los estilos de los textos:

```
children: [
  Text('10', style: TextStyle(fontSize: 160, fontWeight: FontWeight.w100)
  Text('cantidad de Clics', style: TextStyle(fontSize: 25),)
],
```

CAMBIAR EL TEMA

Ir a main.dart y modificarlo de la siguiente manera:

```
11    @override
12    Widget build(BuildContext context) {
13      return MaterialApp(
14        debugShowCheckedModeBanner: false,
15        theme: ThemeData(
16          useMaterial3: true,
17          colorSchemeSeed: Colors.green
18        ), // ThemeData
19        home: const CounterScreen()
20      ); // MaterialApp
21    }
22  }
```

CAMBIAR EL ESTADO DE LA APLICACIÓN

Mantener el cursor en **StatelessWidget** y teclear **Ctrl + .** para convertirlo en un **StatefulWidget**

Definición: Es similar al stateless en cuanto a que es un widget, pero este permite mantener un estado interno y ciclo de vida como su inicialización y destrucción. Se utilizan mucho para las animaciones

```
class CounterScreen extends StatefulWidget {
  const CounterScreen({super.key});

  @override
  State<CounterScreen> createState() => _CounterScreenState();
}
```

Agregar la variable de contador

```
class _CounterScreenState extends State<CounterScreen> {
  int clickCounter = 0;
```

Modificar el texto de la siguiente manera:

```
body: Center(
  child: Column(
    mainAxisAlignment: MainAxisAlignment.center,
    children: [
      Text('$clickCounter', style: const TextStyle(fontSize: 160,
      const Text('cantidad de Clics', style: TextStyle(fontSize: 2
    ],
```

Agregar la función de clic:

```
floatingActionButton: FloatingActionButton(
  onPressed: () {
    clickCounter++;
  },
```

Agregar el estado para el cambio:

```
floatingActionButton: FloatingActionButton(
  onPressed: () {
    setState(() {
      clickCounter++;
    });
  },
```

Modificar los textos

```
child: Column(
  mainAxisAlignment: MainAxisAlignment.center,
  children: [
    Text('$clickCounter', style: const TextStyle(fontSize: 160, fontWeig
    Text('Click${ clickCounter == 1 ? '' : 's'}', style: const TextStyle
  ],
```

APPBAR Y ACCIONES

Crear un nuevo archivo **counter_functions_screen.dart** y copiar todo el contenido de la pantalla anterior

cambiar el nombre: mantener el cursor en la palabra **CounterScreen** y presionar **F2** para renombrar

```
class CounterFunctionScreen extends StatefulWidget {
  const CounterFunctionScreen({super.key});

  @override
  State<CounterFunctionScreen> createState() => _CounterF
}
```

Cambiarlo en **main.dart**

```
Widget build(BuildContext context) {
  return MaterialApp(
    debugShowCheckedModeBanner: false,
    theme: ThemeData(
      colorSchemeSeed: Colors.green
    ), // ThemeData
    home: const CounterFunctionScreen()
  ); // MaterialApp
```

Agregar el leading

```
appBar: AppBar(
  title: const Text('Counter Functions'),
  leading: IconButton(
    icon: const Icon(Icons.refresh_rounded),
    onPressed: () {  },), // IconButton
), // AppBar
```

Para agregarlo a la derecha agregar **actions**

```
appBar: AppBar(
  title: const Text('Counter Functions'),
  actions: [
    IconButton(
      icon: const Icon(Icons.refresh_rounded),
      onPressed: () {  },
    ), // IconButton
  ]
```

Modificar el **OnPressed** para reiniciar el contador:

```
actions: [
  IconButton(
    icon: const Icon(Icons.refresh_rounded),
    onPressed: () {
      setState(() {
        clickCounter = 0;
      });
    },
```

AGREGAR MÚLTIPLES ACCIONES
Modificar el **floatingActionButton**

```dart
floatingActionButton: Column(
  mainAxisAlignment: MainAxisAlignment.end,
  children: [
    FloatingActionButton(
      onPressed: () {
        setState(() {
          clickCounter++;
        });
      },
      child: const Icon( Icons.plus_one ),
    ), // FloatingActionButton
  ],
) // Column
```

Agregar el boton de reiniciar y disminuir (opcional: redondear el botón)

```dart
children: [
  FloatingActionButton(
    shape: const StadiumBorder(),
    onPressed: () {
      setState(() {
        clickCounter = 0;
      });
    },
    child: const Icon( Icons.refresh_rounded ),
  ), // FloatingActionButton
  const SizedBox(height: 10,),
```

```
FloatingActionButton(
  shape: const StadiumBorder(),
  onPressed: () {
    setState(() {
      clickCounter++;
    });
  },
  child: const Icon( Icons.plus_one ),
), // FloatingActionButton
const SizedBox(height: 10,),
FloatingActionButton(
  shape: const StadiumBorder(),
  onPressed: () {
    setState(() {
      clickCounter--;
    });
  },
  child: const Icon( Icons.exposure_minus_1_outlined ),
), // FloatingActionButton
```

WIDGET PERSONALIZADOS
Por el momento eliminar el contenido del OnPressed para generar el widget y luego seleccionar el widget y presionar **Ctrl + .** y **Extract Widget** y llamarlo **CustomButtom**

Modificar la clase **CustomButtom**

```
class CustomButton extends StatelessWidget {

  final IconData icon;
  const CustomButton(/*this.icon, */{
    super.key,
    required this.icon
  });

  @override
  Widget build(BuildContext context) {
    return FloatingActionButton(
      shape: const StadiumBorder(),
      onPressed: () {},
      child: Icon(icon),
    ); // FloatingActionButton
  }
```

Agregar el **final IconData = icon;**
Agregarlo al **constructor** y al **widget**

Modificar el **FloatingActionButton**

```
floatingActionButton: Column(
  mainAxisAlignment: MainAxisAlignment.end,
  children: [
    CustomButton(icon: Icons.refresh_outlined),
    const SizedBox(height: 10,),
    CustomButton(icon: Icons.plus_one),
    const SizedBox(height: 10,),
    CustomButton(icon: Icons.exposure_minus_1_outlined),
  ],
```

VOIDCALLBACK
Agregar la variable **onPressed**:

```
class CustomButton extends StatelessWidget {

  final IconData icon;
  final VoidCallback? onPressed;

  const CustomButton(/*this.icon, */{
    super.key,
    required this.icon,
    required this.onPressed,
  });
```

Agregar la función a todos los widgets

```
CustomButton(
    icon: Icons.refresh_outlined,
    onPressed: () {
      clickCounter = 0;
      setState(() {});
    },
), // CustomButton
const SizedBox(height: 10,),
CustomButton(
    icon: Icons.plus_one,
    onPressed: () {
      clickCounter++;
      setState(() {});
    },
```

Cambiar algunas propiedades del **CustomButtom**

```dart
@override
Widget build(BuildContext context) {
  return FloatingActionButton(
    // shape: const StadiumBorder(),
    enableFeedback: true,
    elevation: 5,
    onPressed: onPressed,
    child: Icon(icon),
  ); // FloatingActionButton
}
```