



# Flutter:


## *Fundamentos y Exploración*



[www.sena.edu.co](http://www.sena.edu.co)

# Nivel Básico: Fundamentos y Exploración



- Instalación y configuración del entorno de desarrollo 
- Sintaxis básica del lenguaje
- Fundamentos de algoritmos y estructuras de datos.
- Fundamentos de Flutter y su arquitectura.
- Diseño de interfaces de usuario con widgets básicos.



Flutter



Dart

https://esflutter.dev

# Crea aplicaciones para cualquier pantalla

The central part of the image is a screenshot of the esflutter.dev website. It features a blue gradient background with the heading "Crea aplicaciones para cualquier pantalla". Below the heading, there are several mockups of applications running on different device screens: a desktop monitor showing a "Vacation Inspiration" gallery, a smartphone displaying a car status app with "ALL GOOD" and a car image, another smartphone showing a weather and commute app with "65° Clear" and "20 min to work", a tablet showing a "Welcome to MGM Resorts" app, and a smaller smartphone showing a fitness app with a bar chart. At the bottom, there is a paragraph in Spanish explaining Flutter's capabilities.

Flutter transforma el proceso de desarrollo de aplicaciones. Cree, pruebe e implemente hermosas aplicaciones móviles, web, de escritorio e integradas desde una única base de código.

# Flutter



Es un marco de desarrollo multiplataforma y de código abierto

Establecido para desarrollar aplicaciones nativas (cross platform) en iOS y Android.

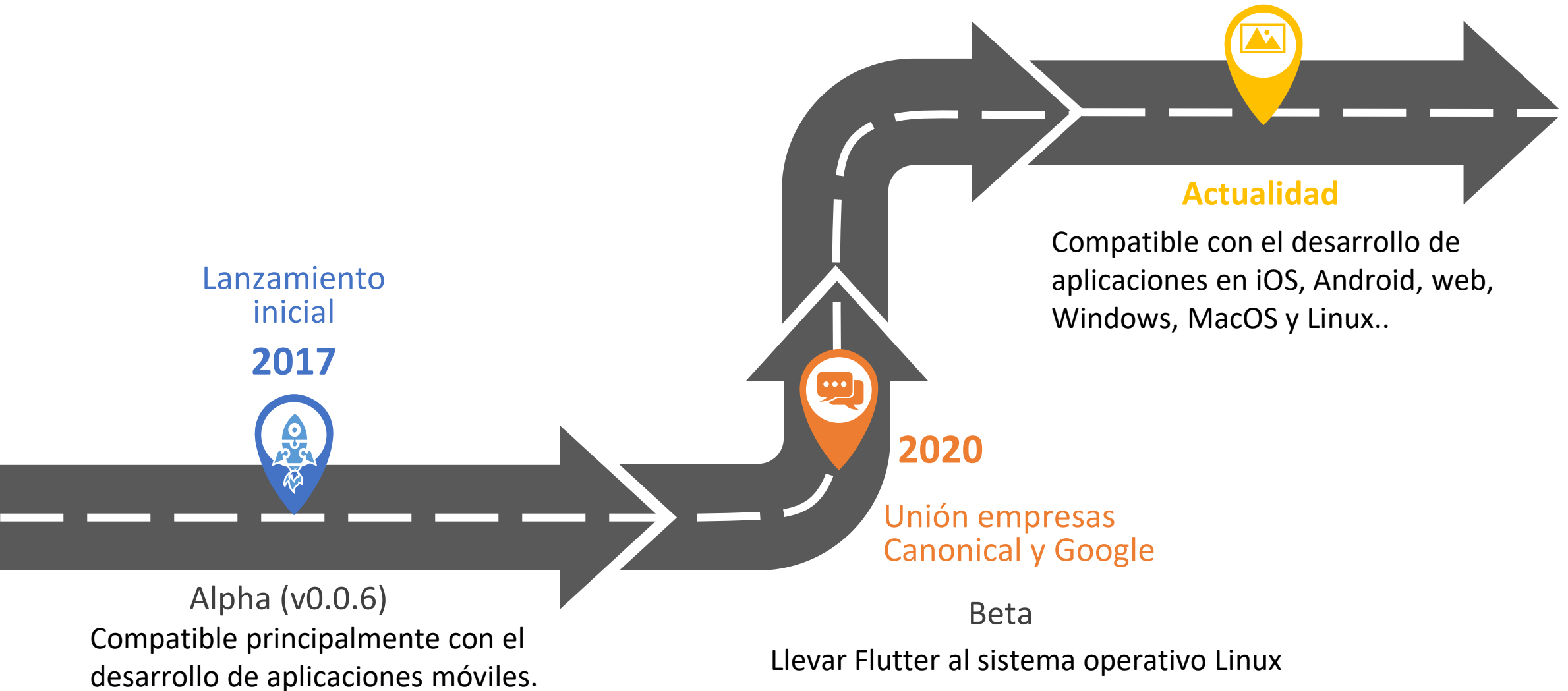
Su principal objetivo es el desarrollo de aplicaciones móviles multiplataforma compiladas de forma nativa a partir de una única base de código.



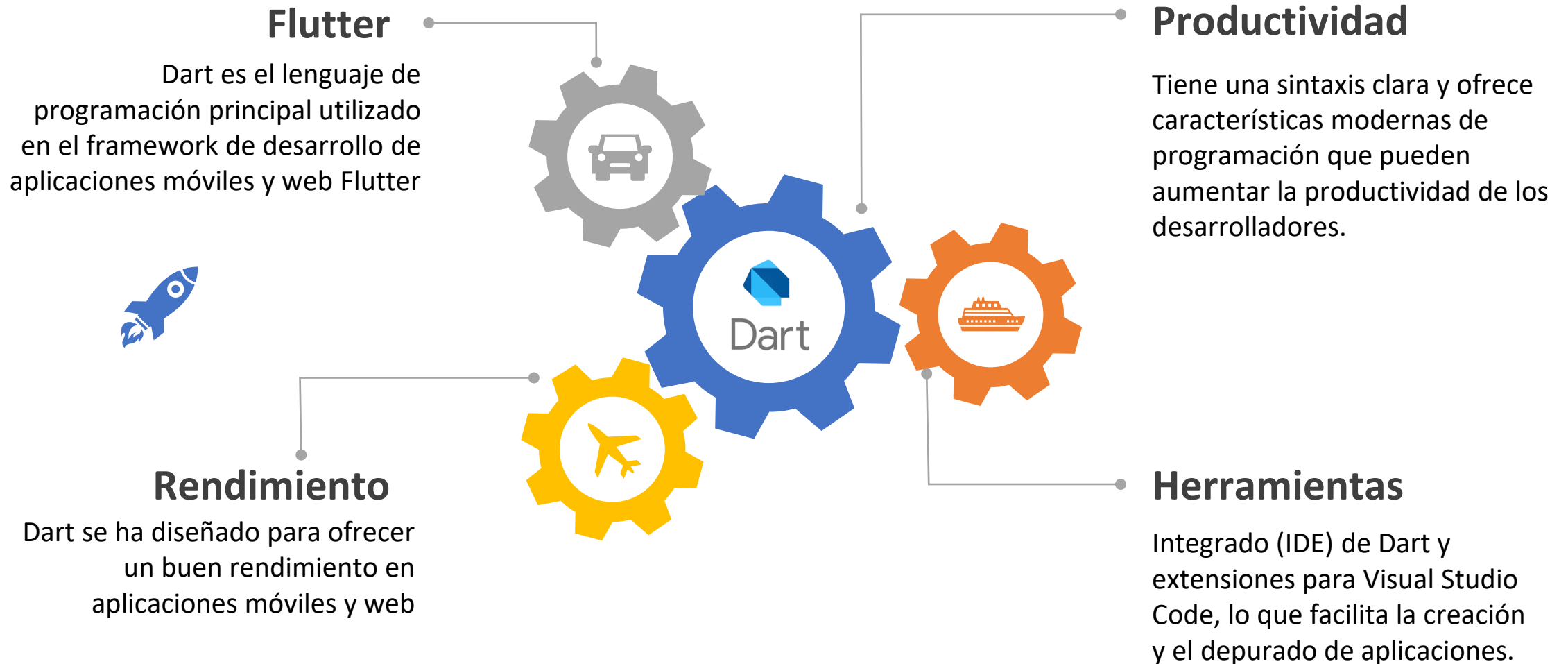
➤ Creado por Google.

➤ Utiliza Dart, un lenguaje de programación optimizado para aplicaciones rápidas en cualquier plataforma, originado por Google y que estaría orientado a aplicaciones móviles y web.

# Historia



# ¿Por qué el lenguaje DART?



# ¿Por qué el lenguaje DART?



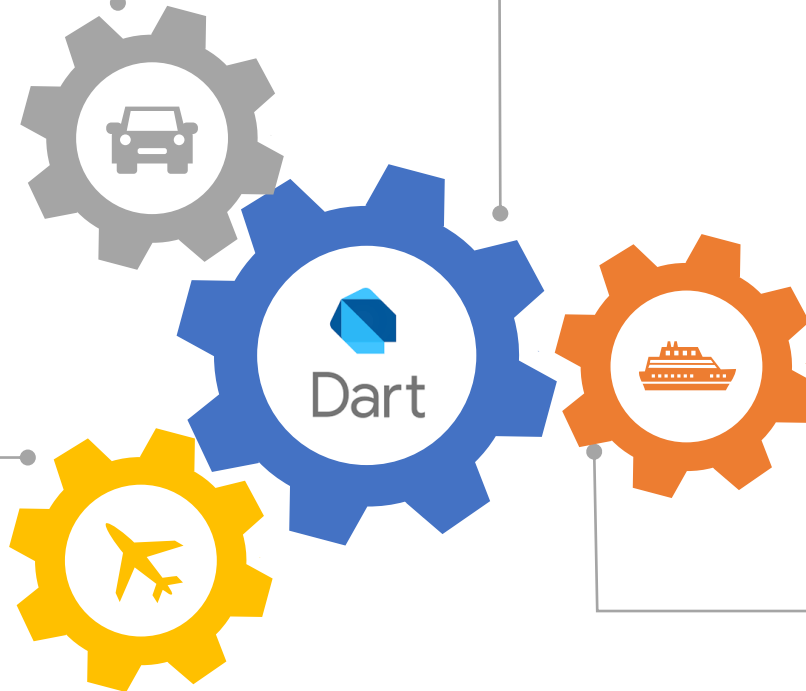
## Comunidad en crecimiento

La comunidad de Dart y Flutter ha estado creciendo y madurando rápidamente, lo que significa que hay recursos, bibliotecas y componentes disponibles para ayudar en el desarrollo de aplicaciones.



## Soporte multiplataforma

Dart y Flutter son ideales para el desarrollo de aplicaciones multiplataforma



## Integración con otros servicios de Google

Si estás trabajando en un proyecto que se integra con otros servicios de Google, como Firebase.

## Empresas que respaldan Dart

El respaldo de grandes empresas como Google.



# Fundamentos DART

A screenshot of the DartPad web application interface. The browser address bar shows 'https://dartpad.dev'. The DartPad logo is on the left, with 'New' and 'Samples' buttons on the right. The code editor contains a Dart program that prints 'hello' followed by numbers 1 through 10. A 'Run' button is visible. The output panel on the right shows the result of the program execution.

```
1 void main() {  
2   for (int i = 0; i < 10; i++) {  
3     print('hello ${i + 1}');  
4   }  
5 }  
6
```

hello 1  
hello 2  
hello 3  
hello 4  
hello 5  
hello 6  
hello 7  
hello 8  
hello 9  
hello 10

# ¿Por qué Flutter?



## Desarrollo multiplataforma

Flutter permite crear aplicaciones para múltiples plataformas a partir de una sola base de código, incluyendo iOS, Android, web y próximamente también aplicaciones de escritorio (Windows, macOS y Linux). Esto ahorra tiempo y esfuerzo en el desarrollo y mantenimiento de aplicaciones para diferentes sistemas operativos.

## Widgets personalizables

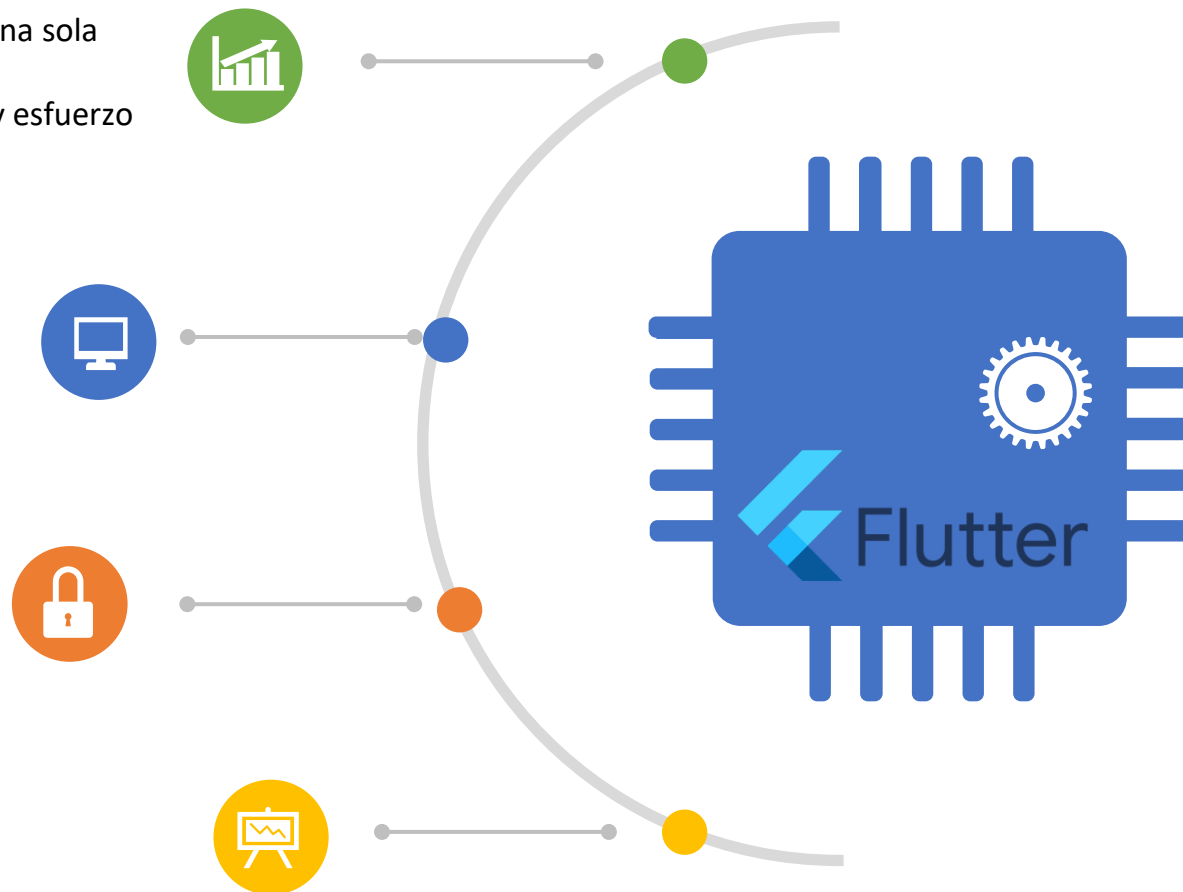
Flutter utiliza un sistema de widgets personalizables y altamente flexibles que permiten diseñar interfaces de usuario atractivas y adaptables. Puedes crear widgets personalizados o personalizar los widgets incorporados para cumplir con tus necesidades de diseño.

## Rendimiento de alta calidad

Flutter utiliza el motor gráfico Skia, Flutter utiliza la compilación en código nativo (AOT) para generar código altamente optimizado, lo que mejora la velocidad de ejecución de la aplicación.

## Recarga en caliente (Hot Reload)

Esta característica permite a los desarrolladores ver los cambios en tiempo real mientras desarrollan la aplicación.



# ¿Por qué Flutter?



## Compatibilidad con material design y Cupertino

Flutter ofrece una amplia gama de widgets que siguen las pautas de diseño de Material Design de Google para Android y el estilo de diseño de Cupertino para iOS.

## Comunidad activa y crecimiento

Flutter cuenta con una comunidad activa de desarrolladores

## Soporte de Google

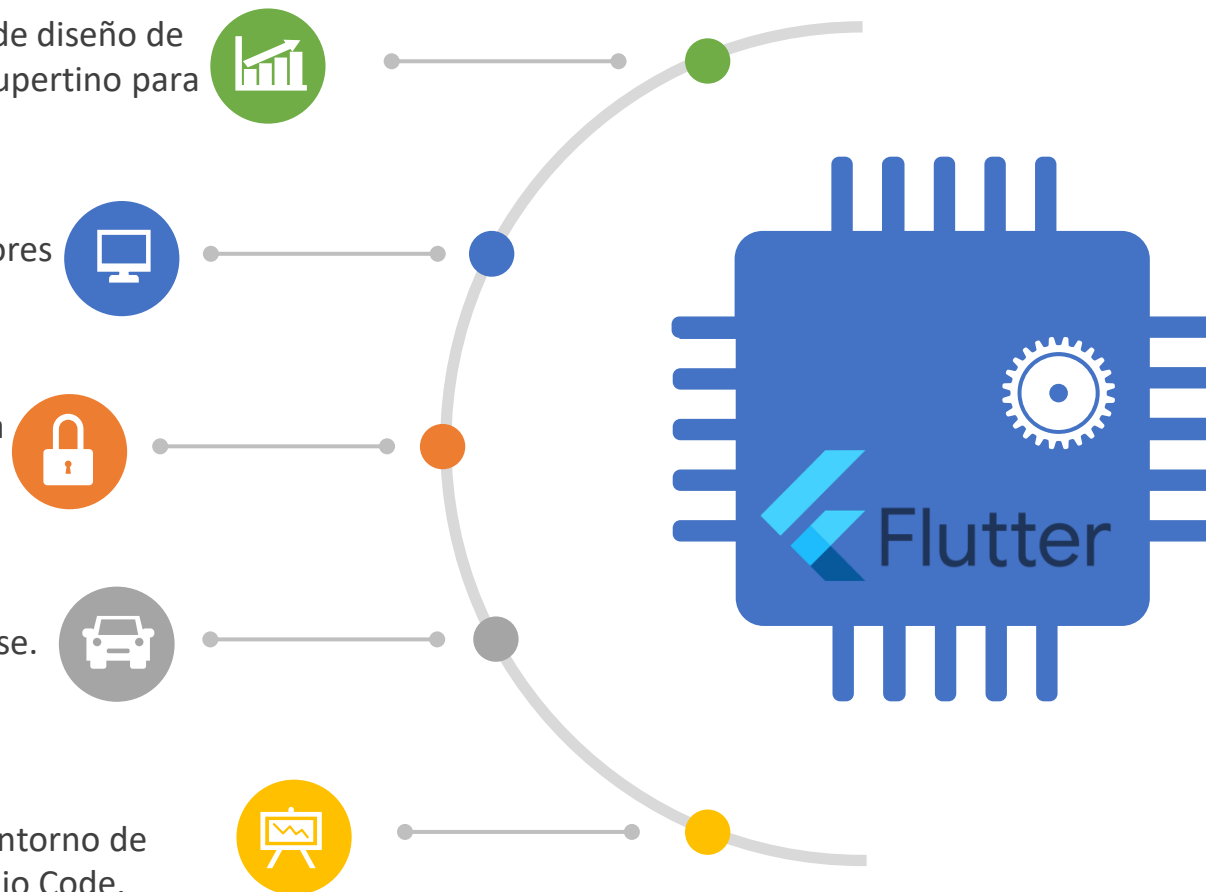
Flutter es respaldado por Google, lo que proporciona confianza en su continuidad

## Integración con Firebase

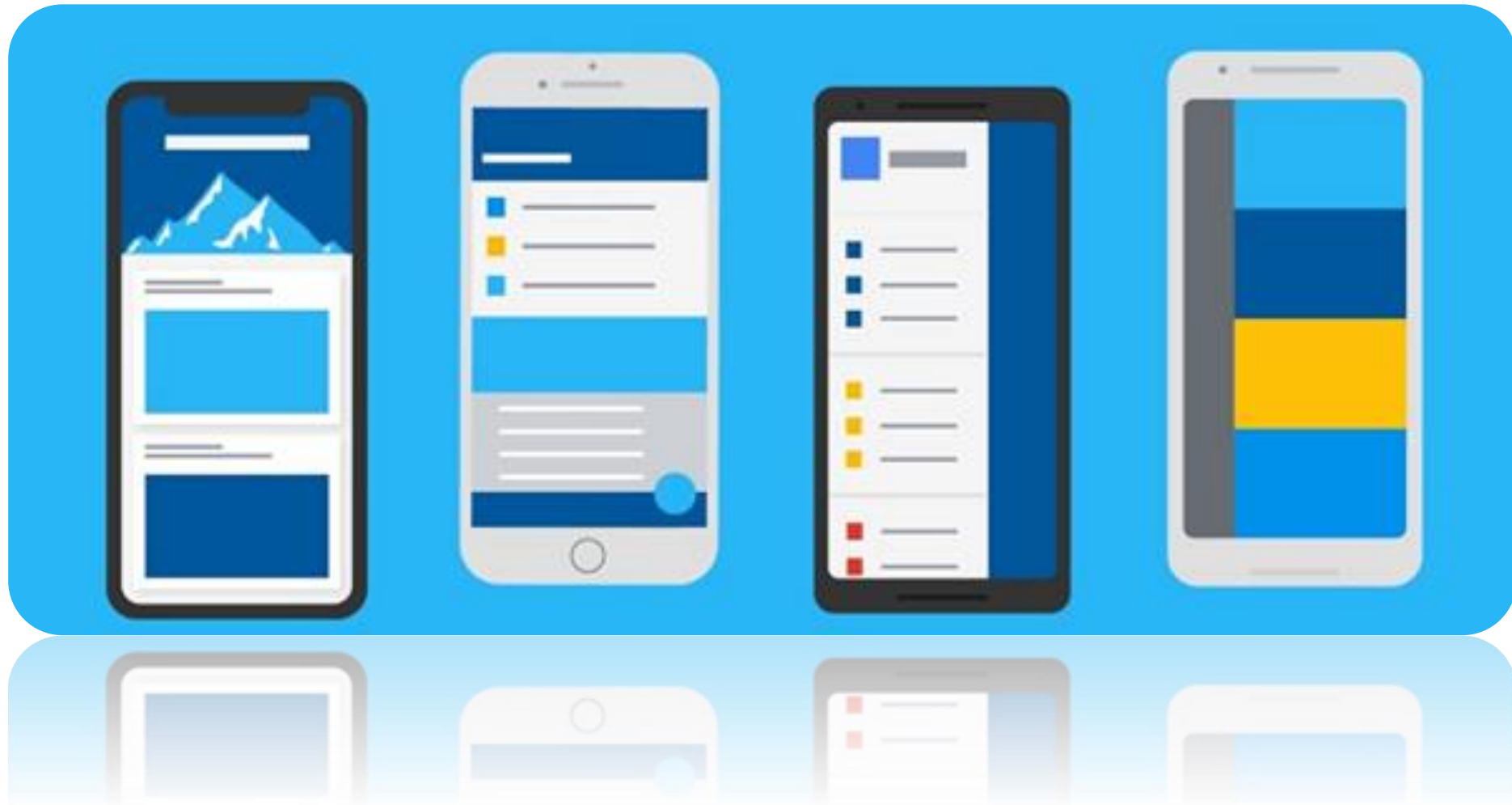
Flutter se integra de manera natural con Firebase.

## Herramientas de desarrollo

Flutter cuenta con herramientas de desarrollo sólidas, como el entorno de desarrollo integrado (IDE) de Dart y extensiones para Visual Studio Code.



# Widgets



# Widgets



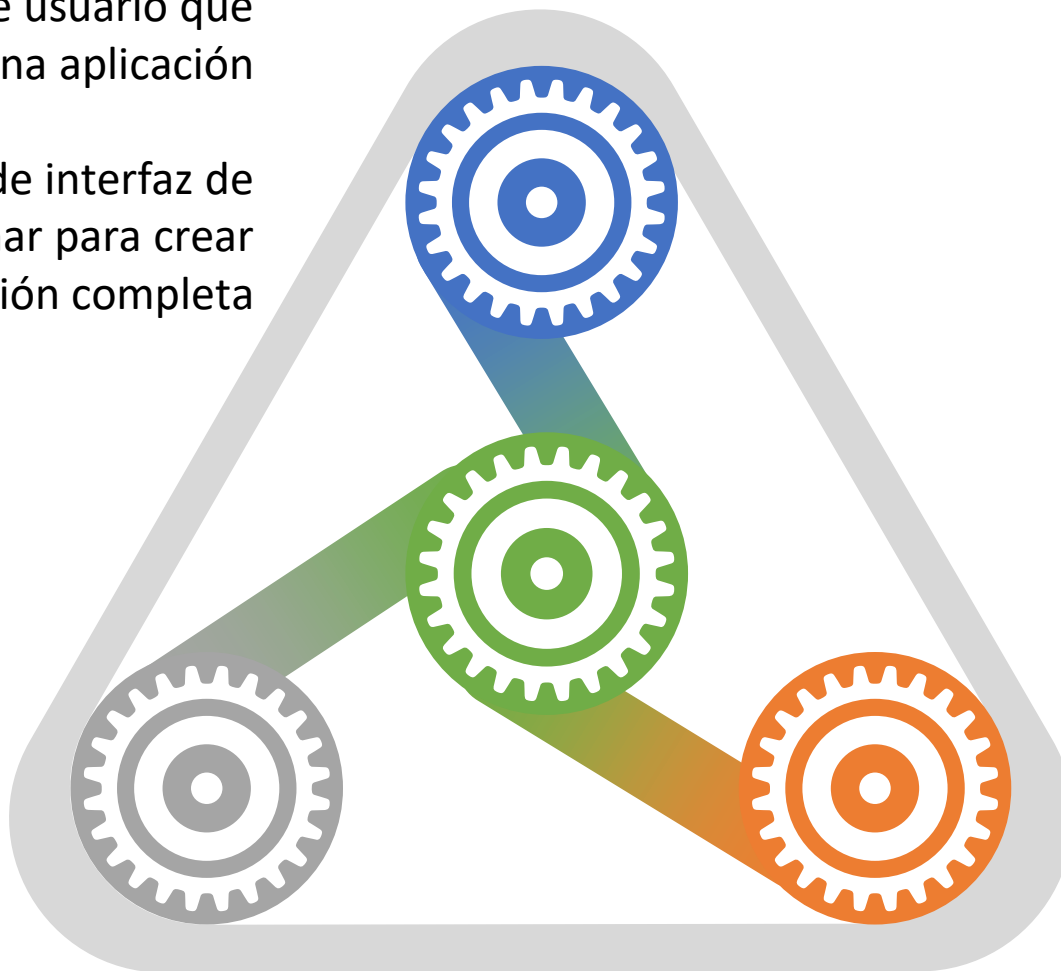
✓ Elementos de la interfaz de usuario que componen una aplicación

✓ Pequeños fragmentos de interfaz de usuario que puede combinar para crear una aplicación completa

## Tipos

✓ Widgets StatelessWidget

✓ Widgets StatefulWidget



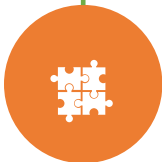
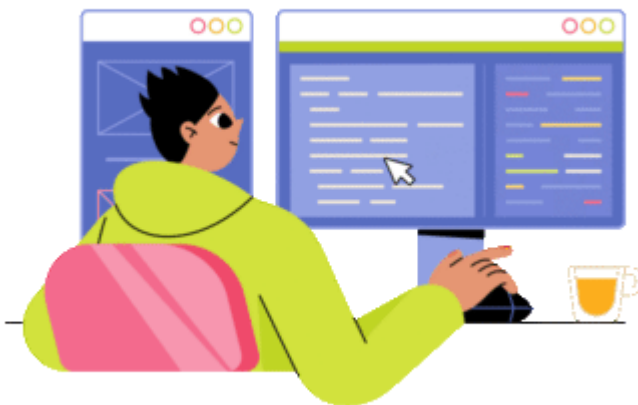
**Todo es un Widget en Flutter**

Los widgets en Flutter son más que solo elementos visuales; también representan la **estructura** y la **lógica** de la interfaz de usuario.

# Widgets

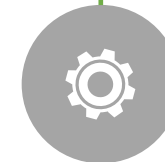


Los widgets de *Flutter* son los bloques de construcción fundamentales para crear interfaces de usuario interactivas y visualmente atractivas en las aplicaciones.



## Widgets StatelessWidget (Inmutables)

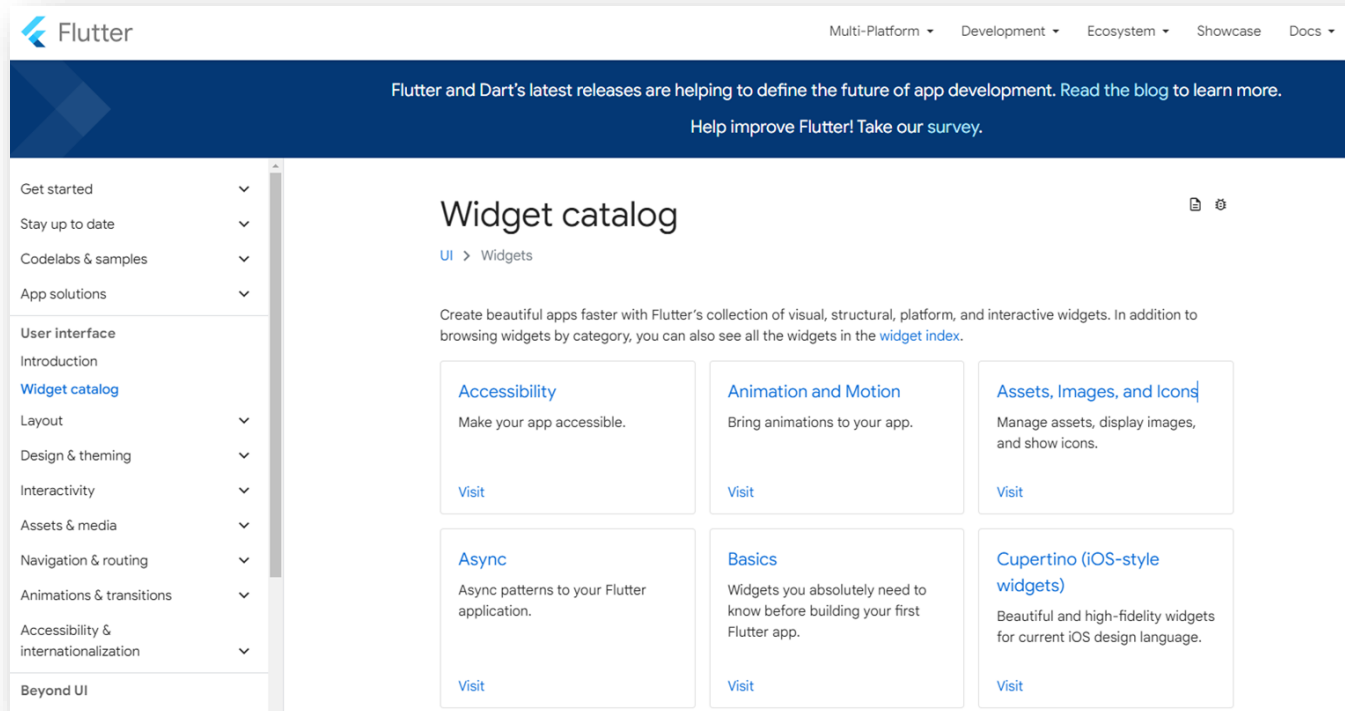
- ✓ No pueden cambiar su estado después de ser creados.
- ✓ Son utilizados principalmente para representar elementos estáticos de la interfaz de usuario.  
(Etiquetas de texto, imágenes o botones que no cambian en respuesta a la interacción del usuario).



## Widgets StatefulWidget (Cambio)

- ✓ Estos son widgets que pueden cambiar su estado a lo largo del tiempo en respuesta a eventos o interacciones del usuario.
- ✓ Son utilizados para representar elementos de la interfaz que pueden cambiar dinámicamente.  
(Listas desplegables, casillas de verificación, formularios, entre otros)

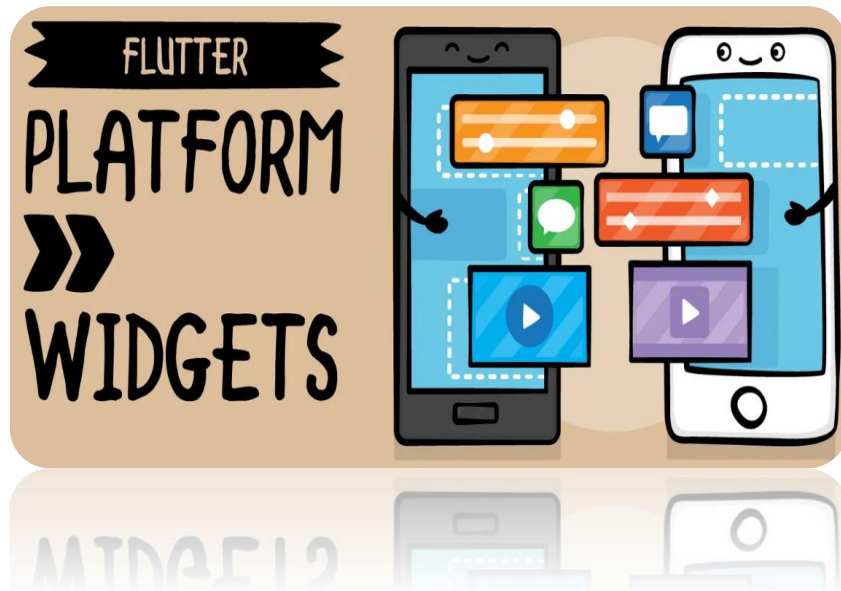
# Catálogo de widgets



Flutter viene con un amplio catálogo de *widgets* desde el momento en que lo descargas. El catálogo tiene 14 categorías, que incluyen estilos, Cupertino (*widgets* estilo iOS) y Material Components (*widgets* que siguen las directrices de Material Design de Google).

<https://docs.flutter.dev/ui/widgets>

# Material Design y Cupertino



- Compatibilidad de Flutter con ***material design*** y ***Cupertino***
- Material Widgets y Cupertino Widgets son dos sistemas de diseño diferentes proporcionados por Flutter para crear interfaces de usuario que se adhieren a los principios de diseño de Android (Material Design) e iOS (Cupertino).
- Flutter ofrece una amplia gama de widgets que siguen las pautas de diseño de Material Design de Google para Android y el estilo de diseño de Cupertino para iOS.



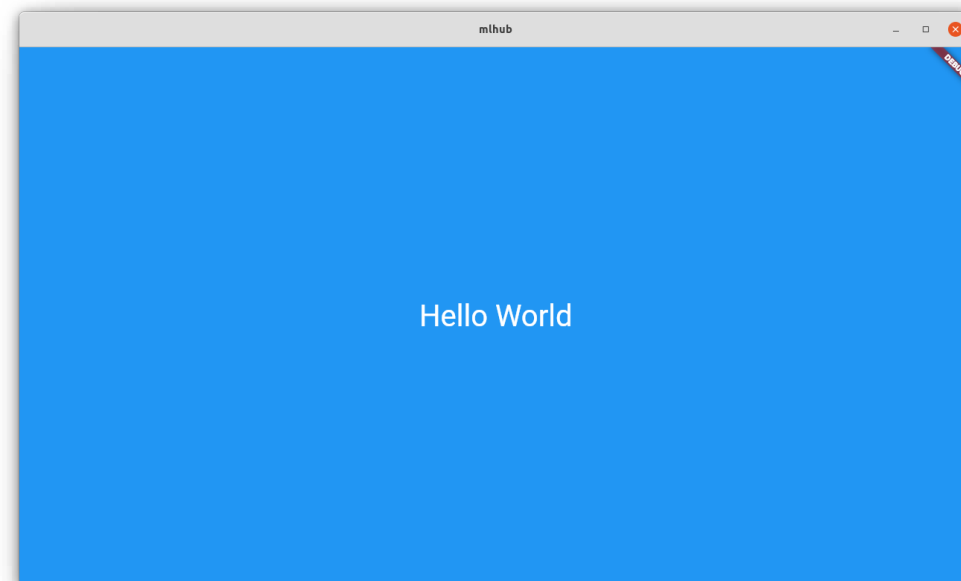
```
@override
Widget build(BuildContext context) {
  return const MaterialApp(
    title: "Ejemplo",
    debugShowCheckedModeBanner: false,
    home: Inicio()
  ); // MaterialApp
```

```
)? \\ wafeljsjvbb
HOME: TUTCTO()
```

# Clase MaterialApp



- La opción de MaterialApp en Flutter se entiende como **una de las clases predefinidas del sistema**.
- Funciona como uno de los componentes centrales o principales del SDK de Flutter.
- El widget MaterialApp proporciona un contenedor para otros ***widgets*** y ***temas*** necesarios para crear una aplicación basada en Material design.
- El Widget MaterialApp ***es el punto de partida de una aplicación***, le dice a Flutter que se usarán componentes de Material Design y se seguirá el diseño de Material design en la aplicación.



<https://api.flutter.dev/flutter/material/MaterialApp-class.html>

# Material Widgets



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

<https://api.flutter.dev/flutter/material/MaterialApp-class.html>

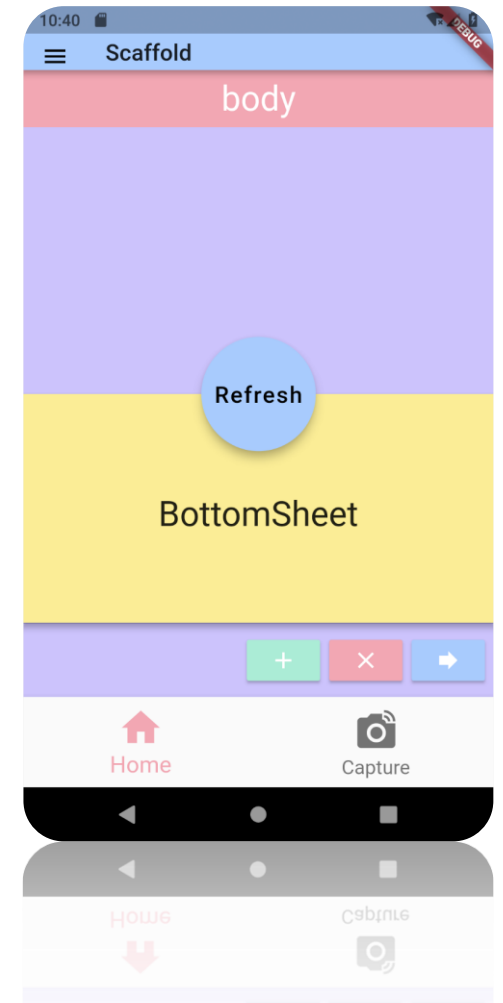
<https://docs.flutter.dev/ui/widgets/material>

# Scaffold



- Es un widget hijo del MaterialApp (que es donde comienza todo) y nos permite emplear elementos (widgets) del Material Design
- Es el widget que implementa la estructura de diseño visual básica de Material Design.
- Scaffold es un widget que proporciona una estructura básica de Material Design para implementar:
  - ✓ Drawers
  - ✓ Snack bars
  - ✓ Bottom sheets
  - ✓ Floating action buttons y más.

<https://api.flutter.dev/flutter/material/Scaffold-class.html>



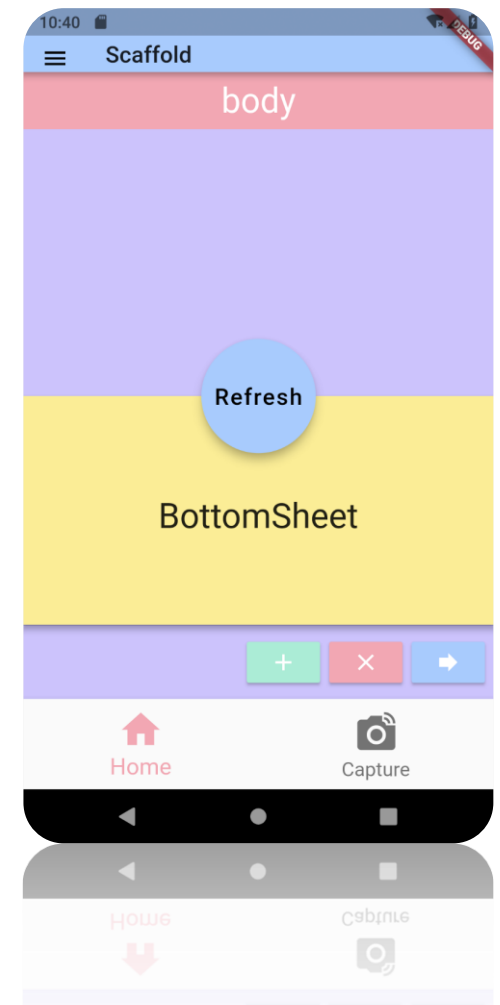
# Scaffold



- Es como un "andamio" que mantiene juntos los componentes visuales estándares de una app
- Scaffold en Flutter se entiende como una clase que ofrece API que permiten mostrar ***bottom sheets*** y ***drawers***.
- No es estrictamente necesario utilizar Scaffold en todas las pantallas.

Sin embargo, si se siguen las directrices de Material Design, el Scaffold es extremadamente útil por la funcionalidad y estructura que ofrece de forma predeterminada.

<https://api.flutter.dev/flutter/material/Scaffold-class.html>

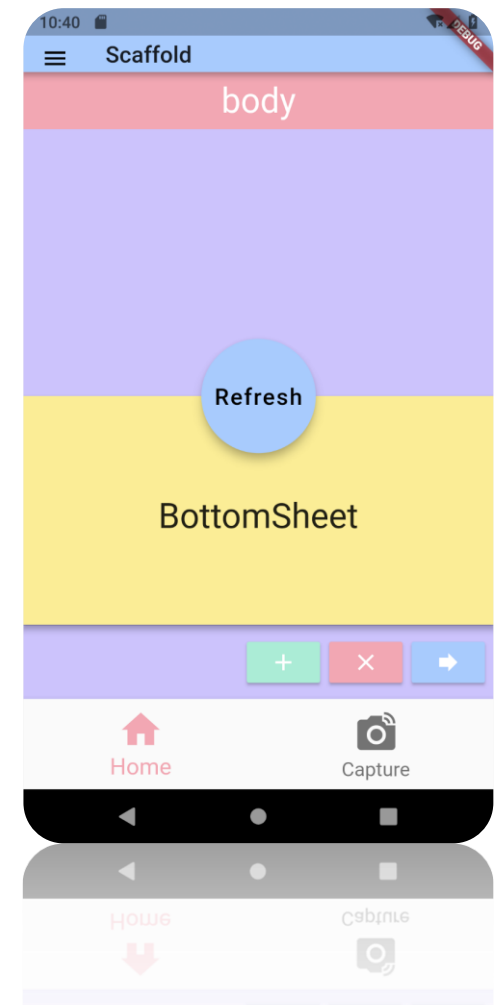


# Scaffold



- El widget Scaffold es una herramienta esencial para desarrollar bajo el framework Flutter.
- Proporciona una estructura de página predeterminada que puede ser enormemente personalizada para satisfacer las necesidades de cualquier aplicación móvil.
- Con el Scaffold, Flutter hace que el desarrollo de UI sea más intuitivo y menos propenso a errores, manteniendo la coherencia del diseño en toda la aplicación.

<https://api.flutter.dev/flutter/material/Scaffold-class.html>



# Mi primera App

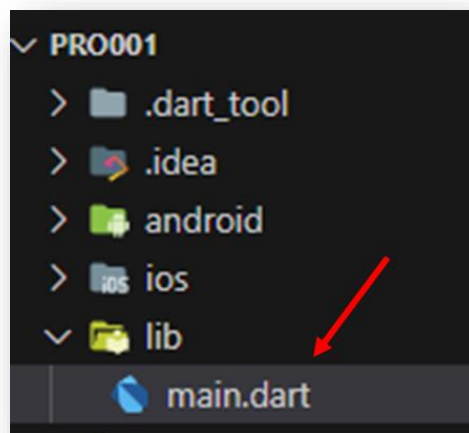


*Abrir command palette:*



1. Buscar **Flutter**
2. Seleccionar **New Project**
3. Seleccionar **Application**
4. Crear la carpeta del proyecto y seleccionarla:
5. Aceptar que se confía en el autor (Evidentemente pues usted es el autor)

*Estructura del proyecto:*



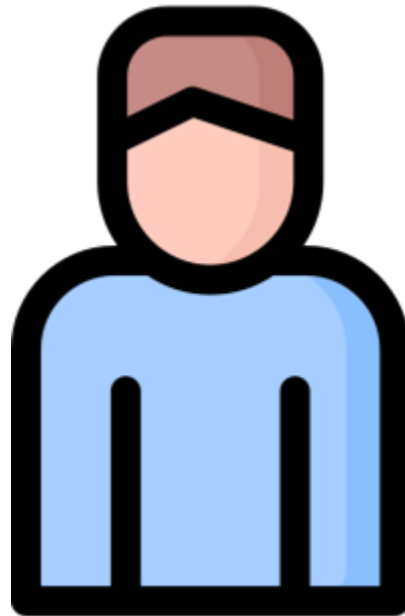


Cuando voy a construir una casa primero requiero el terreno:

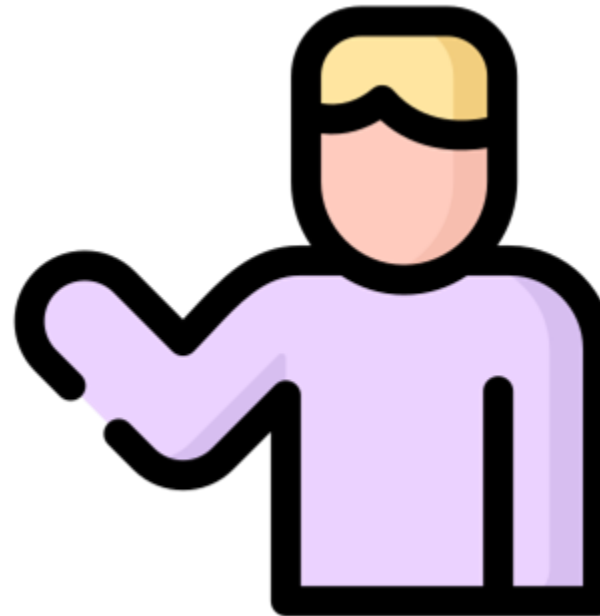
StatefulWidget - StatelessWidget



Amigo, he comprado  
un terreno.



¡Felicidades!  
¿De cuántos metros  
cuadrados?



```
class Principal extends StatefulWidget {  
  const Principal({super.key});  
  
  @override  
  State<Principal> createState() => _PrincipalState();  
}  
  
class _PrincipalState extends State<Principal> {  
  @override  
  Widget build(BuildContext context) {  
    return MaterialApp( );  
  }  
}
```

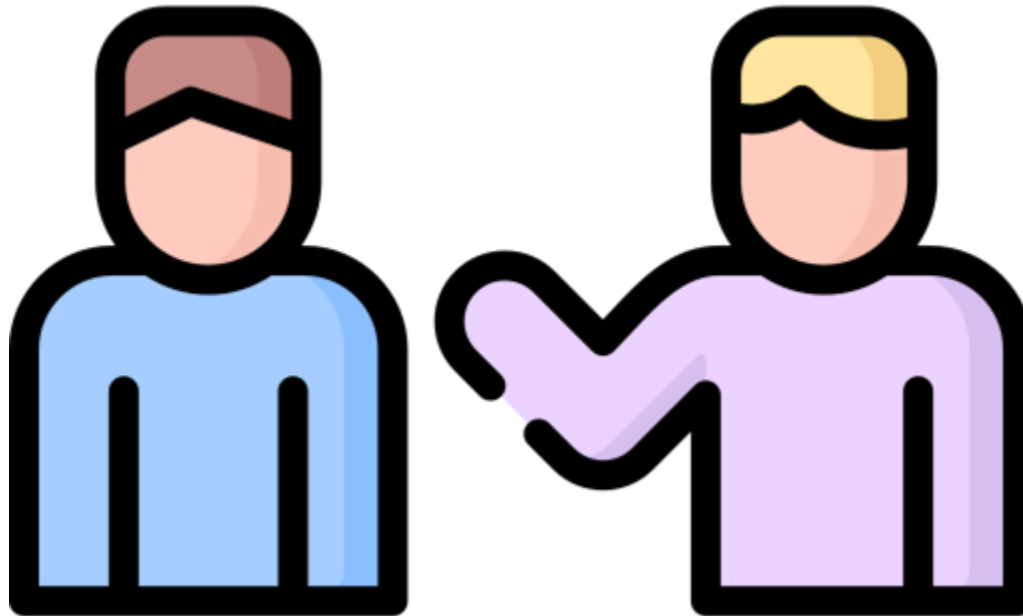
Terreno donde voy a construir

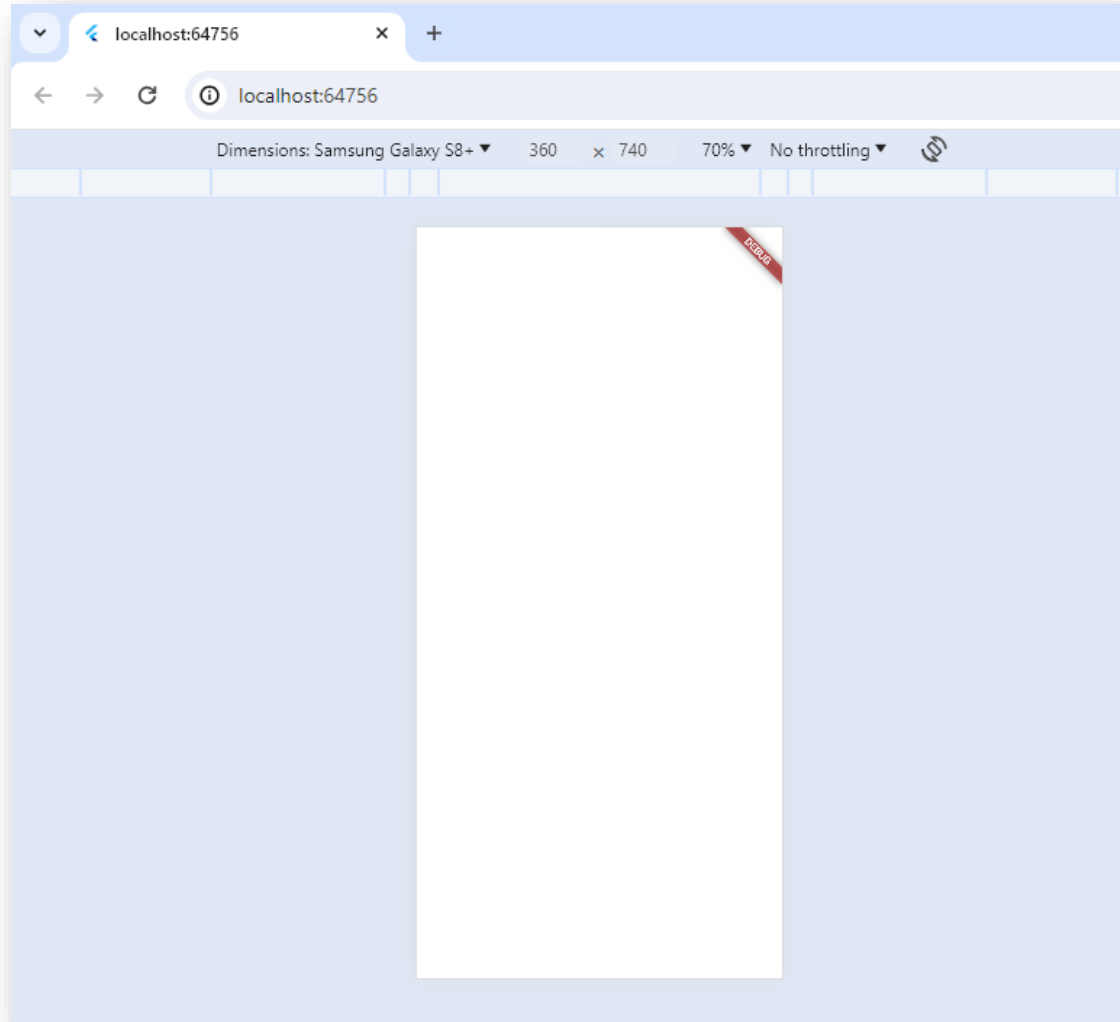
Las dimensiones del terreno

- *MaterialApp muestra unas dimensiones similares a una pantalla*
- *Me permite organizar el direccionamiento (El ancho y alto)*

¡Aún nada!

¿Qué has construido en  
el terreno?





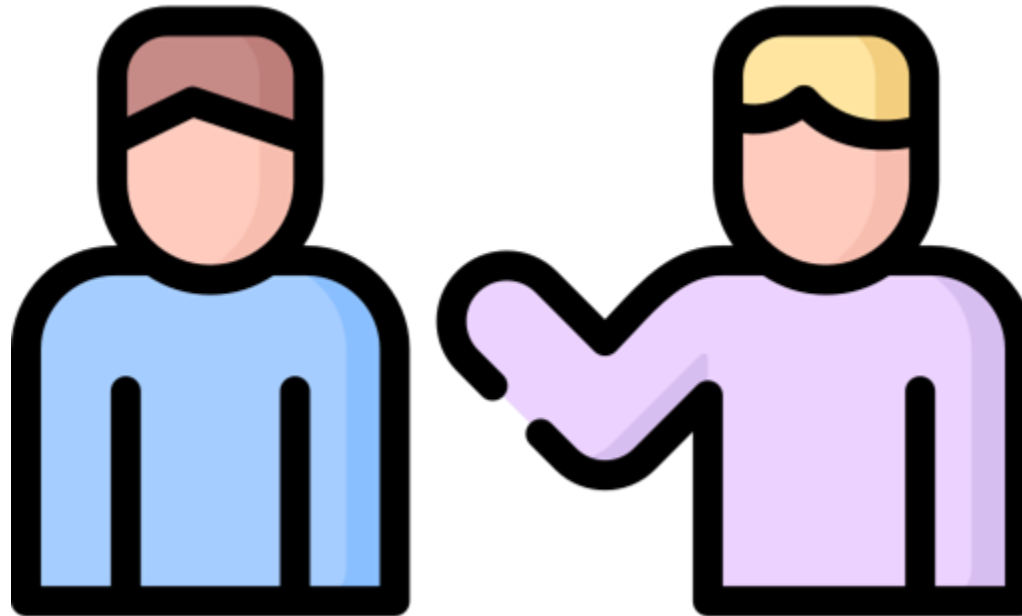
Al ejecutar el programa no tendremos nada aún.

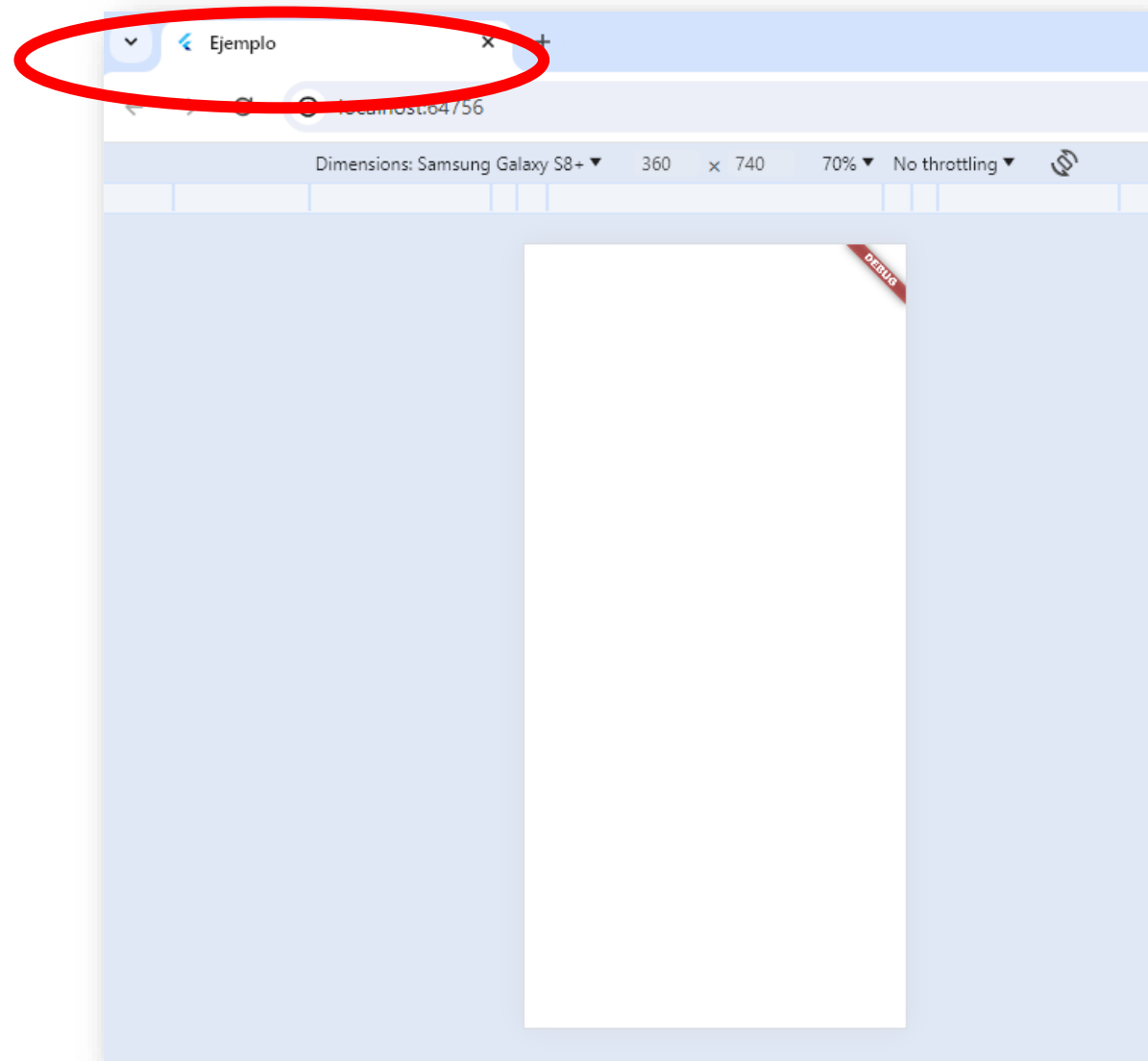
Ahora miremos algunas características:

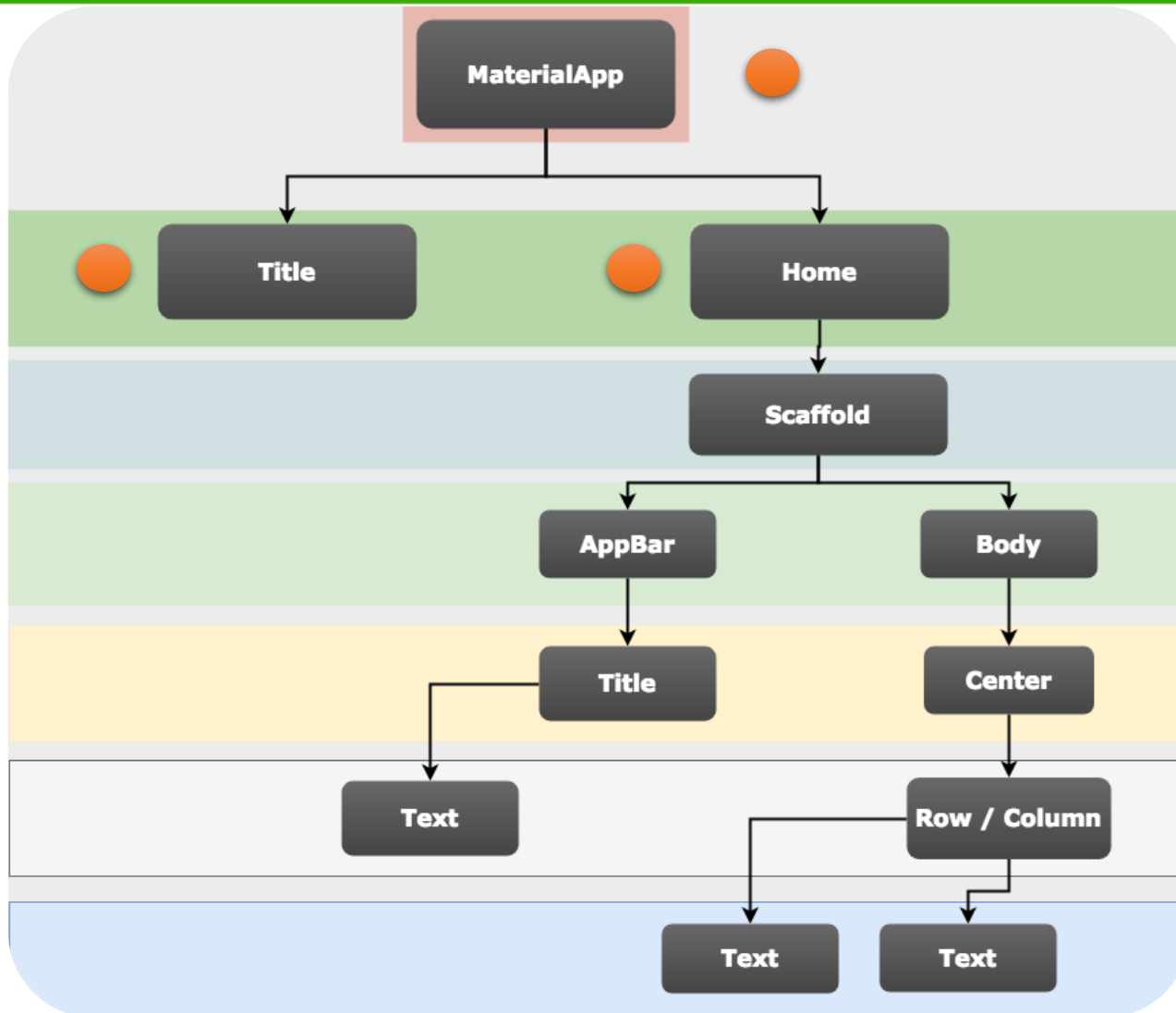
```
main.dart x
lib > main.dart > ...
6
7 class Principal extends StatefulWidget {
8   const Principal({super.key});
9
10  @override
11  State<Principal> createState() => _PrincipalState();
12 }
13
14 class _PrincipalState extends State<Principal> {
15   @override
16   Widget build(BuildContext context) {
17     return MaterialApp(
18       title: "Ejemplo",
19     );
20   }
21 }
```

¡Aún nada!  
Únicamente le coloqué  
un **letrero**.

¿Qué has construido en  
el terreno?







Una app en flutter crece de manera jerárquica.



```
main.dart •
lib > main.dart > main

7  class Principal extends StatefulWidget {
8      const Principal({super.key});
9
10     @override
11     State<Principal> createState() => _PrincipalState();
12 }
13
14 class _PrincipalState extends State<Principal> {
15     @override
16     Widget build(BuildContext context) {
17         return MaterialApp(
18             title: "Ejemplo",
19             home: Text("SENA: Servicio Nacional de Aprendizaje")
20         ) // MaterialApp
21     ;
22 }
23 }
```

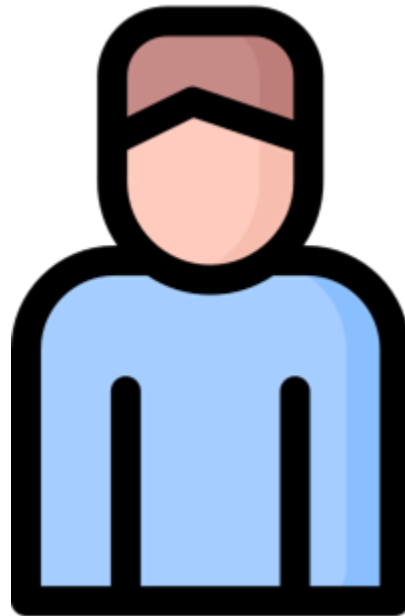
Terreno donde  
voy a construir

Las  
dimensiones  
del terreno

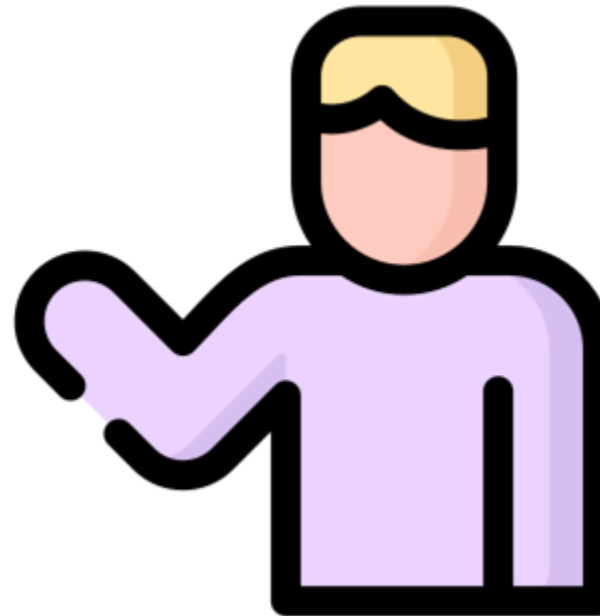
Letrero

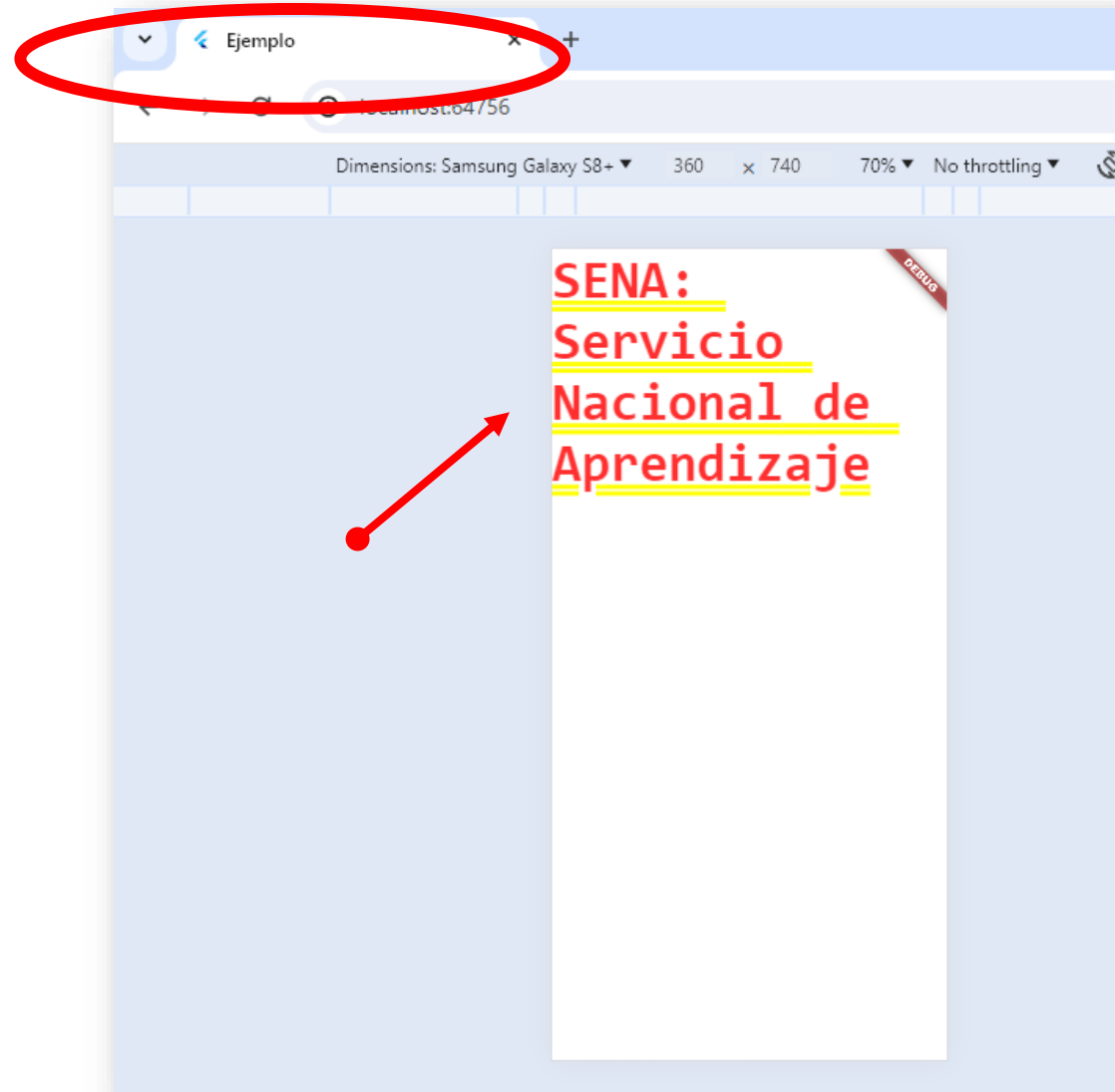
Cercar el  
terreno

¡Aún nada! Pero he  
cercado el terreno.  
Para que sepan que es mi  
terreno



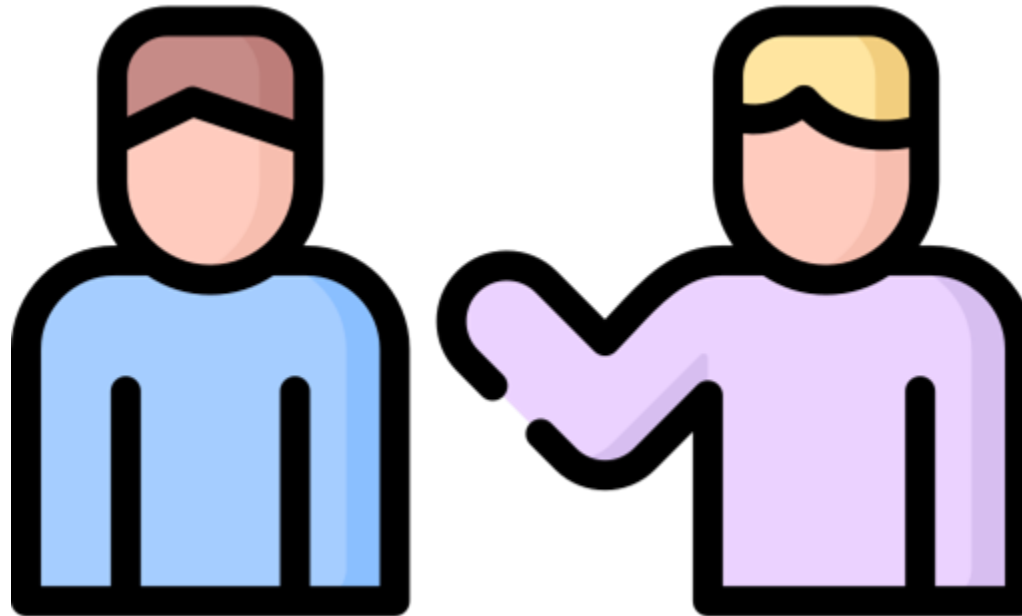
¿Qué has construido en  
el terreno?



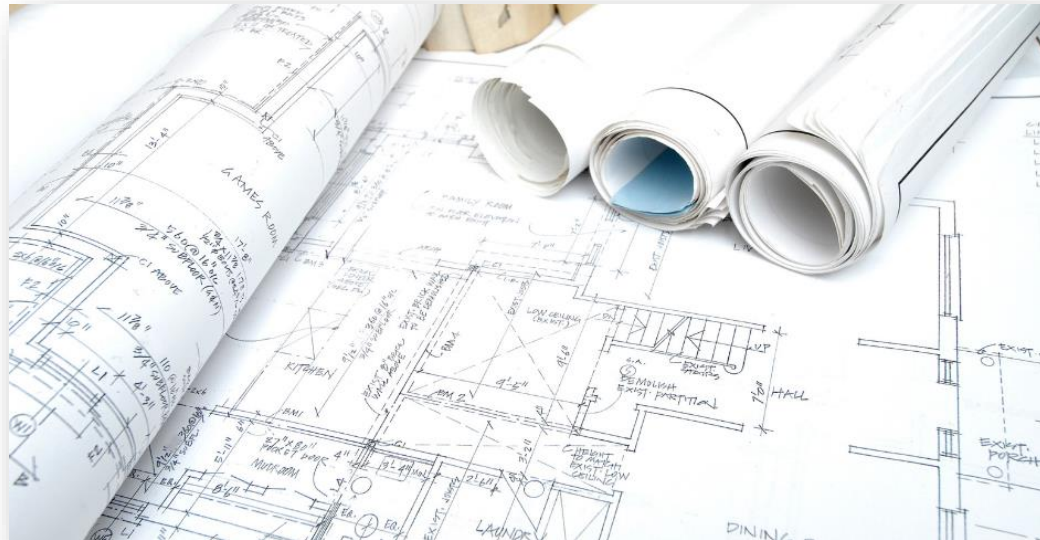


¡Excelente idea!

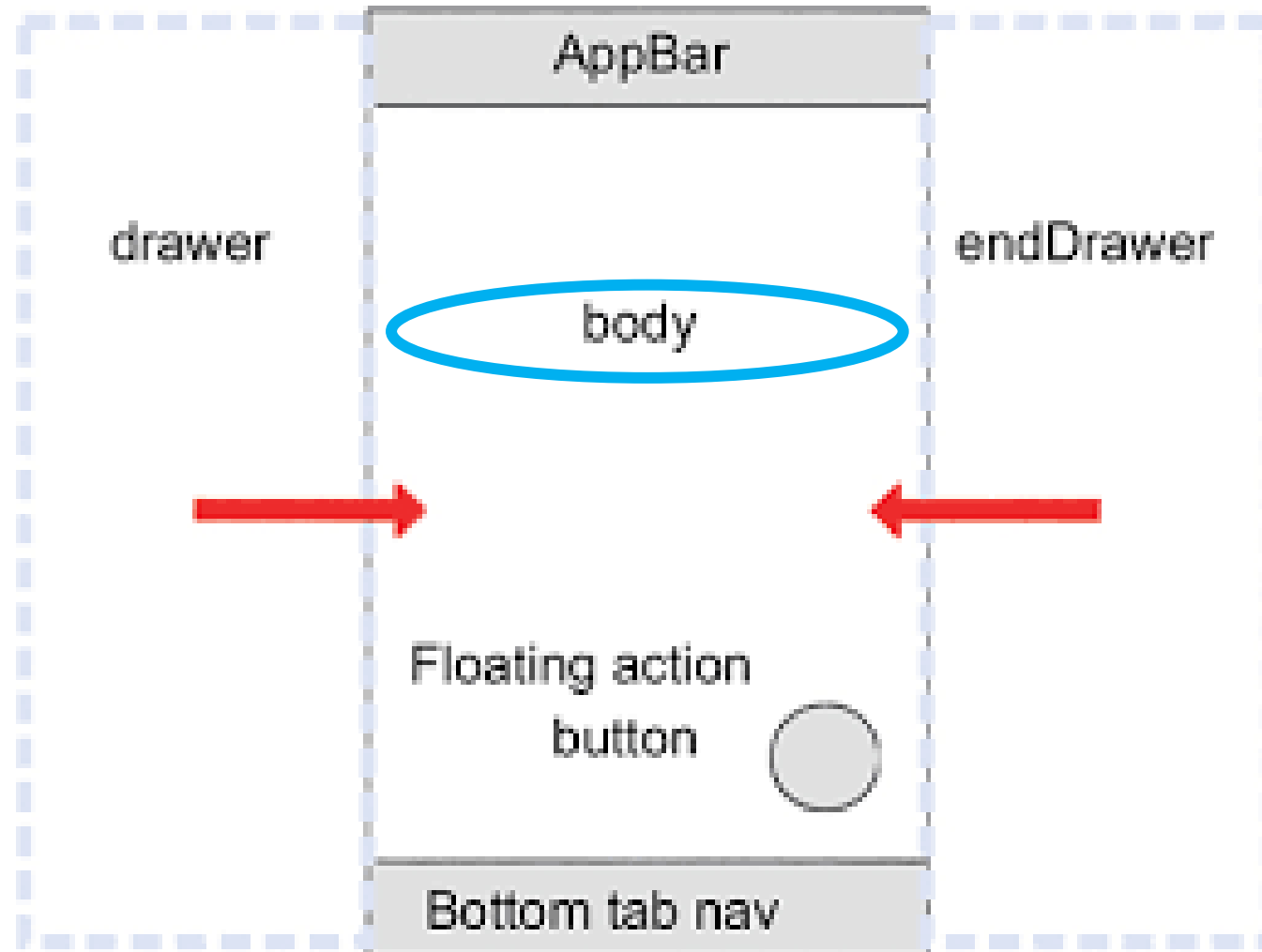
Te recomiendo contactar a un arquitecto experto para que diseñe una estructura de construcción.



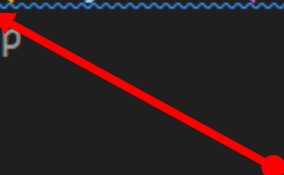
- Dentro del material app utilizado un arquitecto para que me entregue los planos y ese plano se llama ***scaffold***
- En el home irían los planos de la construcción

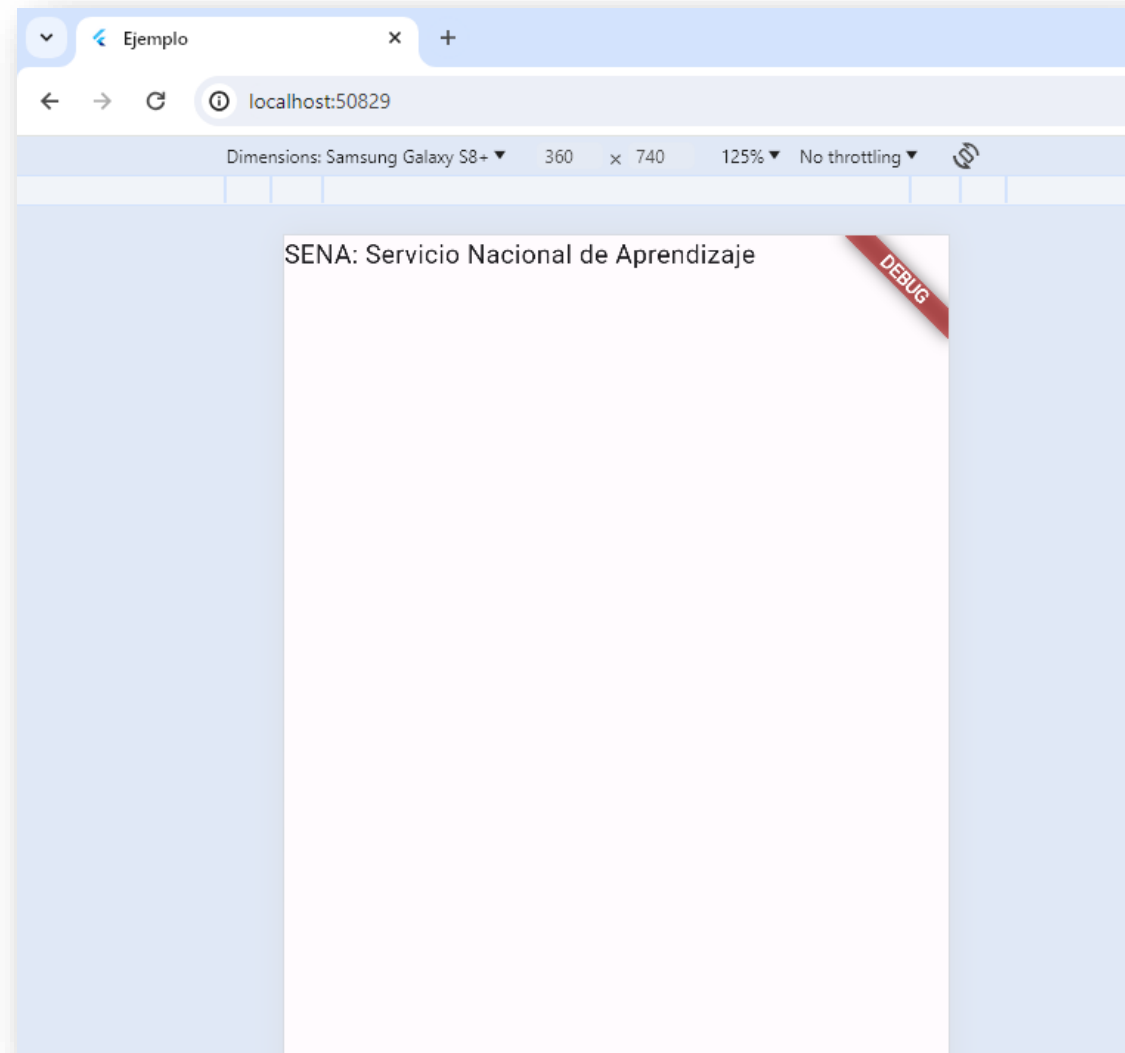


# Scaffold



```
main.dart x
lib > main.dart > main
6
7 class Principal extends StatefulWidget {
8   const Principal({super.key});
9
10  @override
11  State<Principal> createState() => _PrincipalState();
12 }
13
14 class _PrincipalState extends State<Principal> {
15   @override
16   Widget build(BuildContext context) {
17     return MaterialApp(
18       title: "Ejemplo",
19       home: Scaffold(body: Text("SENA: Servicio Nacional de Aprendizaje")),
20     ); // MaterialApp
21   }
22 }
23
```

A red arrow points from the bottom right towards the Scaffold widget in the code, specifically highlighting the Text widget inside its body.







# GRACIAS

Línea de atención al ciudadano: 01 8000 910270  
Línea de atención al empresario: 01 8000 910682



[www.sena.edu.co](http://www.sena.edu.co)