



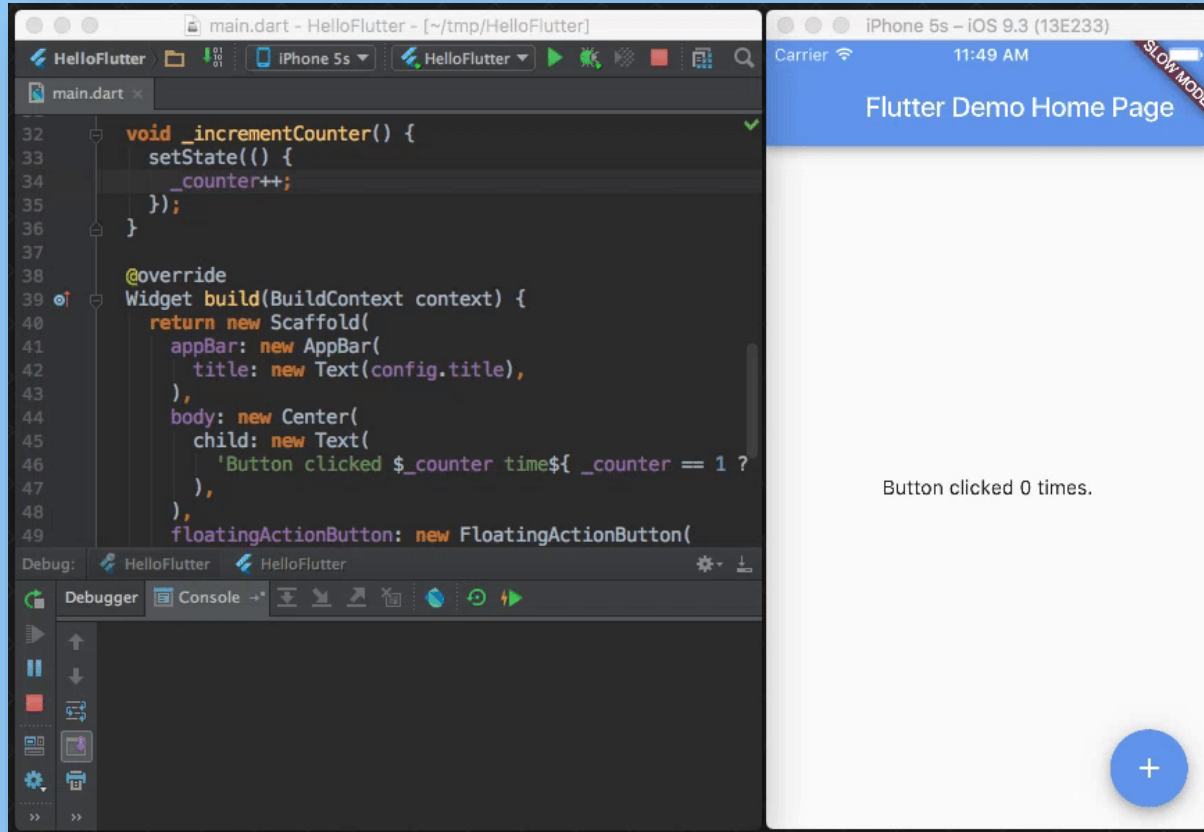
Introducción a Flutter



¿Qué es Flutter?

- Flutter es un framework de desarrollo de aplicaciones multiplataforma desarrollado por Google
- Utiliza el lenguaje de programación Dart
- Permite crear interfaces de usuario de manera rápida y sencilla

¿Por qué utilizar Flutter?

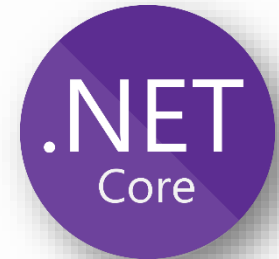


- Permite desarrollar una aplicación para ios, Android, web y próximamente escritorio con una sola base de código
- Desarrollo de interfaces bonitas y rápidas
- Cuenta con una gran comunidad desarrollando librerías

¿Qué es un framework?



- Un Framework o marco de trabajo es un conjunto de librerías, prácticas y conceptos que nos ayudan a desarrollar un software en específico.
- Nos ayudan a:
 - Evitar repetir código
 - Desarrollar más rápido



Tipos de aplicaciones

Nativas

Aplicaciones desarrolladas para un sistema operativo en específico, funcionan en un solo sistema operativo

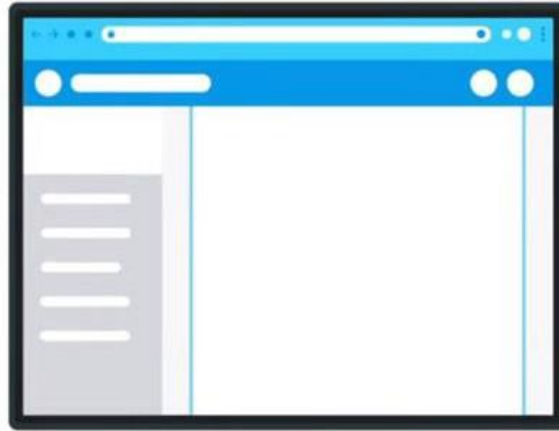
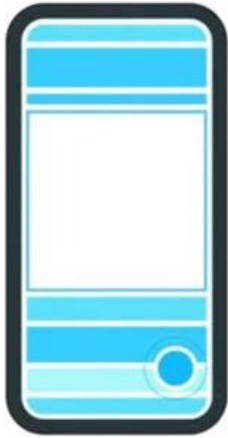
Multiplataforma

Aplicaciones que permiten ser desarrolladas usando una sola base de código con rendimiento muy cercano al nativo

Híbridas

Aplicaciones desarrolladas utilizando tecnologías web, permiten desplegarse a diferentes plataformas

¿Qué tipo de aplicaciones genera Flutter?



Alternativas a Flutter

Nativas



Multiplataforma



Híbridas



Barra de navegación

Lista de Tweets

Barra de navegación
inferior



Historias

Texto

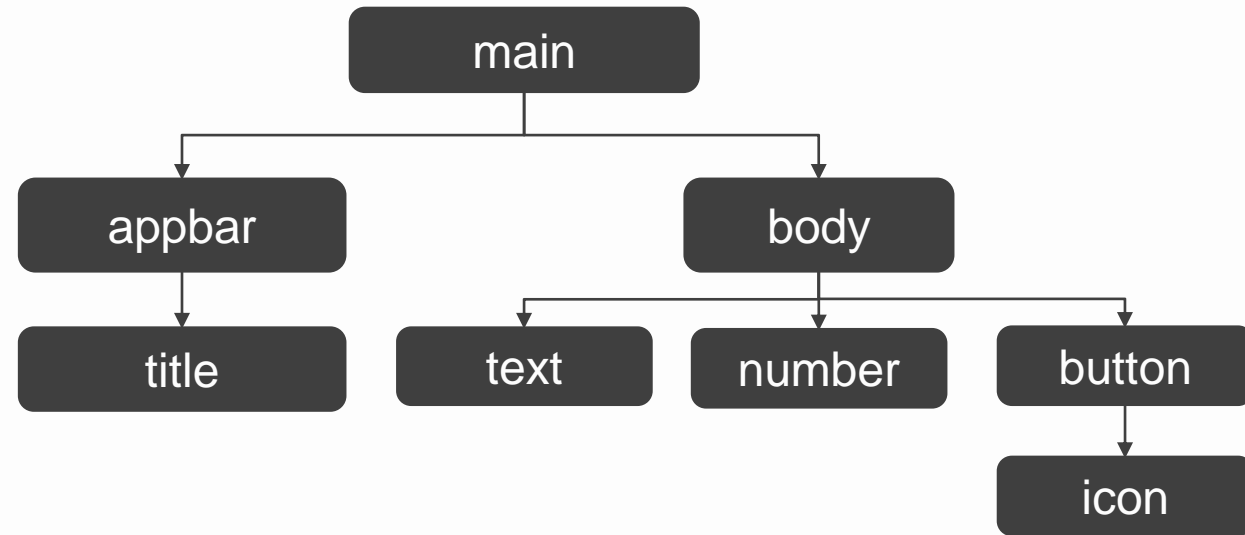
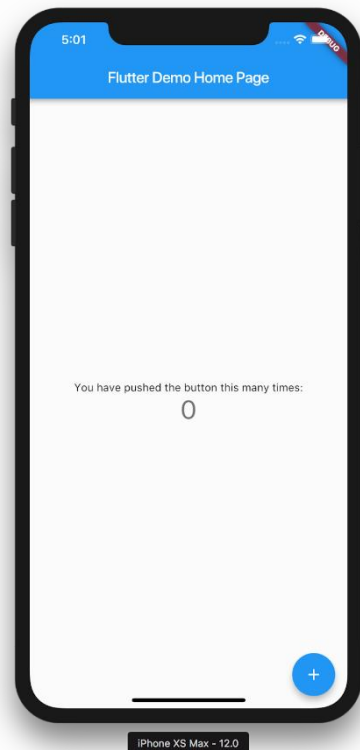
Imagen

Tweet

Botón

Widgets en Flutter


Se le conoce como Widget a cada uno de los elementos de la pantalla en una aplicación desarrollado con Flutter





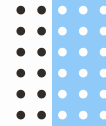
Stateless Widget

Tipo de widget sin estado, la información no puede cambiar




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

class NewWidget extends StatelessWidget{
  @override
  Widget build(BuildContext context){
    return Container();
  }
}
```



Stateful Widget

Tipo de widget con estado donde la información puede cambiar



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

class NewWidget extends StatefulWidget {
  const NewWidget({ Key? key }) : super(key: key);

  @override
  _NewWidgetState createState() => _NewWidgetState();
}

class _NewWidgetState extends State<Widget> {
  @override
  Widget build(BuildContext context) {
    return Container(color: const Color(0xFFFFE306));
  }
}
```



State en Flutter

El state o estado en Flutter se puede considerar como todo lo almacenado en memoria de una aplicación en ejecución, esto incluye todas las variables que Flutter mantiene sobre la interfaz de usuario.

El state es independiente de cada Widget y únicamente puede ser modificado en los stateful widgets

```
var contador = 2;  
  
setState(() {  
  contador++;  
})
```

Programación Orientada a Objetos

Es una forma de programar donde se organiza el código en clases que después utilizamos para crear objetos.

Esta forma de programar nos permite reutilizar código y organizar mejor el código a través de los principios de la POO como son la herencia la abstracción y el polimorfismo.

```
class Calculadora{
    int valor1;
    int valor2;

    Calculadora(int valor1,int valor2){
        this.valor1 = valor1;
        this.valor2 = valor2;
    }

    int calcular(){
        return 0;
    }
}
```

```
class Suma extends Calculadora{

    Suma(int valor1,int valor2)
    :super(valor,valor2);

    @override
    int calcular(){
        return valor1+valor2;
    }
}
```

```
class Resta extends Calculadora{

    Resta(int valor1,int valor2)
    :super(valor,valor2);

    @override
    int calcular(){
        return valor1-valor2;
    }
}
```