

Módulo 1: Introducción a Flutter

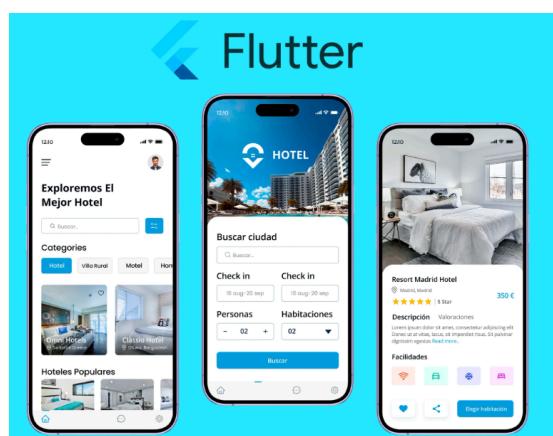
1 Qué es Flutter



Si tu objetivo es crear una aplicación desde cero, entonces debes saber qué es Flutter. Este es un framework de código abierto de Google que necesitarás para comenzar tu proyecto y hacer que funcione para distribuirlo en plataformas como Play Store o App Store.

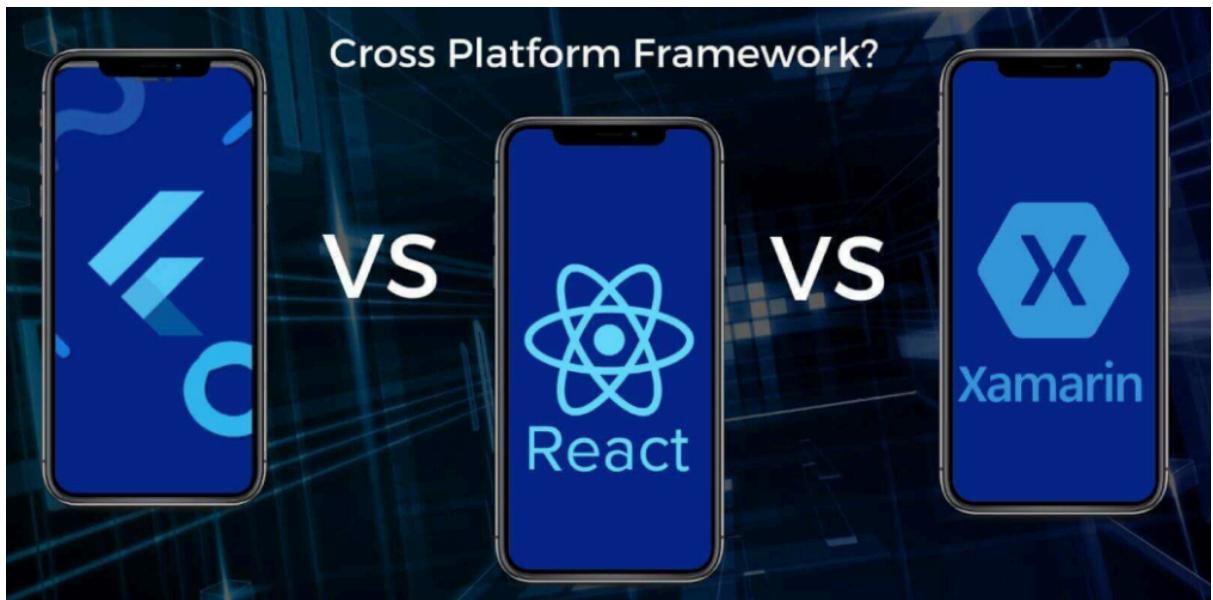
¿Para qué sirve?

Flutter sirve para crear una app gratis de alta calidad, con una interfaz de usuario fluida y personalizable, que se adapta a cualquier tamaño de pantalla y orientación. En cuanto a las directrices del diseño, se siguen los parámetros de Material Design para aplicaciones de Android y Cupertino para apps de iOS.

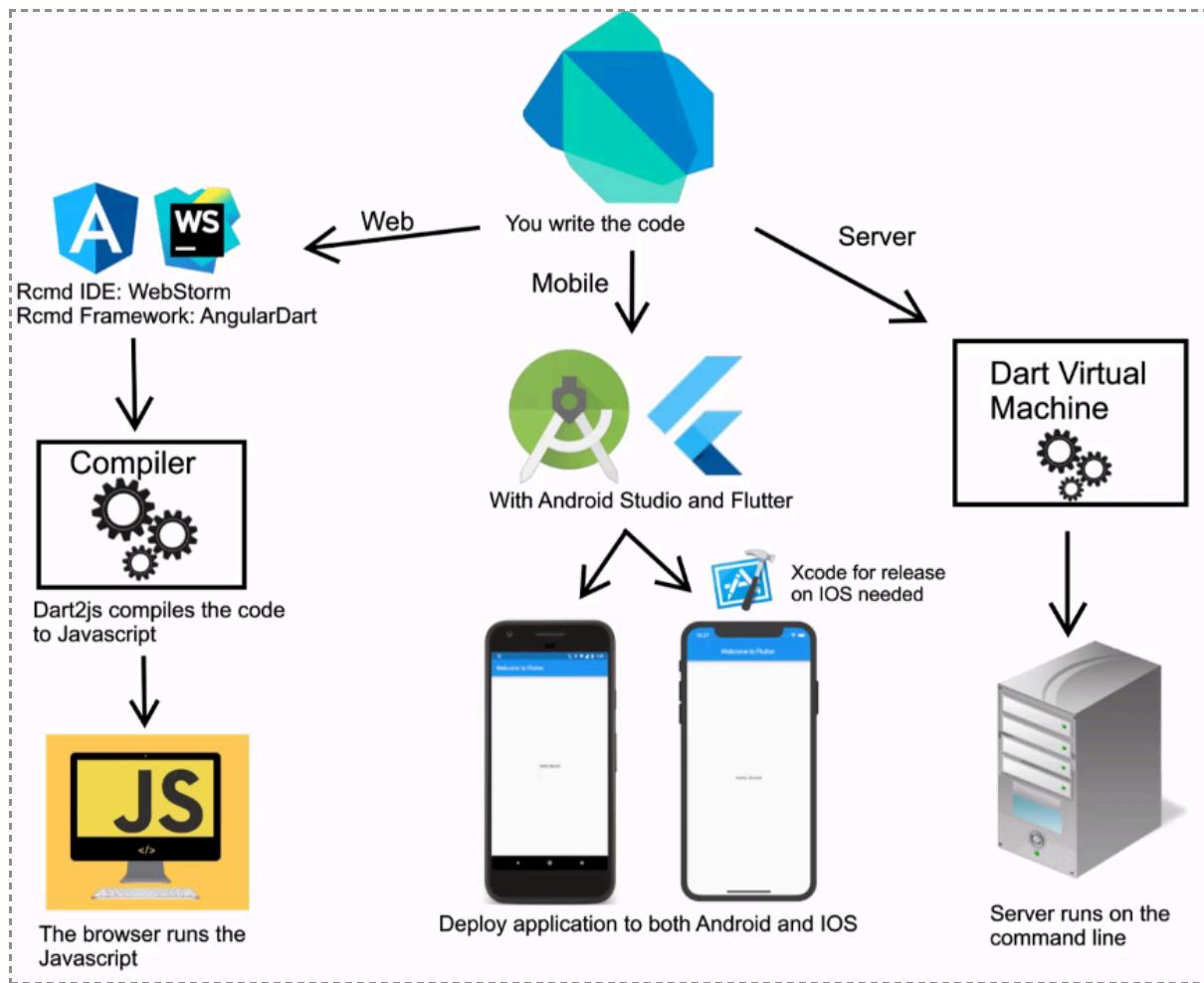
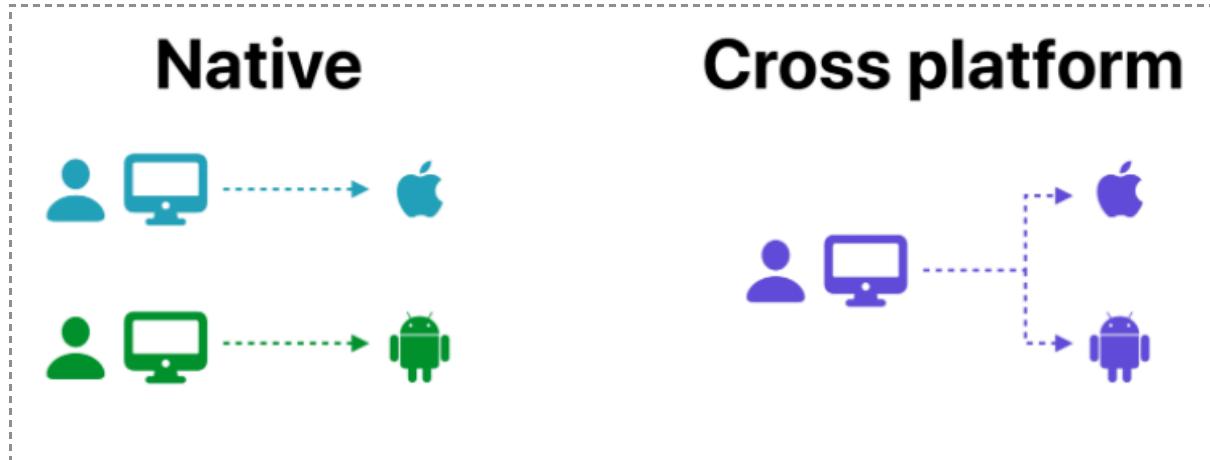


2 Ventajas de Flutter frente a otros frameworks

La elección de la tecnología adecuada para el desarrollo de aplicaciones móviles multiplataforma es una decisión crítica que puede definir el éxito o el fracaso de un proyecto. En 2025, el panorama sigue dominado por dos gigantes: Flutter y React Native. Ambos frameworks prometen eficiencia, reutilización de código y una experiencia de usuario casi nativa, pero sus enfoques, ecosistemas y rendimientos difieren significativamente. ¿Cuál es el más adecuado para su próximo gran proyecto?



[React Native vs Flutter 2025: La comparativa definitiva para desarrollo móvil empresarial](#)



3 Requisitos del sistema

Sistema operativo:

Windows 10 (64-bit) o superior (Windows 11 también es compatible).

Se recomienda la versión Pro, Enterprise o Education (aunque Home también funciona).

Espacio en disco:

Al menos 2.8 GB de espacio libre (más espacio si instalas Android Studio, emuladores, etc.).

RAM:

Mínimo 4 GB, pero se recomienda 8 GB o más para un mejor rendimiento.

Procesador:

Compatible con arquitectura x64.

4 y 5 Configuración del entorno de desarrollo (VS Code / Android Studio)

En Local Instalación de flutter + android en VsCode





-  Mobile
-  Web
-  Desktop
-  Embedded

SIN FLUTTER

DEBES ESCRIBIR EL CÓDIGO MÁS DE UNA VEZ



CONFLUTTER

DEBES ESCRIBIR EL CÓDIGO SOLO UNA VEZ



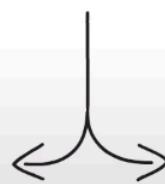
Dart



Flutter



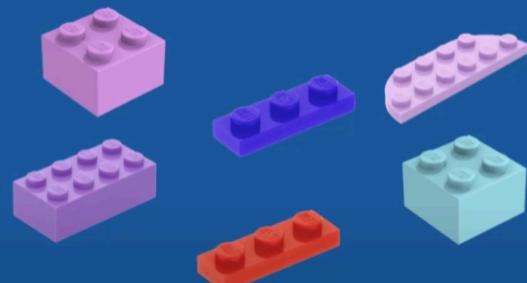
android



Patrón Reactivo

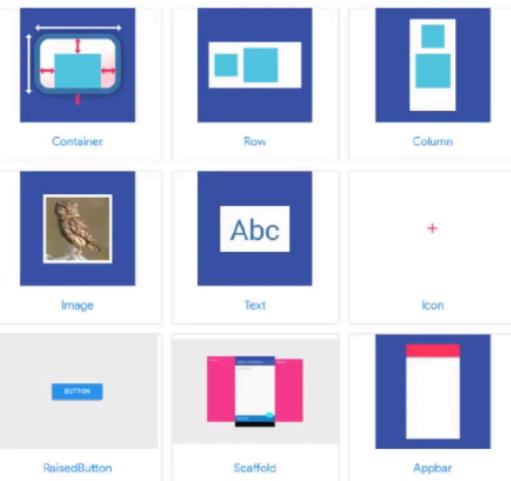
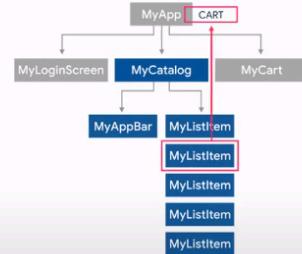
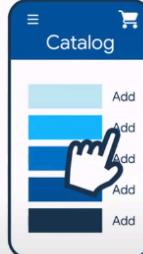
- Todo es un Widget
- Adios, posicionamiento global
- Widget personalizados
- Stateless Widgets
- Stateful Widgets

Todo en Flutter, son widgets



Patrón Reactivo

- Todo es un Widget
- Adios, posicionamiento global
- Widget personalizados
- Stateless Widgets
- Stateful Widgets



LAS APPS
SE COMPONEN
DE WIDGETS



FLUTTER INCLUYE UNA LARGA LISTA DE WIDGETS



No 1 Instala en tu computadora

[Aprenda como instalar flutter página oficial](#)

No 2 Video Instala en tu computadora

[Como Instalar Flutter en Windows \(Paso a Paso 2025\)](#)

6 Espacio de trabajo para Dart, firebase studio y FlutLab

Plataformas para trabajar

No 1 Dartpad

[Editor OnLine lenguaje de programación en Dart](#)

No 2 Espacio de trabajo en Firebase Studio (OnLine)

[Ponte a trabajar rápidamente dondequieras que estés](#)

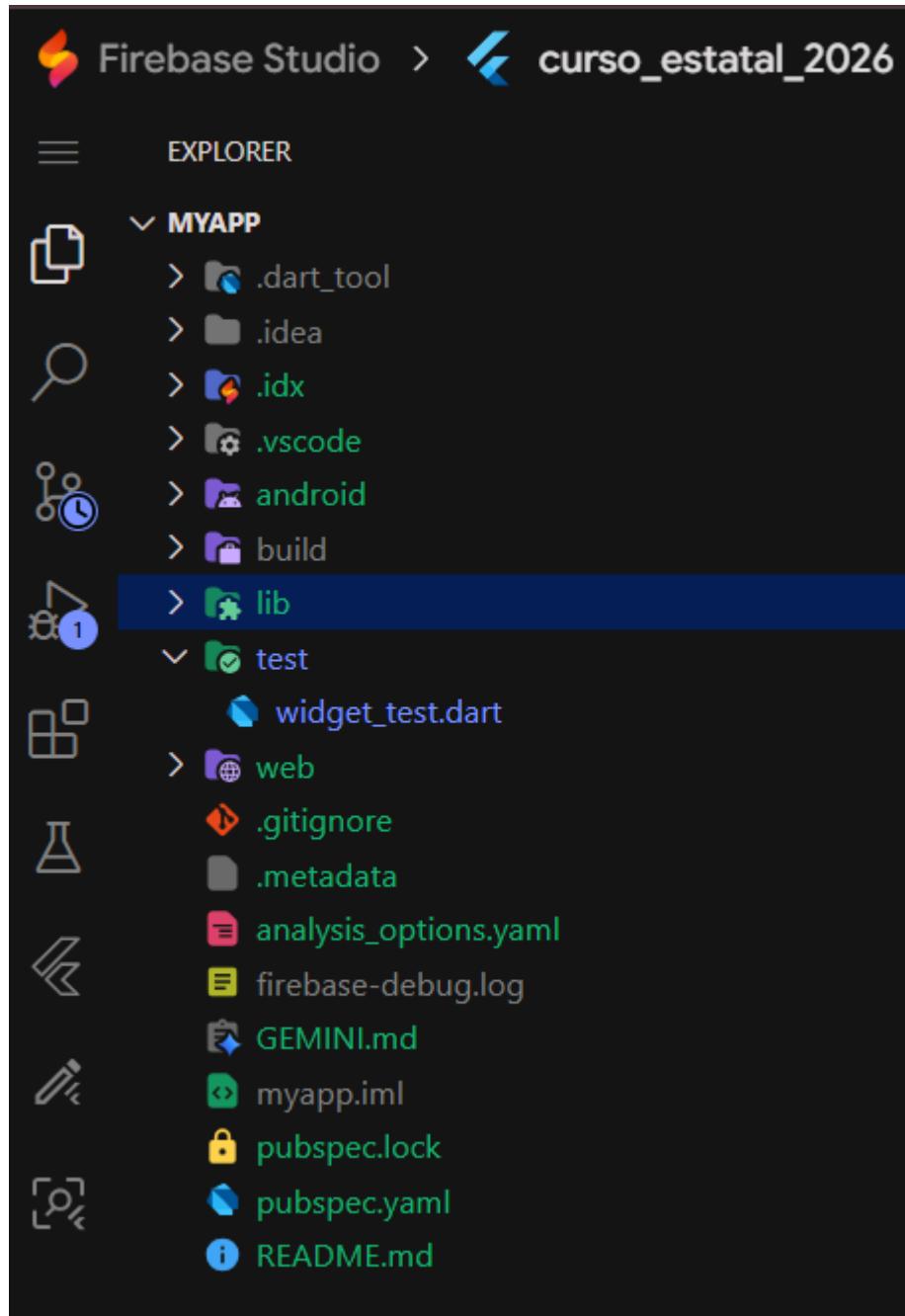
No 3 FlutLab

[FlutLab un IDE OnLine moderno para Flutter](#)

7 Primer proyecto: “Hola Mundo”

- En FlutLab
- Firebase
- <https://flutterstudio.app/>

8 Estructura de un proyecto Flutter



Estructura de un Proyecto Flutter

Un proyecto en Flutter tiene una estructura de carpetas y archivos bien definida que ayuda a mantener el código organizado y escalable. A continuación, se describen los directorios y archivos más importantes.

Directarios Principales

Directorio	Propósito
<code>lib/</code>	El corazón de tu aplicación. Contiene todo tu código Dart. La ejecución de la app comienza aquí.
<code>android/</code>	Contiene el proyecto nativo de Android. Flutter interactúa con este proyecto para construir la app en dispositivos Android. Generalmente, no necesitas editar su contenido a menos que integres funcionalidades nativas específicas.
<code>ios/</code>	Similar a la carpeta <code>android/</code> , pero para el proyecto nativo de iOS.
<code>web/</code>	Contiene los archivos necesarios para construir la versión web de tu aplicación, como el <code>index.html</code> y el manifiesto.
<code>test/</code>	En este directorio se escriben las pruebas para tu aplicación (unitarias, de widgets y de integración) para asegurar que tu código funciona como se espera.

Archivos de Configuración Clave

- **`pubspec.yaml`**: Este es uno de los archivos más importantes. Aquí defines:
 - **Metadatos del proyecto**: nombre, descripción, versión.
 - **Dependencias**: los paquetes de terceros que tu app necesita (ej: `provider`, `http`, `go_router`).
 - **Assets**: recursos como imágenes, fuentes personalizadas y archivos de configuración que la app usará.
- **`analysis_options.yaml`**: Aquí puedes configurar las reglas del analizador de código de Dart (conocido como *linter*). Te ayuda a escribir código más limpio, consistente y a evitar errores comunes.

El Punto de Entrada: `lib/main.dart`

La ejecución de toda aplicación Flutter comienza en el archivo `lib/main.dart`. Este archivo debe contener la función `main()`.

```
// lib/main.dart

import 'package:flutter/material.dart';
// Has creado una carpeta "losWidgets" para organizar tus componentes.
import 'package:myapp/losWidgets/hola.dart';

void main() {
    // runApp() inicia la aplicación con el widget que le pases.
    runApp(const MiApp());
}

class MiApp extends StatelessWidget {
    const MiApp({super.key});

    @override
    Widget build(BuildContext context) {
        // MaterialApp es el widget raíz que configura la app (rutas, tema,
        return MaterialApp(
            title: 'Los widgets en Flutter',
            theme: ThemeData(
                primarySwatch: Colors.blue,
            ),
            // La propiedad "home" define la primera pantalla que se mostrará
            home: InicioHola(),
        );
    }
}
```

Organización del Código Dentro de `lib/`

A medida que tu aplicación crece, no es práctico tener todos los archivos `.dart` directamente en la carpeta `lib/`. La mejor práctica es crear subdirectorios para organizar tu código.

En tu proyecto, ya has comenzado a hacer esto creando la carpeta `lib/losWidgets/`.

- `lib/losWidgets/hola.dart`: Este archivo contiene un widget específico (`InicioHola`), que es la pantalla principal de tu app. Separar los widgets en sus propios archivos hace que el código sea mucho más fácil de leer y mantener.

```
// lib/losWidgets/hola.dart

import 'package:flutter/material.dart';

// Un widget simple que representa una pantalla con una barra de título
class InicioHola extends StatelessWidget {
    const InicioHola({super.key});

    @override
    Widget build(BuildContext context) {
        return Scaffold(
            appBar: AppBar(
                title: Text('Hola mundo'),
            ),
            body: const Center(
                child: Text('Hola mundo cruel!'),
            ),
        );
    }
}
```

Patrón Reactivo

- Todo es un Widget
- Adios, posicionamiento global
- Widget personalizados
- Stateless Widgets
- Stateful Widgets

```
graph TD; MyApp --> MyLoginScreen; MyApp --> MyCatalog; MyApp --> MyCart; MyCatalog --> MyAppBar; MyCatalog --> MyListitem[MyListitem]; MyListitem --- Catalog
```

Recursos recomendados

No 1 **Página oficial**

[Crea aplicaciones para cualquier pantalla](#)

No 2 **Basic widgets**

[Catálogo de widgets básicos en flutter](#)

No 3 **Layout widgets**

[Componentes de diseño widgets](#)

No 4

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