Universidad icesi

Aplicaciones Móviles

DOMICIANO RINCÓN

INGENIERÍA TELEMÁTICA
INGENIRÍA DE SISTEMAS
DISEÑO DE MEDIOS INTERACTIVOS

Universidad ICESI

Composición del curso

1. **UNIDAD 1**
   Fundamentos de programación en Android/Kotlin
2. **UNIDAD 2**
   Diseño, concpeto y prototipado
3. **UNIDAD 3**
   Arquitecturas y cloud
4. **UNIDAD 4**
   Construcción y despliegue

Composición del curso

1. **UNIDAD 1**
   Fundamentos de programación en Android
   - Android Studio
   - Estructura
   - Componentes de una app
   - Elementos de interfaz
2.
3.
4.

Composición del curso

1

2

UNIDAD 2
Diseño, Ideación y prototipado

- Sketch
- Wireframe
- Mockup

3

4

Composición del curso

1

2

3

4

UNIDAD 3
Cloud integration y servicios

Persistencia
Georreferenciación
SaaS, consumo de servicios REST, HTTP
Conexión con Cloud (Firebase SDK)

Composición del curso

1

2

3

4

UNIDAD 4
Mínimo viable y despliegue

Despligue en Firebase
Despliegue en Google Play (Signed ABB/APK)

Calificación

Pitch Elevator 10%

Diseño de base de datos 15%

Prototipo gráfico de alta fidelidad 15%

Sprint 1 15%

Sprint 2 15%

Sprint 3 15%

Exhibición 15%

Próximo evento

Pitch Elevator

Deberá preparar 3 propuestas de proyecto final de aplicaciones móviles. Lo hará en una exposición sucinta en la que el objetivo es convencer a la audiencia. **25 de agosto de 2025**

CLASES

Fuera de clase

- Material del curso
- Implementación
- Instrucciones
- Mecánicas

El cómo

En clase

- Laboratorios
- Conceptos
- Análisis de problemas
- Dudas

El porqué

Comunicación

Whatsapp

Youtube

Contenido

Universidad ICESI

Mentimeter

GO

Universidad ICESI

Clase 1

CONCEPTOS INICIALES
FAMILIARIZACIÓN CON EL SISTEMA

1. Introducción

Relevancia

- Reunir información
- Posicionamiento de marca
- Canal de comunicación
- Publicar un servicio

Gracias al uso masivo de teléfonos inteligentes y a la amplia cobertura de internet, ha surgido el mercado de las aplicaciones móviles.

La portabilidad del Smartphone es un aspecto clave.

Relevancia

- Reunir información
- Posicionamiento de marca
- Canal de comunicación
- Publicar un servicio

Las empresas querrán tener una base de datos de sus clientes y información relacionada con ellos para plantear estrategias de mercado.

Relevancia

- Reunir información
- Posicionamiento de marca
- Canal de comunicación
- Publicar un servicio

→ Mediante una aplicación se puede popularizar una marca. Usando como vitrina la tienda de aplicaciones y atrapando clientes con los servicios ofrecidos.

Relevancia

- Reunir información
- Posicionamiento de marca
- Canal de comunicación
- Publicar un servicio

Una aplicación crea un canal de comunicación entre la empresa y el cliente donde se puede intercambiar información relevante como solicitudes, noticias, cambios o notificaciones entre otros.

Relevancia

- Reunir información
- Posicionamiento de marca
- Canal de comunicación → El accionamiento remoto es muy usado a nivel industrial.
- Publicar un servicio

Relevancia

Reunir información

Posicionamiento de marca

Canal de comunicación

Publicar un servicio → Proveer un servicio a un público objetivo

Sistema Operativo

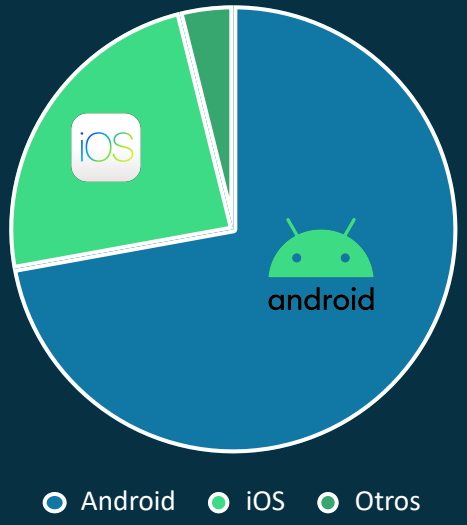
- Es un sistema operativo diseñado para ser ejecutado por dispositivos móviles con pantalla táctil.
- Tiene licencia Apache y GNU GPL que da libertad a cualquiera de usarlo y modificarlo.
- En los últimos años debido a su diseño basado en aplicaciones y su licencia libre, ha sido adoptado por numerosas compañías de electrónica de consumo como el sistema operativo de sus teléfonos
- El lenguaje de desarrollo para aplicaciones en Android es **Kotlin**.

Sistema Operativo

- Es un sistema operativo diseñado para iPhone desarrollado por Apple Inc.
- Tiene una licencia propietaria y de código cerrado, su uso y modificación están controlados por Apple.
- Debido a su diseño basado en aplicaciones, su interfaz intuitiva, la integración con el ecosistema de hardware de Apple y su fuerte énfasis en la privacidad y seguridad, ha sido adoptado para iPhone, iPad, Apple Watch y Apple TV.
- El lenguaje de desarrollo principal para aplicaciones en iOS es **Swift**.

Estadísticas

- La cuota de mercado de Android (72.15%) es superior a iOS (23.96%).
- Sin embargo, iOS no es para nada despreciable.
- Hay gran demanda de desarrolladores móviles nativos
- Las soluciones híbridas también tienen un gran auge. Sobre todo por el ahorro en costos de desarrollo



Fuente: Statita
A 2024

API Level

Lolipop 22	Marshmallow 23	Nougat 24	Nougat 25
Oreo 26	Oreo 27	Pie 28	Quince Tart Android 10 29
Red Velvet Cake Android 11 30	Snow Cone Android 12 31	Snow Cone Android 12L 32	Tiramisú Android 13 33
Upside Down Cake Android 14 34	Vanilla Ice Cream Android 15 35	Baklava Android 16 36	

...

API Level

iOS 7 | iOS 8 | iOS 9 | iOS 10

iOS 11 | iOS 12 | iOS 13 | iOS 14

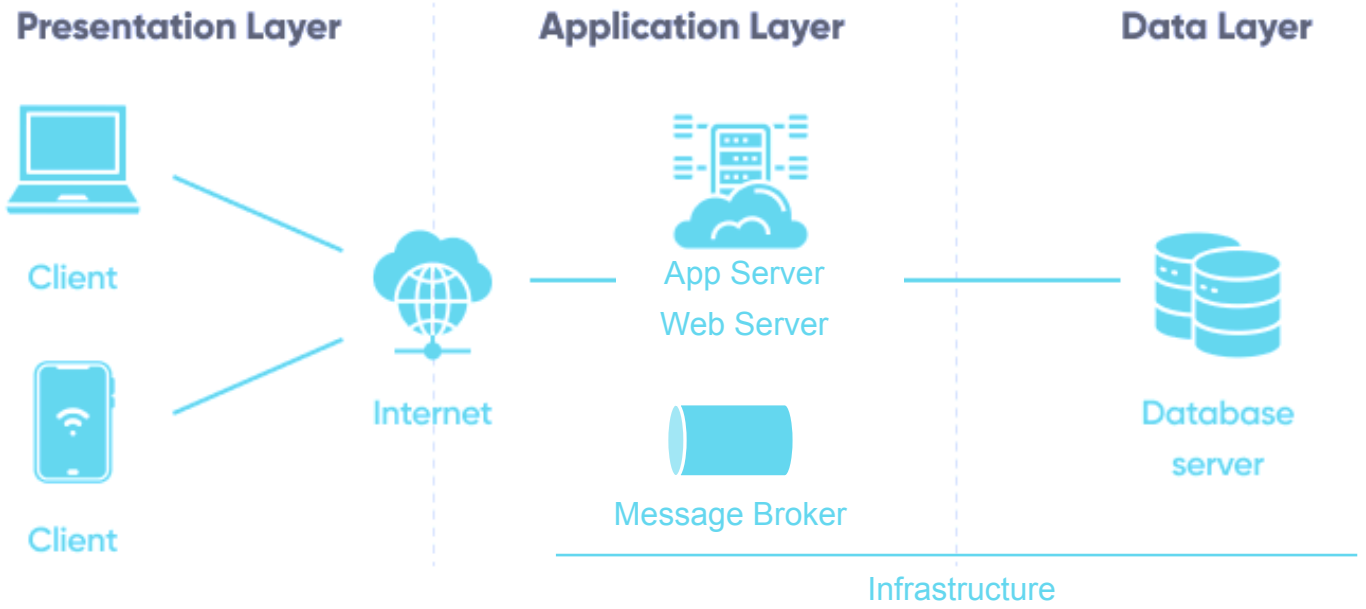
...

iOS 15 | iOS 16 | iOS 17 | iOS 18

iOS 26

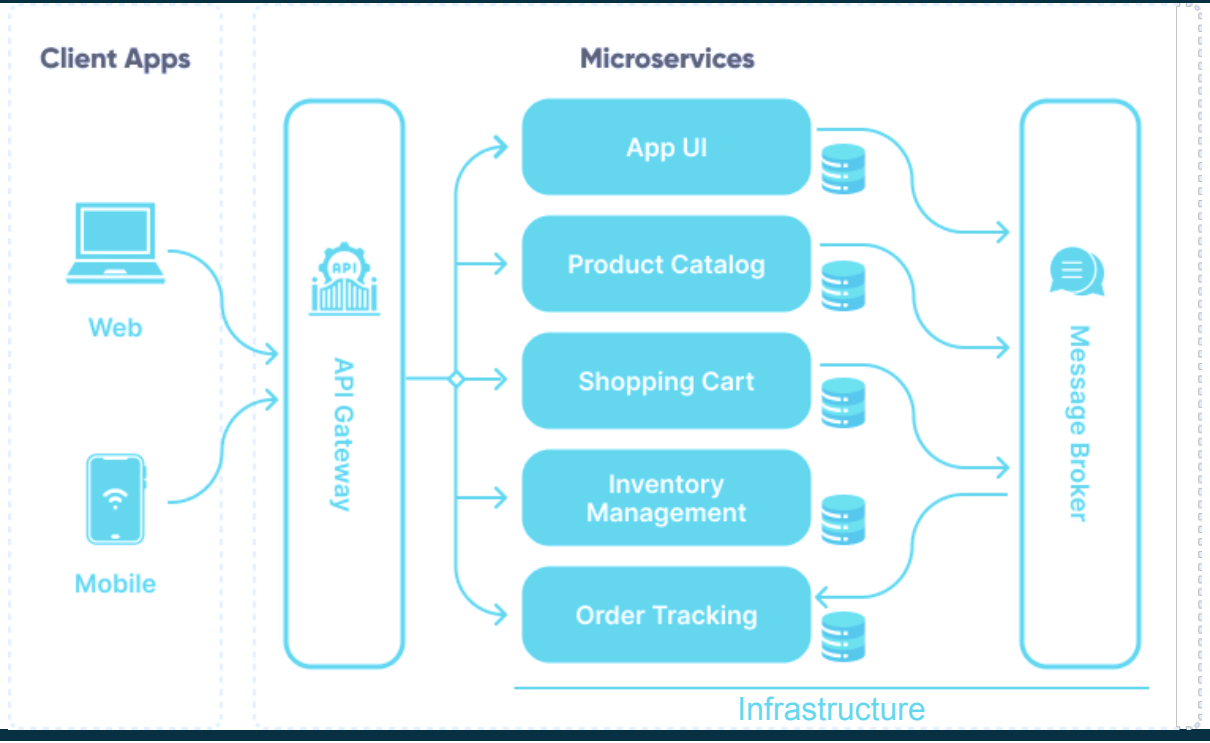
Topologías actuales

Fuente: Softkraft



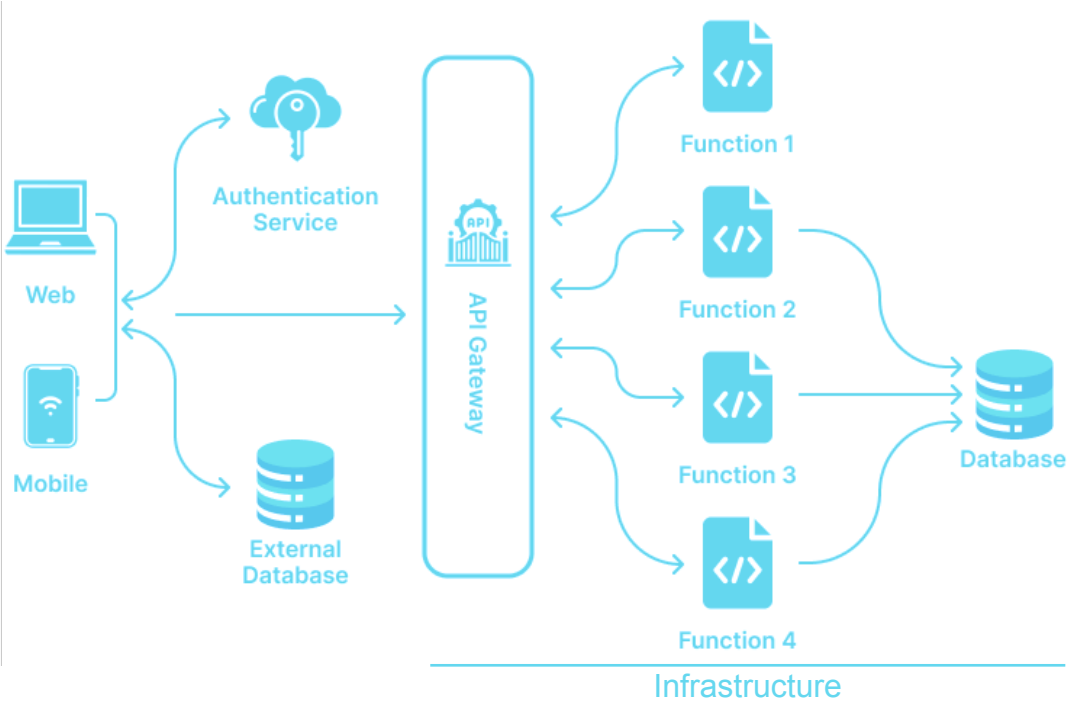
Topologías actuales

Fuente: Softkraft



Topologías actuales

Fuente: Softkraft

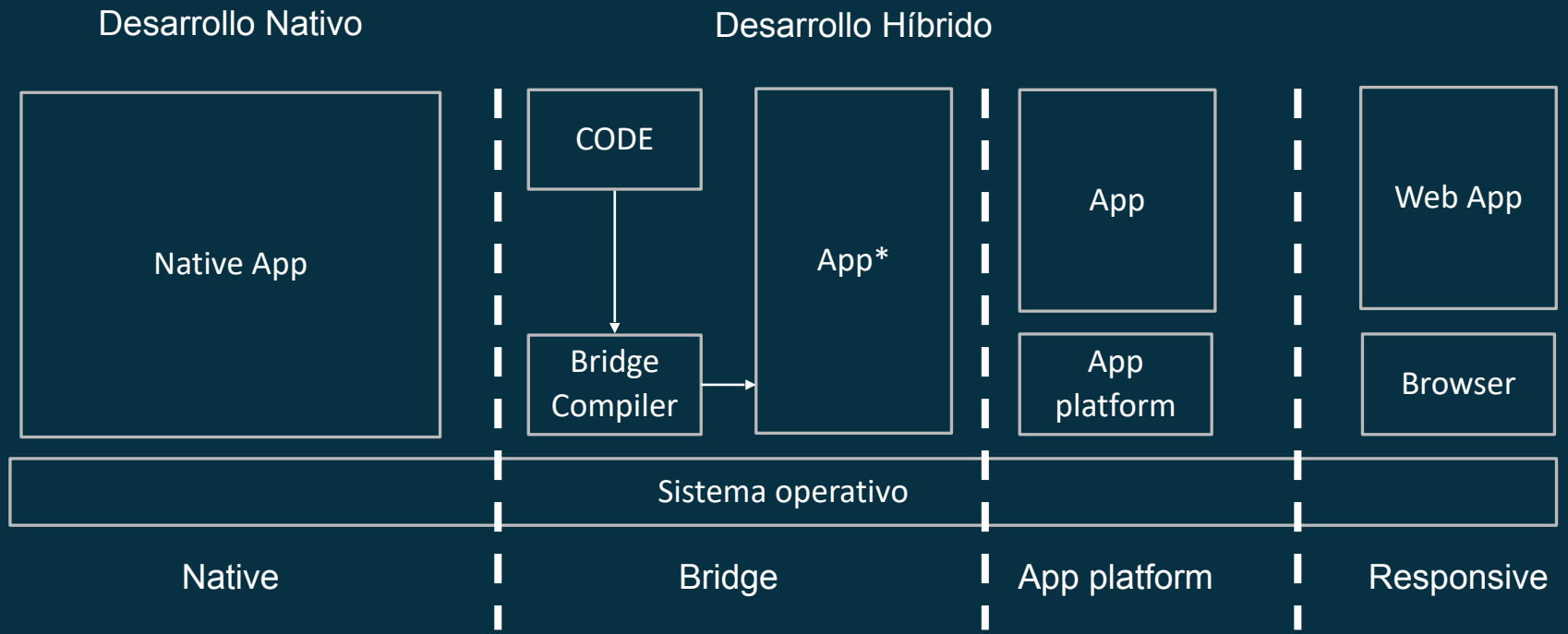


Universidad ICESI

Tendencias del desarrollo móvil

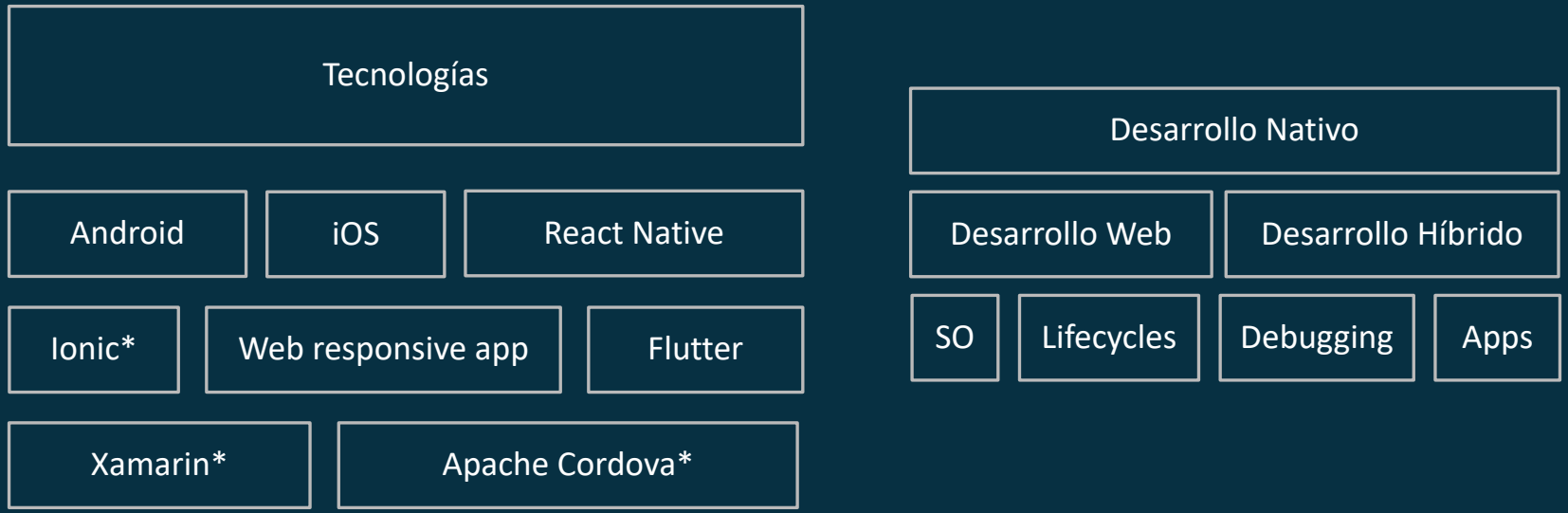
ANDROID

Tendencia de desarrollo



Universidad ICESI

Mobile roadmap



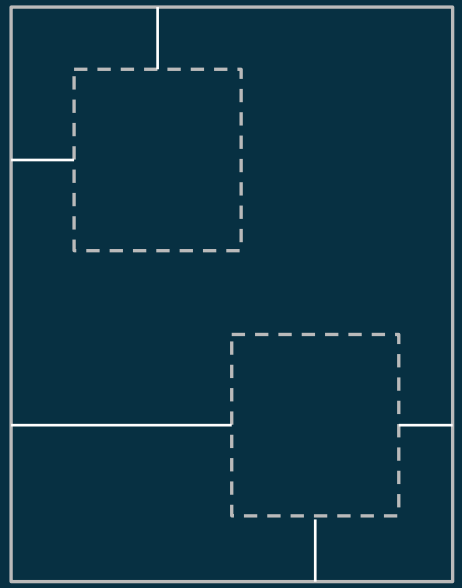
Universidad ICESI

Mobile roadmap

Tendencia de desarrollo

LEGACY

Lenguaje de enmaquetado
+
Herramienta Drag and Drop



NOWDAYS

```
Card {
  Column{
    Image(...)
    Text(...)
    Row{...}
  }
}
```

Vistas declarativas
State Management

Flutter

Jetpack Compose

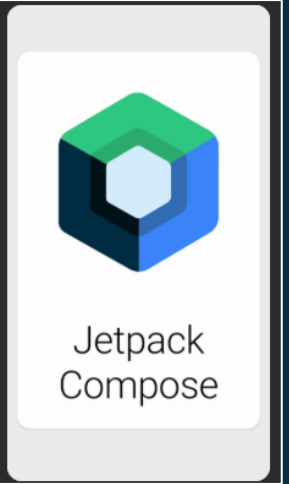
Swift UI

Tendencia de desarrollo

Desarrollo Nativo

Lenguaje declarativo para vistas

```
@Composable
fun JetpackCompose() {
    Card {
        var expanded by remember { mutableStateOf(false) }
        Column(Modifier.clickable { expanded = !expanded }) {
            Image(painterResource(R.drawable.jetpack_compose))
            AnimatedVisibility(expanded) {
                Text(
                    text = "Jetpack Compose",
                    style = MaterialTheme.typography.h2,
                )
            }
        }
    }
}
```



Tendencia de desarrollo

Desarrollo Nativo

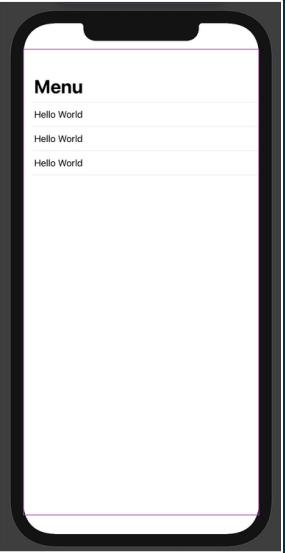
Lenguaje declarativo para vistas

```swift
//  ContentView.swift
//  iDine
//
//  Created by Paul Hudson on 05/02/2021.
//

import SwiftUI

struct ContentView: View {
    var body: some View {
        NavigationView {
            List {
                Text("Hello World")
                Text("Hello World")
                Text("Hello World")
            }
            .navigationTitle("Menu")
        }
    }
}

struct ContentView_Previews: PreviewProvider {
```



Tendencia de desarrollo

Desarrollo Híbrido

Lenguaje declarativo para vistas

```
child: new Column(
  mainAxisAlignment: MainAxisAlignment.end,
  crossAxisAlignment: CrossAxisAlignment.center,
  children: <Widget>[
    new Padding(
      padding: const EdgeInsets.all(6.0),
      child: new Icon(
        category.icon,
        size: 60.0,
        color: isDark ? Colors.white : _kFlutterBlue,
      ), // Icon
    ), // Padding
    const SizedBox(height: 10.0),
    new Container(
      height: 48.0,
      alignment: Alignment.center,
      child: new Text(
        category.name,
```

Universidad ICESI

Características

Característica	Flutter	Nativo
Lenguaje	Dart	Kotlin/Swift (Android/iOS)
Rendimiento	Similar a nativo, pero procesos complejos de la interfaz de usuario pueden ser más lentas	Mejor rendimiento posible
Tiempo de desarrollo	Más rápido debido a la base de código única	Generalmente más lento, requiere código separado para cada plataforma
Mantenimiento	Más fácil, debido a una única base de código	Más complejo, requiere mantener bases de código separadas

Característica	Flutter	Nativo
Comunidad y soporte	Creciente, pero aún más pequeña que las comunidades de desarrollo nativas	Grande y madura, con amplios recursos de soporte
Soporte de paquetes	Bueno y en crecimiento, pero algunos paquetes aún pueden faltar o ser menos maduros	Amplio soporte para diversas funcionalidades a través de numerosas bibliotecas y herramientas
API y características específicas de la plataforma	Se pueden usar, pero se requiere código nativo adicional, lo que puede complicar el desarrollo	Acceso directo a todas las API y características de la plataforma
Soporte de CI/CD	Posible, pero menos herramientas y configuraciones listas para usar	Completo, con muchas herramientas listas para usar como Fastlane, Jenkins, etc.

Universidad ICESI

Casos de uso

ANDROID

Caso de uso	Flutter	Nativo
Prototipado rápido	Perfecto para prototipado rápido gracias a la función de "hot reload" y a la base de código única	Puede no ser tan adecuado debido al mayor tiempo de desarrollo
Restricciones presupuestales	Más rentable debido a la base de código única y al menor tiempo de desarrollo	Puede ser más costoso debido a la necesidad de desarrollo separado para cada plataforma
Acceso a las últimas funciones nativas	Puede no ser la mejor opción si se desea un acceso inmediato a las últimas funciones específicas de la plataforma	La mejor opción si se desea un acceso inmediato a las últimas funciones específicas de la plataforma al ser lanzadas

Caso de uso	Flutter	Nativo
Complejidad de la aplicación	Bueno para aplicaciones de complejidad media, pero puede tener problemas de rendimiento con aplicaciones de alto nivel o aplicaciones con gráficos intensivos	Ideal para crear aplicaciones de alto rendimiento y complejas, como juegos en 3D o aplicaciones con gran carga computacional
Disponibilidad de desarrolladores	La comunidad de Flutter está creciendo, pero aún es más pequeña que las comunidades de desarrollo nativas	Más fácil encontrar desarrolladores con experiencia, ya que Java, Kotlin, Swift y Objective-C son ampliamente utilizados
UX/UI específica de la plataforma	Menos adecuado si se quiere seguir estrictamente los lenguajes de diseño específicos de la plataforma (Material Design en Android, Human Interface en iOS)	Ideal si se quiere seguir estrictamente los lenguajes de diseño específicos de la plataforma