Construye aplicaciones para iOS y Android 100% en Kotlin con Compose Multiplatform

CMP for Mobile Native Developers

Presentación

Santiago Mattiauda Verificar

Mobile Software Engineer

Montevideo, Montevideo, Uruguay · **Información de contacto**

Mercado Libre

Universidad de la Empresa

Contexto

El desarrollo de aplicaciones móviles multiplataforma permite crear una única base de código que funcione en Android e iOS, **optimizando tiempo**, **esfuerzo** y **recursos**, frente al desafío de desarrollar versiones separadas para cada plataforma.

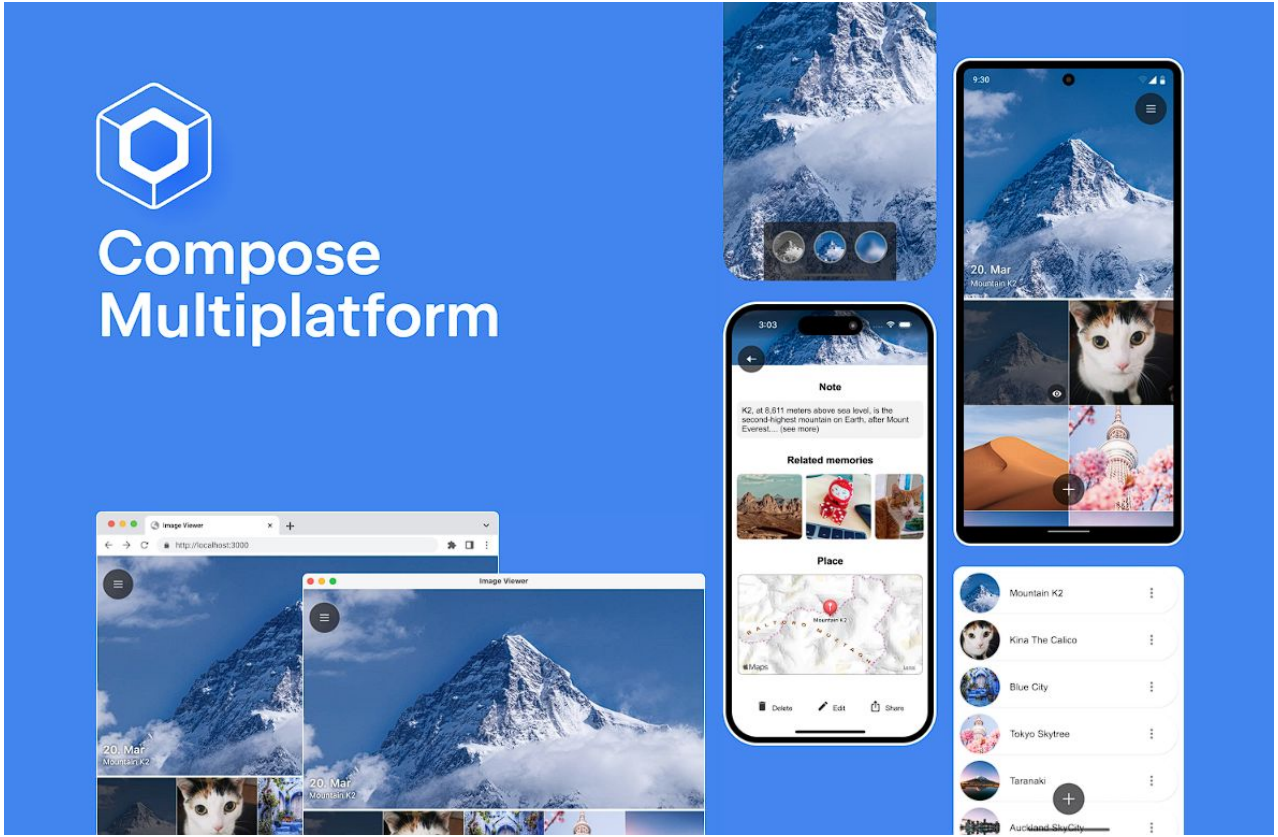
💪 Motivación

- **Eficiencia y consistencia**: Permiten desarrollar la lógica de negocio y la interfaz de usuario una sola vez, asegurando consistencia y reduciendo el tiempo de desarrollo en todas las plataformas.
- **Mantenimiento simplificado**: Una base de código unificada facilita el mantenimiento y la actualización, permitiendo implementar correcciones y nuevas funcionalidades simultáneamente en todas las plataformas.
- **Aprovechamiento de recursos nativos**: Permiten acceder a las funcionalidades nativas de Android e iOS cuando es necesario, garantizando un rendimiento óptimo.
- **Velocidad de desarrollo**: Al gestionar una única aplicación, las empresas pueden acelerar el ciclo de desarrollo y llegar al mercado más rápidamente.
- **Flexibilidad y escalabilidad**: Esta solución permite compartir la lógica y la interfaz de usuario, manteniendo la capacidad de adaptar aspectos específicos a cada plataforma según sea necesario.

⏳ Evolución de las herramientas multiplataforma

- Comienza con WebView y Apache Cordova, que permitían empaquetar aplicaciones web en contenedores nativos, aunque con rendimiento inferior.
- Xamarin y React Native, que mejoraron la experiencia de usuario al permitir el uso de **C#** y **JavaScript**.
- Flutter desarrollado por Google, que utiliza **Dart** para crear aplicaciones nativas de alta calidad.

Compose Multiplatform

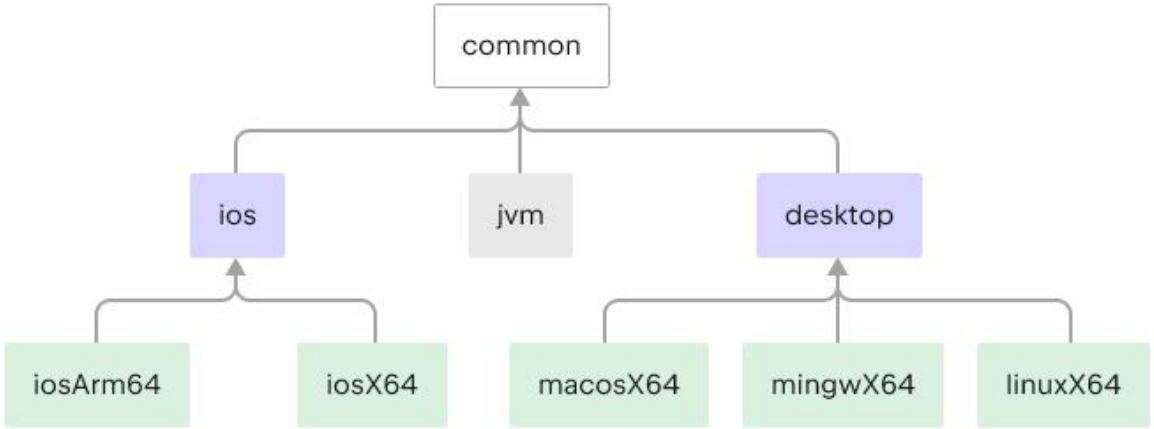


Objetivos de la charla

- Entender qué es Compose Multiplatform.
- Conocer sus características y beneficios.
- Explorar ejemplos prácticos de su uso.

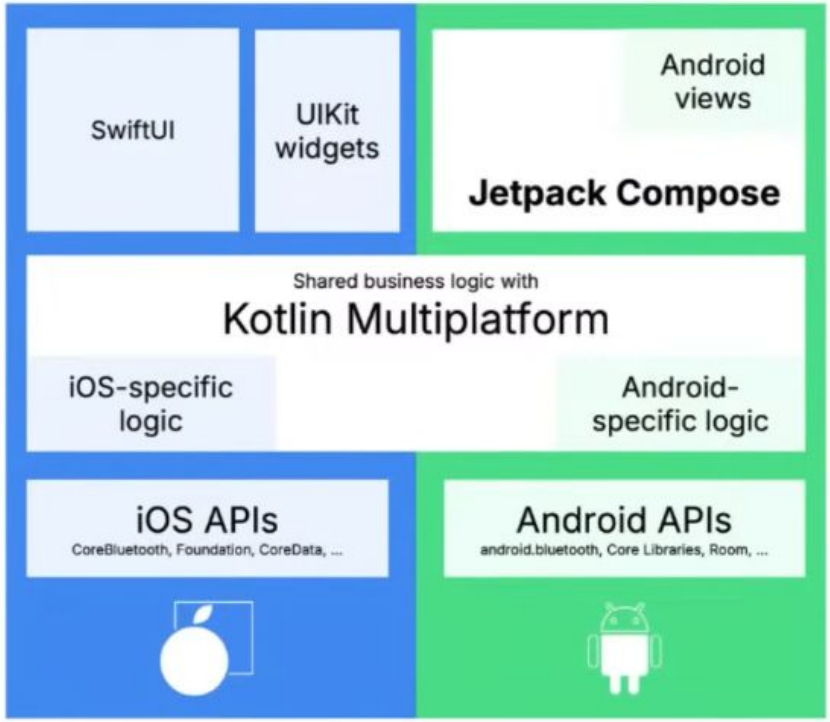
¿Qué es Kotlin Multiplatform?

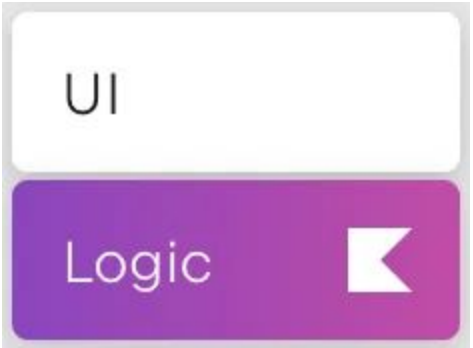
Kotlin Multiplatform es una tecnología desarrollada por JetBrains que permite a los desarrolladores escribir código en el lenguaje de programación Kotlin y compartirlo entre múltiples plataformas, como Android, iOS, Web y Escritorio.



KMP for Mobile Native Developer — Part. 1: The beginning

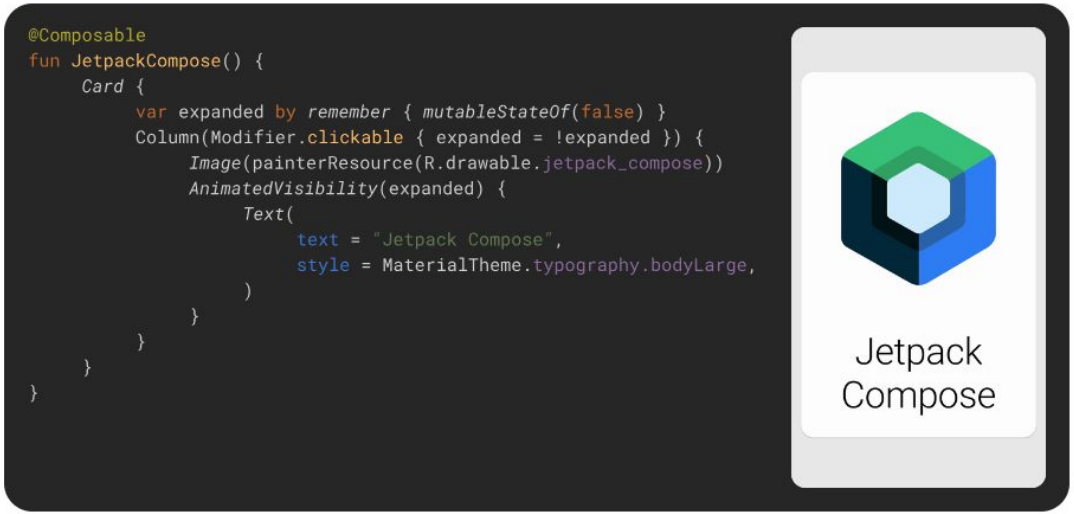
¿Qué es Kotlin Multiplatform? (2)





¿Qué es Jetpack Compose?

Jetpack Compose es un framework de UI declarativa desarrollado por Google para construir interfaces de usuario en aplicaciones Android. Jetpack Compose permite a los desarrolladores diseñar y gestionar la interfaz de usuario directamente en código Kotlin, lo que simplifica la creación y actualización de las vistas.



¿Qué es Jetpack Compose? (2)

XML

example.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <Button
        android:id="@+id/button"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="My Button" />
</LinearLayout>
```

Jetpack Compose

Example.kt

```kotlin
@Composable
fun Example(modifier: Modifier = Modifier) {
    Column {
        Button(onClick = { /*TODO*/ }) {
            Text(text = "My Button")
        }
    }
}
```

¿Qué es Compose Multiplatform?

Kotlin Multiplatform

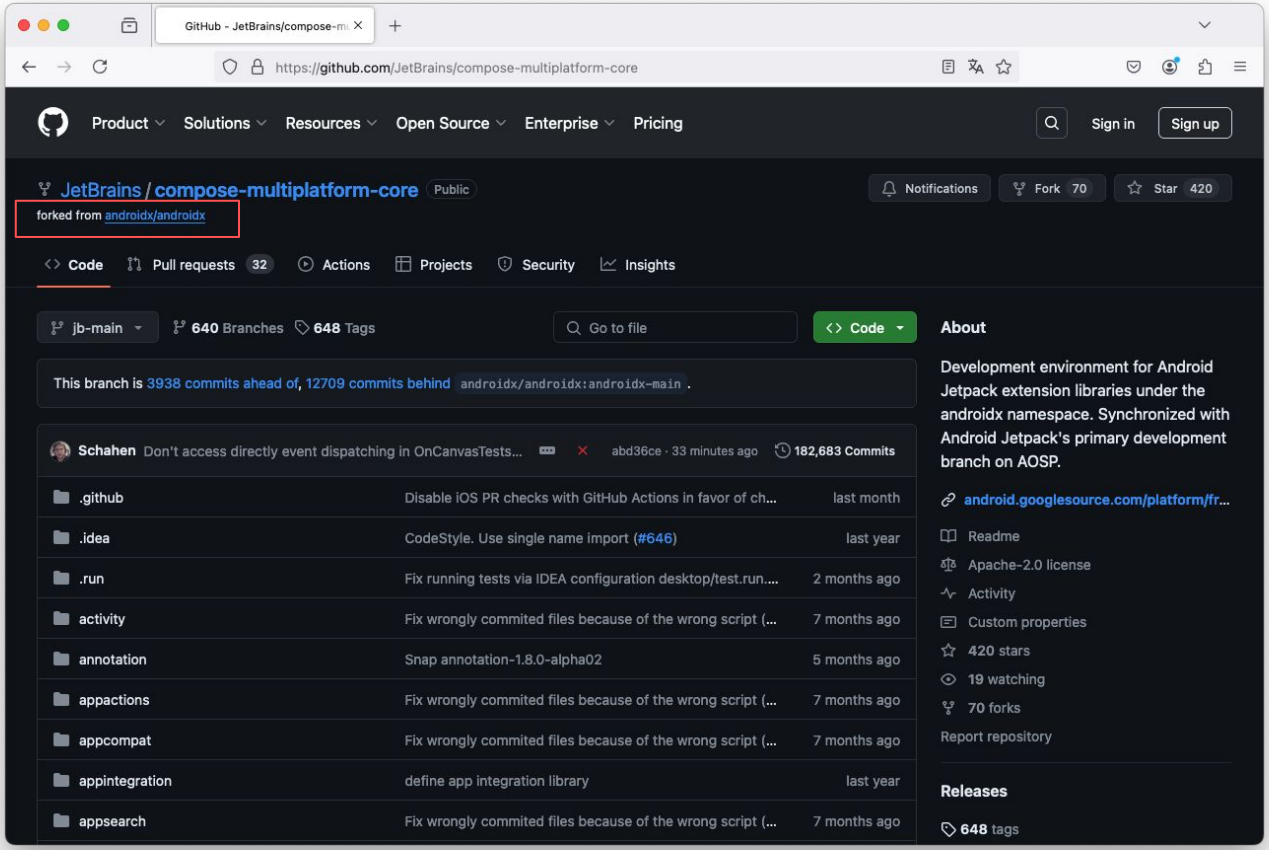
+

Jetpack Compose

=

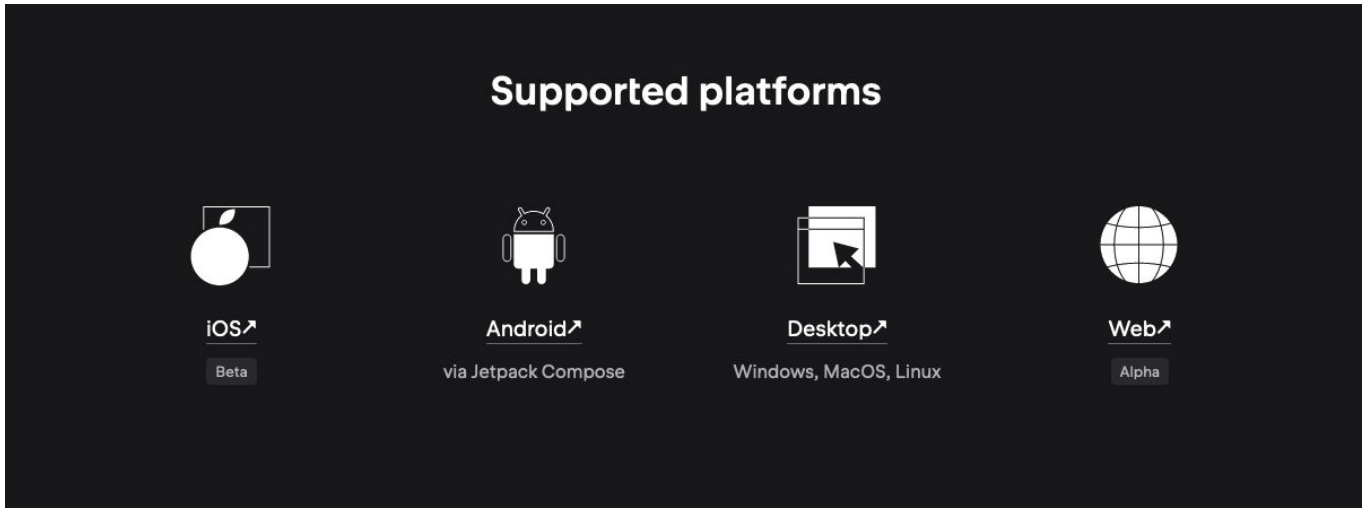
Compose Multiplatform

¿Qué es Compose Multiplatform? (2)

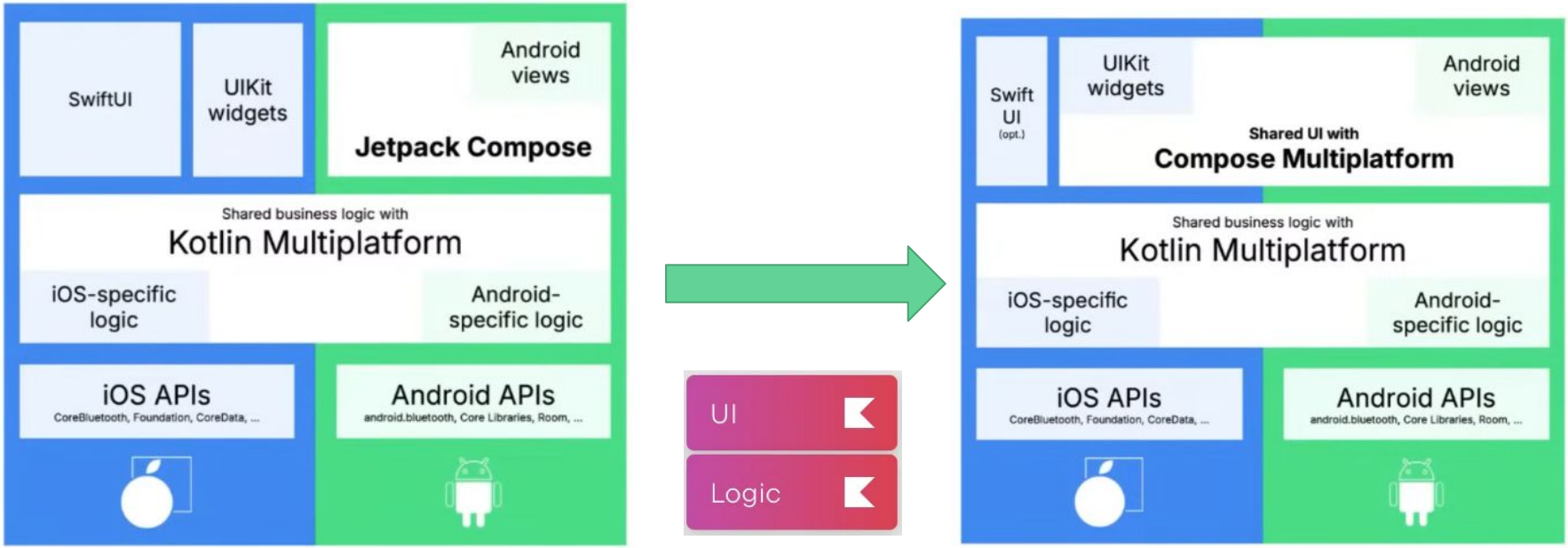


¿Qué es Compose Multiplatform? (3)

Compose Multiplatform es una extensión de Jetpack Compose que permite a los desarrolladores crear interfaces de usuario reutilizables para múltiples plataformas (Android, iOS, escritorio y web) utilizando una sola base de código en Kotlin, simplificando y unificando el desarrollo de UI.



¿Qué es Compose Multiplatform? (4)



Propósito de Compose Multiplatform

Compose Multiplatform busca resolver uno de los principales desafíos del desarrollo de software moderno: **la fragmentación de plataformas**. Al permitir la creación de una interfaz de usuario única que pueda adaptarse y funcionar en diferentes entornos, Compose Multiplatform no solo **reduce el tiempo de desarrollo**, sino que también garantiza que los usuarios de diversas plataformas reciban una **experiencia coherente** y de **alta calidad**.

🎯 Compose Multiplatform: Arquitectura General (Core)

Tendremos el core de Jetpack Compose:

- **UI Components**: Define componentes básicos y funciones de UI (botones, textos, listas, etc.) que son comunes a todas las plataformas.
- **Compose Runtime**: Determina cómo y cuándo actualizar la UI en función de cambios en el estado.

🎯 Compose Multiplatform: Arquitectura General (Platform Abstractions)

Para extender a Jetpack Compose, Compose Multiplataforma incluye implementaciones abstractas sobre las plataformas, por ejemplo:

- **Android**: Extiende el framework original de Jetpack Compose con algunas adaptaciones necesarias para la integración multiplataforma.
- **iOS**: Utiliza Kotlin/Native para compilar código Kotlin a binarios nativos y se integra con UIKit para construir interfaces de usuario.

Expect - Actual

WebView.common.Kt.kt

```kotlin
//WebView.common.kt
@Composable
expect fun WebView(modifier: Modifier = Modifier, url: String)

```

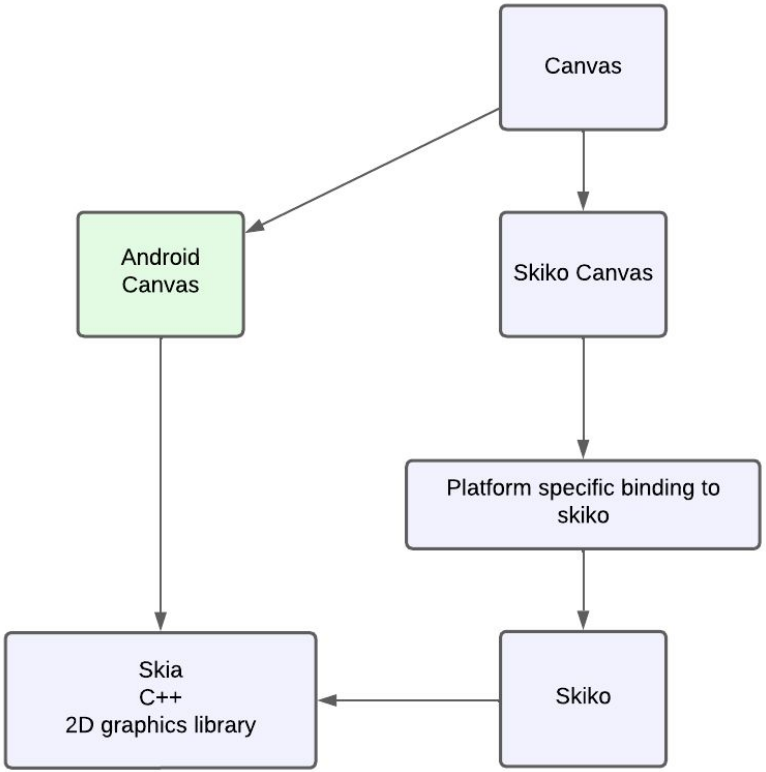
WebView.android.Kt.kt

```kotlin
//WebView.android.kt
@Composable
actual fun WebView(modifier: Modifier, url: String) {

    AndroidView(factory = {
        AndroidWebView(it).apply {
            layoutParams = ViewGroup.LayoutParams(
                ViewGroup.LayoutParams.MATCH_PARENT,
                ViewGroup.LayoutParams.MATCH_PARENT
            )
            loadUrl(url)
            webViewClient = CustomWebViewClient()
        }
    }, update = {})
}
```

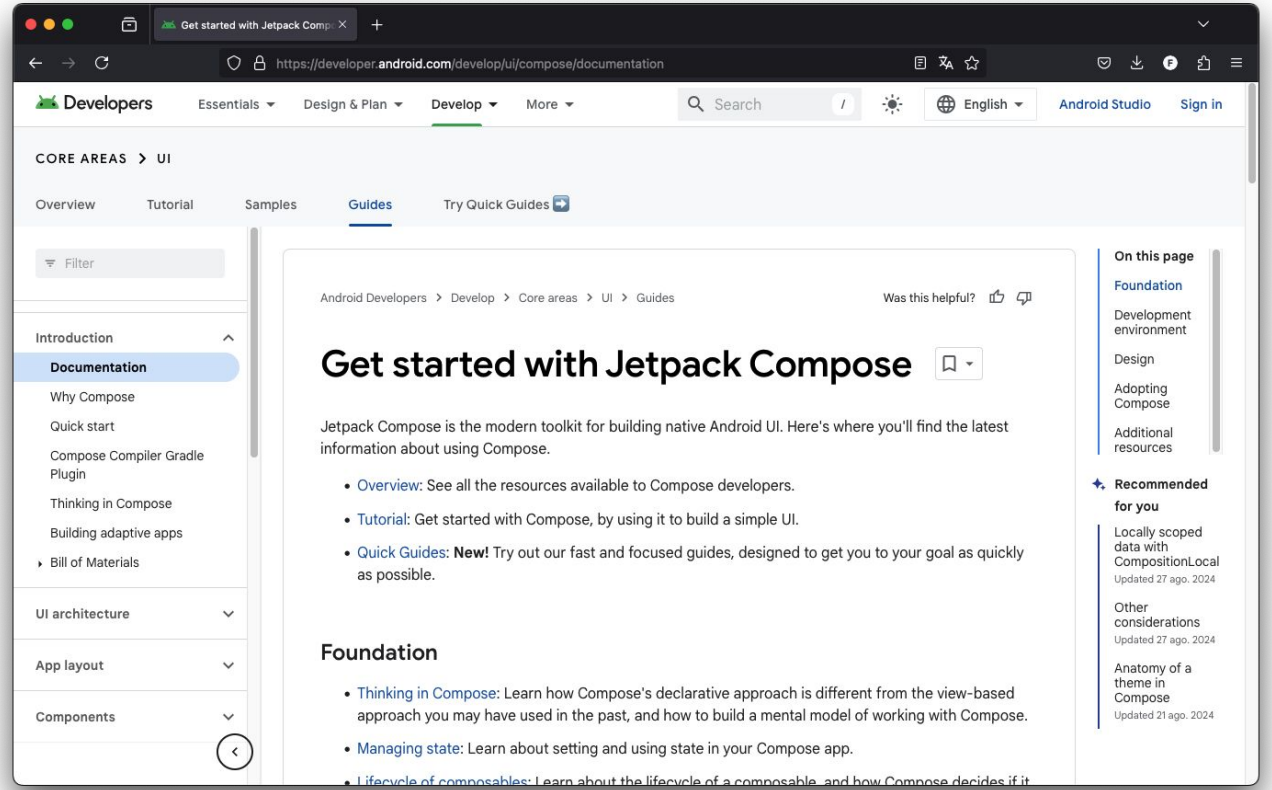
WebView.ios.Kt.kt

```kotlin
//WebView.ios.kt
@OptIn(ExperimentalForeignApi::class)
@Composable
actual fun WebView(modifier: Modifier, url: String) {

    val nativeViewFactory = LocalNativeViewFactory.current

    UIKitViewController(
        modifier = Modifier
            .fillMaxSize(),
        factory = { nativeViewFactory.createWebView(url) },
        update = {}
    )
}
```

⚙️ Compose Multiplatform: Engine



Compose Multiplatform

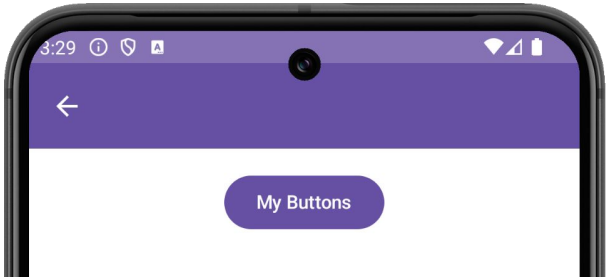


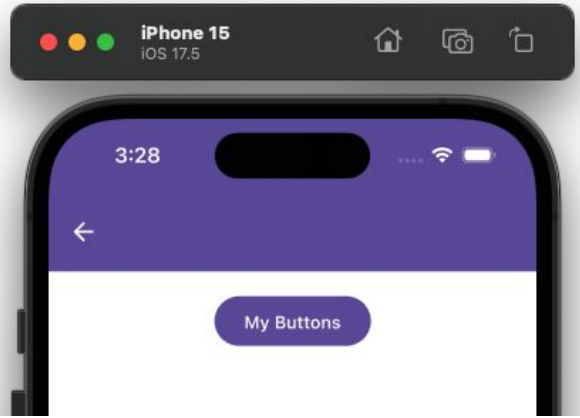
https://developer.android.com/develop/ui/compose/documentation

🛠️ Detalles del Diseño Interno: **Composables**

Example.kt

```kotlin
@Composable
fun Example(modifier: Modifier = Modifier) {
    Column {
        Button(onClick = { /*TODO*/ }) {
            Text(text = "My Button")
        }
    }
}
```

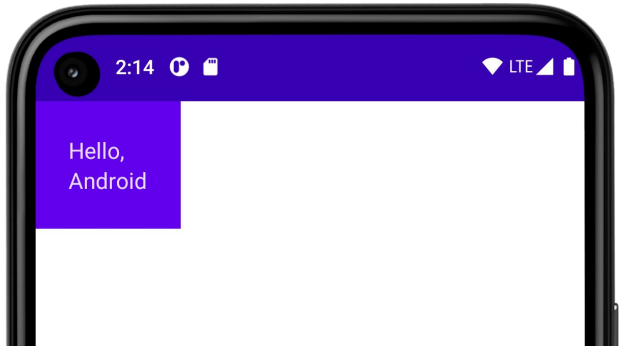


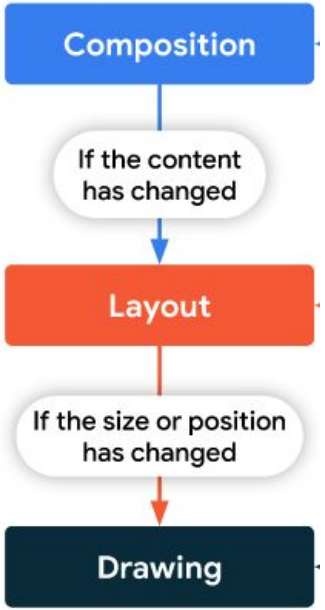


🛠️ Detalles del Diseño Interno: **Modifiers**

Example.kt

```kotlin
@Composable
private fun Greeting(name: String) {
    Column(modifier = Modifier.padding(24.dp)) {
        Text(text = "Hello,")
        Text(text = name)
    }
}
```



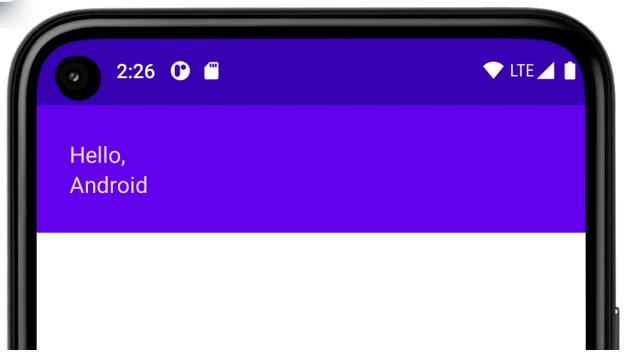


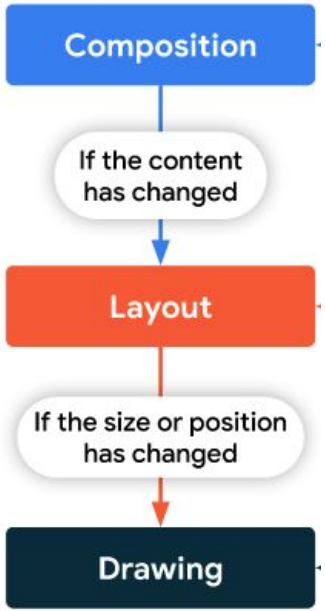
🛠️ Detalles del Diseño Interno: **Modifiers (2)**

Example.kt

```kotlin
@Composable
private fun Greeting(name: String) {
    Column(
        modifier = Modifier
            .padding(24.dp)
            .fillMaxWidth()
    ) {
        Text(text = "Hello,")
        Text(text = name)
    }
}
```

👈

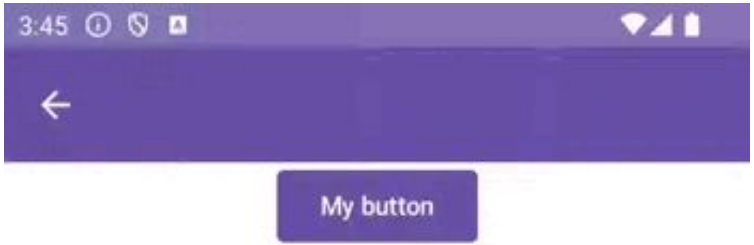


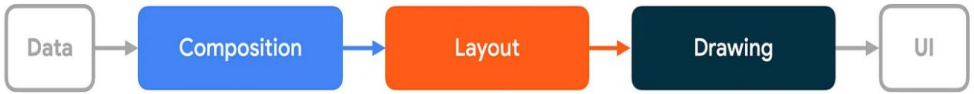


🔃 Detalles del Diseño Interno: **State Management**

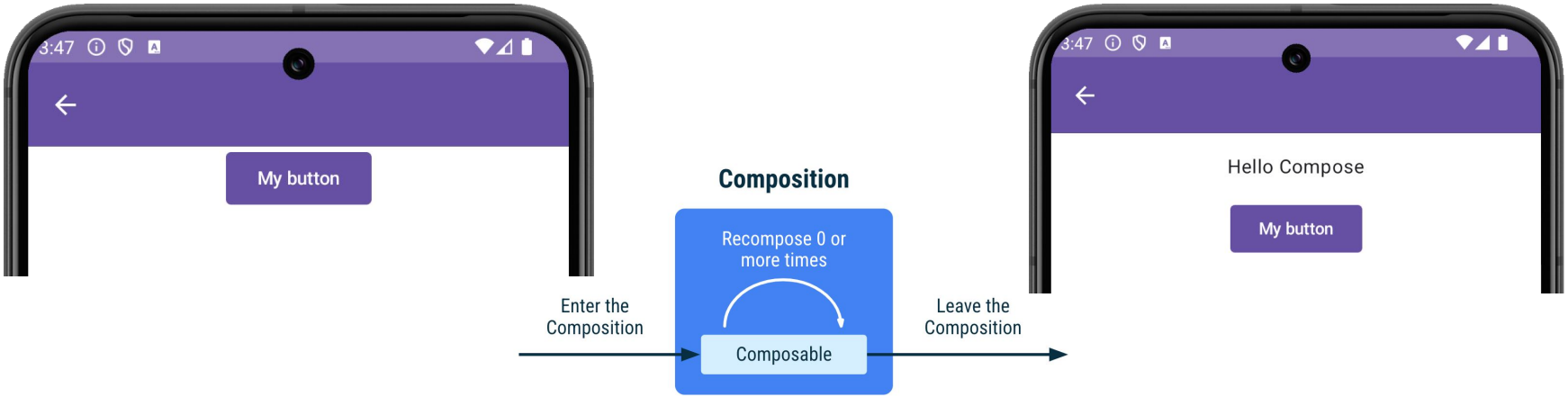
Example.kt

```kotlin
@Composable
fun Example(modifier: Modifier = Modifier) {
    var isVisible by remember {
        mutableStateOf(false)
    }
    Column {
        if (isVisible) {
            Text(text = "Hello Compose")
        }
        Button(onClick = { isVisible = !isVisible }) {
            Text(text = "My Button")
        }
    }
}
```





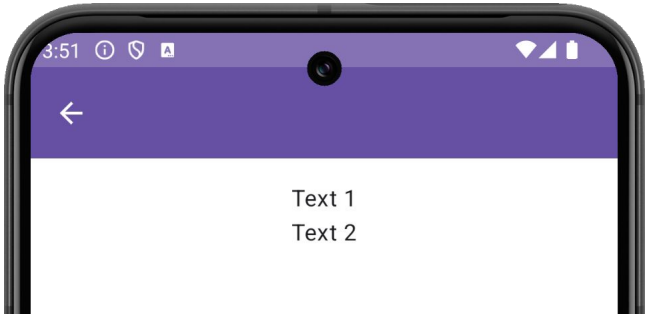
♻️ Detalles del Diseño Interno: **Recomposition**

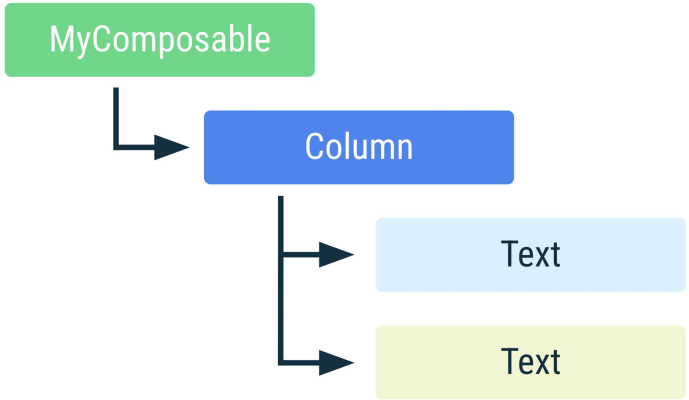


🏗️ Detalles del Diseño Interno: **Layout System**

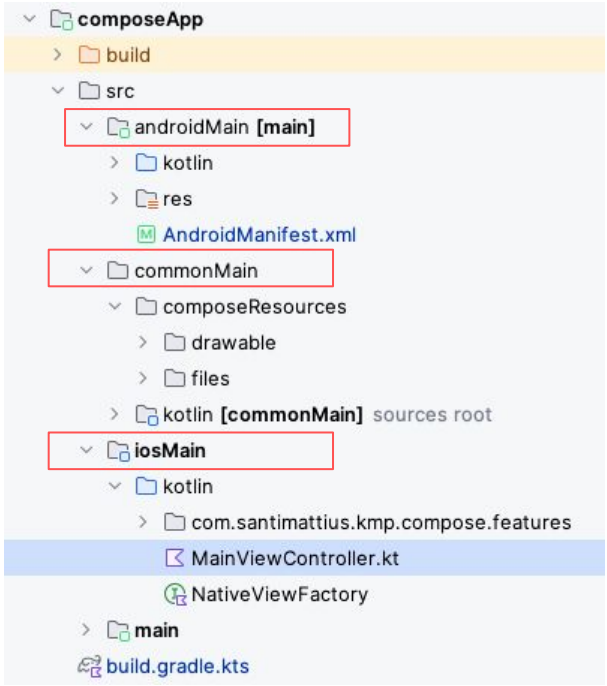
Example.kt

```kotlin
@Composable
fun Example3(modifier: Modifier = Modifier) {
    Column(
        modifier = modifier.fillMaxSize(),
        horizontalAlignment = Alignment.CenterHorizontally,
        verticalArrangement = Arrangement.Center
    ) {
        Text("Text 1")
        Text("Text 2")
    }
}
```





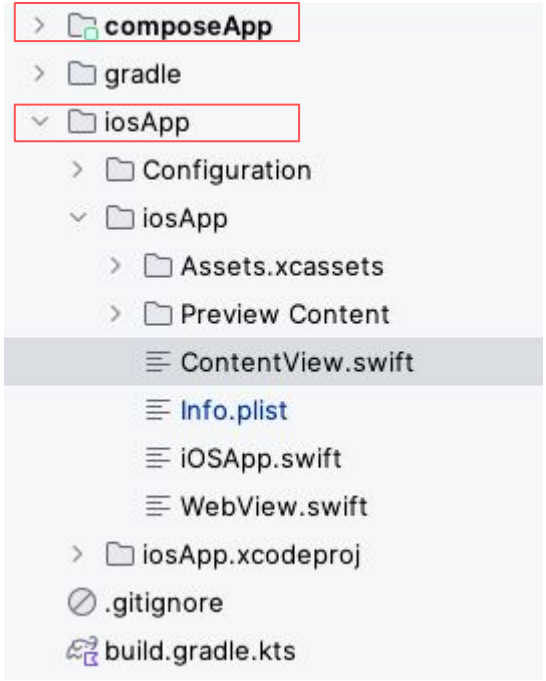
Compose en iOS



MainViewController.Kt.kt

```kotlin
//MainViewController.Kt
import androidx.compose.ui.window.ComposeUIViewController
import platform.UIKit.UIViewController

fun MainViewController(): UIViewController {
    return ComposeUIViewController { App() }
}

//Common code
@OptIn(ExperimentalCoilApi::class)
@Composable
fun App() {
    AppTheme {
        RootScreen()
    }
}
```

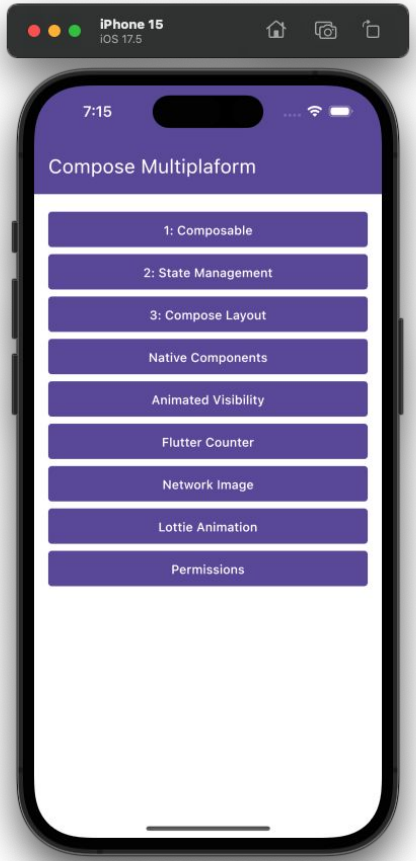
📱 Compose en iOS (2) : Swift + Kotlin

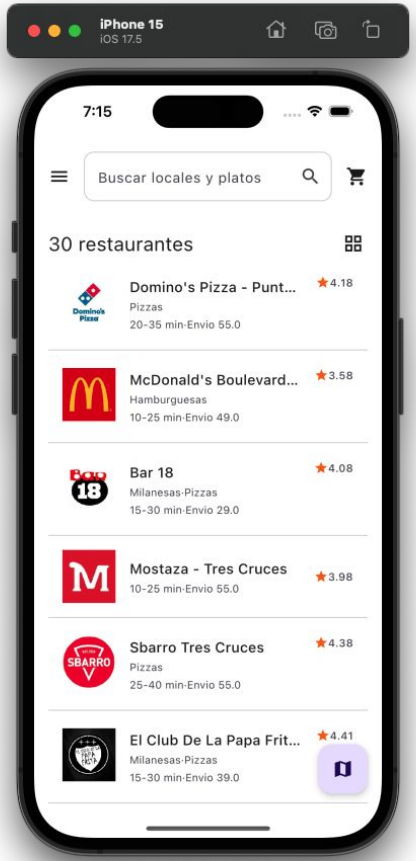


ContentView.swift

```swift
import UIKit
import SwiftUI
import ComposeApp

struct ComposeView: UIViewControllerRepresentable {
    func makeUIViewController(context: Context) -> UIViewController {
        MainViewControllerKt.MainViewController()
    }

    func updateUIViewController(_ uiViewController: UIViewController, context: Context) {}
}

struct ContentView: View {
    var body: some View {
        ComposeView()
            .edgesIgnoringSafeArea(.top) // Compose has own keyboard handler
    }
}
```

Demo


iPhone 15
iOS 17.5
7:15
Compose Multiplaform
1: Composable
2: State Management
3: Compose Layout
Native Components
Animated Visibility
Flutter Counter
Network Image
Lottie Animation
Permissions


iPhone 15
iOS 17.5
7:15
Buscar locales y platos
30 restaurantes
Domino's Pizza - Punt...
Pizzas
20-35 min·Envio 55.0
4.18
McDonald's Boulevard...
Hamburguesas
10-25 min·Envio 49.0
3.58
Bar 18
Milanesas·Pizzas
15-30 min·Envio 29.0
4.08
Mostaza - Tres Cruces
10-25 min·Envio 55.0
3.98
Sbarro Tres Cruces
Pizzas
25-40 min·Envio 55.0
4.38
El Club De La Papa Frit...
Milanesas·Pizzas
15-30 min·Envio 39.0
4.41

Ventajas de Compose Multiplatform

- Reutilización de Código ✅
- Consistencia en la Experiencia de Usuario ✅
- Mantenimiento Simplificado ✅
- Desarrollo Más Rápido ✅
- Uso de Kotlin ❤️
- Interoperabilidad con Código Nativo. ✅
- UI Declarativa y Reactiva ✅
- Escalabilidad y Flexibilidad ✅
- Reducción de Costos 🤔
-

Compose Multiplatform vs. Flutter

Aspecto	Compose Multiplatform	Flutter
Lenguaje	Kotlin	Dart
Soporte Multiplataforma	Android, iOS, Desktop (macOS, Windows, Linux)	Android, iOS, Web, Desktop (macOS, Windows, Linux)
UI Declarativa	Sí (Jetpack Compose)	Sí (Flutter Widgets)
Desempeño	Alto, cercano al nativo en Android, con optimizaciones continuas	Alto, cercano al nativo en todas las plataformas
Integración con código nativo	Excelente integración con código nativo de Android	Buena integración, requiere plugins para interactuar con código nativo
Comunidad y Ecosistema	Creciente, con fuerte respaldo de JetBrains y Google	Amplia y madura, con gran cantidad de paquetes y plugins
Curva de Aprendizaje	Más fácil si ya tienes experiencia en Kotlin o Android	Más fácil si no tienes experiencia previa en Kotlin
Despliegue Web	En desarrollo, no completamente estable	Estable y soportado

🏭 Impacto en la Industria

Tanto **empresas** como **desarrolladores** están adoptando esta tecnología para reducir costos y mejorar la eficiencia. Además, promueve una experiencia de usuario coherente en diversos dispositivos, lo cual puede aumentar la aceptación y satisfacción del usuario.

- **Startups y Pequeñas Empresas**: Las startups, con recursos limitados, pueden beneficiarse enormemente de Compose Multiplatform al reducir el tiempo y los costos asociados al desarrollo para diversas plataformas.
- **Grandes Empresas**: Las empresas con aplicaciones complejas y de amplio uso pueden aprovechar Compose Multiplatform para mantener la consistencia en la experiencia del usuario y agilizar el proceso de desarrollo.
- **Proyectos Open Source**: La comunidad de código abierto también se beneficia al colaborar en proyectos multiplataforma sin la barrera de tener que manejar múltiples códigos.

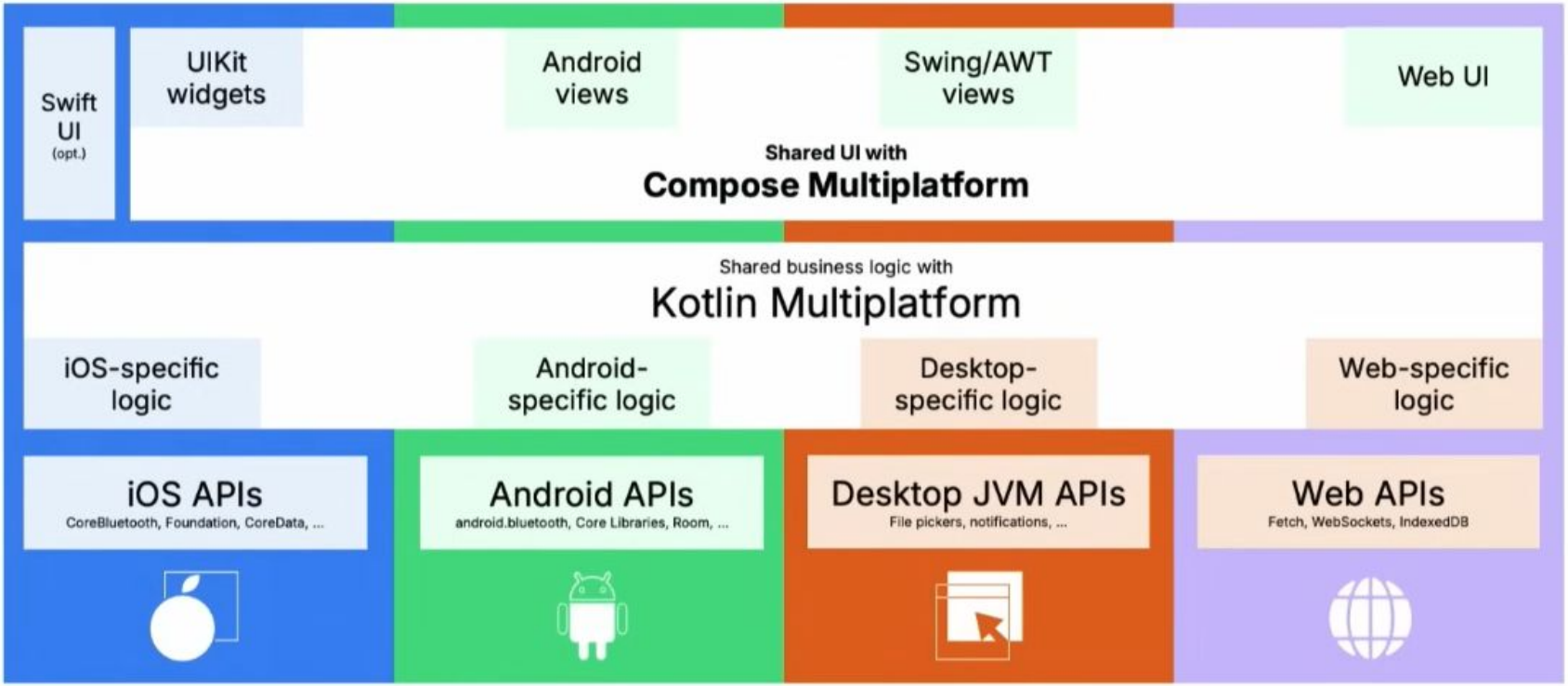
Desafíos y Consideraciones

Desafíos y Consideraciones: Más Info

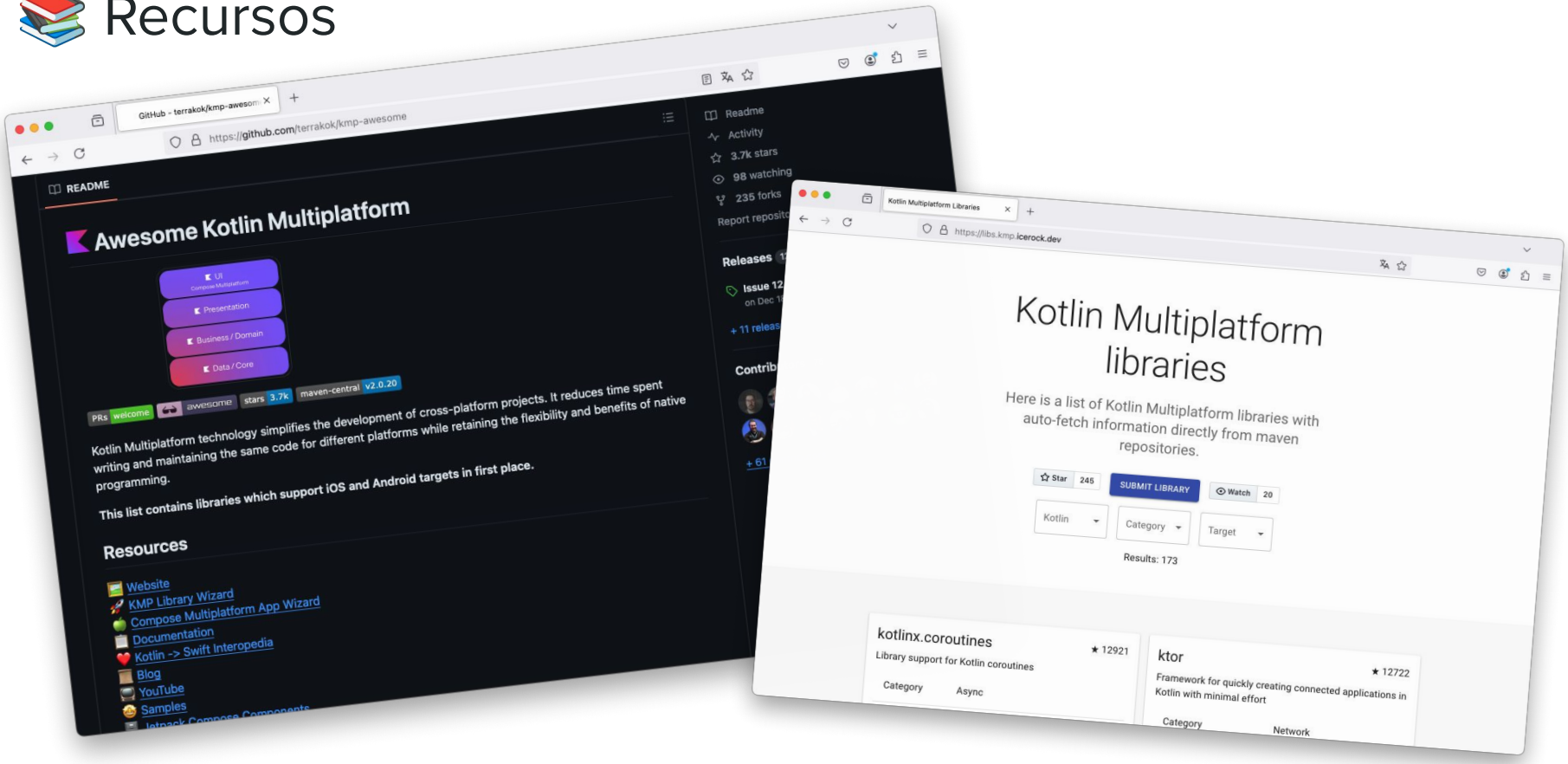
[Compose Multiplatform performance on iOS | Elijah Semyonov]

[Guide to Improving Compose Performance | Mohit Sarveiya]

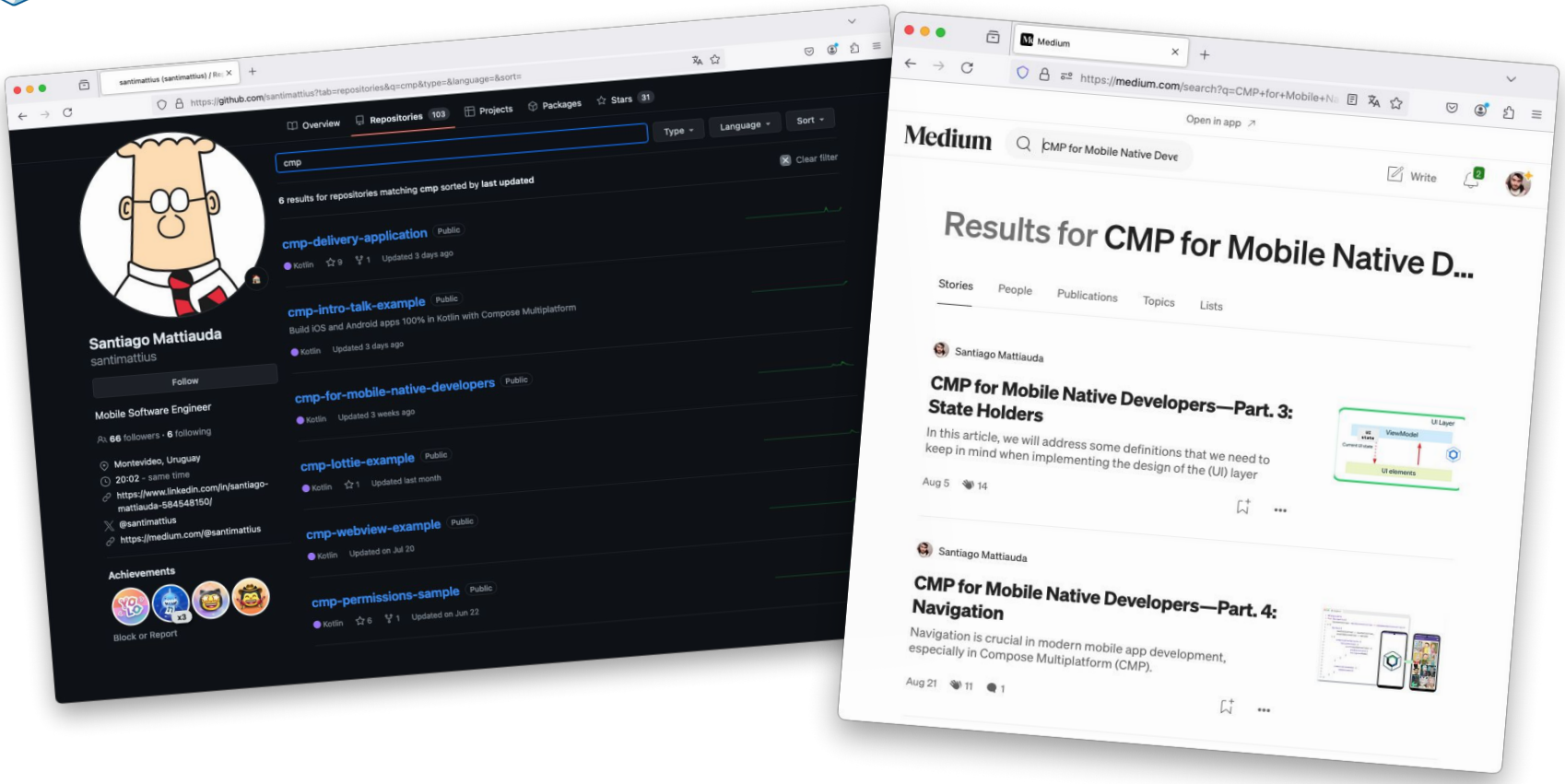
Futuro de Compose Multiplatform



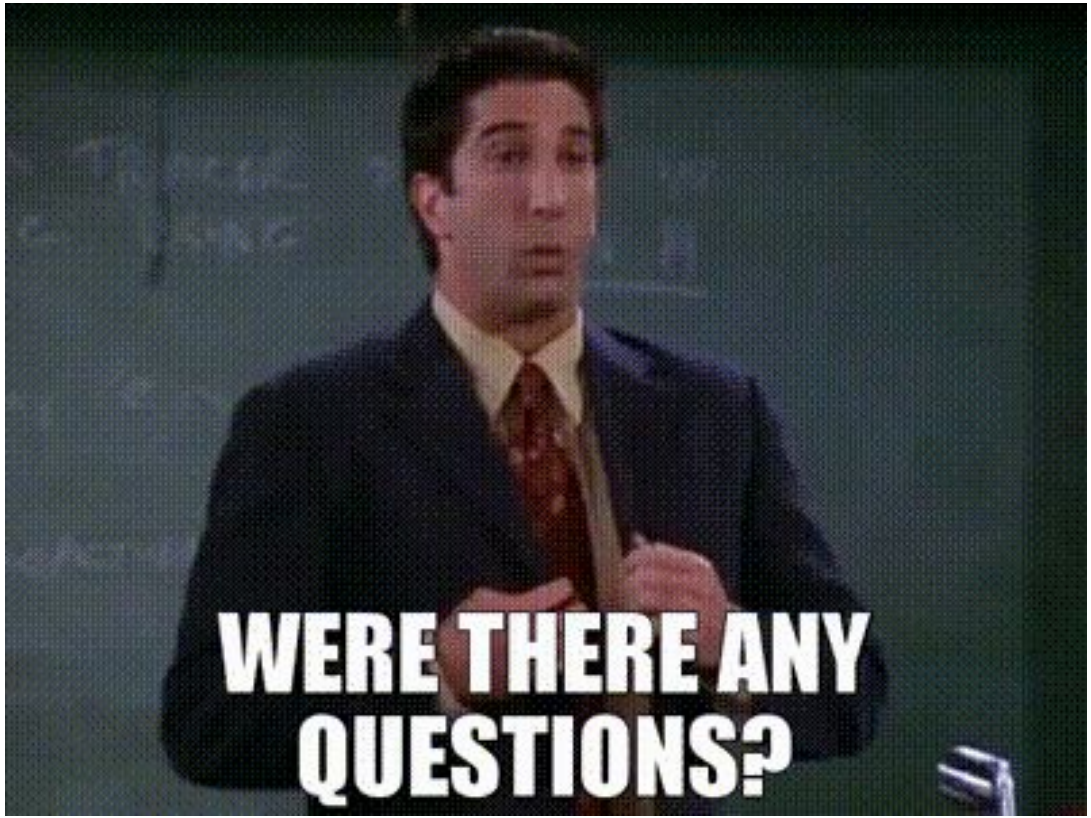
Recursos


README
Awesome Kotlin Multiplatform
UI
Presentation
Business / Domain
Data / Core
Kotlin Multiplatform technology simplifies the development of cross-platform projects. It reduces time spent writing and maintaining the same code for different platforms while retaining the flexibility and benefits of native programming.
This list contains libraries which support iOS and Android targets in first place.
Resources
Website
KMP Library Wizard
Compose Multiplatform App Wizard
Documentation
Kotlin -> Swift Interopedia
Blog
YouTube
Samples
Kotlin Multiplatform libraries
Here is a list of Kotlin Multiplatform libraries with auto-fetch information directly from maven repositories.
SUBMIT LIBRARY
Results: 173
kotlinx.coroutines
Library support for Kotlin coroutines
ktor
Framework for quickly creating connected applications in Kotlin with minimal effort

📚 Mis Recursos



Preguntas



Gracias 🙏