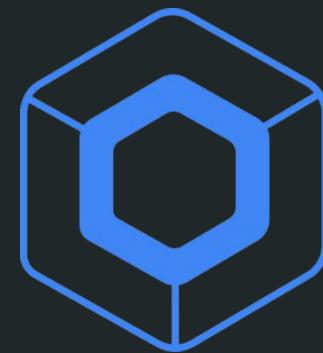
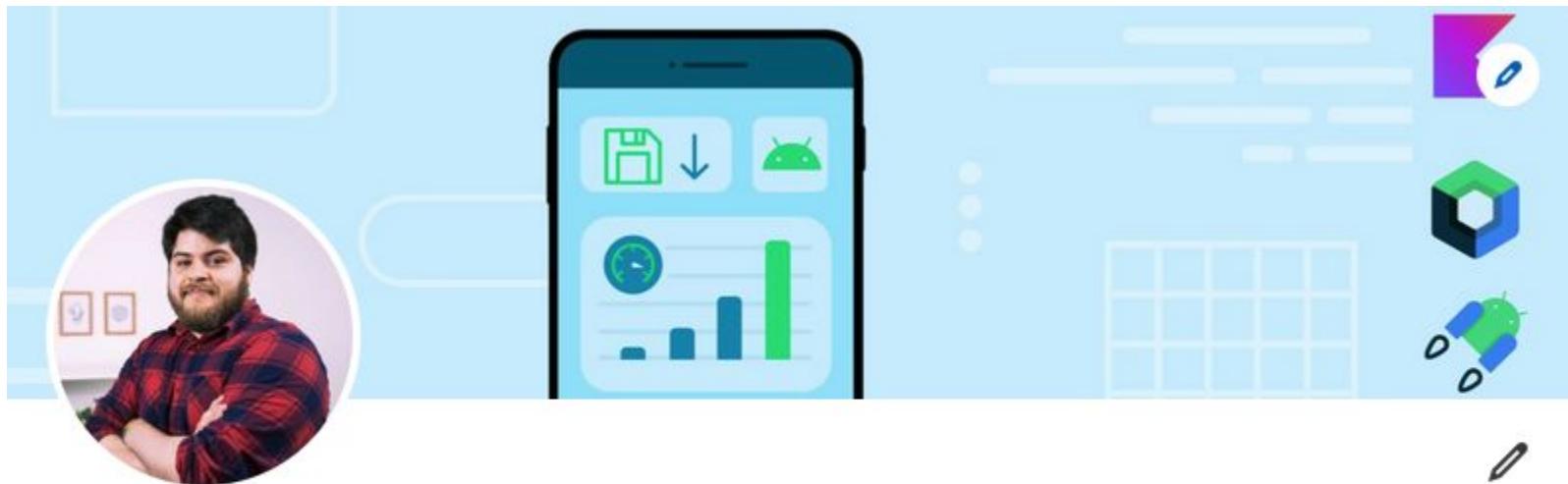


Construye aplicaciones para iOS y Android 100% en Kotlin con Compose Multiplatform

CMP for Mobile Native Developers



Presentación



Santiago Mattiauda 

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.

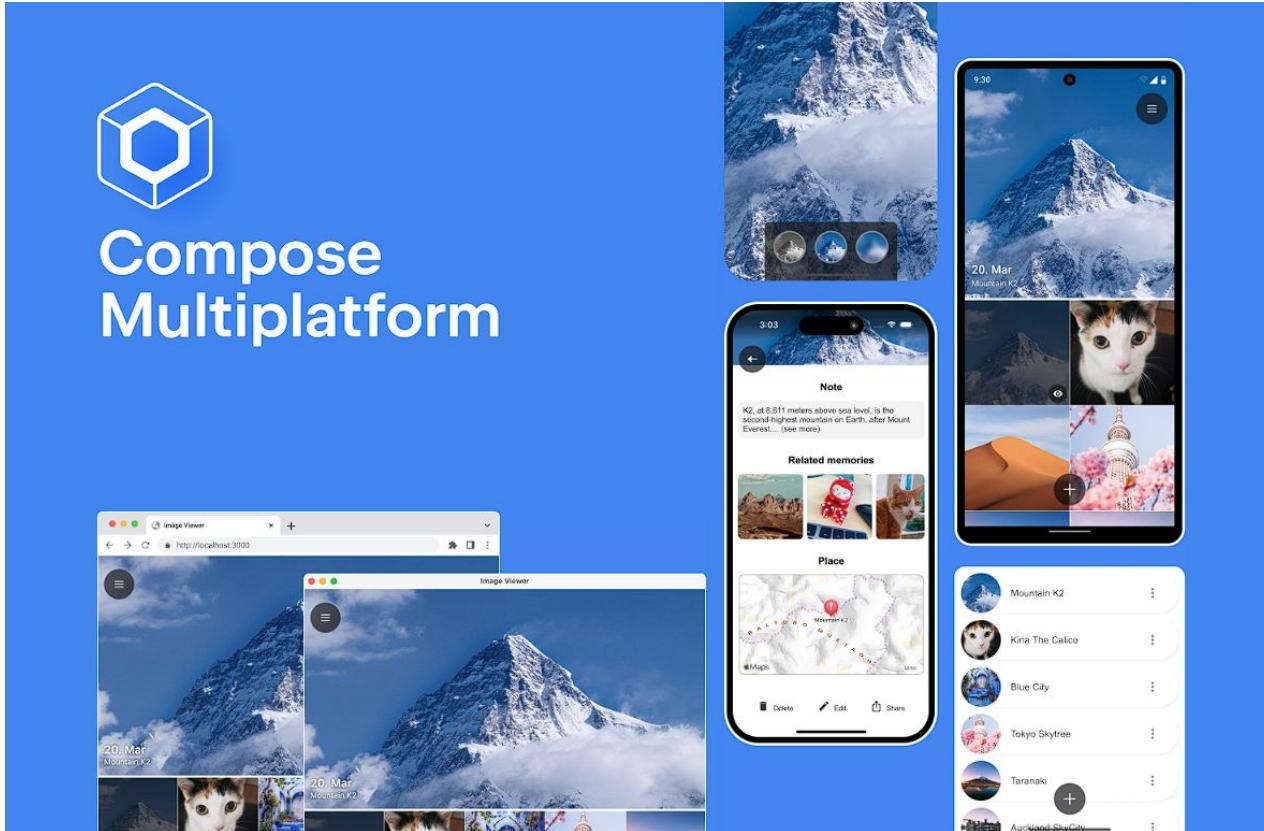


Evolución de las herramientas multiplataforma

- 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





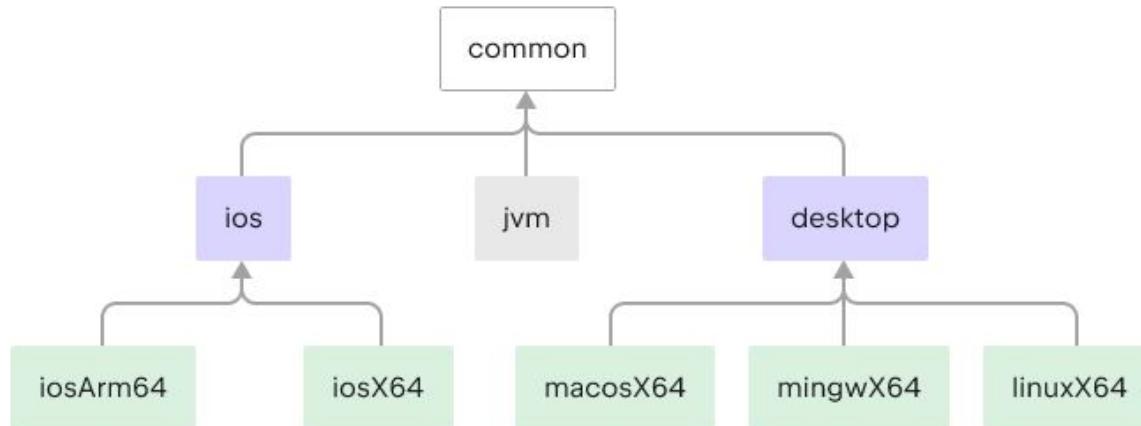
Objetivos de la charla

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

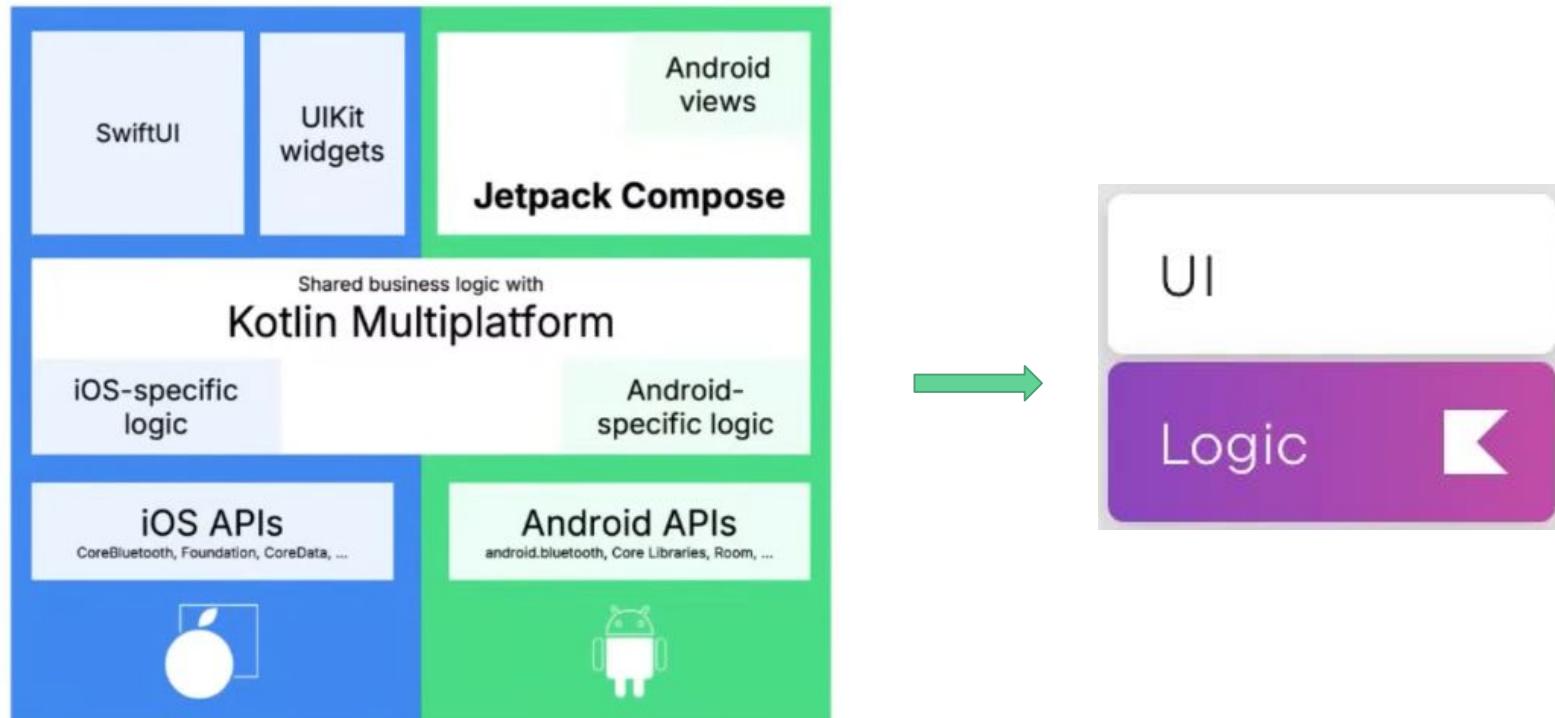


¿Qué es Kotlin Multiplatform?

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.



¿Qué es Kotlin Multiplatform? (2)

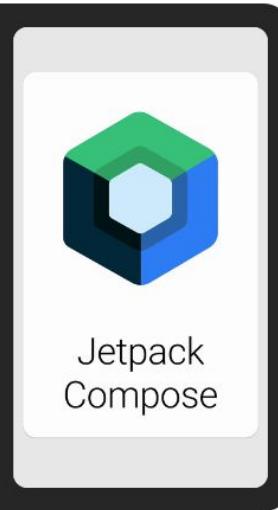




¿Qué es Jetpack Compose?

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.

```
@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.bodyLarge,
                )
            }
        }
    }
}
```





¿Qué es Jetpack Compose? (2)

XML

```
example.xml
1 <?xml version="1.0" encoding="utf-8"?>
2 <LinearLayout
3     xmlns:android="http://schemas.android.com/apk/res/android"
4     android:orientation="vertical"
5     android:layout_width="match_parent"
6     android:layout_height="match_parent">
7
8     <Button
9         android:id="@+id/button"
10        android:layout_width="match_parent"
11        android:layout_height="wrap_content"
12        android:text="My Button" />
13 </LinearLayout>
```

Jetpack Compose

```
Example.kt
1 @Composable
2 fun Example(modifier: Modifier = Modifier) {
3     Column {
4         Button(onClick = { /*TODO*/ }) {
5             Text(text = "My Button")
6         }
7     }
8 }
```



¿Qué es Compose Multiplatform?



Kotlin Multiplatform



Jetpack Compose



Compose Multiplatform

¿Qué es Compose Multiplatform? (2)

The screenshot shows the GitHub repository page for `JetBrains/compose-multiplatform-core`. The repository is public and has been forked from `androidx/androidx`. It features a dark theme with a red border around the forked from information. The main navigation bar includes links for Product, Solutions, Resources, Open Source, Enterprise, Pricing, Sign in, and Sign up. Below the header, there's a banner stating "This branch is 3938 commits ahead of, 12709 commits behind androidx/androidx:androidx-main". The repository stats show 640 branches and 648 tags. The code tab is selected, showing a list of recent commits by Schahen, such as "Disable iOS PR checks with GitHub Actions in favor of ch..." and "CodeStyle. Use single name import (#646)". To the right, there's an "About" section describing the repository as a development environment for Android Jetpack extension libraries under the androidx namespace, synchronized with the AOSP. It also lists links to Google's platform repository, Readme, Apache-2.0 license, Activity, Custom properties, stars (420), watching (19), forks (70), and a report repository link. The releases section indicates 648 tags.

GitHub - JetBrains/compose-multiplatform-core

Product Solutions Resources Open Source Enterprise Pricing

Notifications Fork 70 Star 420

Code Pull requests 32 Actions Projects Security Insights

jb-main 640 Branches 648 Tags Go to file

This branch is 3938 commits ahead of, 12709 commits behind androidx/androidx:androidx-main.

Schahen Don't access directly event dispatching in OnCanvasTests... abd36ce · 33 minutes ago 182,883 Commits

.github Disable iOS PR checks with GitHub Actions in favor of ch... last month

.idea CodeStyle. Use single name import (#646) last year

.run Fix running tests via IDEA configuration desktop/test.run.... 2 months ago

activity Fix wrongly committed files because of the wrong script (...) 7 months ago

annotation Snap annotation-1.8.0-alpha02 5 months ago

appactions Fix wrongly committed files because of the wrong script (...) 7 months ago

appcompat Fix wrongly committed files because of the wrong script (...) 7 months ago

appintegration define app integration library last year

appsearch Fix wrongly committed files because of the wrong script (...) 7 months ago

About

Development environment for Android Jetpack extension libraries under the androidx namespace. Synchronized with Android Jetpack's primary development branch on AOSP.

[android.googlesource.com/platform/fr...](#)

Readme

Apache-2.0 license

Activity

Custom properties

420 stars

19 watching

70 forks

Report repository

Releases

648 tags



¿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.

Supported platforms


[iOS↗](#)
Beta

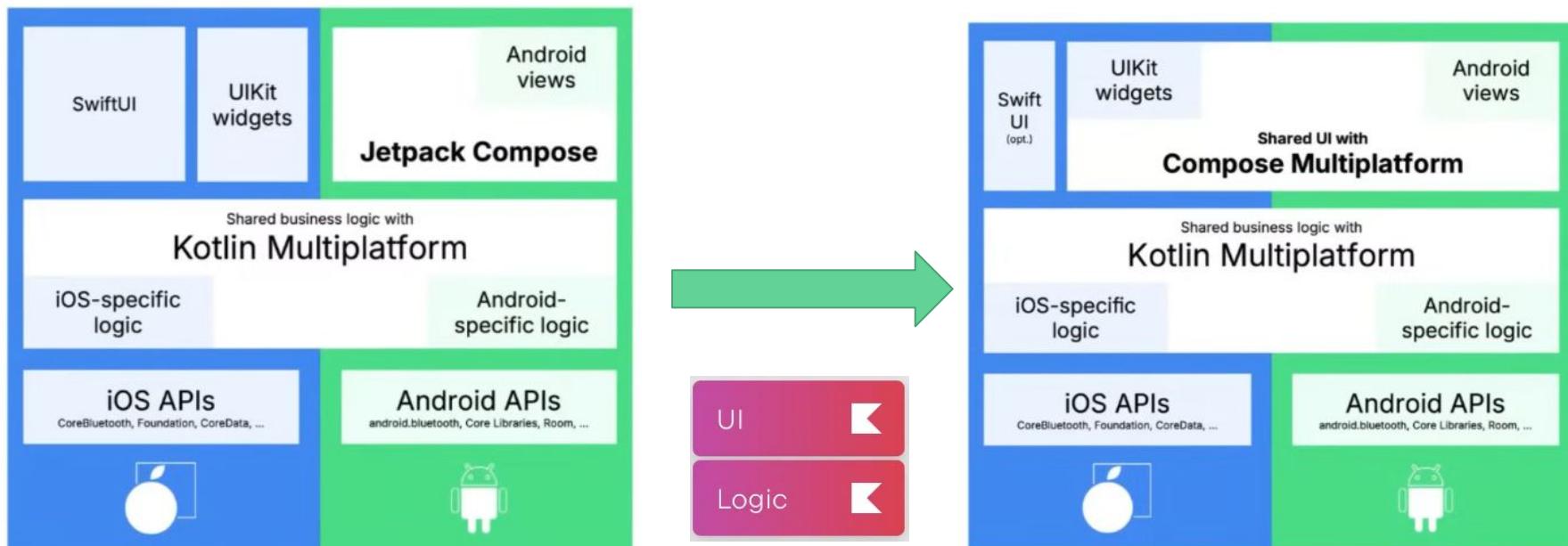

[Android↗](#)
via Jetpack Compose


[Desktop↗](#)
Windows, MacOS, Linux


[Web↗](#)
Alpha



¿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

```
1 //WebView.common.kt
2 @Composable
3 expect fun WebView(modifier: Modifier = Modifier, url: String)
4
```



WebView.android.Kt.kt

```
1 //WebView.android.kt
2 @Composable
3 actual fun WebView(modifier: Modifier, url: String) {
4
5     AndroidView(factory = {
6         AndroidWebView(it).apply {
7             layoutParams = ViewGroup.LayoutParams(
8                 ViewGroup.LayoutParams.MATCH_PARENT,
9                 ViewGroup.LayoutParams.MATCH_PARENT
10            )
11            loadUrl(url)
12            webViewClient = CustomWebViewClient()
13        }
14    }, update = {})
15 }
```



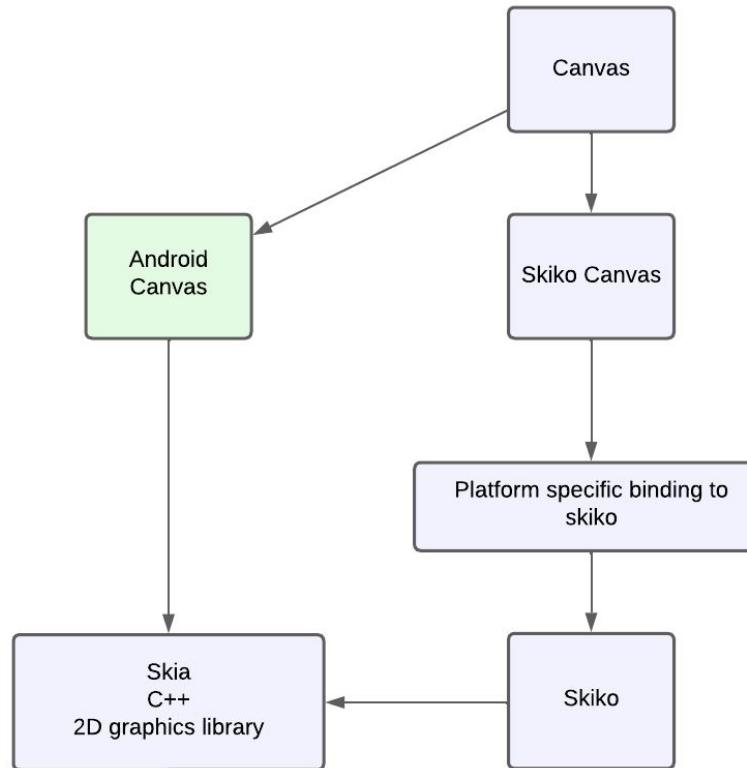
WebView.ios.Kt.kt

```
1 //WebView.ios.kt
2 @OptIn(ExperimentalForeignApi::class)
3 @Composable
4 actual fun WebView(modifier: Modifier, url: String) {
5
6     val nativeViewFactory = LocalNativeViewFactory.current
7
8     UIKitViewController(
9         modifier = Modifier
10            .fillMaxSize(),
11         factory = { nativeViewFactory.createWebView(url) },
12         update = {}
13     )
14 }
```





Compose Multiplatform: Engine



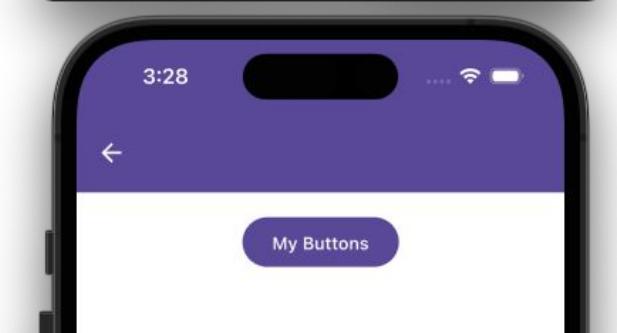
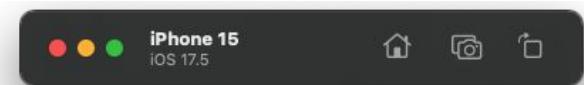
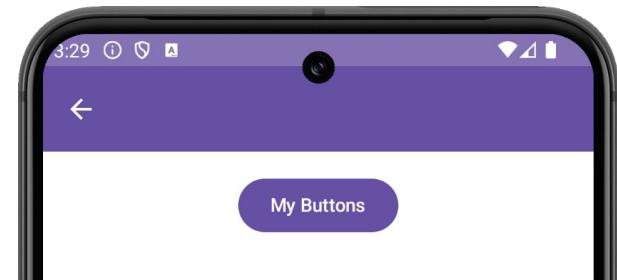
Compose Multiplatform

The screenshot shows a web browser displaying the 'Get started with Jetpack Compose' documentation on the Android Developers website. The URL is <https://developer.android.com/develop/ui/compose/documentation>. The page is part of the 'Develop' section under 'Core areas > UI'. The 'Guides' tab is selected. The main content area features a large heading 'Get started with Jetpack Compose' and a brief introduction about the toolkit. Below this, there's a bulleted list of resources: Overview, Tutorial, and Quick Guides. The 'Quick Guides' section is highlighted with a blue box and the text 'New! Try out our fast and focused guides, designed to get you to your goal as quickly as possible.' To the right, a sidebar titled 'On this page' lists categories like Foundation, Development environment, Design, Adopting Compose, and Additional resources. Another sidebar titled 'Recommended for you' lists 'Locally scoped data with CompositionLocal', 'Other considerations', and 'Anatomy of a theme in Compose'. The left sidebar contains a navigation tree for 'Documentation' (Why Compose, Quick start, etc.), 'UI architecture', 'App layout', and 'Components'.



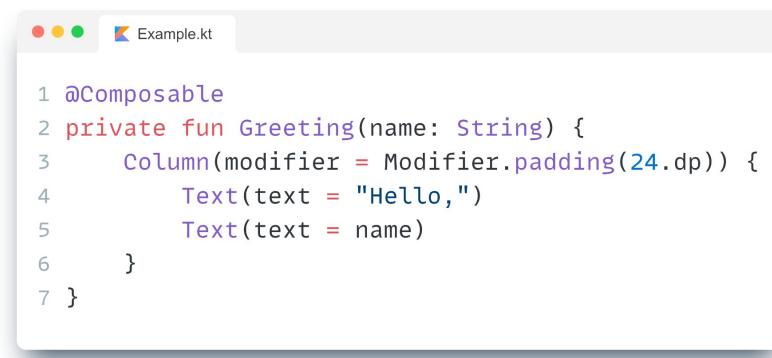
🛠️ Detalles del Diseño Interno: Composables

```
1 @Composable
2 fun Example(modifier: Modifier = Modifier) {
3     Column {
4         Button(onClick = { /*TODO*/ }) {
5             Text(text = "My Button")
6         }
7     }
8 }
```

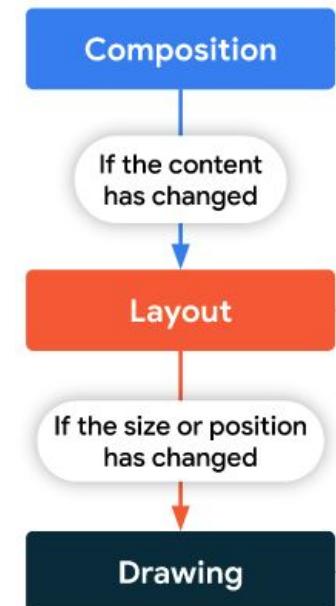
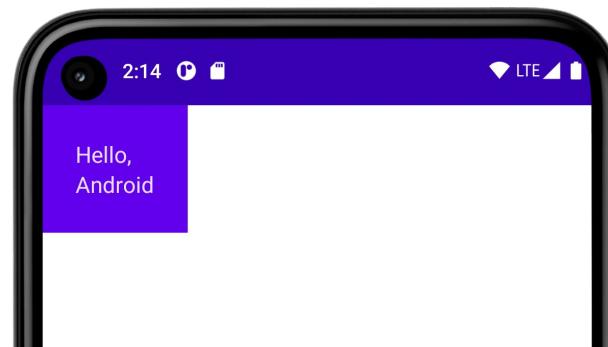




🛠️ Detalles del Diseño Interno: **Modifiers**



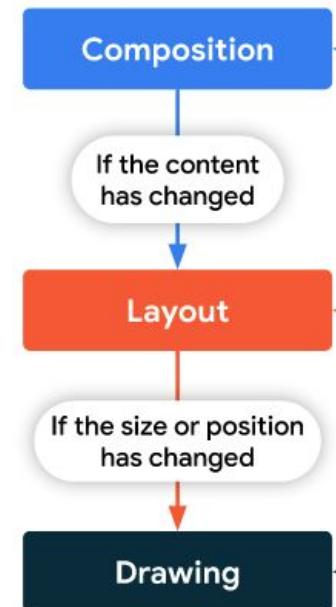
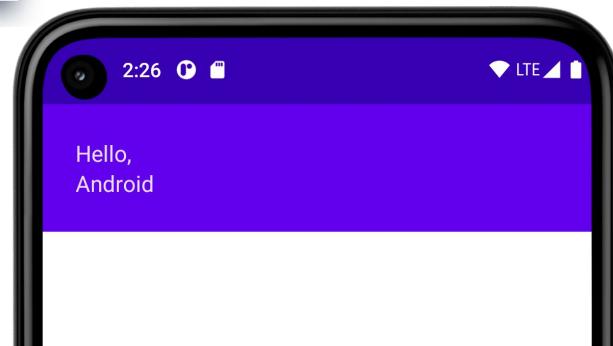
```
1 @Composable
2 private fun Greeting(name: String) {
3     Column(modifier = Modifier.padding(24.dp)) {
4         Text(text = "Hello,")
5         Text(text = name)
6     }
7 }
```





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

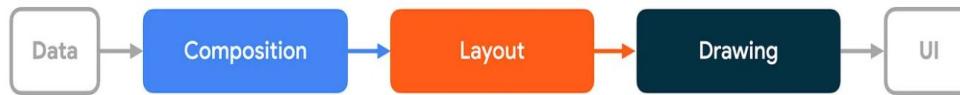
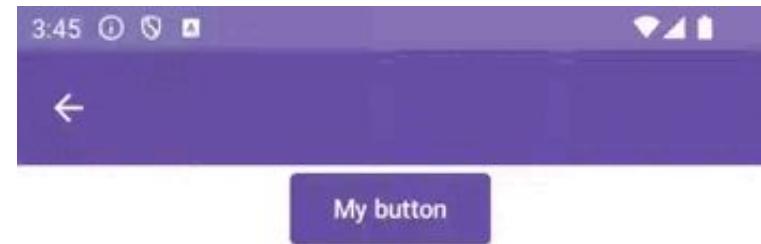
```
1 @Composable
2 private fun Greeting(name: String) {
3     Column(
4         modifier = Modifier
5             .padding(24.dp)
6             .fillMaxWidth())
7     {
8         Text(text = "Hello,")
9         Text(text = name)
10    }
11 }
```





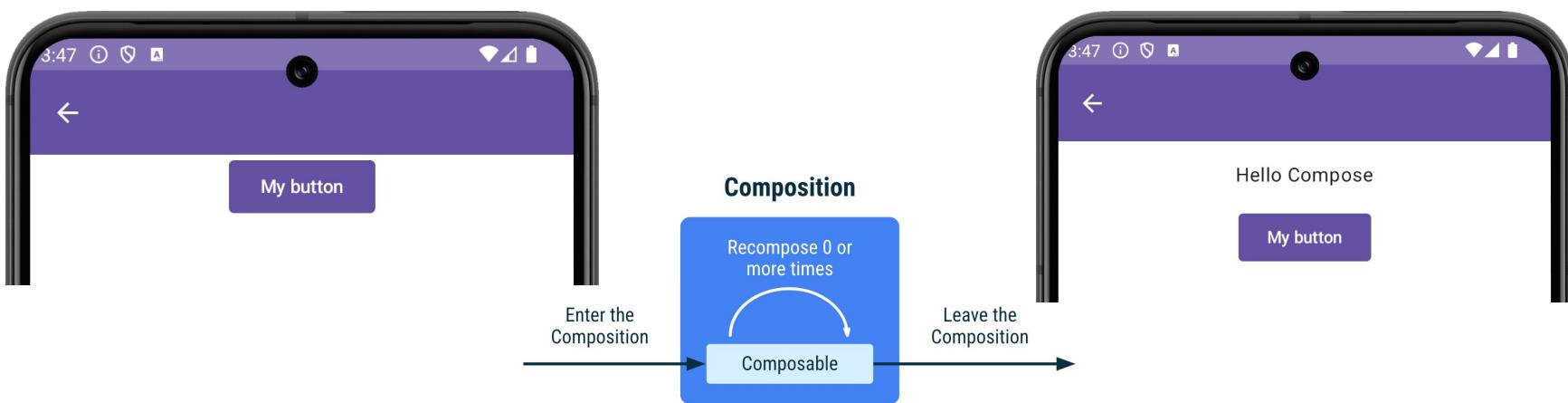
Detalles del Diseño Interno: State Management

```
1 @Composable
2 fun Example(modifier: Modifier = Modifier) {
3     var isVisible by remember {
4         mutableStateOf(false)
5     }
6     Column {
7         if (isVisible) {
8             Text(text = "Hello Compose")
9         }
10        Button(onClick = { isVisible = !isVisible }) {
11            Text(text = "My Button")
12        }
13    }
14 }
```



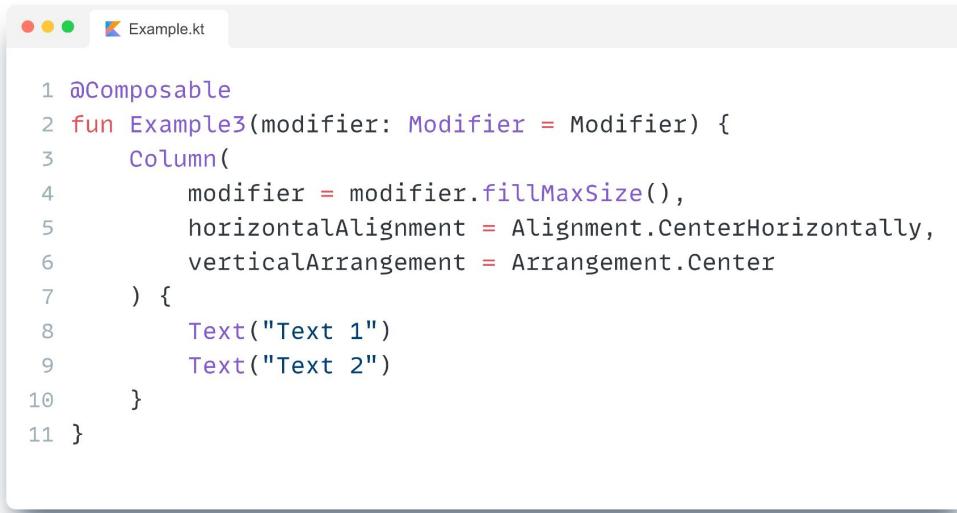


Detalles del Diseño Interno: **Recomposition**

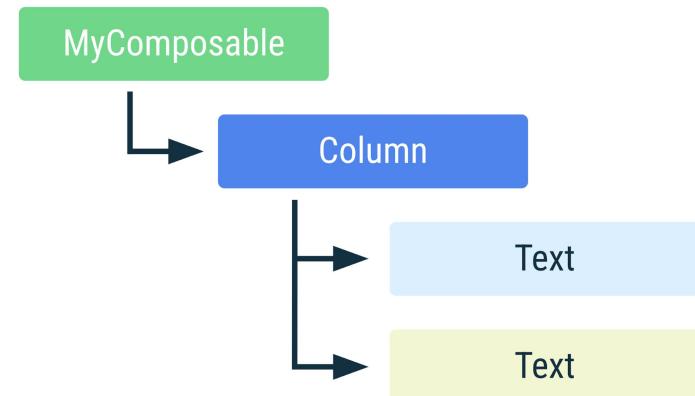
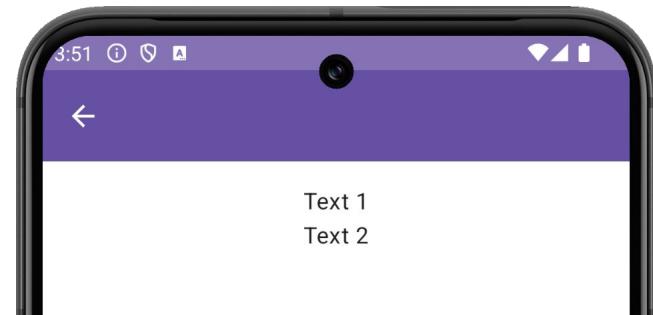




Detalles del Diseño Interno: Layout System

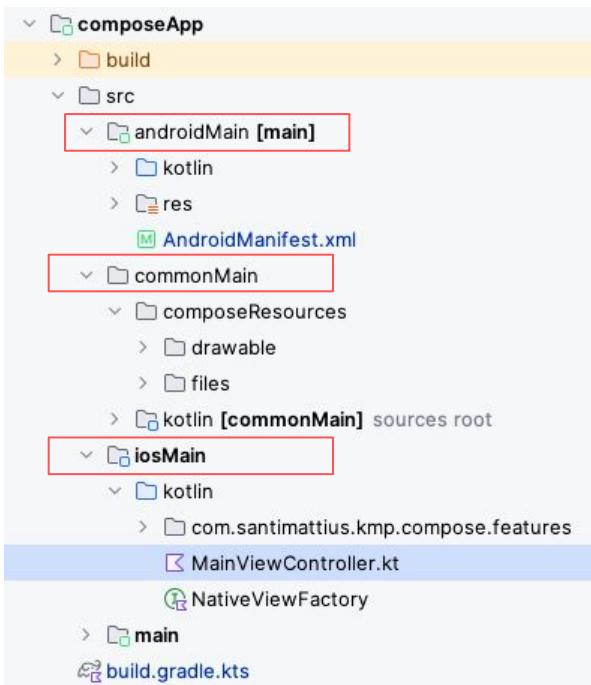


```
1 @Composable
2 fun Example3(modifier: Modifier = Modifier) {
3     Column(
4         modifier = modifier.fillMaxSize(),
5         horizontalAlignment = Alignment.CenterHorizontally,
6         verticalArrangement = Arrangement.Center
7     ) {
8         Text("Text 1")
9         Text("Text 2")
10    }
11 }
```





Compose en iOS



```
1 //MainViewController.Kt
2 import androidx.compose.ui.window.ComposeUIViewController
3 import platformUIKit.UIViewController
4
5 fun MainViewController(): UIViewController {
6     return ComposeUIViewController { App() }
7 }
8
9 // Common code
10 @OptIn(ExperimentalCoilApi::class)
11 @Composable
12 fun App() {
13     AppTheme {
14         RootScreen()
15     }
16 }
```



Compose en iOS (2) : Swift + Kotlin

The screenshot shows the Xcode Project Navigator with the following structure:

- > **composeApp** (highlighted with a red border)
- > **gradle**
- < **iosApp** (highlighted with a red border)
 - > Configuration
 - < **iosApp** (highlighted with a red border)
 - > Assets.xcassets
 - > Preview Content
 - ☰ ContentView.swift (highlighted with a grey background)
 - ☰ Info.plist
 - ☰ iOSApp.swift
 - ☰ WebView.swift- > **iosApp.xcodeproj**
- ∅ .gitignore
- ∅ build.gradle.kts

The screenshot shows the Xcode Editor displaying the `ContentView.swift` file. The code is as follows:

```
1 import UIKit
2 import SwiftUI
3 import ComposeApp
4
5 struct ComposeView: UIViewControllerRepresentable {
6     func makeUIViewController(context: Context) -> UIViewController {
7         MainViewControllerKt.MainViewController()
8     }
9
10    func updateUIViewController(_ uiViewController: UIViewController, context: Context) {}
11 }
12
13 struct ContentView: View {
14     var body: some View {
15         ComposeView()
16             .edgesIgnoringSafeArea(.top) // Compose has own keyboard handler
17     }
18 }
```

Two yellow hand icons point to the `import ComposeApp` line and the `MainViewControllerKt.MainViewController()` call.



Demo

The image displays two iPhone 15 devices side-by-side, both running iOS 17.5. The left phone shows a list of Compose Multiplatform features, while the right phone displays a food delivery application interface.

iPhone 15 Left Screen (Compose Multiplatform):

- 1: Composable
- 2: State Management
- 3: Compose Layout
- Native Components
- Animated Visibility
- Flutter Counter
- Network Image
- Lottie Animation
- Permissions

iPhone 15 Right Screen (Food Delivery App):

Buscar locales y platos

30 restaurantes

Restaurante	Categoría	Calificación
Domino's Pizza - Punt...	Pizzas	4.18
McDonald's Boulevard...	Hamburguesas	3.58
Bar 18	Milanesas-Pizzas	4.08
Mostaza - Tres Cruces	10-25 min-Envio 55.0	3.98
Sbarro Tres Cruces	Pizzas	4.38
El Club De La Papa Frit...	Milanesas-Pizzas	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.

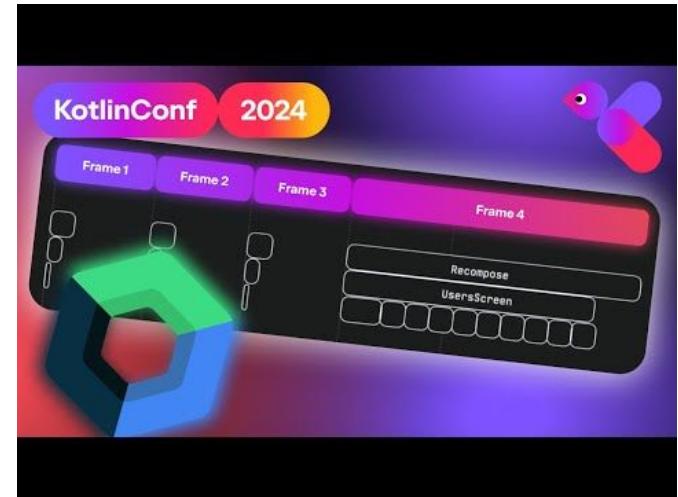


Desafíos y Consideraciones





Desafíos y Consideraciones: Más Info

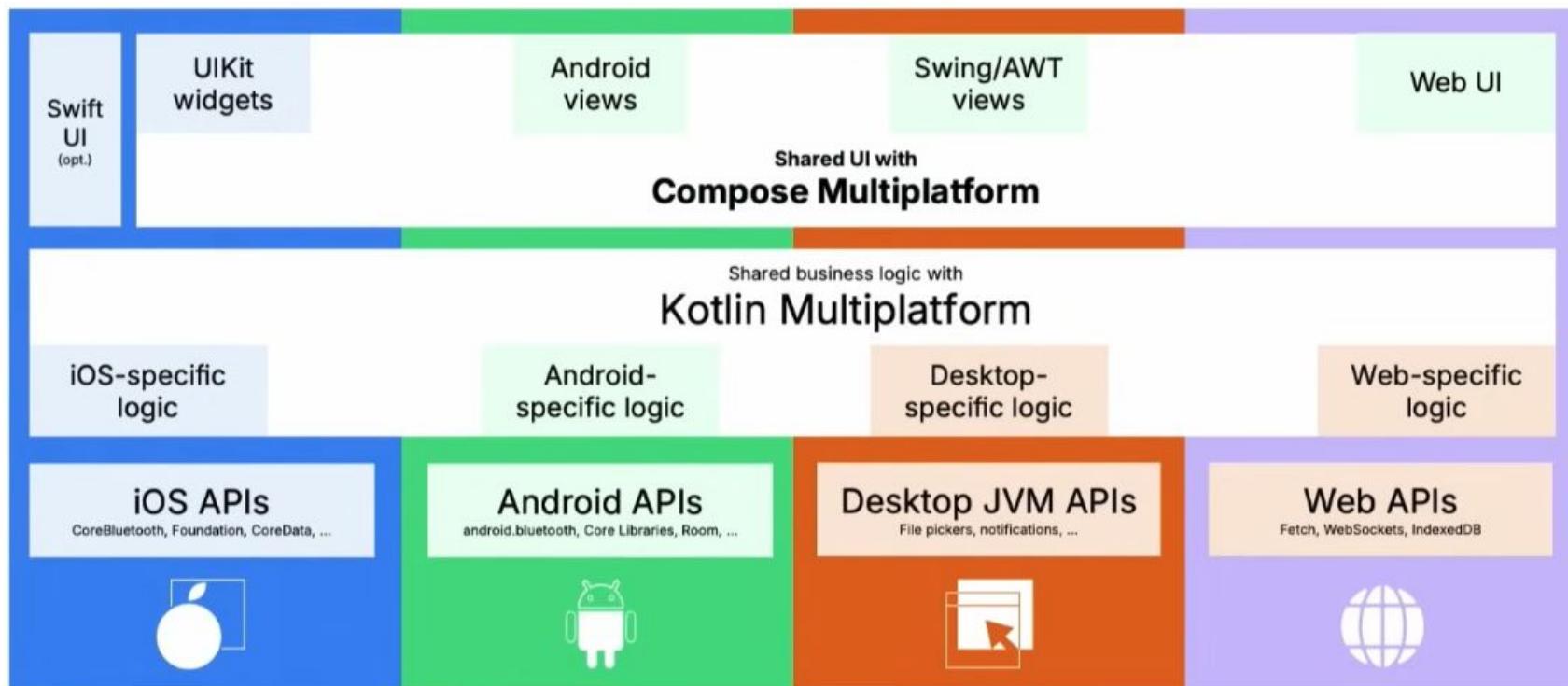


[Compose Multiplatform performance on iOS | Elijah Semyonov](#)

[Guide to Improving Compose Performance | Mohit Sarveiya](#)



Futuro de Compose Multiplatform





Recursos

The image displays two screenshots of web pages related to Kotlin Multiplatform:

Screenshot 1: GitHub - terrakok/kmp-awesome

This screenshot shows the README page of a GitHub repository named "kmp-awesome". The page features a purple header with the text "Awesome Kotlin Multiplatform". Below the header is a purple button labeled "Awesome". A sidebar on the left contains sections for "UI", "Presentation", "Business / Domain", and "Data / Core". The main content area includes a green "PTS welcome" button, a purple "awesome" button, and a "stars 3.7k" badge. It also mentions "maven-central v2.0.20". A descriptive paragraph explains that Kotlin Multiplatform technology simplifies cross-platform development by writing and maintaining the same code for different platforms while retaining native programming benefits. A note at the bottom states: "This list contains libraries which support iOS and Android targets in first place." A "Resources" section at the bottom lists links to the website, KMP Library Wizard, Compose Multiplatform App Wizard, Documentation, Kotlin -> Swift Interopedia, Blog, YouTube, Samples, and Jetpack Compose Components.

Screenshot 2: Kotlin Multiplatform Libraries

This screenshot shows a list of Kotlin Multiplatform libraries. The title is "Kotlin Multiplatform libraries". Below the title, a paragraph states: "Here is a list of Kotlin Multiplatform libraries with auto-fetch information directly from maven repositories." There are filters for "Star" (245), "SUBMIT LIBRARY", "Watch" (20), "Category" (Kotlin), "Target", and a search bar showing "Results: 173". Two library entries are visible:

- kotlinx.coroutines**: Library support for Kotlin coroutines. Category: Async. Star count: 12921.
- ktor**: Framework for quickly creating connected applications in Kotlin with minimal effort. Category: Network. Star count: 12722.



Mis Recursos

https://github.com/santimattius?tab=repositories&q=cmp&type=&language=&sort=

Santiago Mattiauda
santimattius

Follow

Mobile Software Engineer
66 followers · 6 following

Montevideo, Uruguay
20:02 - same time
<https://www.linkedin.com/in/santiago-mattiauda-584548150/>
<https://medium.com/@santimattius>

Achievements

Block or Report

Repositories 103

cmp

6 results for repositories matching cmp sorted by last updated

- cmp-delivery-application** Public
Kotlin Updated 3 days ago
- cmp-intro-talk-example** Public
Build iOS and Android apps 100% in Kotlin with Compose Multiplatform
Kotlin Updated 3 days ago
- cmp-for-mobile-native-developers** Public
Kotlin Updated 3 weeks ago
- cmp-lottie-example** Public
Kotlin Updated last month
- cmp-webview-example** Public
Kotlin Updated on Jul 20
- cmp-permissions-sample** Public
Kotlin Updated on Jun 22

Medium

https://medium.com/search?q=_CMP+for+Mobile+Native+Deve

Results for CMP for Mobile Native D...

Stories People Publications Topics Lists

Santiago Mattiauda

CMP for Mobile Native Developers—Part. 3: State Holders

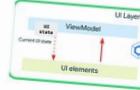
In this article, we will address some definitions that we need to keep in mind when implementing the design of the (UI) layer

Aug 5 14

CMP for Mobile Native Developers—Part. 4: Navigation

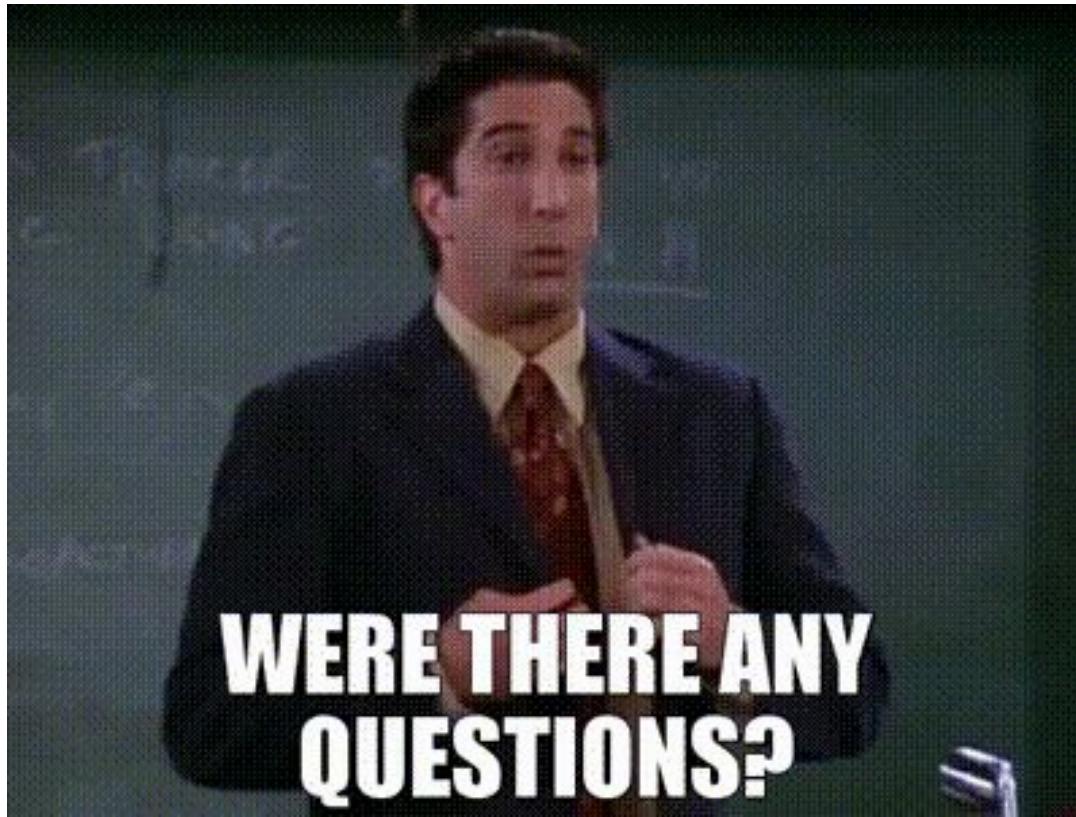
Navigation is crucial in modern mobile app development, especially in Compose Multiplatform (CMP).

Aug 21 11





Preguntas



Gracias

