Tabla de contenido

[Introducción 2](#_Toc199245143)

[Contexto Histórico 2](#_Toc199245144)

[¿Qué es Jetpack Compose? 2](#_Toc199245145)

[Principios Fundamentales 2](#_Toc199245146)

[Arquitectura y Flujo de Trabajo 3](#_Toc199245147)

[Características Clave 3](#_Toc199245148)

[Ventajas 3](#_Toc199245149)

[Instalación 4](#_Toc199245150)

[Guía de implementación en el proyecto 4](#_Toc199245151)

[Curiosidades o cosas a resaltar sobre la implementación 4](#_Toc199245152)

[Análisis de rendimiento 4](#_Toc199245153)

# Introducción

## Contexto Histórico

Antes de Jetpack Compose, el desarrollo de interfaces de usuario (UI) en Android se basaba en un enfoque **imperativo** utilizando XML para definir layouts y código Java/Kotlin para manipular vistas (View y ViewGroup). Este enfoque presentaba desafíos como:

1. **Boilerplate code**: Manipular vistas requería múltiples pasos (inflar layouts, buscar referencias con findViewById, actualizar estados manualmente).
2. **Complejidad en la sincronización de estados**: Mantener la coherencia entre la lógica y la UI ante cambios era propenso a errores.
3. **Falta de flexibilidad**: Sistemas de vistas jerárquicas (como ConstraintLayout) podían volverse difíciles de mantener en pantallas complejas.

En respuesta, Google introdujo **Jetpack Compose** en 2019 como parte de **Android Jetpack**, un conjunto de herramientas modernas para simplificar el desarrollo. Su versión estable (1.0) se lanzó en julio de 2021.

## ¿Qué es Jetpack Compose?

Jetpack Compose es un **kit de herramientas declarativo** para construir UIs nativas en Android, escrito completamente en Kotlin. En lugar de XML, defines la UI mediante **funciones componibles** (componentes reutilizables), que describen cómo debe verse la interfaz en función de su estado actual.

Principios Fundamentales:

1. **Declarativo**: Describes *qué* quieres mostrar, no *cómo* hacerlo (ejemplo: "muestra un botón rojo" vs. "instancia un Button, configura su color y añádelo a un LinearLayout").
2. **Funciones Componibles**: Bloques de construcción reutilizables escritos como funciones Kotlin anotadas con @Composable.
3. **Gestión Reactiva de Estados**: La UI se actualiza automáticamente cuando cambian los datos subyacentes.

## Arquitectura y Flujo de Trabajo

* **Composición**: Compose convierte tus funciones componibles en una estructura de nodos (árbol de UI).
* **Recomposición**: Mecanismo que actualiza solo las partes de la UI afectadas por cambios de estado, optimizando el rendimiento.
* **Interoperabilidad**: Puedes integrar Compose en proyectos existentes que usen Views tradicionales.

## Características Clave

1. **Kotlin-First**: Aprovecha características del lenguaje como lambdas, corrutinas y null-safety.
2. **Previews Interactivas**: Previsualiza componentes en Android Studio sin necesidad de ejecutar la app.
3. **Material Design Integrado**: Componentes predefinidos (Button, TextField, Card) alineados con Material Design 3.
4. **State Hoisting**: Patrón para centralizar la gestión de estados en componentes padres.
5. **Animaciones Simplificadas**: APIs declarativas para transiciones y efectos complejos.

## Ventajas

* **Menos Código**: Elimina XML y reduce el boilerplate.
* **UI Dinámica**: Facilita la creación de interfaces reactivas (ejemplo: dark mode, cambios de idioma).
* **Testing Simplificado**: Las funciones componibles son más fáciles de aislar y probar.
* **Compatibilidad**: Funciona en Android 5.0 (API 21) y superiores.

# Instalación

**1. Requisitos Previos**

* **Android Studio Hedgehog (2023.1.1)** o superior.
* **Android Gradle Plugin (AGP) 8.2** o superior.
* **Kotlin 1.9.20** o superior.
* **SDK mínimo**: API 21 (Android 5.0).

**2. Crear un Proyecto con Jetpack Compose**

1. **Abre Android Studio** y haz clic en **"New Project"**.
2. En la ventana de plantillas, selecciona **"Empty Compose Activity"** (o cualquier otra plantilla de Compose).
3. Configura el proyecto:
   * **Nombre**: Elige un nombre para tu app.
   * **Paquete**: Define tu package name.
   * **Lenguaje**: Asegúrate de que esté en **Kotlin**.
   * **Mínimum SDK**: API 21 o superior.
4. Haz clic en **"Finish"**.

**3. Configurar el Archivo build.gradle (Módulo :app)**

Abre el archivo build.gradle del módulo app (app/build.gradle) y verifica:

Plugins y Configuración Básica

Texto

El contenido generado por IA puede ser incorrecto.

**4. Dependencias de Jetpack Compose**

Captura de pantalla de un celular

El contenido generado por IA puede ser incorrecto.

# Guía de implementación en el proyecto

# Curiosidades o cosas a resaltar sobre la implementación

# Análisis de rendimiento

# Referencias

*JetPack Compose UI App Development Toolkit - Android Developers*. (s. f.). Android Developers. <https://developer.android.com/compose>