

Desarrollo de Aplicaciones para Dispositivos Móviles

Ingeniero Walter Medina
walter.medina@correounivalle.edu.co



Un poco de historia



- ✓ El primer teléfono móvil de la historia fue el **Motorola DynaTAC 8000X**, desarrollado en 1973 por la empresa Motorola.
- ✓ Pesaba 800 gramos y medía 33 cm de alto por 4.5 cm de ancho y 8.9 cm de grosor. Cuando salió al mercado costaba US\$3995, es decir unos 16 millones de pesos.
- ✓ La batería le duraba solo 1 hora.
- ✓ Martin Cooper, exdirector general de la división de sistemas de Motorola, dirigió un equipo que produjo el DynaTAC 8000x.
- ✓ En 1973 en Nueva York, Martin Cooper fue la primera persona de la historia en hacer una llamada telefónica desde un celular móvil.

Un poco de historia

...



- ✓ **El IBM Simon Personal Communicator** es un teléfono móvil fabricado por IBM y distribuido por BellSouth en 1992. Se le considera el primer teléfono móvil inteligente en la historia.
- ✓ Permitía mandar correos electrónicos, contaba con calendario, agenda, ofrecía programas (ahora apps) descargables.
- ✓ Cuando salió al mercado costaba US\$899, es decir unos 3 millones de pesos.
- ✓ Tenía 1 MB de memoria RAM.
- ✓ Cuando salieron más teléfonos inteligentes al mercado, El Simon se actualizó para poder ejecutar aplicaciones de otros fabricantes y se hacía por medio de tarjetas
- ✓ El precio elevado y la limitada duración de su batería contribuyeron a su pronta desaparición del mercado, apenas dos años después de su lanzamiento.

Un poco de historia...

El desarrollo de los teléfonos inteligentes se parte en dos.



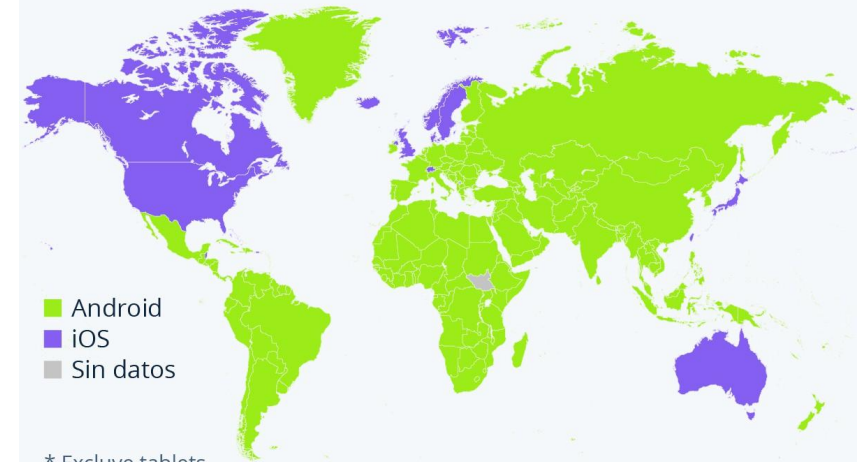
- ✓ La compañía Apple, bajo la dirección de Steve Jobs crea el **iPhone** en el 2007. Fue nombrado “El Invento del año” por la revista Time.
- ✓ Carece de un teclado físico, con pantalla táctil, cámara de 2 megapíxeles, Bluetooth, reproductor de música, software para enviar y recibir mensajes de texto y de voz. Ofrece servicios de Internet como leer correo electrónico, cargar páginas web y conectividad por Wi-Fi
- ✓ Ram de 128 MB, Capacidad de almacenamiento de 16 GB, con un peso de 135 g, dimensiones de 115 x 61 x 11,6 mm, más de 16 horas de batería.
- ✓ La supremacía de Apple no duraría mucho, ya que en el año 2008 la compañía HTC lanzó el HTC Dream, el primer dispositivo con Android como sistema operativo que poseía Bluetooth, Wi-Fi, SMS, entre otros. Desde entonces, iPhone y Android comienzan una guerra comercial hasta el día de hoy.



Android mantiene en la actualidad su posición como sistema operativo móvil líder a nivel mundial, con una cuota de mercado del **71%**, mientras que iOS representa el **28%**

El mapa mundial de Android e iOS

Sistema operativo móvil* con la mayor cuota de mercado por país (marzo de 2023)



■ Android
■ iOS
■ Sin datos

* Excluye tablets.
Fuente: StatCounter



statista



Desarrollo móvil nativo



iOS:

- IDE: Xcode
- Lenguaje: Swift

Android:

- IDE: Android Studio
- Lenguajes: Kotlin y Java

Desarrollo móvil híbrido

Flutter:

- IDE: Android Studio, Visual Studio COde, Emacs, etc
- Lenguaje: Dart

React Native:

- IDE: Visual Studio Code, Atom, Sublime Text, etc
- Lenguajes: JavaScript



Desarrollo móvil nativo

Ventajas:

- Las aplicaciones nativas suelen tener un rendimiento superior porque están optimizadas para la plataforma específica.
- Tienen acceso completo a las características y APIs del dispositivo, como cámara, GPS, sensores, etc.

Desventajas:

- Desarrollar y mantener dos bases de código separadas para iOS y Android puede ser costoso.
- Requiere más tiempo porque hay que desarrollar y probar cada plataforma por separado.



Desarrollo móvil híbrido

Ventajas:

- Un solo código base puede funcionar en múltiples plataformas, lo que reduce el costo de desarrollo y mantenimiento.
- El tiempo de desarrollo generalmente es más rápido porque se desarrolla una vez y se despliega en ambas plataformas.

Desventajas:

- Las aplicaciones híbridas pueden no ser tan rápidas o fluidas como las nativas.
- El acceso a funcionalidades del dispositivo puede ser limitado o requerir plugins adicionales.



La elección final depende de tus prioridades y recursos específicos

¿Por qué nos vamos por Desarrollo Móvil Nativo Android?

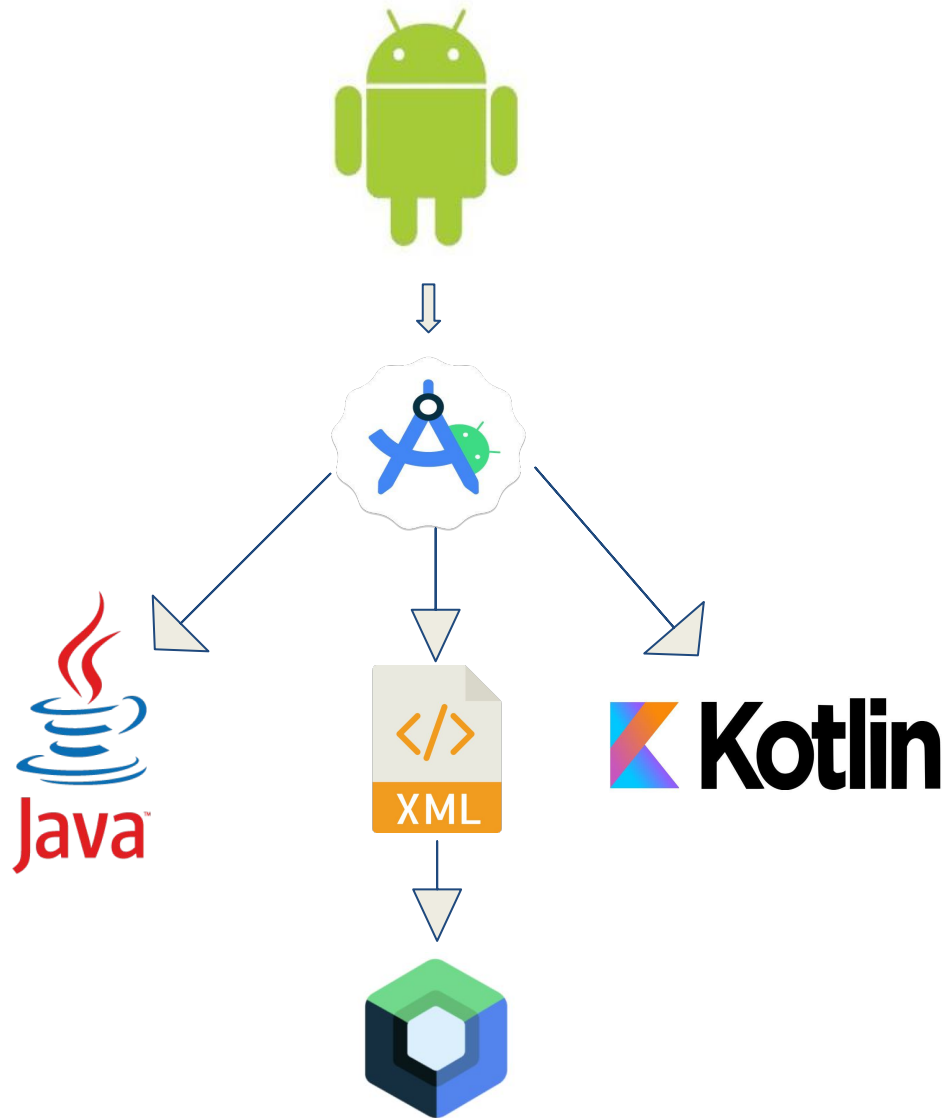
- Acceso completo a las características y APIs del dispositivo, como cámara, GPS, sensores, etc.
- Rendimiento más rápido y eficiente en comparación con aplicaciones híbridas.
- Para desarrollar aplicaciones para iOS o híbrido es recomendable tener una computadora Mac.
- Muchas librerías son open source para el sistema operativo android
- Android es compatible con una amplia variedad de dispositivos y fabricantes
- **Kotlin Multiplataforma:** Es una tecnología que permite crear aplicaciones para varias plataformas y reutilizar eficazmente el código en todas ellas conservando las ventajas de la programación nativa. Las aplicaciones multiplataforma funcionan en diferentes sistemas operativos, como iOS, Android, macOS, Windows, Linux y otros.

<https://www.jetbrains.com/es-es/kotlin-multiplatform/>



android

Desarrollo móvil nativo Android



- ✓ Java y Kotlin pueden convivir sin ningún problema, es decir, puedes hacer algún módulo de la app en java y otro en kotlin.
- ✓ Debido a disputas legales entre Google dueño de android y Oracle propietario de Java, Google decide asociarse con la empresa JetBrains, creadora de kotlin, y desde ese momento kotlin es el lenguaje oficial para desarrollar aplicaciones móviles para Android.
- ✓ Para realizar las vistas de una aplicación se puede hacer con XML y con Jetpack Compose.

Aprendiendo kotlin

Declaración e inicialización de variables

```
var numero: Int = 10
numero = 15

var texto: String = "hola mundo"
texto = "hola estudiantes"

var booleano: Boolean = true
booleano = false

//variables sin usar el tipo de dato:
var texto = "hola mundo"
var numero = 12
var booleano = true
```

Aprendiendo kotlin

Variables con inicializaciones tardías

```
var texto: String //lanza error, kotlin te obliga a inicializarla
lateinit var miTexto:String //lateinit no se usa para tipos primitivos

fun main() {
    miTexto = "hola mundo"
}
```

lateinit es un modificador que se utiliza para declarar variables que se inicializarán más adelante, pero no en el momento de su declaración.

Aprendiendo kotlin

Nulabilidad de variables

```
var texto:String = null // Error
var miTexto:String? = null // ?: para hacer nutable una variable
var numero:Int? = null

fun main() {
    miTexto = "hola mundo"
    print(miTexto)
}
```

La **nulabilidad** permite asignar un valor nulo a una variable. Para utilizar la nulabilidad en Kotlin es necesario hacer uso del identificador “?” con el propósito de que el programa pueda comprobar que hace referencia a un valor nulo.

Aprendiendo kotlin

Constantes

```
val miConstante: Int = 10
miConstante = 20 //Error

const val VARIABLE_PI = 3.1459

const val PI = miConstante // ERROR, se debe asignar un valor directamente en tiempo de compilación
```

La diferencia principal entre **val** y **const** en Kotlin es que **val** se utiliza para declarar constantes cuyo valor se establece en **tiempo de ejecución**, mientras que **const** se utiliza para declarar constantes que deben tener un valor en **tiempo de compilación**.

Aprendiendo kotlin

Condicionales

```
if (edad >= 18) {  
    println("Eres mayor de edad")  
} else {  
    println("Eres menor de edad")  
}
```

```
val respuesta: String = if (edad >= 18) {  
    "Eres mayor de edad"  
} else if (edad < 18) {  
    "Eres menor de edad"  
} else {  
    "Edad incorrecta"  
}  
  
println(respuesta)
```

Aprendiendo kotlin

when es el equivalente a **switch** en java

```
fun main() {  
    val x = 2  
    when (x) {  
        1 -> println("x es igual a 1")  
        2 -> println("x es igual a 2")  
        3 -> println("x es igual a 3")  
        else -> println("x no es igual a ninguna opción")  
    }  
}
```

Aprendiendo kotlin

listas y estructura for

```
//Lista inmutable, no se puede modificar  
val miLista: List<Int> = listOf(1, 2, 3, 4, 5)  
  
for (i in miLista) {  
    println(i)  
}  
  
//Lista mutable, se puede modificar  
val miListaMutable: MutableList<Int> = mutableListOf()  
  
for (i in 1..10) {  
    miListaMutable.add(i)  
}
```

Aprendiendo kotlin

while y do-while

```
//ejemplo while:
var numero = 0
while (numero < 5) {
    println("valor de numero es: $numero")
    numero++
}

//ejemplo do-while
var numero = 0
do {
    println("valor de numero es: $numero")
    numero++
} while (numero < 5)
```

Aprendiendo kotlin

Funciones, clases y objetos

```
class MiClase {  
    fun main() {  
        val persona = Persona(  
            nombre = "juan",  
            apellido = "gonzalez",  
            edad = 25  
        )  
        println(  
            "nombre:${persona.nombre} \n"+  
            "apellido:${persona.apellido}\n"+  
            "edad:${persona.edad}"  
        )  
    }  
}  
  
class Persona (val nombre: String, val apellido: String, val edad: Int)
```

*En kotlin se define directamente el constructor en la declaración de la clase

*En kotlin no es necesario usar getter y setter, el lenguaje internamente lo hace por nosotros

Aprendiendo kotlin

Funciones lambda con parámetros (retorno)

Las funciones lambda son expresiones que **se pueden pasar como argumentos a otras funciones o asignarlas a variables.**

Tienen una sintaxis compacta y son útiles cuando necesitas pasar una función como parámetro o cuando quieres escribir código de manera más concisa.

```
val name: (Int, Int) -> Unit = { a, b ->
    println(a + b)
}
```

```
fun main() {
    val lambda = Lambda()
    lambda.requestPassword(passwordRequest = "1234")
}

class Lambda {

    fun requestPassword(passwordRequest: String) {
        responseData(passwordRequest) { response ->
            print("response: `${response}`")
        }
    }

    fun responseData(
        passwordRequest: String,
        responseData: (response: String) -> Unit
    ) {
        val passwordSuccessful = "1234"
        if (passwordSuccessful == passwordRequest) {
            responseData.invoke(response: "contraseña correcta")
        } else {
            responseData.invoke(response: "contraseña incorrecta")
        }
    }
}
```

Aprendiendo kotlin

Funciones lambda sin parámetros

```
val name: () -> Unit = {  
    //puedo hacer aquí cualquier implementación  
}
```

```
fun main() {  
    val lambda = Lambda()  
    lambda.lambdaSinParametro()  
}  
  
class Lambda {  
  
    fun lambdaSinParametro() {  
        responseData {  
            val num1 = 5  
            val num2 = 10  
            val result = num1 + num2  
            print("suma: ${result}")  
        }  
    }  
  
    fun responseData(  
        responseData: () -> Unit  
    ) {  
        responseData.invoke()  
    }  
}
```

Aprendiendo kotlin

Ejercicio en clase

Realice un clase llamada **Estudiante** cuyos atributos sean **nombre**, **edad** y **curso**, luego cree una función llamada **main** donde llamará a un método llamado **listStudentResponse** que recibe por parámetro una edad y una función lambda.

La función lambda retornará o responderá una lista de estudiantes de acuerdo a la edad ingresada, es decir, si ingresa una edad ≥ 18 , la lambda responderá con una lista de estudiantes que tengan una edad ≥ 18 , si ingresa una edad < 18 , la lambda responderá con una lista de estudiantes que tengan una edad < 18 . Finalmente imprima la lista de estudiantes con su nombre, edad y curso.

Para el ejercicio puede crear una lista manual de 5 estudiantes.

IDE Online :

<https://play.kotlinlang.org/>

Lecturas recomendadas:

- <https://developer.android.com/kotlin/learn?hl=es-419>
- <https://www.w3schools.com/kotlin/index.php>
- <https://developer.android.com/codelabs/basic-android-kotlin-compose-function-types-and-lambda?hl=es-419#0>



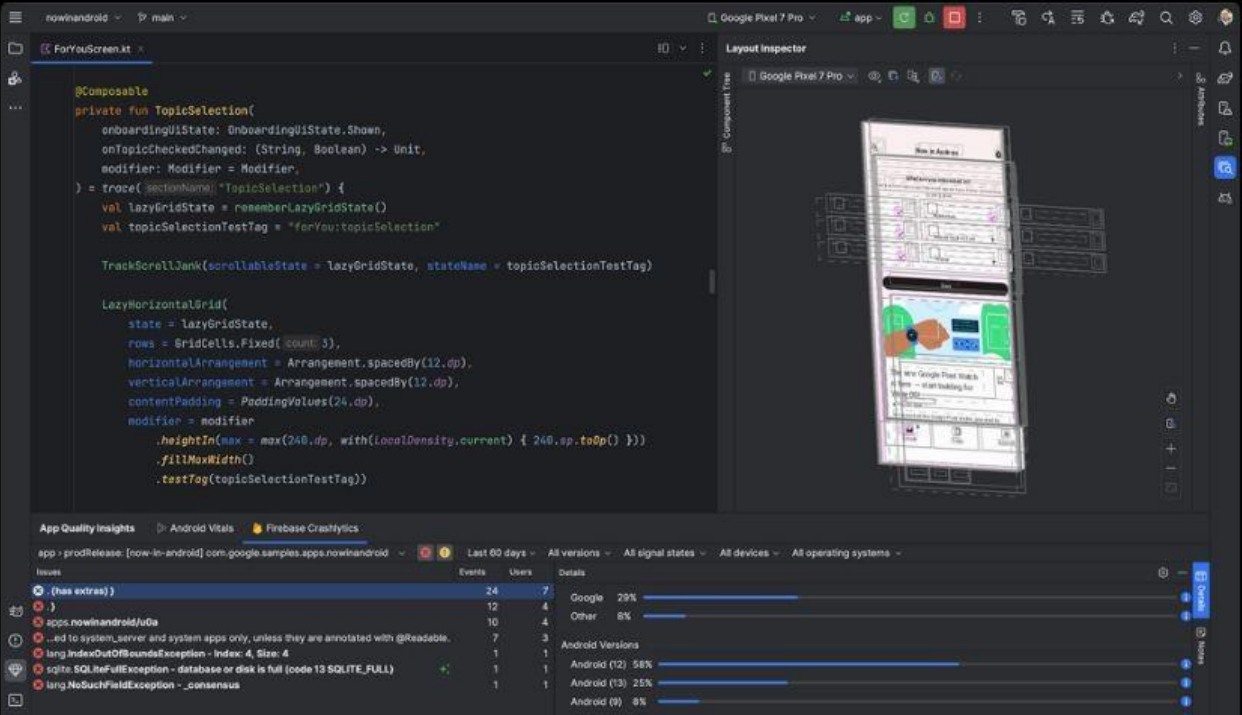
Descargar el IDE Android Studio

Android Studio

Get the official Integrated Development Environment (IDE) for Android app development.

[Download Android Studio Giraffe](#)

[Read release notes](#)



Sitio web oficial:

<https://developer.android.com/studio>

Instrucciones Instalación:

<https://developer.android.com/codelabs/basic-android-kotlin-compose-install-android-studio?hl=es-419#0>

GRACIAS !!

*“El esfuerzo vence al talento cuando el
talento no se está esforzando”.*
Anónimo



Bibliografía:

- ❖ <https://history-computer.com/largest-smartphone-companies-in-the-world/>
- ❖ https://es.wikipedia.org/wiki/Motorola_DynaTAC
- ❖ https://es.wikipedia.org/wiki/IBM_Simon
- ❖ [https://es.wikipedia.org/wiki/IPhone_\(1.%C2%AA_generaci%C3%B3n\)](https://es.wikipedia.org/wiki/IPhone_(1.%C2%AA_generaci%C3%B3n))
- ❖ <https://gs.statcounter.com/os-market-share/mobile/worldwide>
- ❖ <https://www.w3schools.com/kotlin/index.php>
- ❖ <https://developer.android.com/kotlin/learn?hl=es-419>
- ❖ <https://www.jetbrains.com/es-es/kotlin-multiplatform/>

