



# CBTIS 44



## Manual de Android estudio

Programación 6I

- Desarrollo aplicaciones móviles para Android
- Desarrollo aplicaciones móviles para iOS

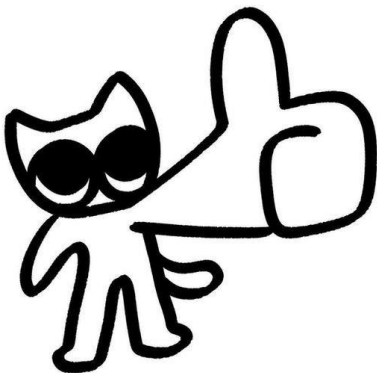
Marisela Hernández Morales



Marlene Faridy González Perdomo

# Índice

- 1.** Introducción
- 2.** Instalación de Android Studio
- 3.** Conociendo la interfaz
- 4.** Creación de tu primer proyecto en Kotlin
- 5.** Estructura de carpetas y archivos
- 6.** Uso del emulador
- 7.** Escribir y ejecutar tu primera app
- 8.** Edición de interfaz: XML y Compose
- 9.** Tips para principiantes
- 10.** Atajos y buenas prácticas
- 11.** Recursos adicionales



## *1.Introducción*

Android Studio es un entorno de desarrollo integrado (IDE, por sus siglas en inglés) creado específicamente para el **desarrollo de aplicaciones móviles en el sistema operativo Android**. Es una herramienta potente y ampliamente utilizada por desarrolladores de todo el mundo.

Es una plataforma de desarrollo de software que proporciona un conjunto de herramientas y recursos para crear aplicaciones móviles para Android. Es desarrollado y mantenido por Google, y se ha convertido en la opción preferida de muchos desarrolladores debido a su amplia gama de características y su estrecha integración con el ecosistema de Android.

**Entre las principales características de Android Studio se encuentran:**

- **Editor de código inteligente:** Ofrece un editor de código inteligente con funciones como autocompletado, sugerencias de código y resaltado de sintaxis que facilitan la escritura y la depuración de código.
- **Emulador de Android:** Incluye un emulador de Android que permite probar y depurar aplicaciones en diferentes dispositivos virtuales sin la necesidad de tener un dispositivo físico.
- **Administrador de dependencias:** Facilita la gestión de las dependencias de tu proyecto a través de su integración con Gradle, un sistema de compilación potente y flexible.
- **Herramientas de diseño de interfaz de usuario:** Ofrece un conjunto de herramientas para diseñar y crear interfaces de usuario intuitivas y atractivas, como el editor de diseño visual y la vista previa en tiempo real.
- **Depuración y perfilado:** Proporciona herramientas avanzadas de depuración y perfilado que permiten identificar y solucionar problemas en el rendimiento y el funcionamiento de las aplicaciones.

## 2. Instalación de Android Studio

### Pasos:

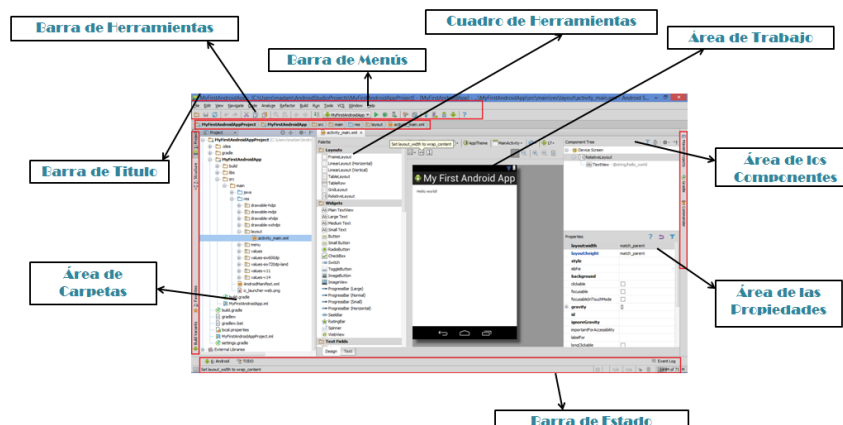
1. Ve a <https://developer.android.com/studio>
2. Descarga el instalador correspondiente a tu sistema operativo.
3. Ejecuta el instalador y sigue las instrucciones.
4. Asegúrate de instalar también el Android SDK, el emulador y Kotlin Plugin si se ofrece.

### Requisitos recomendados:

- RAM: 8 GB o más
- Espacio: al menos 4 GB libres
- Procesador: Intel i5 o superior

## 3. Conociendo la interfaz

- Barra de herramientas: para compilar, ejecutar y acceder a funciones rápidas.
- Panel del proyecto: muestra la estructura de carpetas.
- Editor de código: donde escribes Kotlin y XML.
- Consolas y registros: muestran errores, mensajes del sistema y más.
- Diseñador de interfaces (Layout Editor): para diseñar pantallas visualmente.



## *4. Creación de tu primer proyecto en Kotlin*

1. Abre Android Studio > New Project.
2. Elige “Empty Activity”.
3. En el lenguaje, selecciona Kotlin.
4. Escribe un nombre para tu app y elige dónde guardarla.
5. Presiona Finish y espera a que se configure el entorno.

## *5. Estructura de carpetas y archivos*

- app/src/main/java: Aquí va el código Kotlin.
- app/src/main/res/layout: Aquí están los archivos XML para las interfaces.
- AndroidManifest.xml: Archivo que declara tu app, sus componentes y permisos.
- build.gradle: Configura las dependencias y compilación.

## *6. Uso del emulador*

1. Ve a Tools > Device Manager.
2. Clic en “Create device”.
3. Selecciona un dispositivo virtual (por ejemplo, Pixel 5).
4. Elige una versión de Android.
5. Presiona Finish y luego Run.




## 7. Escribir y ejecutar tu primera app

En MainActivity.kt, modifica el método onCreate así:

kotlin

 Copiar

 Editar

```
override fun onCreate(savedInstanceState: Bundle?) {  
    super.onCreate(savedInstanceState)  
    setContentView(R.layout.activity_main)  
    Toast.makeText(this, "¡Hola desde Kotlin!", Toast.LENGTH_SHORT).show()  
}
```

## 8. Edición de interfaz: XML y Compose

XML (modo tradicional):

- Abre activity\_main.xml.
- Usa etiquetas como <TextView>, <Button>, etc.

Jetpack Compose (moderno):

- Permite crear interfaces directamente en Kotlin.

kotlin

```
@Composable  
fun MiApp() {  
    Text(text = "Hola Mundo", fontSize = 24.sp)  
}
```

## *9. Tips para principiantes*

- Guarda cambios frecuentemente: Ctrl+S o Cmd+S.
- Usa Logcat para ver errores o mensajes del sistema.
- Si algo falla, intenta Clean Project y Rebuild Project.
- Lee los mensajes de error cuidadosamente.
- Personaliza el tema desde Settings > Appearance & Behavior.

## *10. Atajos y buenas prácticas*

Atajos:

Windows/Linux - Mac

Ejecutar proyecto: Shift+F10 - Control+R

Autocompletar código: Ctrl+Space - Control+Space

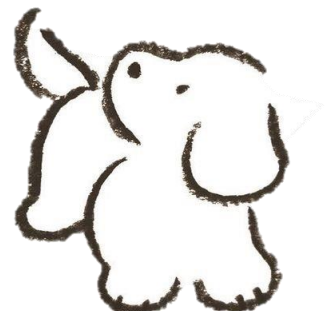
Buscar archivos: Ctrl+Shift+N - Cmd+Shift+O

Ir a definición: Ctrl+B - Cmd+B

Comentar línea: Ctrl+/ - Cmd+/

Buenas prácticas:

- Nombra tus variables y funciones con claridad.
- Usa comentarios útiles.
- Organiza tu código por responsabilidades.



## 11. Recursos adicionales



- Documentación oficial: <https://developer.android.com>
- Curso gratuito de Kotlin: <https://kotlinlang.org/docs/home.html>
- Comunidad en español: foros y grupos de Telegram sobre Android y Kotlin

## Zonas y botones de la interfaz

### 1. Panel del proyecto (izquierda)

- Muestra la estructura de carpetas y archivos.
- Puedes alternar entre diferentes vistas (Project, Android, Packages, etc.) usando el ícono de carpeta en la parte superior.
- Aquí puedes ver carpetas como `app`, `src`, `main`, `res`, y archivos como `AndroidManifest.xml`, `build.gradle.kts`, etc.

### 2. Archivos abiertos en pestañas

- En la parte superior se ven los archivos abiertos:  
`activity_main.xml`, `MainActivity.kt`.  
Puedes hacer clic para cambiar entre ellos.







### 3. Editor de código (centro)

- Área donde se edita el código. En tu caso, está mostrando el archivo `MainActivity.kt` con código en Kotlin.
- Líneas con `onCreate()` y `setContentView()` son las que inicializan la pantalla principal de tu app.




## 4. Botones de la barra superior

Aquí están varios íconos importantes:

Ícono	Función
 (Play)	Ejecuta tu aplicación. Abre el emulador o dispositivo físico.
 (Build)	Construye tu proyecto (Build Project). Útil después de cambios grandes.
 (Sync)	Sincroniza el proyecto con Gradle. Esto soluciona muchos errores comunes.
 (Buscar)	Permite buscar texto, clases o archivos.
 (Settings)	Abre la configuración de Android Studio.
 (Cuenta)	Cambia o gestiona tu cuenta de Google para Android Studio.

## 5. Panel lateral derecho - Running Devices

- Aquí puedes:
  -  Ejecutar tu app en un dispositivo físico o virtual.
  - Hacer **mirror** de un teléfono conectado por USB o WiFi.
  - Crear un emulador nuevo con el botón +.

## 6. Pestañas inferiores (no visibles en tu captura completa)

- Suelen mostrar:
  - Logcat: muestra errores y mensajes del sistema.
  - Terminal: consola de comandos.
  - Build: progreso y errores de compilación.

---

## Recomendaciones rápidas

- Si el mensaje dice "**Gradle project sync in progress...**", espera a que termine antes de ejecutar tu app.
- Para correr tu app, asegúrate de tener un emulador o dispositivo listo desde el panel derecho.

## Código de playground

```
import java.text.DecimalFormat

fun main() {

    val calculadora = Calculadora()

    // Ejemplo de uso

    calculadora.seleccionarNumero("8")

    calculadora.cambiarOperador("+")

    calculadora.seleccionarNumero("4")

    calculadora.igual()

    println("Resultado: ${calculadora.tvResult}")

}

class Calculadora {

    val SUMA = "+"

    val RESTA = "-"

    val MULTIPLICACION = "*"

    val DIVISION = "/"

    val PORCENTAJE = "%"

    var operacionActual = ""

    var primerNumero: Double = Double.NaN

    var segundoNumero: Double = Double.NaN

    var tvTemp: String = ""

    var tvResult: String = ""

    val formatoDecimal = DecimalFormat("#.#####")
```

```

fun cambiarOperador(operador: String) {

    if (tvTemp.isNotEmpty() || !primerNumero.isNaN()) {

        calcular()

        operacionActual = when (operador.trim()) {

            "÷" -> "/"

            "X" -> "*"

            else -> operador.trim()

        }

    }

    if (tvTemp.isEmpty()) {

        tvTemp = tvResult

    }

    tvResult = formatoDecimal.format(primerNumero) + operacionActual

    tvTemp = ""

}

}

fun calcular() {

    try {

        if (!primerNumero.isNaN()) {

            if (tvTemp.isEmpty()) {

                tvTemp = tvResult

            }

            segundoNumero = tvTemp.toDouble()

            tvTemp = ""

            primerNumero = when (operacionActual) {

                "+" -> primerNumero + segundoNumero

                "-" -> primerNumero - segundoNumero

                "*" -> primerNumero * segundoNumero

                "/" -> primerNumero / segundoNumero

                "%" -> primerNumero % segundoNumero

                else -> primerNumero

            }

        }

    }

}

```

```

    } else {

        primerNumero = tvTemp.toDouble()

    }

} catch (e: Exception) {

    println("Error en el cálculo: ${e.message}")

}

}

fun seleccionarNumero(numero: String) {

    tvTemp += numero

}

fun igual() {

    calcular()

    tvResult = formatoDecimal.format(primerNumero)

    operacionActual = ""

}

fun borrar(tipo: String) {

    when (tipo.trim()) {

        "C" -> {

            if (tvTemp.isNotEmpty()) {

                tvTemp = tvTemp.dropLast(1)

            }

        }

    }

} else {

    primerNumero = Double.NaN

    segundoNumero = Double.NaN

    tvTemp = ""

    tvResult = ""

}

}

```

```
"CA" -> {  
  
    primerNumero = Double.NaN  
  
    segundoNumero = Double.NaN  
  
    tvTemp = ""  
  
    tvResult = ""  
  
}  
  
}  
  
}  
  
}
```

```
// Ejemplo de uso  
calculadora.seleccionarNumero("8")  
calculadora.cambiarOperador("+")  
calculadora.seleccionarNumero("4")  
calculadora.igual()  
println("Resultado: ${calculadora.tvResult}")  
}  
class Calculadora {
```

```
Resultado: 12
```