AWAKELAB

#programmingbootcamp

MI PRIMERA APP

¿QUÉ VAMOS A VER?

- Crear un proyecto con plantilla.
- Ejecutar Android Emulator
- Ejecutar Apps en un dispositivo local.
- Layouts

¡VAMOS A COMENZAR!

AWAKELAB

CREAR UN PROYECTO CON PLANTILLA

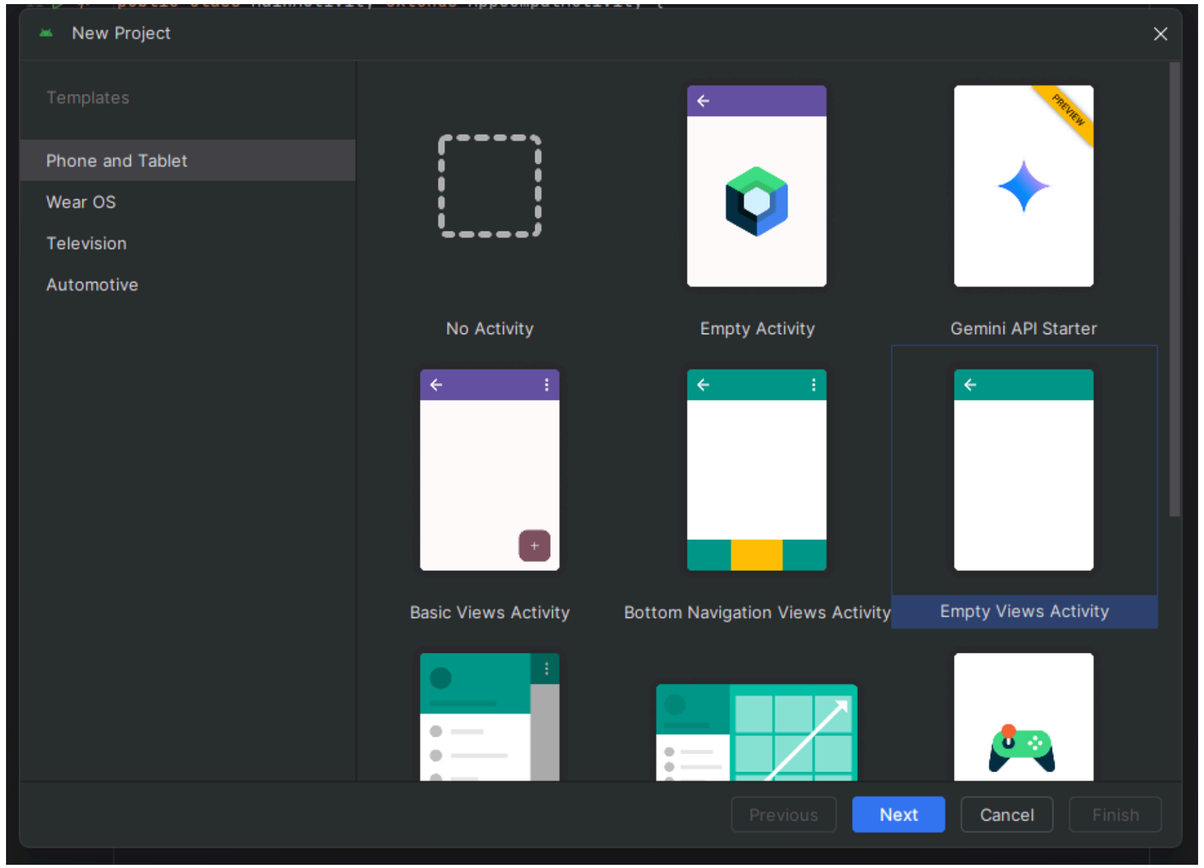
AWAKELAB

CREANDO UN PROYECTO

- Para comenzar, vamos a crear un proyecto utilizando la plantilla **Empty Views Activity**.

- **¿Por qué no utilizamos directamente Empy Activity con JetPack Compose?**

- Si bien, **JetPack Compose** es lo más actual en Android para el diseño de interfaces gráficos, solamente es compatible con Kotlin.



CREANDO UN PROYECTO

- Una vez seleccionada la plantilla, se nos abrirá una ventana en dónde vamos a rellenar con la configuración de nuestro proyecto.

- **Nombre:** Vamos a mantener el uso de PascalCase para nombrar las clases y CamelCase para los atributos/variables.

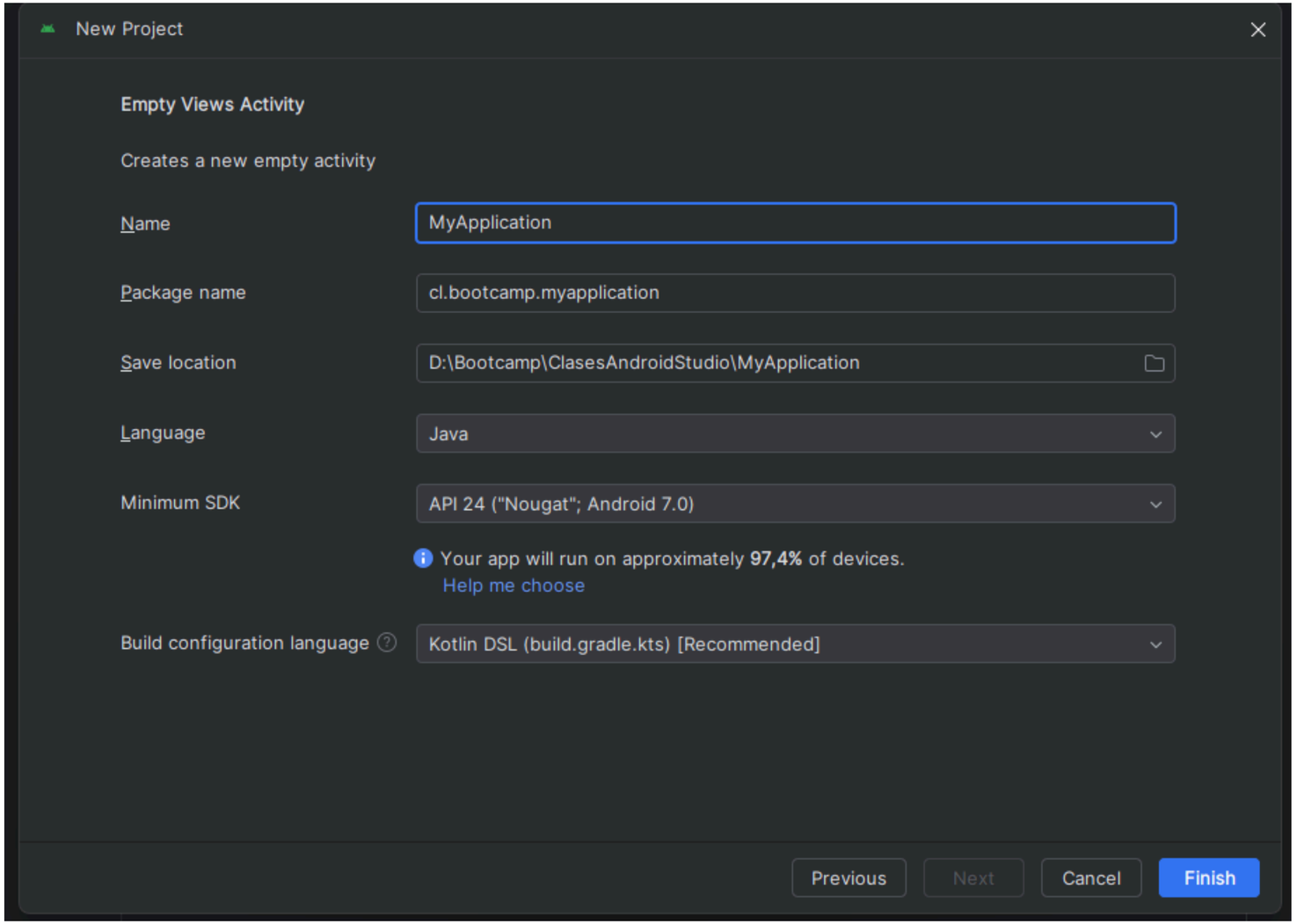
- **Paquete:** Lo ideal en este punto es dar una buena referencia de dónde está siendo creado el proyecto. (ej. *país.empresa.proyecto*).

- **Ruta:** Lugar dónde estará almacenado el proyecto.

- **Lenguaje:** Comenzaremos utilizando ***Java*** como nuestro lenguaje de programación.

- **SDK:** Ya que Android lo recomienda, comenzaremos con ***Nougat***.

CREANDO UN PROYECTO

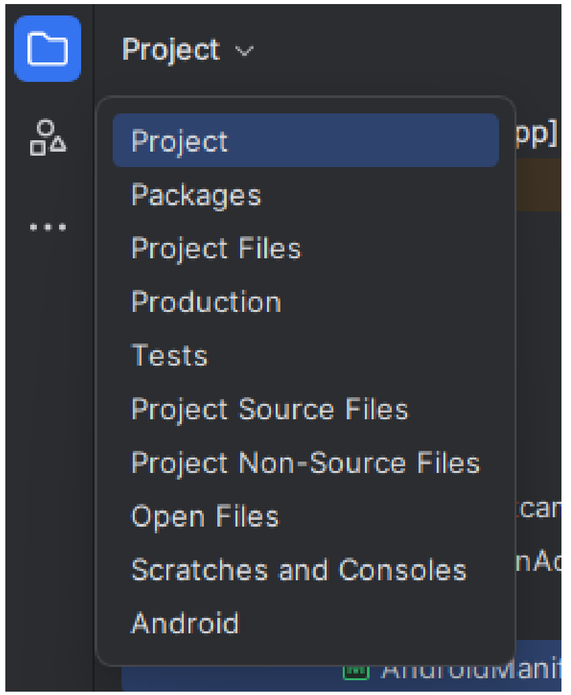


ORDEN DE CARPETAS

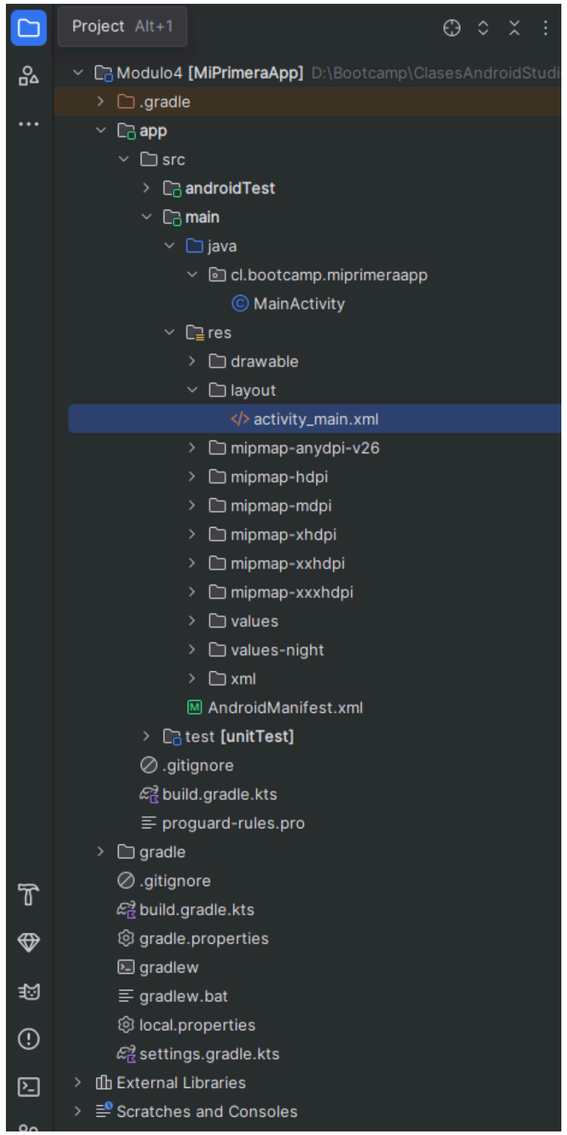
- Una vez las dependencias de Gradel estén instaladas, nos encontraremos con que Android Studio nos organizó todo nuestro proyecto en base a la plantilla y lenguaje seleccionados.

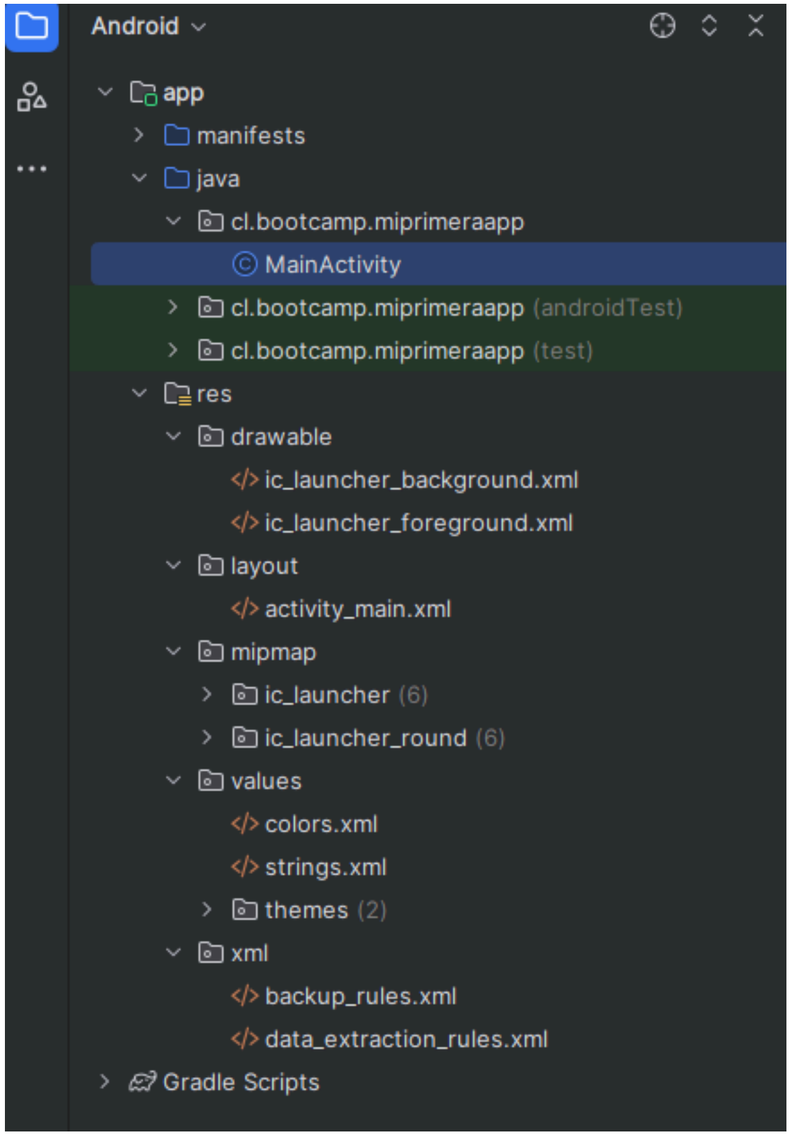
- En el lateral izquierdo nos vamos a encontrar con todos los campos de nuestro interés, en dónde también podemos elegir la organización de carpetas que nos muestra el IDE.

- El método **Project** se centra en todas las carpetas que realmente necesitamos, mientras que **Android** es una opción más acotada, pero que ocasiones también viene muy bien.



ORDEN DE CARPETAS





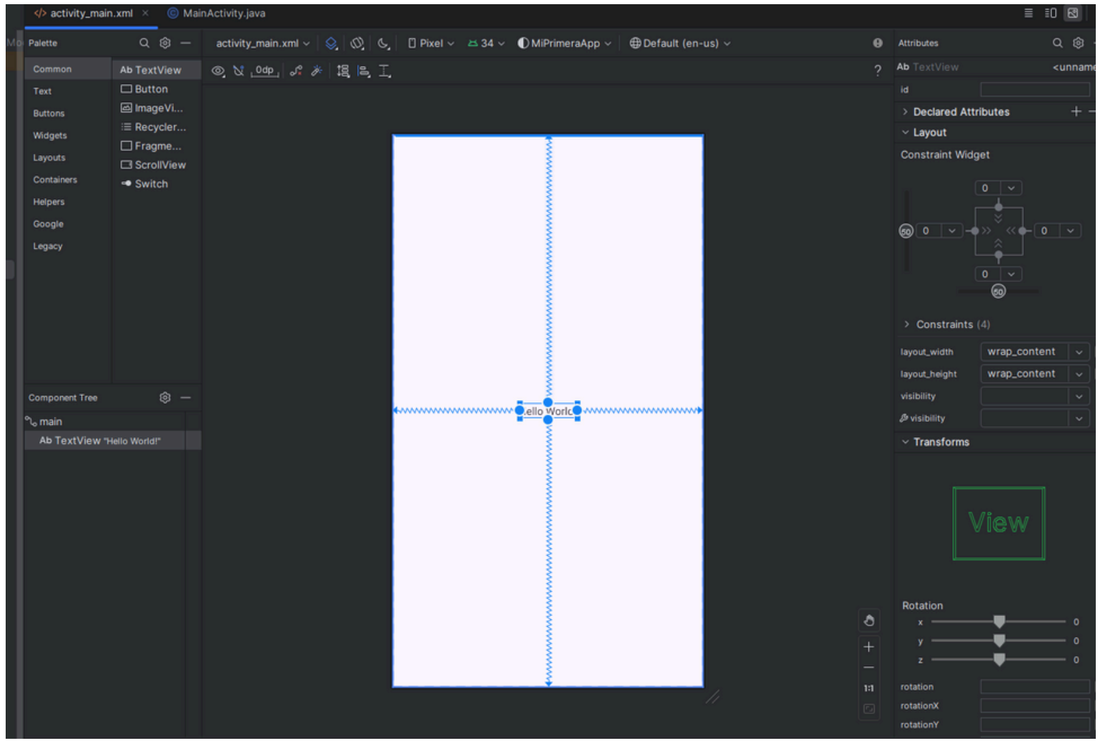
activity_main.xml

- Uno de los archivos que se creará por defecto es la plantilla base de nuestro diseño.

- Quizás en un comienzo solo vean código, pero en la esquina superior derecha se podrán encontrar diferentes opciones para visualizar el modo **Diseño**.

- A veces por defecto también viene activo el modo contraste, es posible que por eso les aparezca una pestaña en blanco y otra en un color azulino.

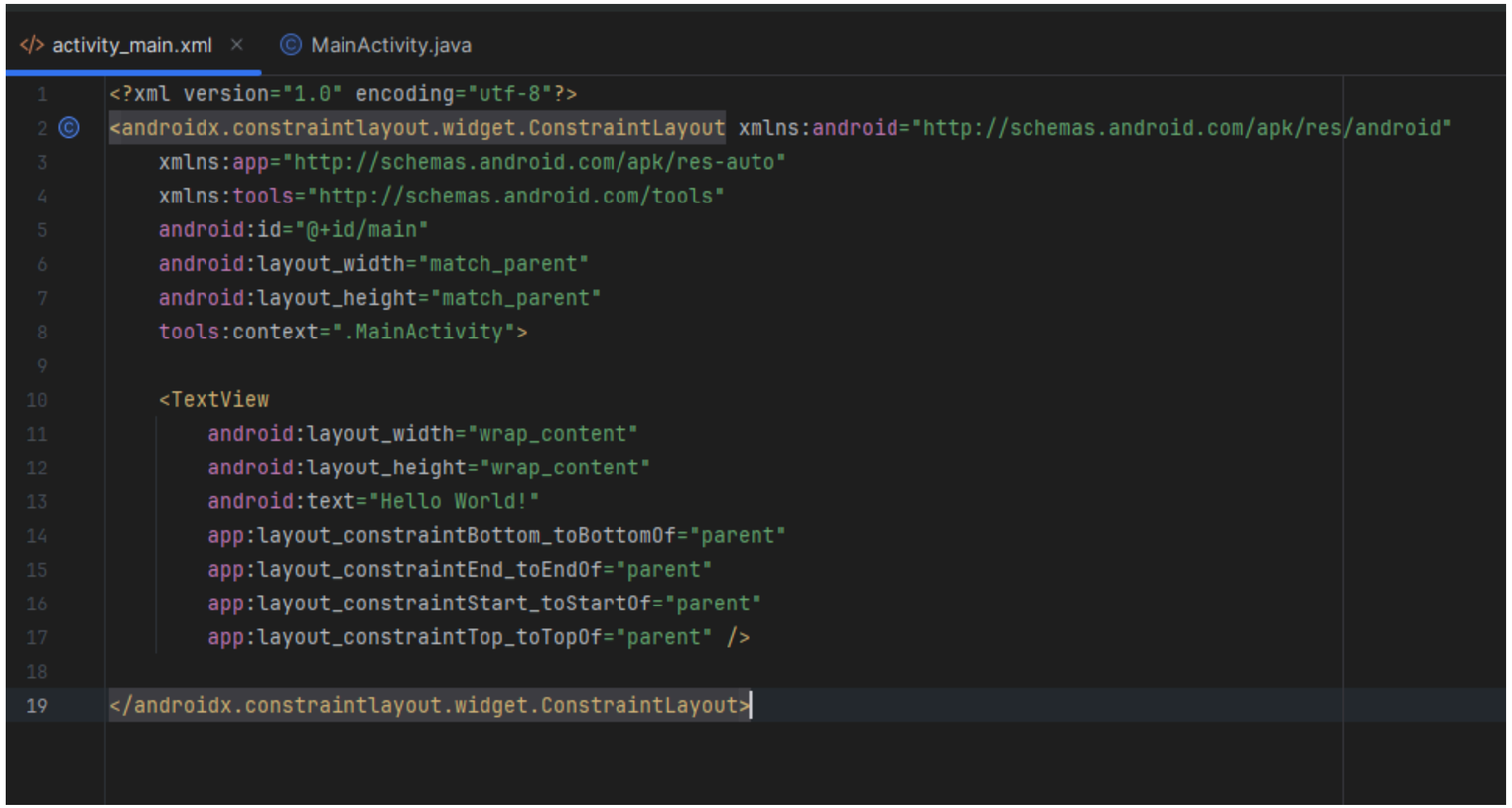
- Es desde esta pestaña en dónde podemos crear nuestro primer diseño solo utilizando las herramientas que nos proporciona el IDE.



activity_main.xml

- También tenemos la opción de generar nuestro propio código y de esa forma añadir los diferentes elementos.

- Ninguna forma es mejor que la otra, aunque la recomendación es que siempre va a ser optimo conocer el código que estamos creando para no tener problema a la hora de manipularlo o leerlo sin ver la parte gráfica.



EJECUTAR ANDROID EMULATOR

AWAKELAB

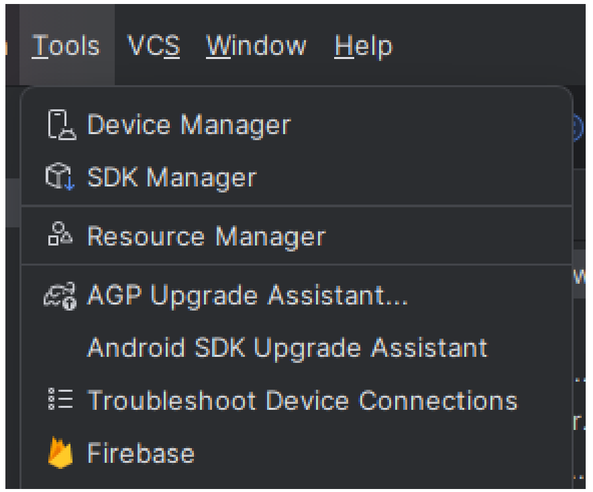
ANDROID EMULATOR

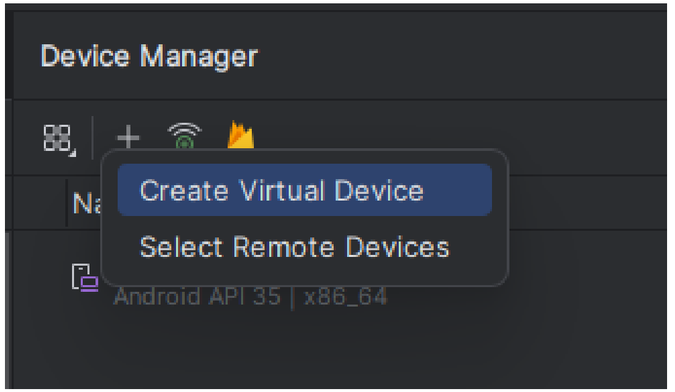
- Para generar nuestro propio dispositivo dentro de Android Studio, lo primero es dirigirse a **Tools > Device Manager**

- Se abrirá una pestaña en la derecha que corresponde al **Device Manager**. Actualmente, por defecto, Android Studio genera un dispositivo *Pixel Fold API 35*.

- Vamos a crear un dispositivo nuevo, por lo tanto vamos a símbolo de **+** y seleccionamos **Create Virtual Device**.

- Con eso se abrirá la configuración del dispositivo virtual.





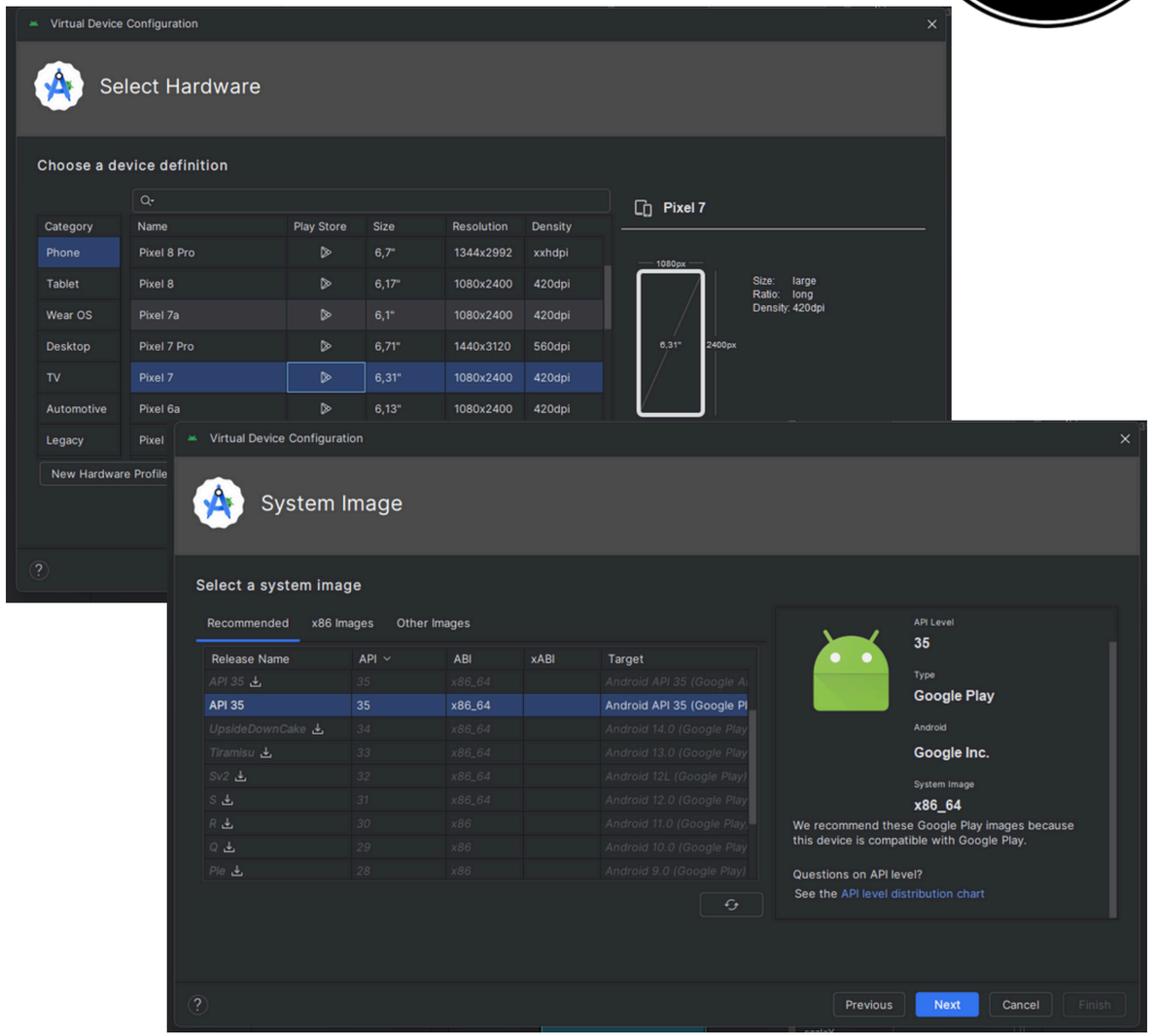
ANDROID EMULATOR

- Para este ejemplo seleccionaremos el **Pixel 7**.

- Se nos abrirá la selección de **System Image**, en dónde tendremos muchas opciones. Por esta vez, nos aprovecharemos la que la versión Koala trae por defecto. **API 35**.

- En la pestaña de configuración pueden cambiar el nombre del dispositivo y darle a **Finish**.

- Con eso, ya tendremos creado un nuevo dispositivo virtual para visualizar nuestras aplicaciones.



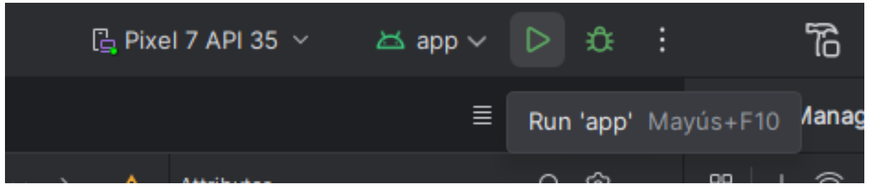
ANDROID EMULATOR

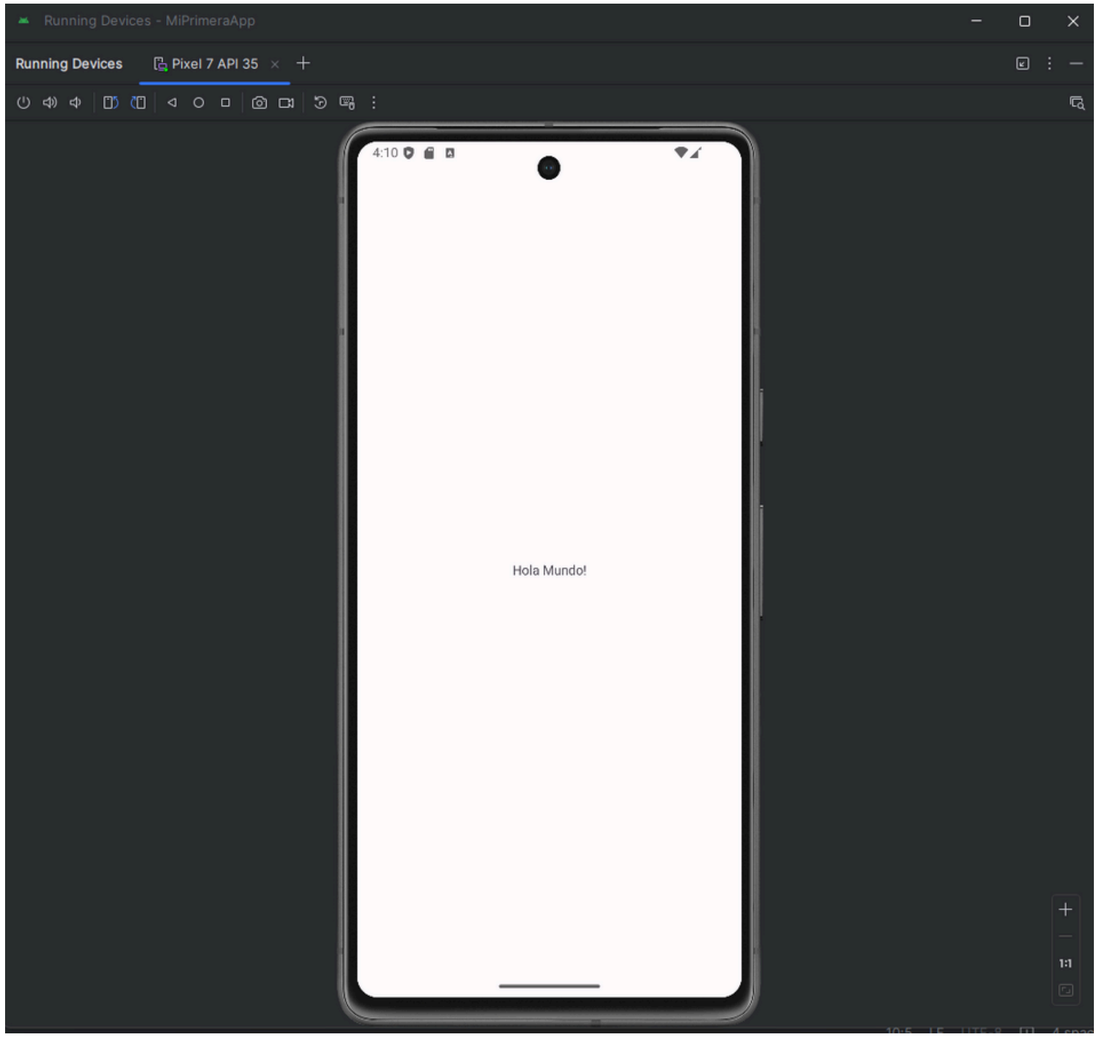
- Para poner en marcha nuestra aplicaciones, deben pinchar sobre la **flecha verde** o **Mayús + F10**.

- La primera vez que ejecuten un dispositivo virtual o quieran correr una nueva aplicación, tardará un poco más de lo normal porque debe configurar el equipo e instalar la aplicación como tal.

- Luego de la espera, se les abrirá el emulador y con el, la aplicación.

- **¿Es la mejor opción?** Depende, si tu equipo va limitado de RAM, es mejor recurrir a un dispositivo físico que a uno virtual.



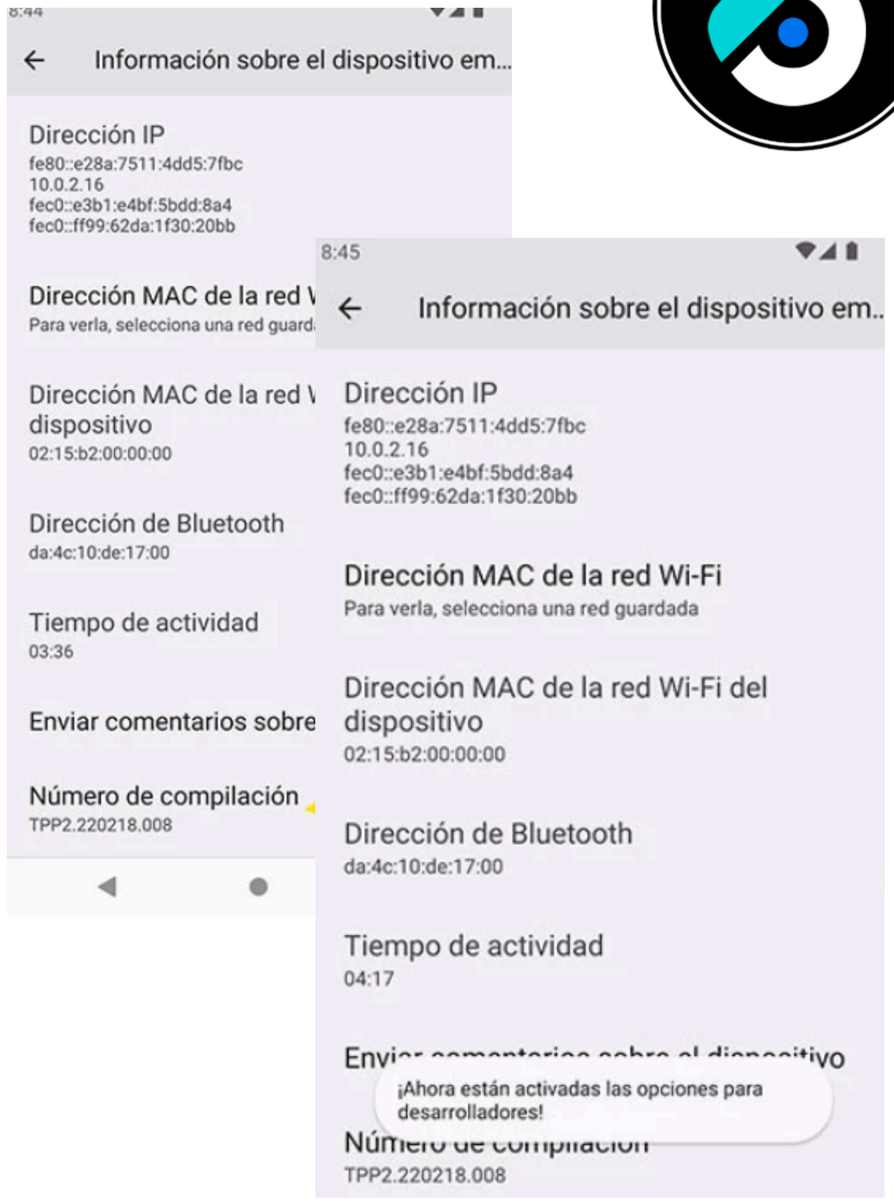


EJECUTAR APPS EN UN DISPOSITIVO LOCAL

AWAKELAB

DISPOSITIVO LOCAL

- **Opción más recomendada.**
- Lo primero, es contar con un dispositivo Android que tenga un SO igual o superior al que escogimos cuando creamos nuestro proyecto,
- *** Los pasos a seguir pueden variar un poco según el dispositivo ***
- Deben dirigirse a **Ajustes > Acerca del teléfono > Información de software > Número de compilación.**
- Una vez estén ahí, deben dar **clic 7 veces sobre el Número de compilación**. El dispositivo les avisará una vez las opciones de desarrollador fueron activadas



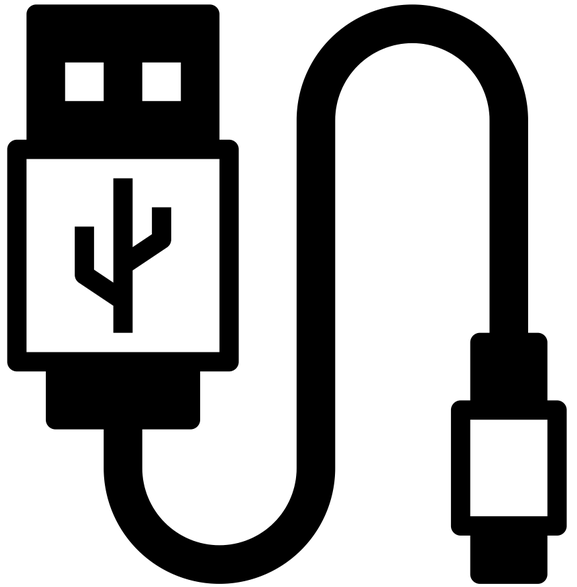
DISPOSITIVO LOCAL

- Ahora, deben acceder a las **Opciones de desarrollador** y buscar la casilla de **Depuración**.

- Les saldrá una casilla de **Depuración por USB** que deben activar. Es probable que les muestre una alerta, simplemente le dan en **Aceptar**.

- **Importante:** Este método es para conectar el equipo por medio de USB, por esa razón ahora deben conectar su celular al pc.

- Una vez lo conecten, les saldrá un mensaje para confirmar que desean permitir la depuración por USB entre ambos dispositivos, le dan en **Permitir**.

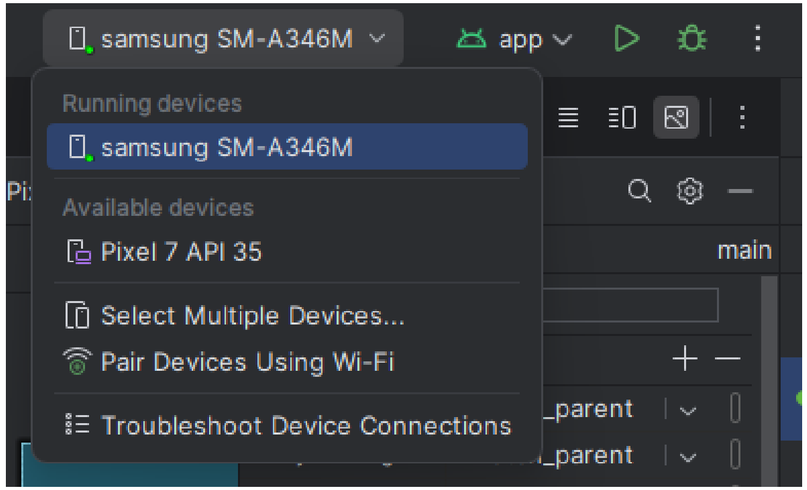


DISPOSITIVO LOCAL

- Ahora que tiene el dispositivo conectado y debidamente configurado. Android Studio lo va a reconocer y les dará la opción de usarlo para correr sus aplicaciones.

- Para ejecutar la aplicación simplemente le dan en **Run** y queda esperar que los equipos hagan su magia.

- **Importante:** Mientras las opciones de desarrollador estén activas en el dispositivo móvil, es probables que las apps de su banco no funcionen. Por eso, la recomendación es solo tener activo el modo desarrollador mientras se esté utilizando.



LAYOUTS

LENGUAJE DE MARCADO EXTENSIBLE (XML)

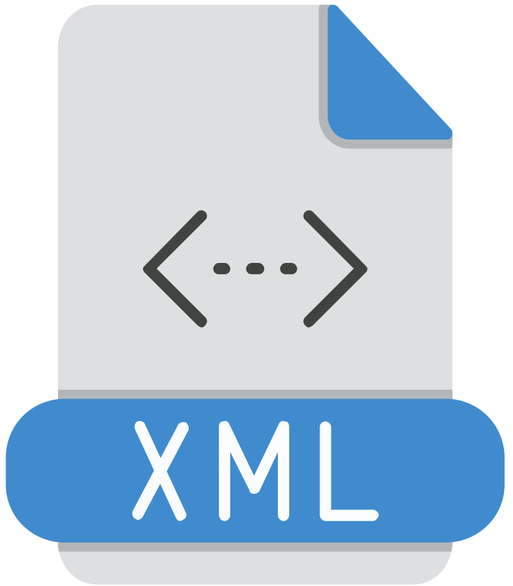
- Cuando nos adentramos en el diseño, vamos a tener a nuestra disposición diferentes estructuras que nos van a permitir “dibujar” en nuestra aplicación.

- Cada una de estas estructuras tiene un nombre en especifico y se encuentra encerrada por flechas.

- **<Etiqueta>** *contenido* **</Etiqueta>**

- También debemos entender que cada etiqueta va a tener diferentes atributos que nos van a permitir darle más detalles al diseño.

- Se diferencia de HTML porque no tiene etiquetas predefinidas.



FRAME LAYOUT

- **android:layout_width** y **android:layout_height** van a definir el ancho y alto de nuestra estructura.

- **match_parent:** Abarca todo el espacio que tenga disponible.

- **wrap_content:** Utiliza solo el espacio que necesite, es decir, depende mucho del contenido que tengamos dentro de nuestro layout.

- **0dp:** Son dimensiones en especificas, en dónde 0 va a corresponder al tamaño que nosotros queramos darle a nuestro objeto.

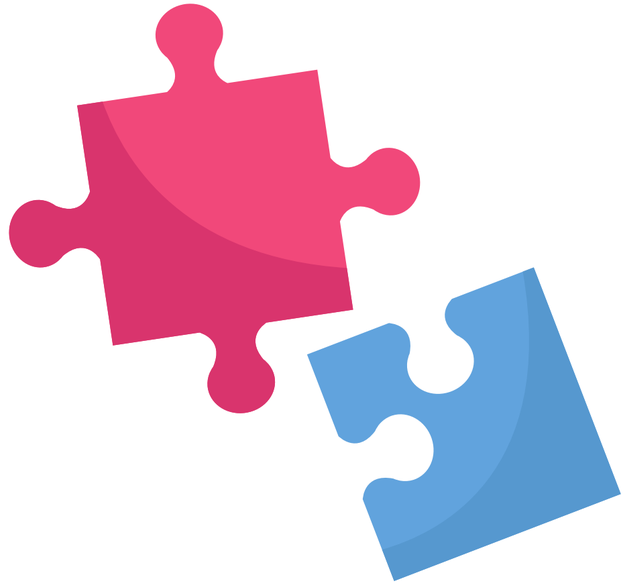
FRAME LAYOUT

- **<View>:** Es una vista como tal. Se utiliza para delimitar un contenido previamente.

- **android:background:** Nos permite modificar el fondo de un objeto.

- Cuando queramos agregar color a un objeto, podemos utilizar código hexadecimal o aprovechar la etiqueta **@color**.

- **android:layout_gravity:** Nos permite posicionar un objeto dentro del contenido padre.
    - **center**: centro.
    - **top**: arriba.
    - **bottom**: abajo.
    - **start**: izquierda.
    - **end**: derecha.



FRAME LAYOUT

- Dentro de **gravity_layout** podemos darle una instrucción doble a un objeto, por ejemplo **bottom|end**, para indicarle de que debe estar abajo del todo y a la derecha del todo.

- **¡Ahora ustedes!**

- Con lo que acabamos de aprender, poseen las herramientas para realizar la imagen de la derecha. ¡Manos al código!

LINEAR LAYOUT

- A diferencia del FrameLayout, **LinearLayout** ordena las estructuras una al lado de otra, por eso cada vez que lo vamos a utilizar, debemos indicar la orientación que va a poseer.

- **<LinearLayout> </LinearLayout>**

- **android:orientation:** Por defecto es horizontal, pero se le puede cambiar.

- **android:weight:** Nos va a permitir organizar nuestras vistas por *“peso”* dentro de un espacio en especifico. Cuando tenemos **weight**, dependiendo si es horizontal o vertical, **width** o **heigth** debe ser **0dp**.

LINEAR LAYOUT

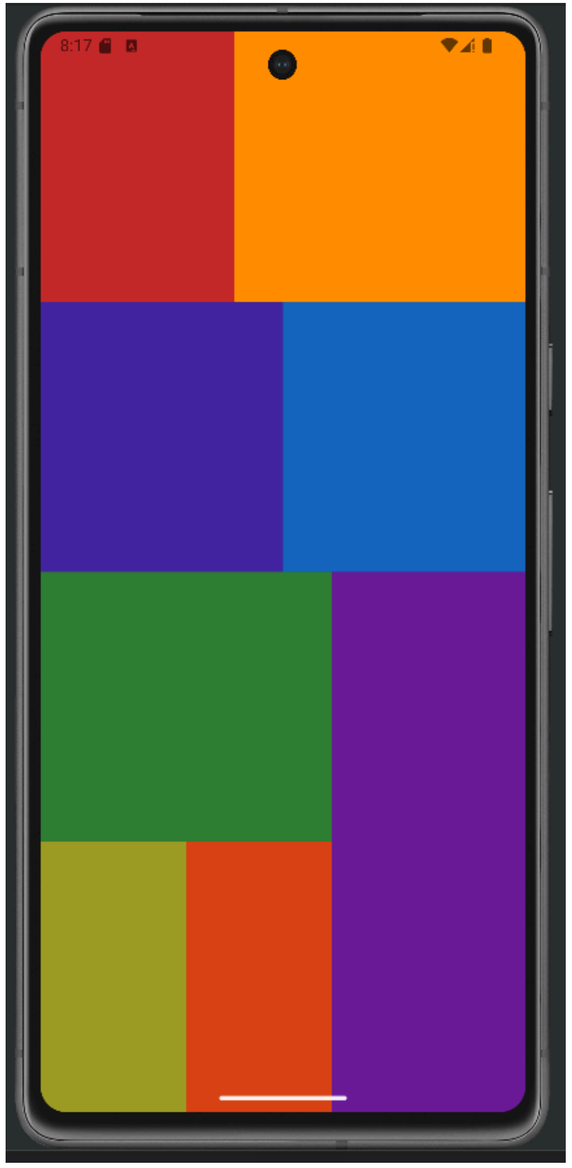
- Es posible **anidar** un LinearLayout dentro de otro, pero se debe tener cuidado con esto, ya que consume más memoria en nuestra aplicación.

- Es más, actualmente anidar LinearLayouts no es una practica recomendada, existen otras estructuras más complejas que pueden hacer un trabajo similar, pero a modo de ejercicio para entender el manejo de la pantalla, cumple con el objetivo.

- **¡Manos al código!**

¡MÁS CÓDIGO!

AWAKELAB


8:17

GRACIAS POR LA ATENCIÓN

AWAKELAB