

Software para Dispositivos Móviles

Miguel Sánchez Santillán
sanchezsmiguel@uniovi.es

Sobre las prácticas

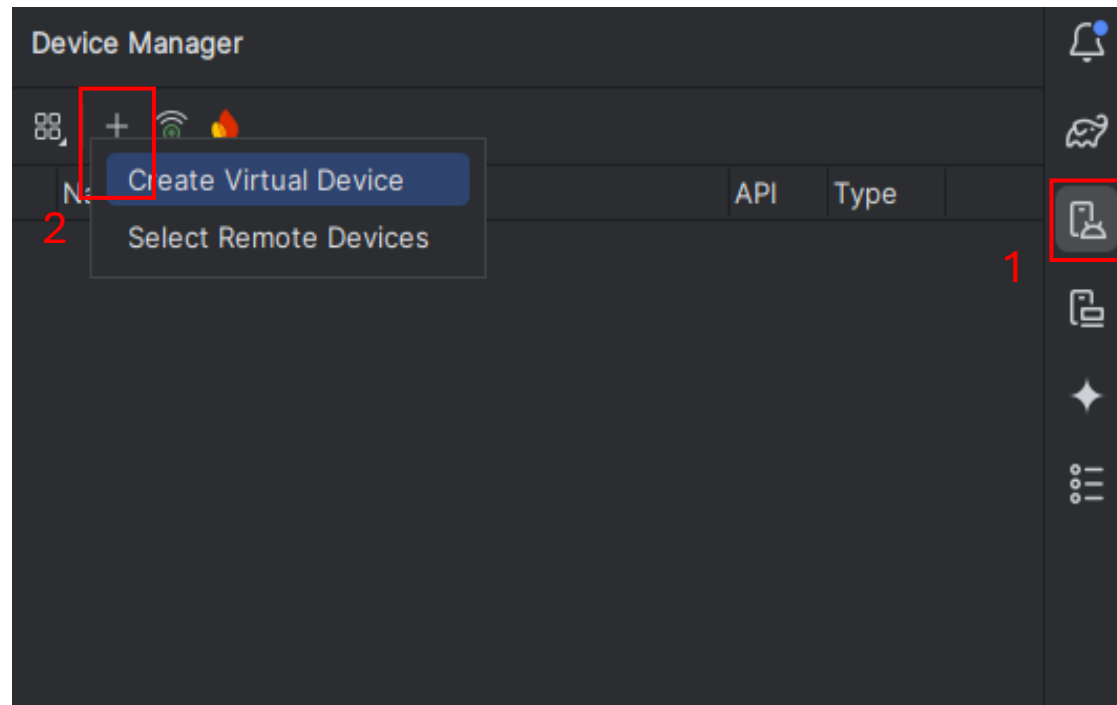
- **Asistencia del 80%**
 - Mínimo 11 sesiones de 13
- **Tutorías**
 - Facultad de Geología, 5ª planta, despacho 5.1
 - Contacto previo: sanchezsmiguel@uniovi.es
- Yo os daré 4 sesiones prácticas de las 13

IDE

- Android Studio (<https://developer.android.com/studio>)
 - Koala Feature Drop | 2024.1.2
- Aplicaciones con **compatibilidad mínima** Android 14 (**API 34**)
 - Upside Down Cake
- Desarrollaremos para Android 15 (**API 35**)
 - Vanilla Ice Cream

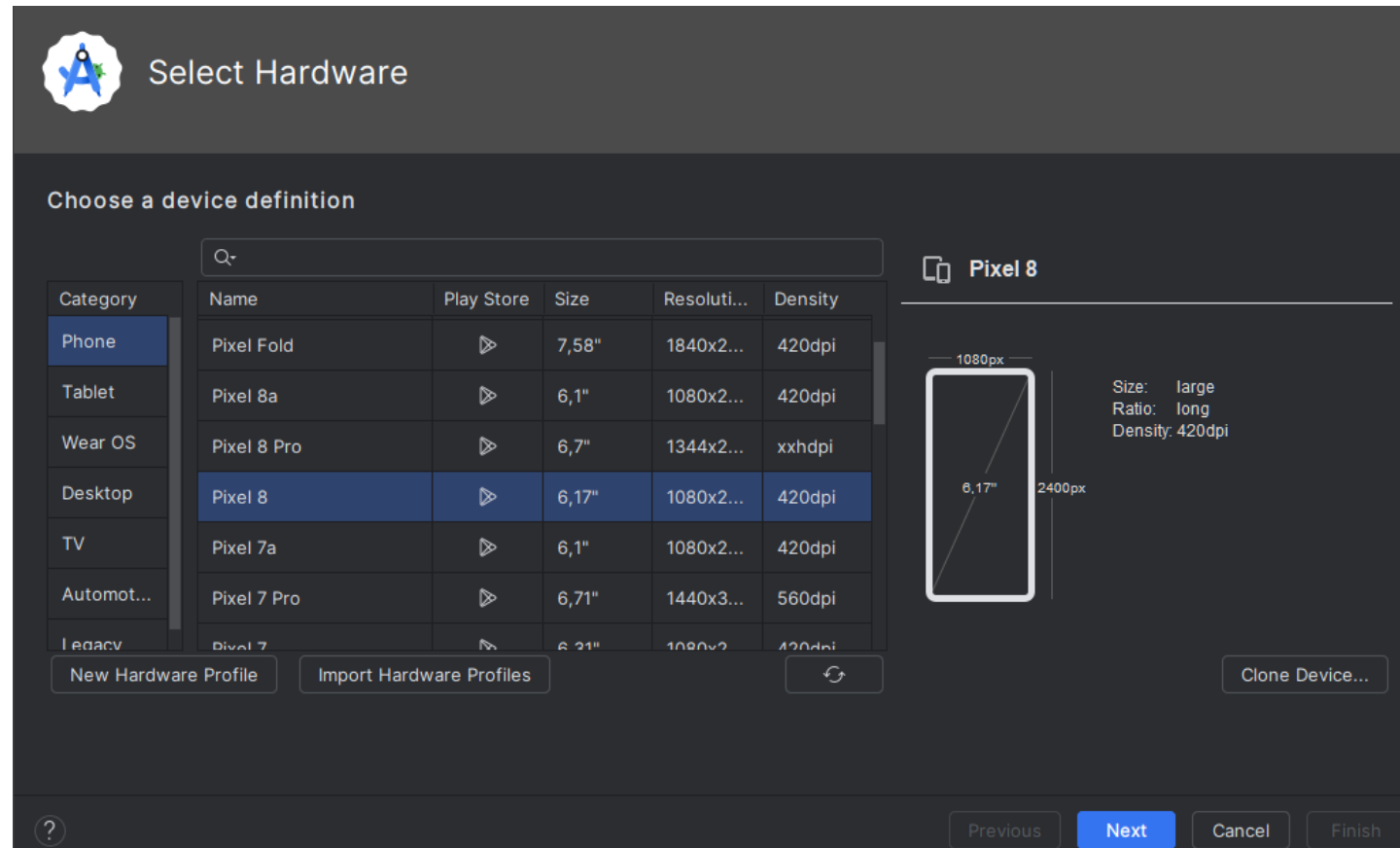
Crear dispositivo emulado I

- Device Manager → Create Virtual Device



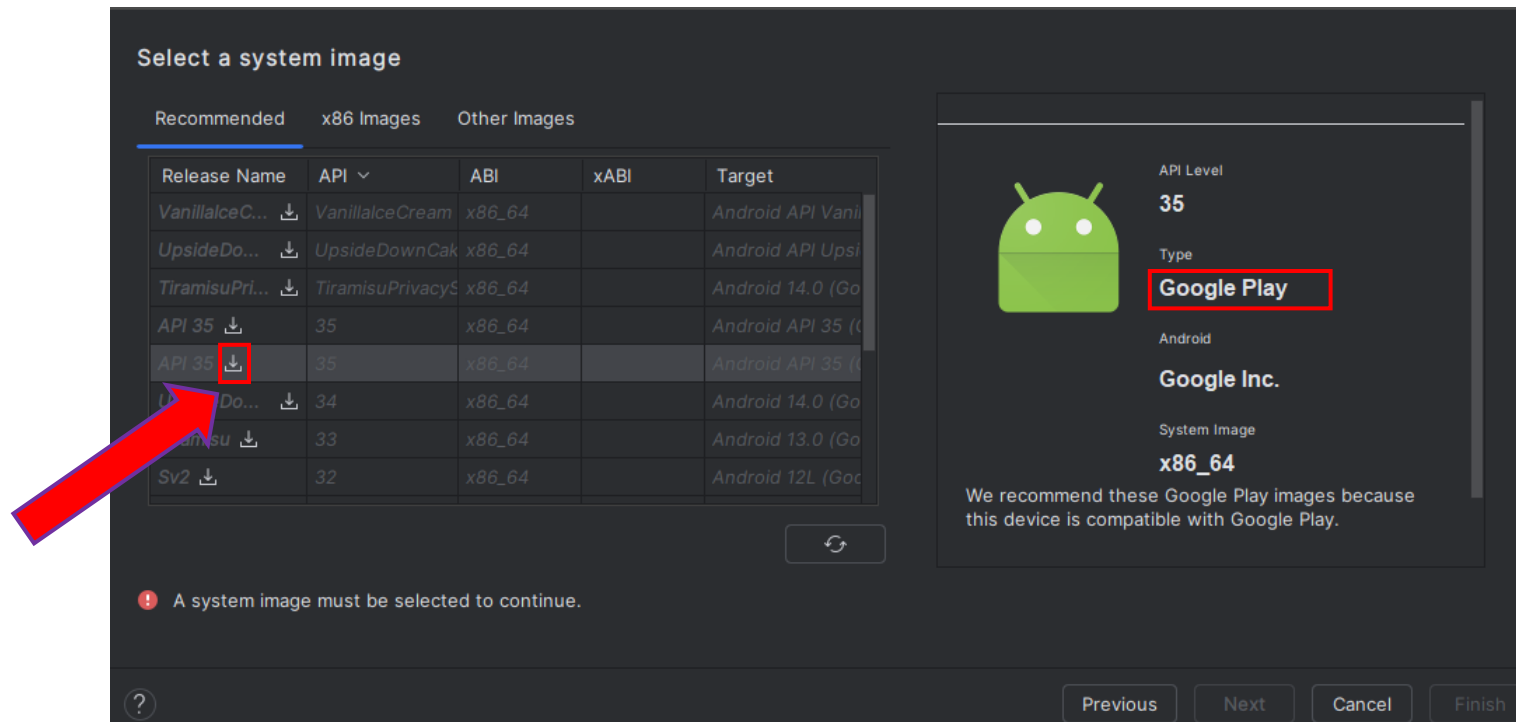
Crear dispositivo emulado II

- Selecciona: **Pixel 8 6,17"** 1080x2400 420dpi.



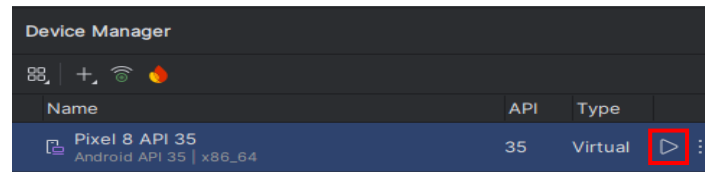
Crear dispositivo emulado III

- Imagen **API 35 (Google Play)**.



Crear dispositivo emulado y IV

- **Arranca** el dispositivo por primera vez



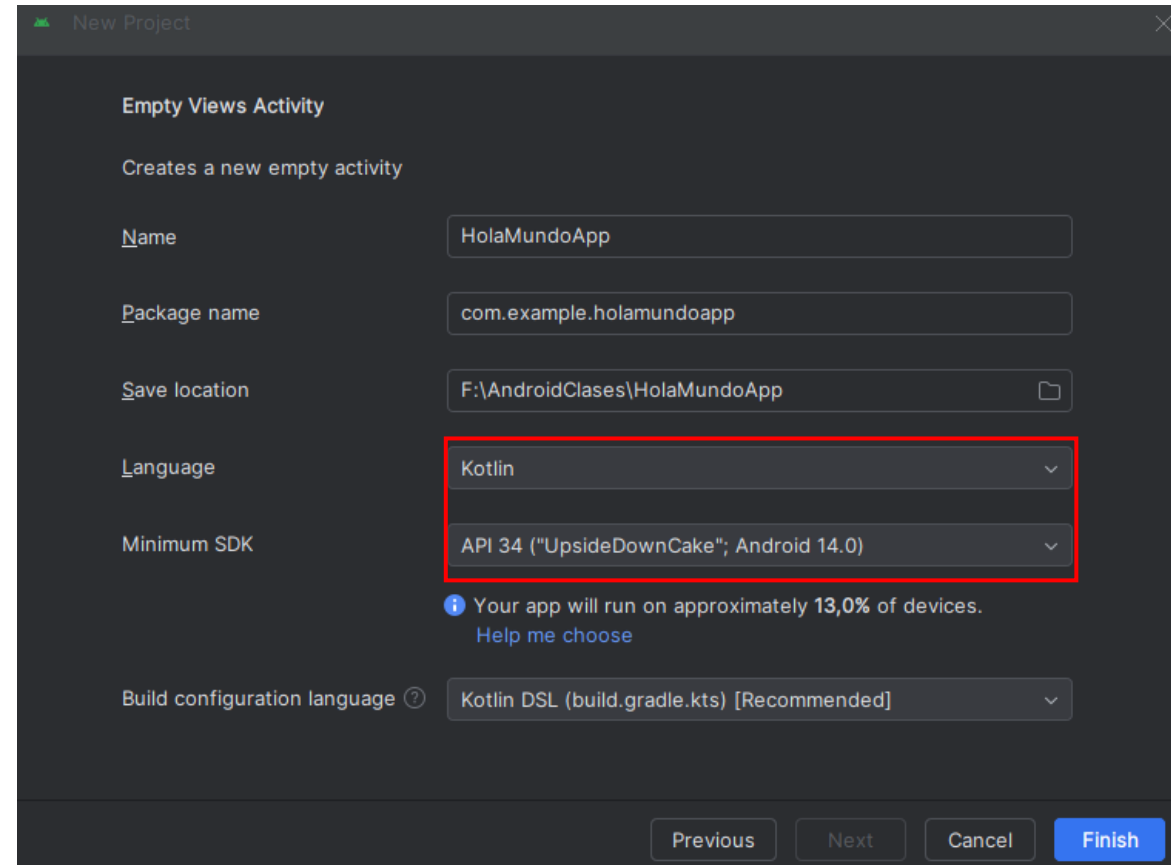
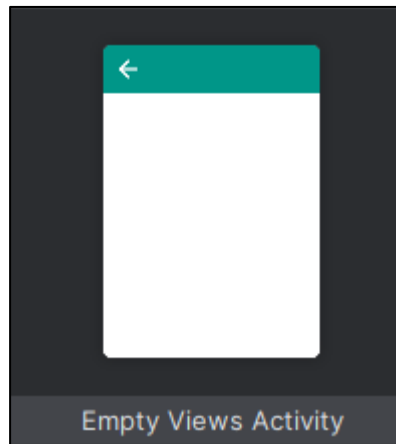
- Accede al dispositivo y **configura lo que consideres**
 - Idioma y teclado, por ejemplo.



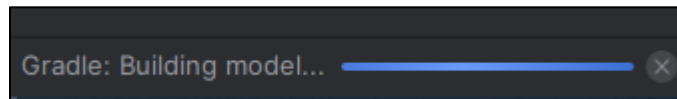
Primer proyecto – *HolaMundoApp*

- File → New Project...

Empty Views Activity

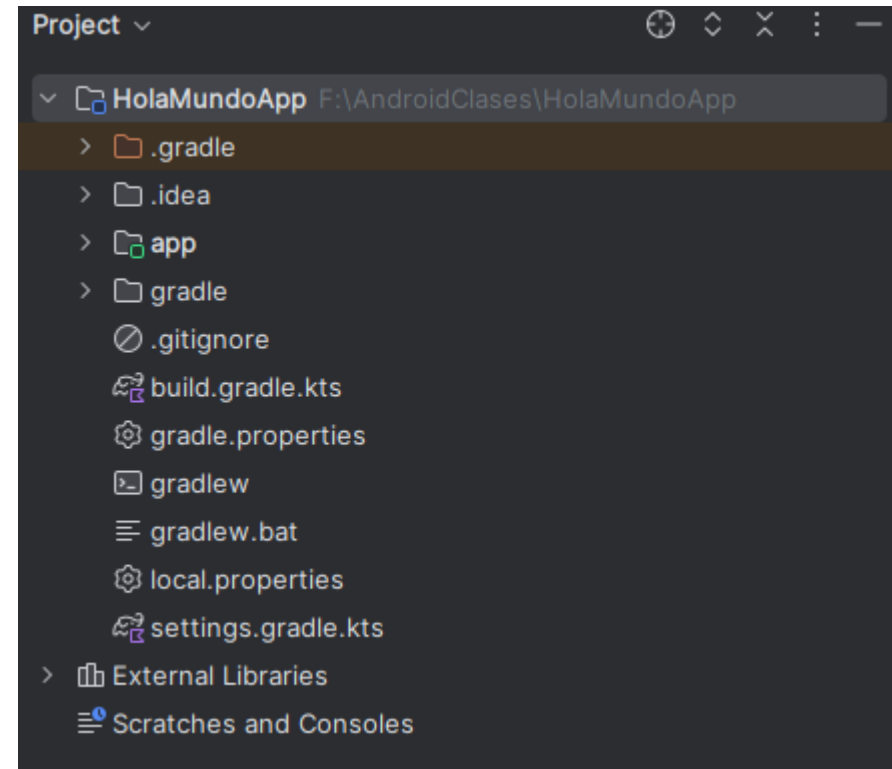
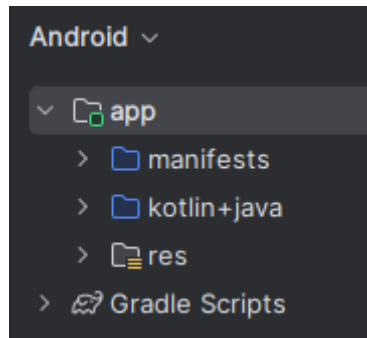
A screenshot of the "New Project" dialog in an IDE. The dialog is titled "New Project" and has a close button (X) in the top right corner. It contains the following fields and options:

- Empty Views Activity**: Subtitle "Creates a new empty activity".
- Name**: Text field containing "HolaMundoApp".
- Package name**: Text field containing "com.example.holamundoapp".
- Save location**: Text field containing "F:\AndroidClases\HolaMundoApp" with a folder icon on the right.
- Language**: Dropdown menu with "Kotlin" selected. This field is highlighted with a red rectangle.
- Minimum SDK**: Dropdown menu with "API 34 ('UpsideDownCake'; Android 14.0)" selected. This field is also highlighted with a red rectangle.
- Information**: A blue information icon followed by the text "Your app will run on approximately 13,0% of devices." and a link "Help me choose".
- Build configuration language**: Dropdown menu with "Kotlin DSL (build.gradle.kts) [Recommended]" selected.
- Buttons**: "Previous", "Next", "Cancel", and "Finish" buttons at the bottom right.

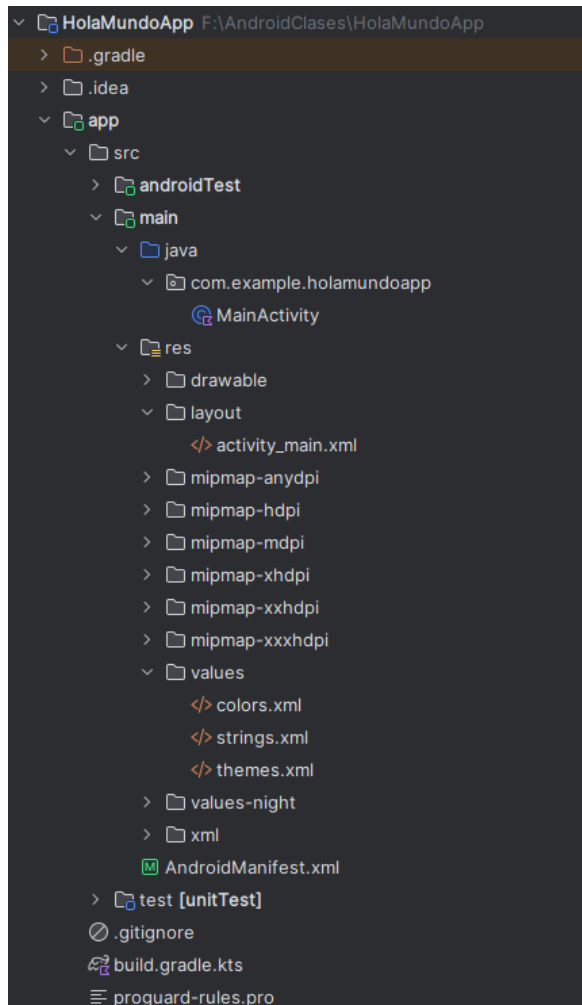


Estructura del proyecto - Vistas

- Vista **Android** (por defecto) VS Vista **Project**



Estructura del proyecto – Qué es qué



- **app/src/main/java** : Código
- **app/src/main/res** : Recursos
 - **layout** → **Vistas de la app** en formato XML
 - **drawable** y **mipmap** → imágenes e iconos **
 - **values** → Textos, colores...
- Ficheros **gradle** → Dependencias y construcción
- **AndroidManifest.xml** → Info fundamental: permisos, temas, **actividades**....


```
<color name="black">#FF000000</color>
```

HolaMundoApp - MainActivity

- Una ***Activity*** representa una **pantalla de la aplicación**.
- Las aplicaciones se componen de **una o varias actividades****.
- **Se definen en el Manifiesto**, estableciendo (entre otros) cuál es la principal.


Vamos a analizar el código de MainActivity

Vistas – activity_main.xml

- **setContentView(R.layout.activity_main)**
 - Ruta a la vista (**R**)recurso.carpeta **layout.activity_main.xml**
- Abrimos el fichero activity_main.xml
 -  Vistas de: código, **código + diseño** y solamente diseño
- Usa un **Layout** (ConstraintLayout) y contiene un **TextView**

Vistas – Dimensiones elementos (y textos)

- En el **TextView** podemos ver cómo se define ancho y alto:
 - **layout_width** (ancho) y **layout_height** (alto).
- Si utilizamos el valor **wrap_content**, ajustará al contenido:
- Si utilizamos el valor **match_parent**, ajusta al contenedor (**hazlo**)
- Un valor específico con la **unidad dp**. Asígnale **150dp x 50dp**
- Para el **tamaño de texto** se utiliza la **unidad sp**



Hello World!

FrameLayout - Simple

- Cambiamos a **FrameLayout**
- Elimina referencias sobrantes
- Asígnale una id al TextView
textViewHola

```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

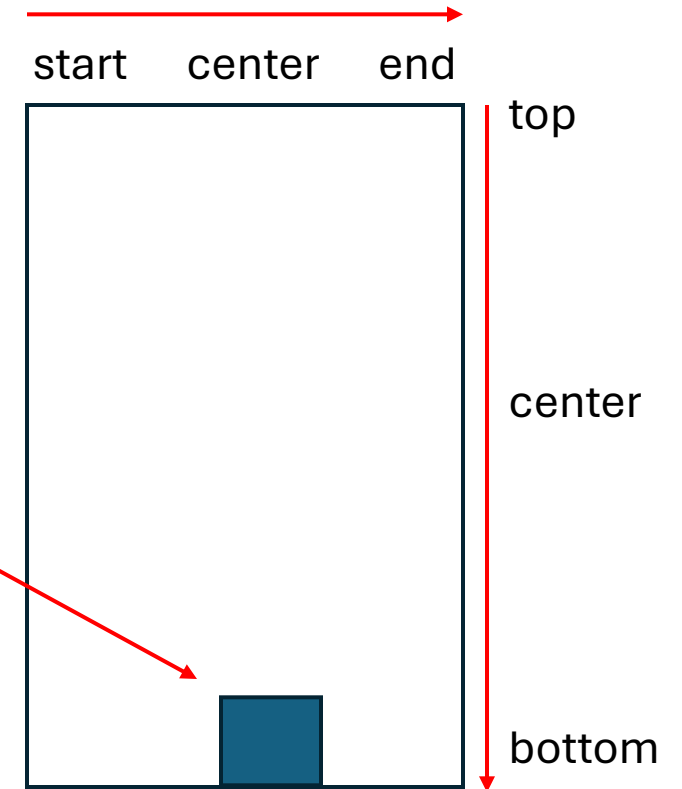
    <TextView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:text="Hello World!"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

</FrameLayout>
```

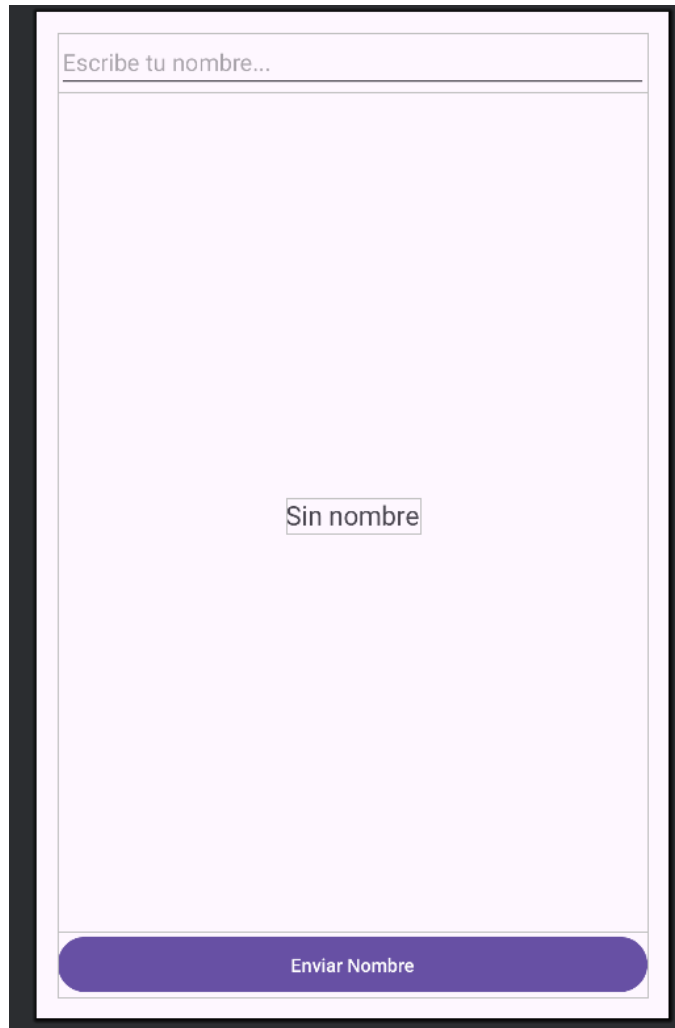
En la próxima práctica trabajaremos con el ConstraintLayout

FrameLayout – Posicionamiento relativo

- Mediante la propiedad android:**layout_gravity**
- Pueden combinarse mediante el símbolo |
 - Ejemplo: `layout_gravity="center|bottom"`
- Podemos distanciar elementos entre sí con las propiedades **layout_margin---** (Top, Bottom...)



FrameLayout – Diseña esta interfaz



- Consta de tres elementos:
 - **EditText**, **TextView** y un **Button**
- Asigna **ids**
- Colócalos en la **posición** adecuada
- Fíjate en los **márgenes**
- Uso **dos propiedades no explicadas**

Logcat - Depuración

- Sacar información mediante el uso de la clase **Log**
 - `Log.d("miapp", "prueba")`
 - `.debug`, `.error`, `.info`, `.warning`, `.verbose`
- **Añade la línea de log anterior al final del método onCreate**

-  Reinicia la app y búscala en el **Logcat** con:



```
package:mine tag:miapp
```

Kotlin – Accediendo a elementos de las vistas

- *lateinit var* vs *val*

- Fíjate en el **findViewById()**

```
class MainActivity : AppCompatActivity() {  
    private lateinit var buttonEnviarNombre : Button  
    /* Declara los otros dos elementos */  
  
    private fun iniciarElementos() {  
        buttonEnviarNombre = findViewById(R.id.buttonEnviarNombre)  
        /* Inicializa los otros dos elementos */  
    }  
}
```

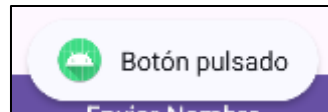
- **Declara e inicializa los otros dos elementos**
- **Invoca el método `iniciarElementos` al final del `onCreate`**
- **Mueve la línea de log al final de este método o añade una nueva.**

Kotlin – Añadiendo un listener al botón

- Uso de expresiones lambda. Fíjate en el subrayado de view

```
buttonEnviarNombre.setOnClickListener(){ view ->  
    Toast.makeText(context: this, text: "Botón pulsado", Toast.LENGTH_SHORT).show()  
}
```

- En este caso estamos mostrando un Toast, pasándole:
 - El **contexto** (la activity), el **texto** a mostrar y la **duración**

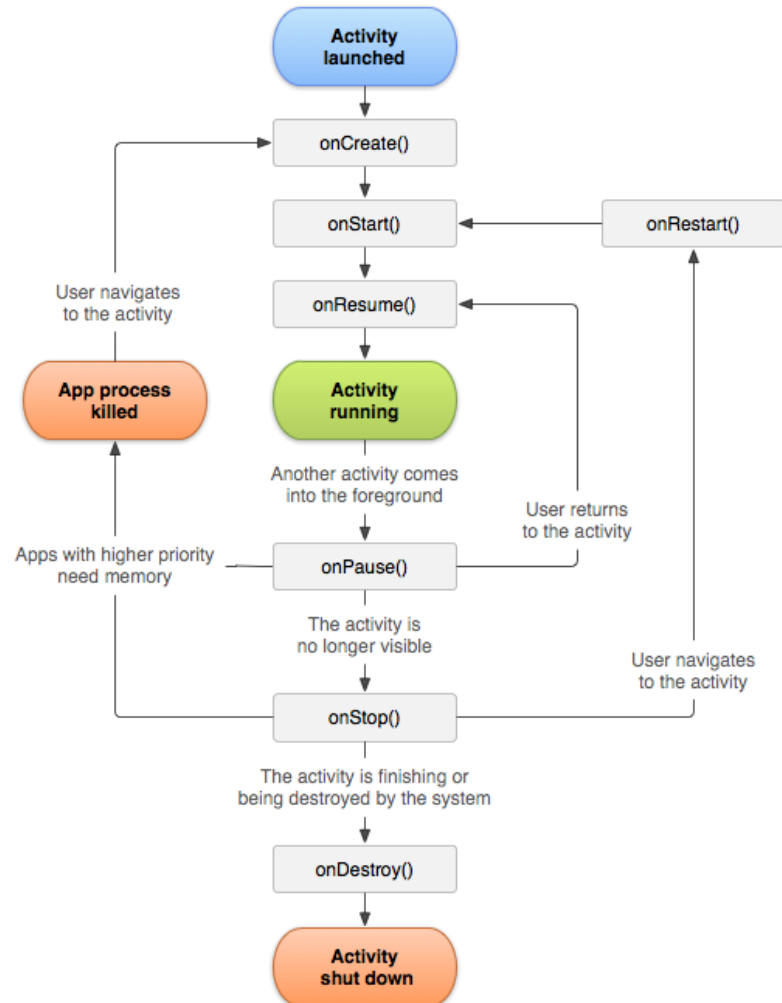


Ejercicio

- Haz que al pulsar el botón se muestre en el TextView el texto “Hola **nombre**” . Siendo nombre el **texto que hay en el EditText**
- **Cambia el texto del EditText a “”**
- Cuando funcione, añade una **comprobación**:
 - Si el tamaño del texto es inferior a 2, muestra un Toast con un error.
- **Mueve el código del listener a un método privado saludar.**

Envía un nombre válido y rota el dispositivo

Ciclo de una Actividad



- **Redefine los métodos que faltan** y haz que se genere un mensaje de log en cada uno de ellos.
- **¿En qué momento deberíamos almacenar la información relevante?**
¿Otros ciclos de vida?
- **No será en esta práctica**

Ejercicio – Acierta la palabra

- Adivinar una palabra de **5 letras** obtenida de una **lista** predefinida
- Se mostrará en pantalla el texto: _ _ _ _ _
- Cuando acierte una letra se pondrá la letra en lugar de _
- Se mostrará el número de intentos y letras utilizadas
- Si pasan más de 5 intentos se pierde la partida (toast con derrota)
- En caso de acertar la palabra, se gana (toast con victoria)
- Se reinicia el juego con nueva palabra y la interfaz reiniciada.