CENTRE ESPECÍFIC
D´EDUCACIÓ A DISTÀNCIA DE
LA COMUNITAT VALENCIANA

PRÁCTICA 2.1

INTRODUCCIÓN AL INTERFAZ GRÁFICO

PROGRAMACIÓN MULTIMEDIA Y DISPOSITIVOS MÓVILES 24/25
CFGS DAM

Autor: Mara Vañó
m.vanoalonso@edu.gva.es
Fecha: 2024/2025

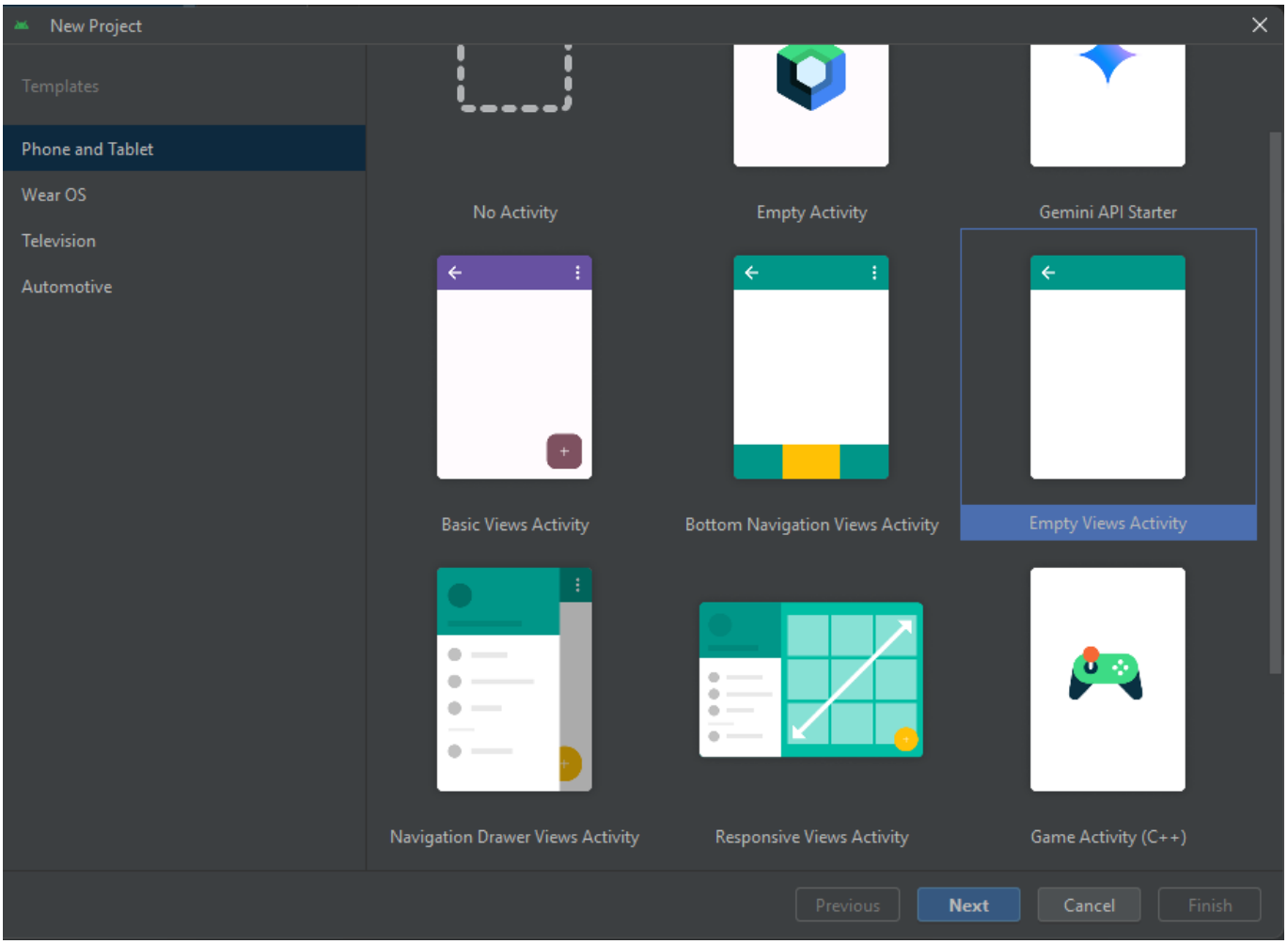
ÍNDICE

1. OBJETIVOS DE LA PRÁCTICA

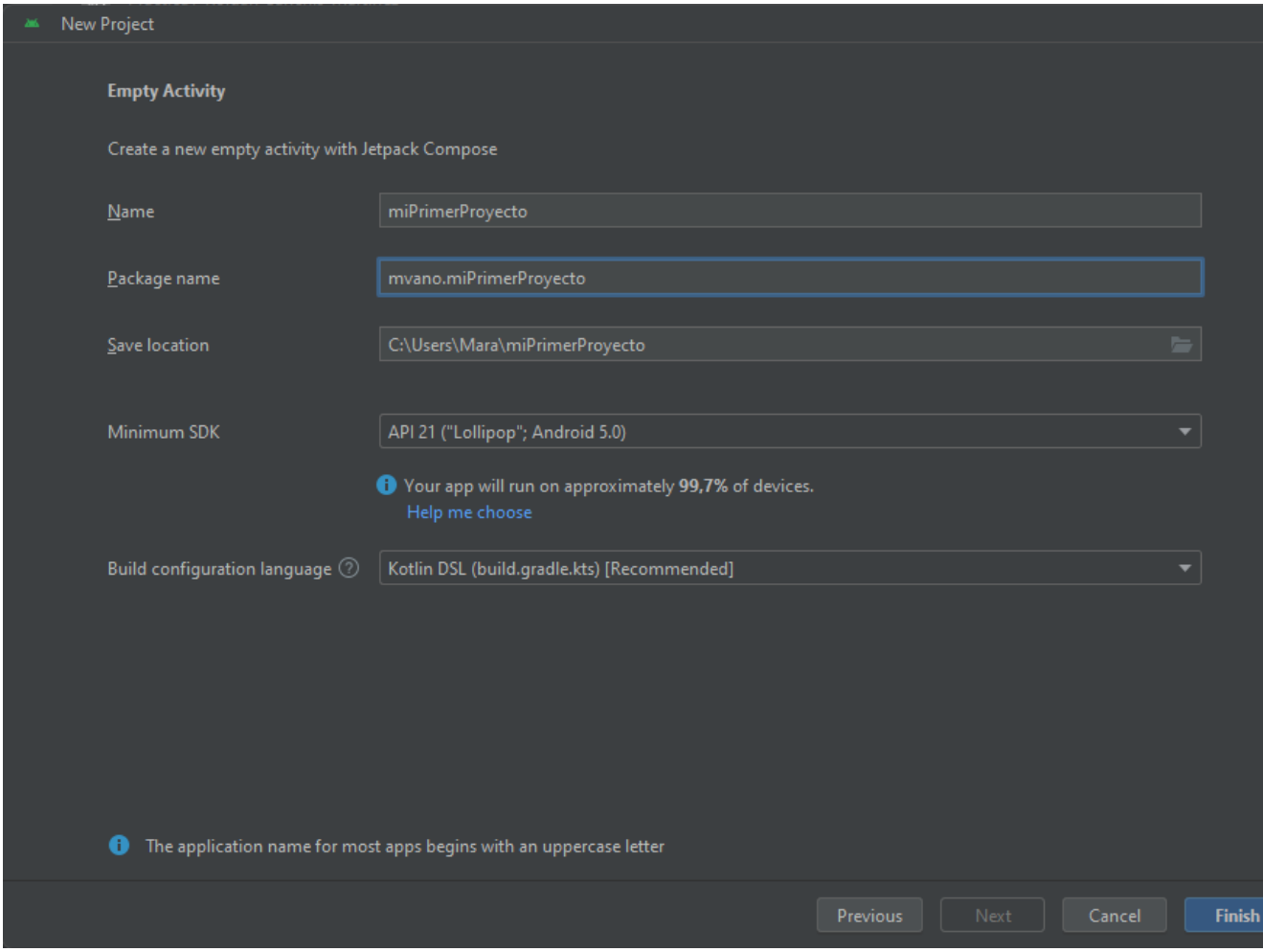
- Aprender a crear nuestra primera interfaz
- Familiarizar con el nuevo entorno de desarrollo
- Asociar funcionalidad a los elementos gráficos
- Realizar ejercicios repasar el funcionamiento de Kotlin

2. EJERCICIO 0. CREAR UN PROYECTO VACÍO.

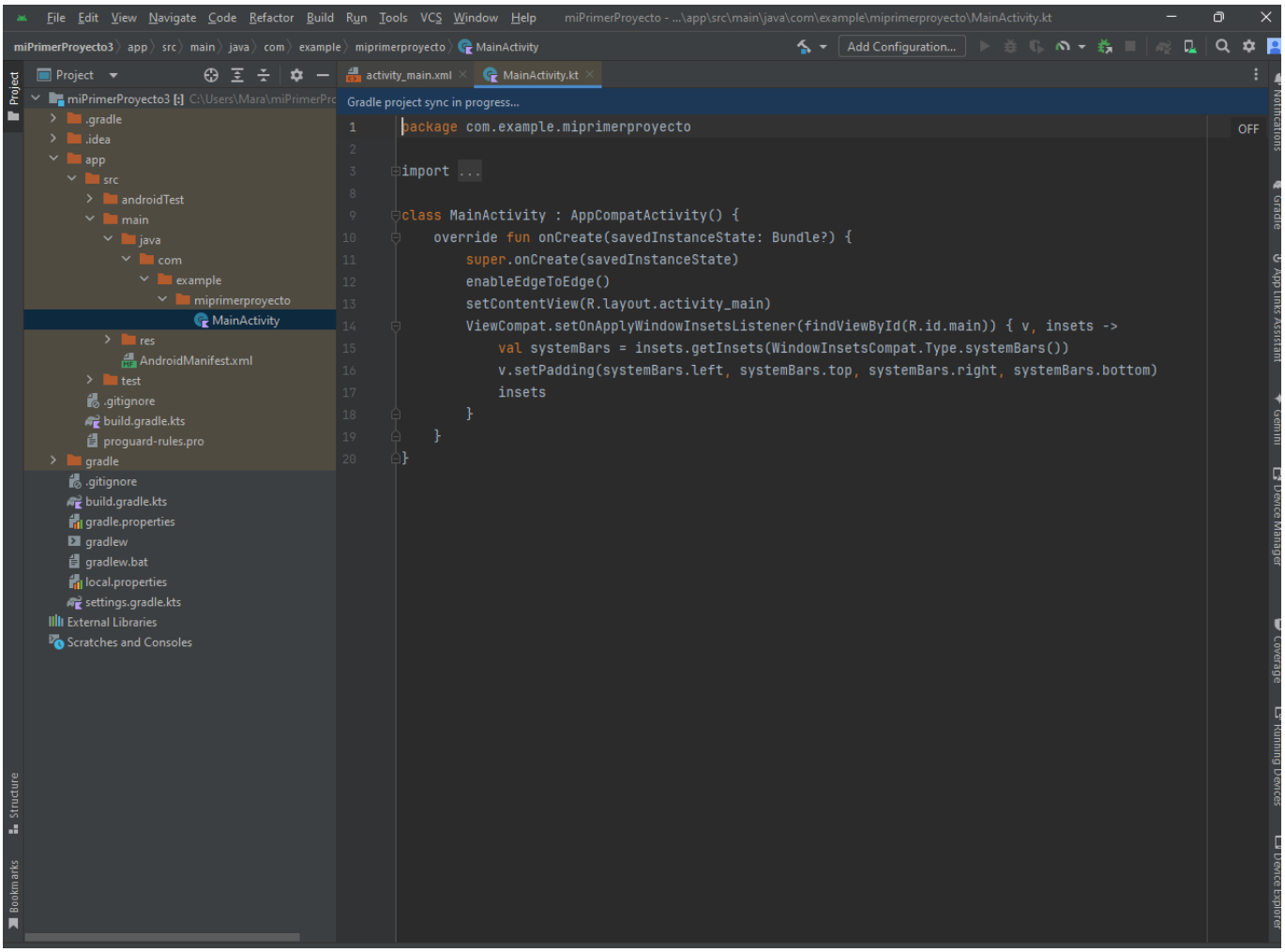
Para crear un proyecto en Android Studio iremos a File → New… → New Proyect y elegiremos Empty Views Activity. Tenemos diferente plantilla predefinidas dependiendo del uso que queramos dar a nuestra aplicación, pero en nuestro caso como estamos aprendiendo desde cero creamos un proyecto vacío.



- En la configuración básica del proyecto sugiero nombrar el proyecto con el nombre de la práctica utilizando la nomenclatura **Camel Case** donde pondremos en minúscula la primera palabra y posteriormente en mayúscula la primera de cada palabra que concatenemos. Ej.: **mi**Primer**P**royecto. Tened en cuenta que Kotlin es un lenguaje de programación que diferencia entre mayúsculas y minúsculas.
- En el apartado **Package name**, pondremos nuestro nombre o el de nuestra organización, será el nombre del directorio donde se incluirán todos los elementos de la aplicación.
- Seleccionamos **Kotlin** como lenguaje preferido para programar.
- Finalmente, en el apartado minimun SDK seleccionaremos la versión **API 21** ya que es soportada por el **97,7%** de los usuarios en la actualidad. A lo largo del módulo, si necesitásemos alguna función muy específica para nuestra aplicación, podremos seleccionar una versión más actual.

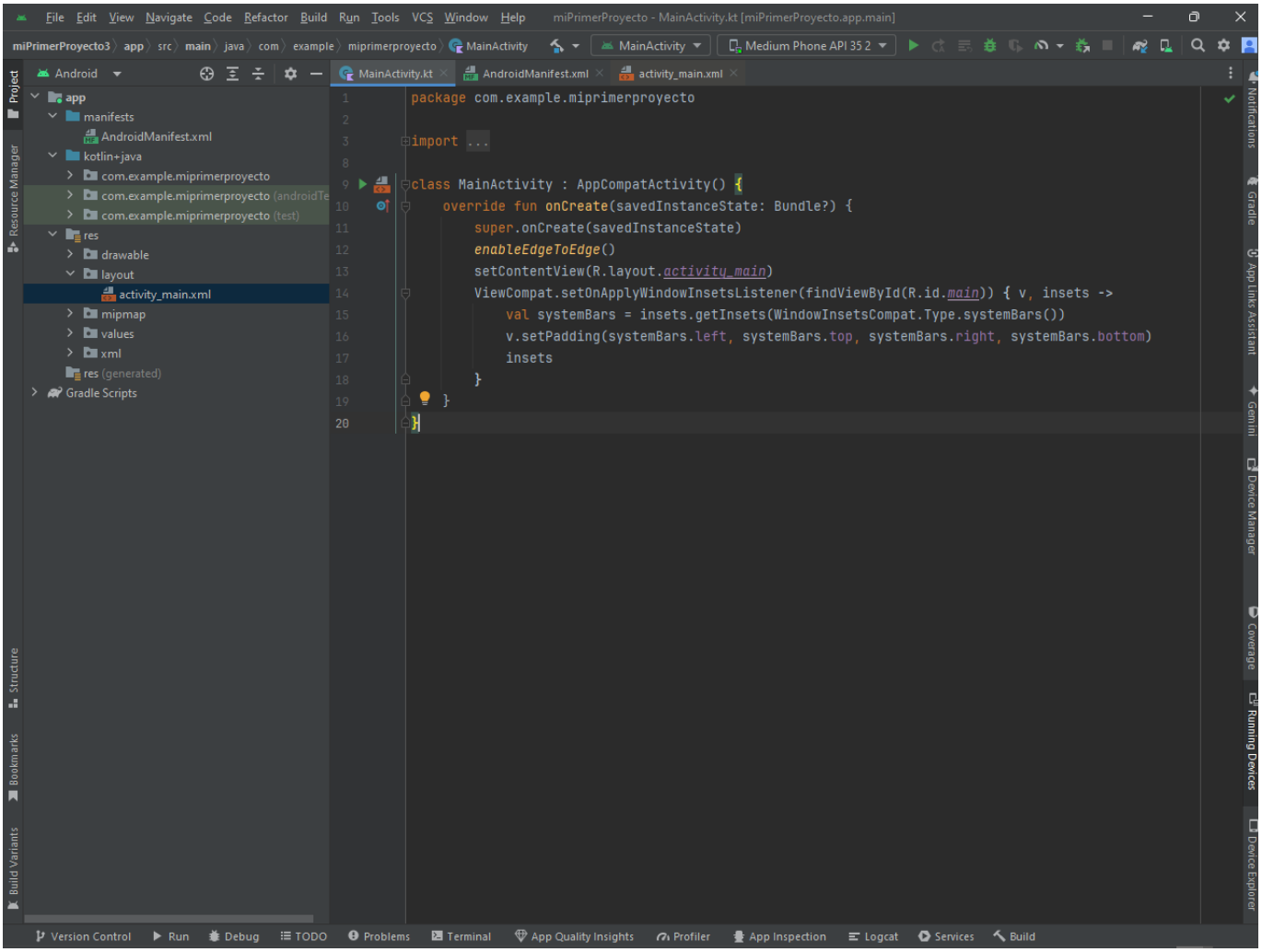


En la parte central de Android Studio encontraremos **Main Activity**; esta es la parte principal de nuestro programa y donde empezaremos a desarrollar nuestra aplicación.



En la pestaña Run nos aparecerá toda la salida del proceso de compilación de nuestro código.

Mi consejo es que tengáis la pantalla de Main Activity y la de Run una al lado de la otra para poder ver por pantalla el resultado de nuestras instrucciones, tal y como aparece en la siguiente imagen:

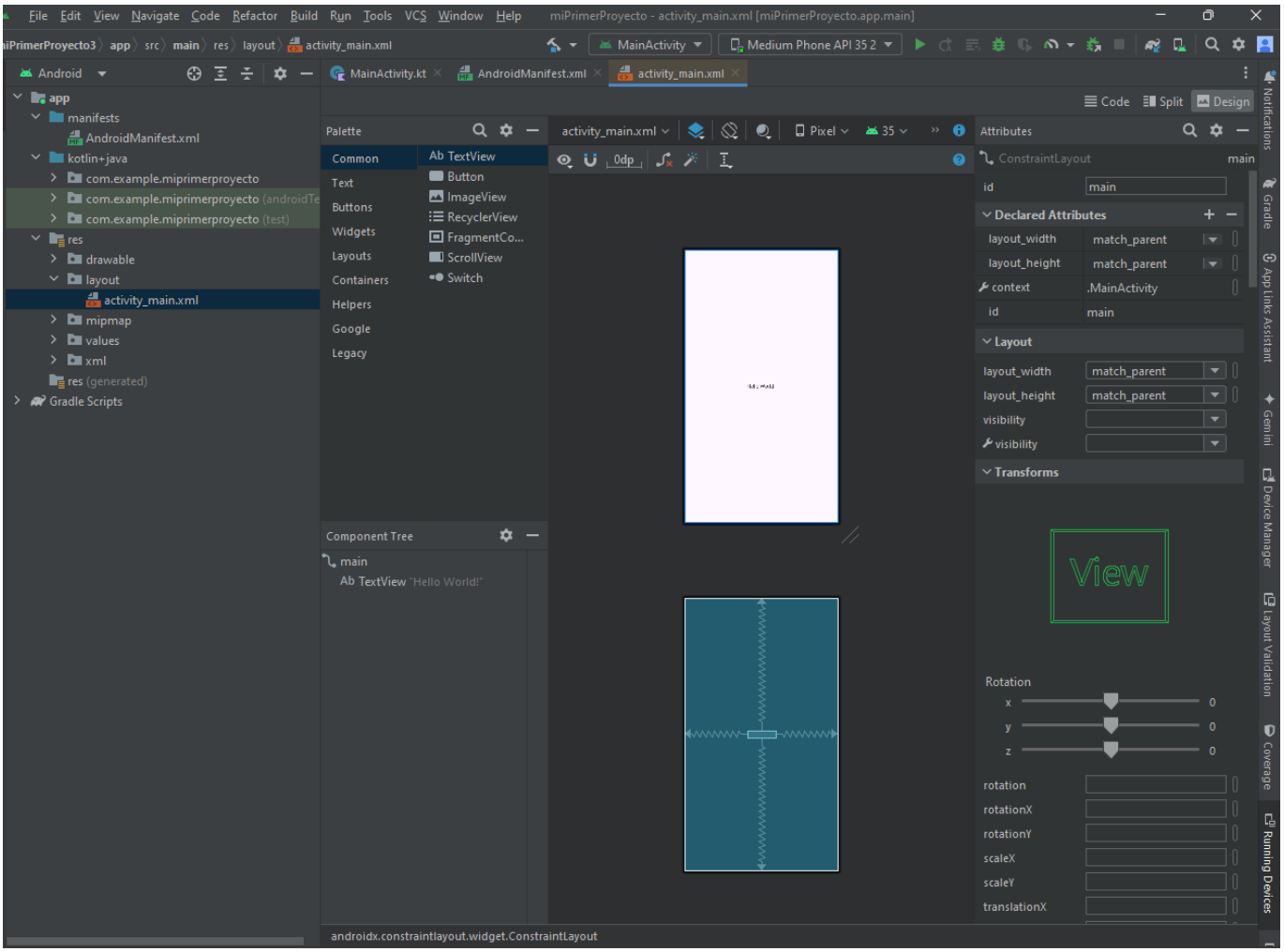


3. EJERCICIO 1. AÑADIR ELEMENTOS EN LAYOUT.

Para el diseño de interfaces Android Studio cuenta con un apartado denomina res, en dicho directorio encontraremos diferentes elementos gráficos.

El elemento principal del directorio res lo podemos encontrar en el directorio de layout. En este directorio encontraremos un archivo .xml con el nombre activity_main donde añadiremos elementos gráficos a la pantalla.

Existen diferentes métodos para añadir elementos, pero en esta parte del tema vamos a utilizar la parte gráfica de creación de interfaces. Utilizando este método el código se escribe automáticamente en el archivo activity_main y podremos editarlo con posterioridad.

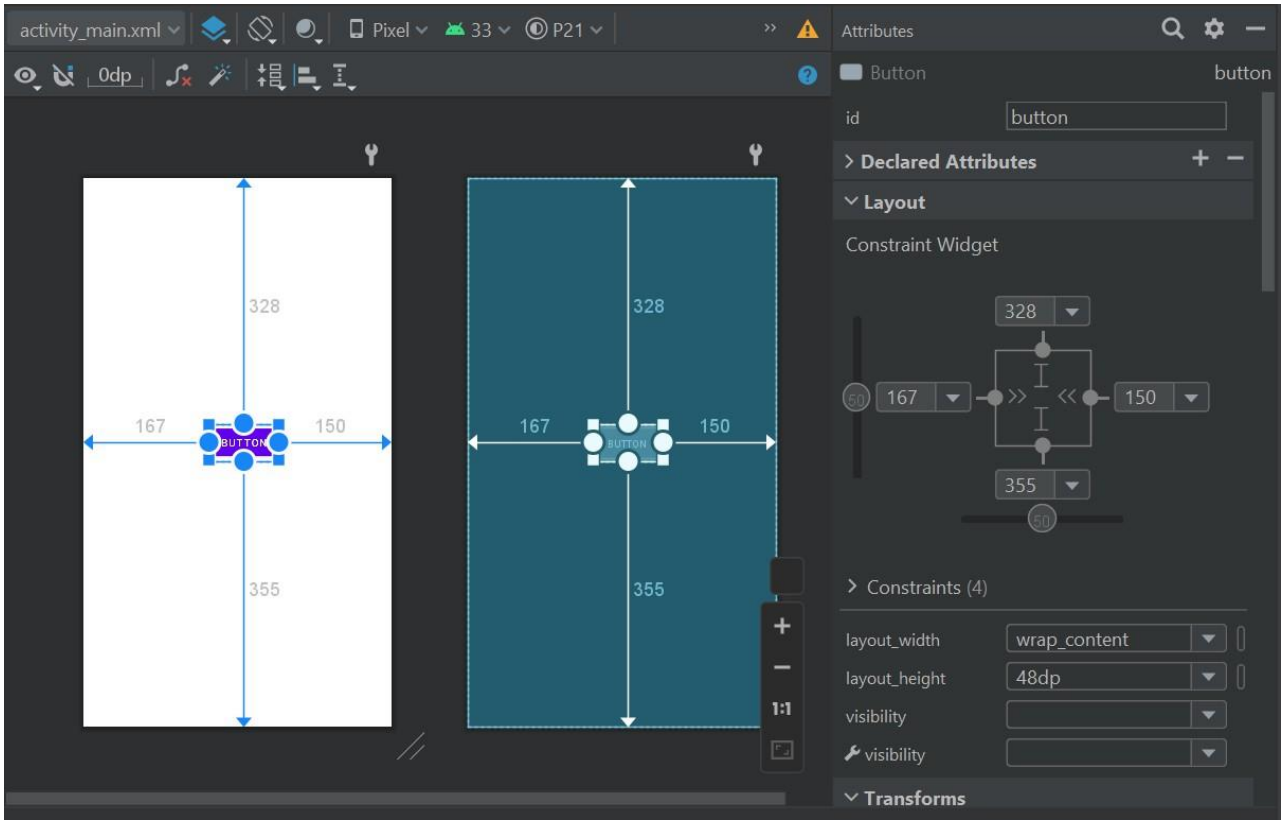


En esta práctica nos centraremos en el apartado de Common donde aparecen de forma resumida los elementos gráficos más utilizados en el desarrollo de aplicaciones móviles.

Para añadir un elemento simplemente selecciona con el ratón y arrastra a la pantalla.

Elemento Button:

En el apartado atributos puedes editar parámetros para adaptar el botón al uso que le queramos dar:



En el apartado de código aparecen los atributos que añadamos de manera gráfica y podremos editarlos también mediante la edición de dicho código utilizando una sintaxis en xml. Por ejemplo, cambiando en el apartado text el texto que aparece dentro del botón.

```
<Button
    android:id="@+id/button"
    android:layout_width="wrap_content"
    android:layout_height="48dp"
    android:layout_marginStart="167dp"
    android:layout_marginTop="328dp"
    android:layout_marginEnd="150dp"
    android:layout_marginBottom="355dp"
    android:text="Button"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent" />
```

Elemento Text:

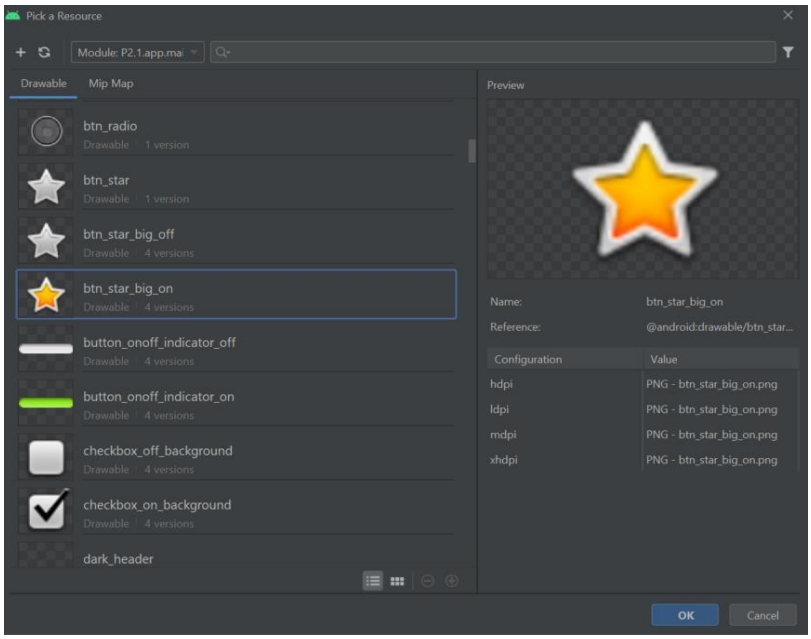
De igual modo podemos añadir el elemento texto.

```
<TextView
    android:id="@+id/textView"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginStart="176dp"
    android:layout_marginTop="196dp"
    android:layout_marginEnd="177dp"
    android:layout_marginBottom="113dp"
    android:text="Práctica 2"
    app:layout_constraintBottom_toTopOf="@+id/button"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="1.0"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintVertical_bias="1.0" />
```

Elemento Imagen:

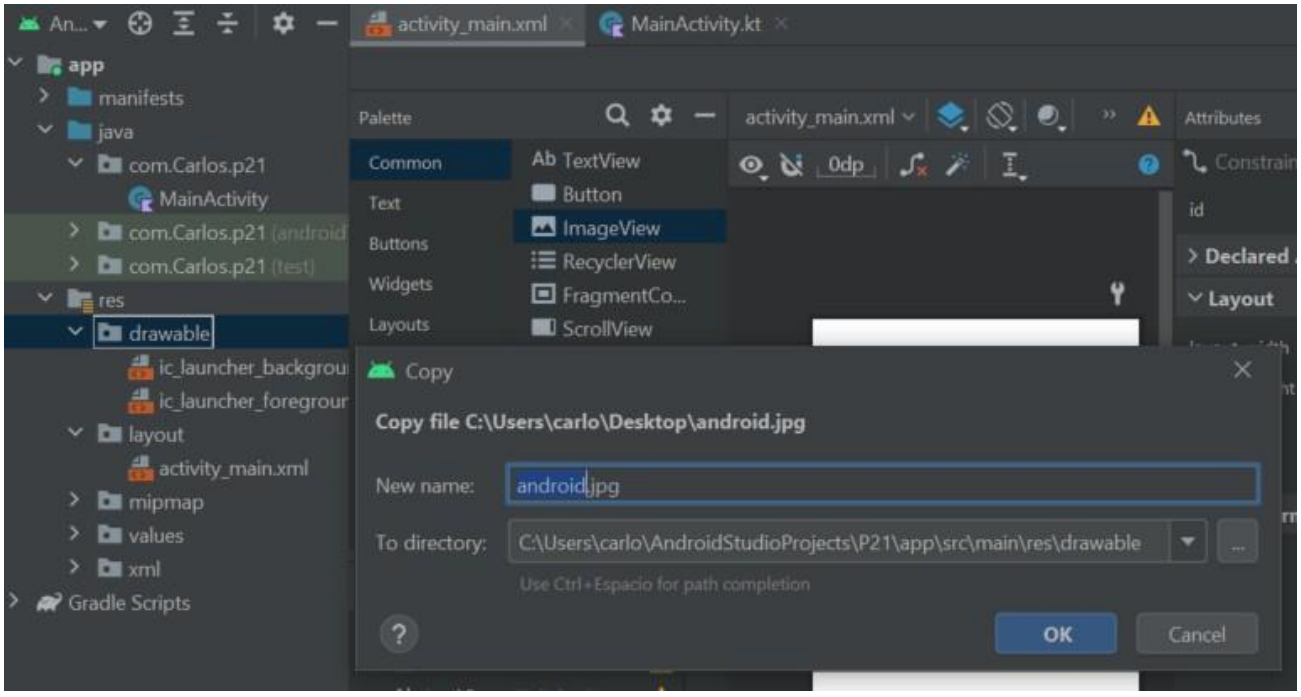
El elemento imagen es un poco peculiar poco debemos seleccionar la imagen que queremos que aparezca. Al añadir una imagen utilizando este método la estamos introduciendo como un elemento propio, es decir, al igual que otros elementos de layout puede moverse por la pantalla y adaptarse al contenido o quedarse fija.

En el momento que añadamos la imagen aparece una pantalla con imágenes que ya nos incluye el propio Android Studio como iconos, flechas o elementos útiles para aplicaciones sencillas.



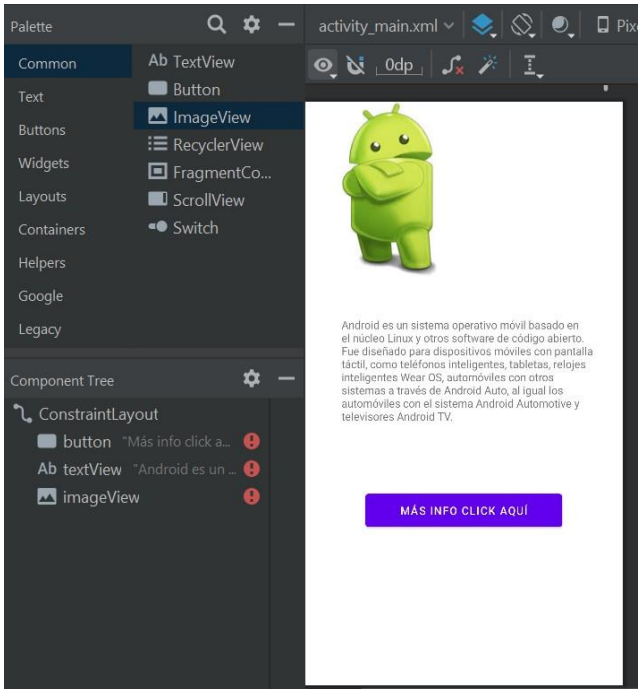
En caso de querer añadir una imagen propia necesitaremos guardarla en nuestro proyecto.

Los elementos gráficos como imágenes deben estar guardados en la carpeta drawable que podemos encontrar en el apartado res. Para añadir una imagen simplemente arrastramos la imagen desde nuestro equipo a dicha carpeta.



3.1. Enunciado Ejercicio 1:

Crea un Layout donde presentes un personaje de actualidad. Incluye una imagen, una pequeña descripción y un botón que diga click aquí para más información.



4. EJERCICIO 2. CAMBIAR FONDO DE PANTALLA.

Para cambiar el fondo de nuestra app debemos añadir una imagen a la carpeta drawable. Después utilizar esta instrucción en el activity_main del layout.

```
android:background="@drawable/fondo">
```

El apartado marcado en amarillo es el nombre del archivo de imagen que hemos subido a la carpeta drawable. Si no lo llamamos igual aparecerá un error.

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity"
    android:background="@drawable/fondo">
```

4.1. Enunciado Ejercicio 2:

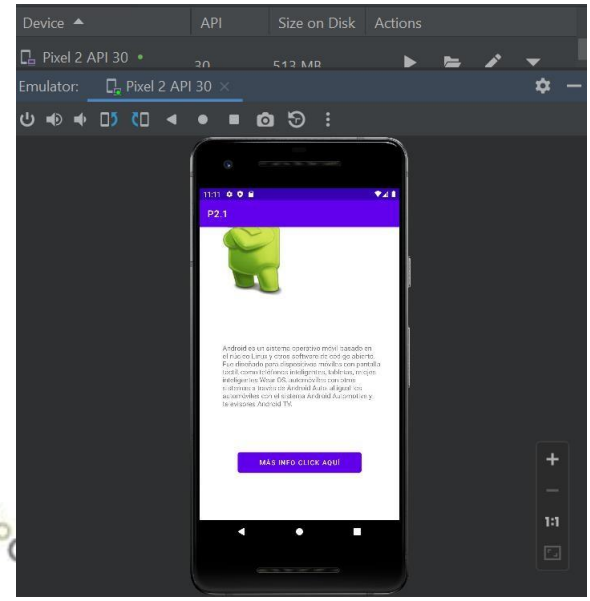
Cambia el fondo de pantalla de tu app por una imagen que más te guste. Te recomiendo utilizar imágenes con colores claros y lisos para poder visualizar el texto correctamente.

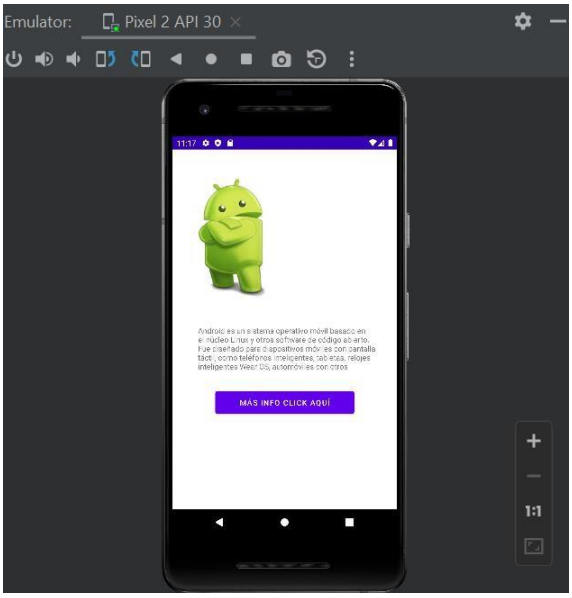
5. EJERCICIO 3. OCULTAR BARRA DE ESTADO DE NUESTRA APP

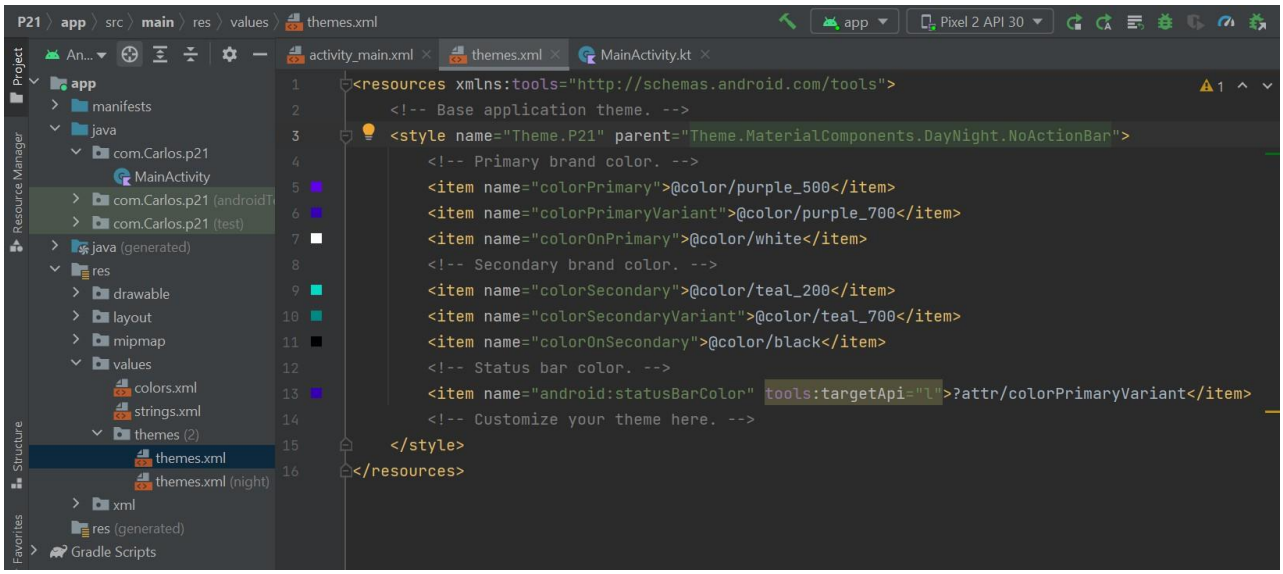
Podemos editar parámetro de la visualización mediante la edición de estilos; dichos estilos se encuentran en la carpeta values-> themes en el directorio res.

Para eliminar la barra de acciones superior debemos simplemente cambiar esta línea del código:

```
<style name="Theme.MyApplication22" parent="Theme.MaterialComponents.DayNight.NoActionBar">
```







5.1. *Enunciado Ejercicio 3:*

Oculta la barra de acciones de tu aplicación.

6. EJERCICIO 4: AÑADIR FUNCIONALIDAD A LOS ELEMENTOS DEL LAYOUT:

Genera la varible lateinit botón y utiliza la función **findViewById** y **setOnClickListener**:

```
package com.Carlos.p21

import ...

class MainActivity : AppCompatActivity() {

    lateinit var boton1: Button

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        boton1 = findViewById(R.id.button)
        boton1.setOnClickListener {  it: View!

            /*Aquí añadiremos las instrucciones
             que queremos que se ejecuten al pulsar el botón
             */

        }

    }
}
```

6.1. Enunciado Ejercicio 4

Crea una aplicación que muestre por pantalla un botón en el que aparezcan las veces que ha sido pulsado.