



# Programacion multimedia y dispositivos móviles

## UT-1. Instalación de Android Studio



# Introducción

Para programar aplicaciones que se ejecuten en SO Android, lo primero que vamos a necesitar es aprender Android Studio e instalar lo que se conoce como entorno de desarrollo.

El entorno de desarrollo es un software que tiene todas las herramientas necesarias para programar tus aplicaciones, y en el caso de Android, Android Studio es la herramienta oficial (y la más extendida en el mercado).

Instalar Android Studio es muy sencillo. Tan solo tienes que ir a la página de descarga oficial de Android Studio y podrás ver un botón verde para descargar tu entorno de desarrollo (había una versión portable sin instalación, pero ahora parece que no está)



# Introducción

El entorno de desarrollo de Android cuenta con un editor visual que nos dejará ver como va quedando nuestra aplicación.

Un analizador que mostrará que tan optimizada está quedando nuestra aplicación.

Un emulador que nos permitirá ejecutar y probar nuestra aplicación desde nuestra pc. Gracias a esta herramienta no tenemos que ejecutar nuestra aplicación directamente desde nuestro teléfono a la hora de hacer pruebas, sino que simulamos un dispositivo virtual. Además ponemos emular dispositivos con diferente hardware.

Por último el editor de código que nos ayuda a agilizar el desarrollo. Estas no son las únicas herramientas que hay, existen muchas otras que se van aprendiendo a manejar en medida que se va usando.



# Instalación

Descargamos Android Studio de su página oficial: <https://developer.android.com/studio>

Es más sencillo hacerlo desde el fichero .exe que va guiando la instalación.

Si la máquina no tiene instalado el JDK, es necesario instalarlo..

La instalación de Android Studio se encarga de instalar el SDK de Android.

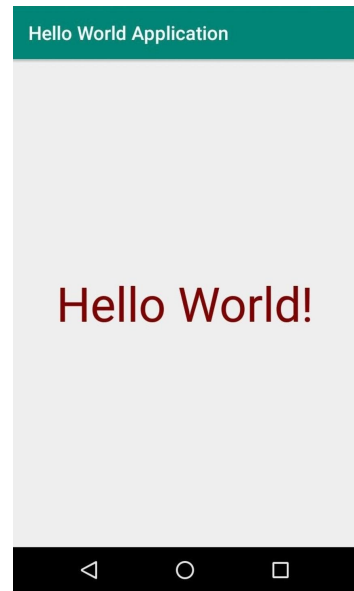
Junto con Android Studio instalaremos el dispositivo virtual de Android (permite probar con emulador)

Si nos pregunta de si queremos plugin para Kotlin, respondemos que si.



# Aplicación Android "Hola mundo"

Vamos a familiarizarnos con el entorno creando una aplicación simple





# Aplicación Android "Hola mundo"

Según la documentación proporcionada por Android Developers: una Activity es un componente de la aplicación que contiene una pantalla con la que los usuarios pueden interactuar para realizar una acción, como marcar un número telefónico, tomar una foto, enviar un correo electrónico o ver un mapa. A cada actividad se le asigna una ventana en la que se puede dibujar su interfaz de usuario. La ventana generalmente abarca toda la pantalla, pero en ocasiones puede ser más pequeña que esta y quedar "flotando" encima de otras ventanas.

Una aplicación generalmente consiste en múltiples actividades vinculadas de forma flexible entre sí. Normalmente, una actividad en una aplicación se especifica como la actividad "principal" que se presenta al usuario cuando este inicia la aplicación por primera vez. Cada actividad puede a su vez iniciar otra actividad para poder realizar diferentes acciones. Cada vez que se inicia una actividad nueva, se detiene la actividad anterior, pero el sistema conserva la actividad en una pila (la "pila de actividades"). Cuando se inicia una actividad nueva, se la incluye en la pila de actividades y capta el foco del usuario. La pila de actividades cumple con el mecanismo de pila "el último en entrar es el primero en salir", por lo que, cuando el usuario termina de interactuar con la actividad actual y presiona el botón Atrás, se quita de la pila (y se destruye) y se reanuda la actividad anterior.



## Aplicación Android "Hola mundo"

Así pues, de forma un poco simple, podremos asociar el concepto de Activity al de pantalla o ventana de la aplicación, de forma que todas las diferentes pantallas que pudiéramos ver en una aplicación tendrán asociadas una actividad.

Las actividades estarán compuestas por una parte lógica y una parte gráfica.

La parte lógica se corresponderá con un archivo `.kt`, que contiene la clase para poder interactuar desde esa actividad, a través de los diferentes métodos de que disponga, e incluso apoyándose en otros ficheros `.kt` que contengan clases que también interactúen entre sí.

Por otro lado, la parte gráfica normalmente se corresponderá con un fichero `.xml` en el que se define la disposición (layout) de los elementos que se mostrarán. Así, en dicho fichero, mediante etiquetas XML específicas, se podrán incorporar todos aquellos elementos que queramos que se muestren en mi interfaz.



# Aplicación Android "Hola mundo"

Vamos a crear un nuevo proyecto. Lo primero que nos pregunta Android es como vamos a diseñar nuestras actividades (las activitys es como llama Android a las pantallas de móvil)

Hasta hace un par de años todas las activitys se creaban con Views, vistas. Ya veremos lo que es.

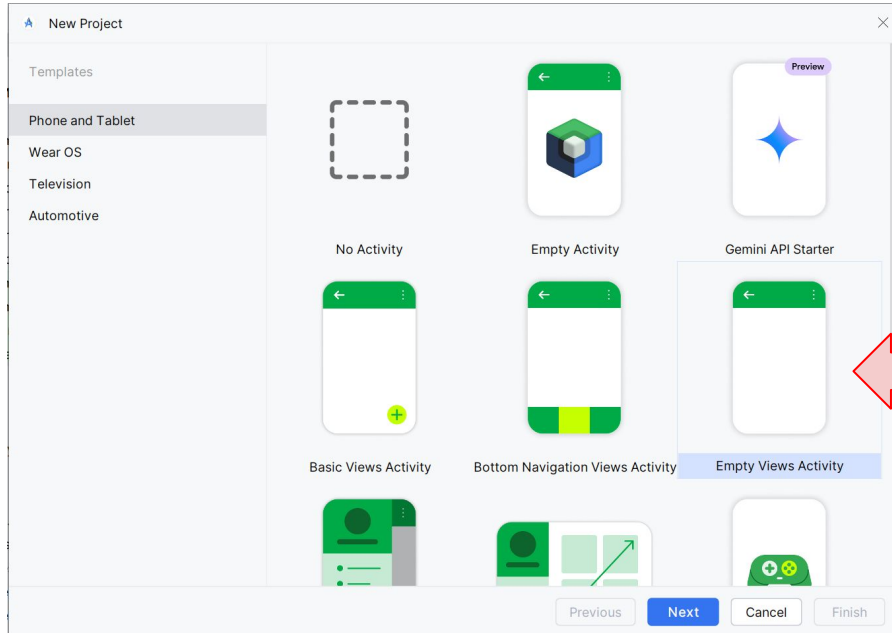
Hace un par de años Google introdujo una nueva manera de hacer las activitys, [compose](#).

Vamos a empezar creando apps con vistas y, espero, que en algún momento veamos también las actividades con compose.

Siempre escogeremos la actividad vacía (empty activity). El resto suelen traer mucho código ya hecho que hay que conocer y entender. En este punto, no nos merece la pena.



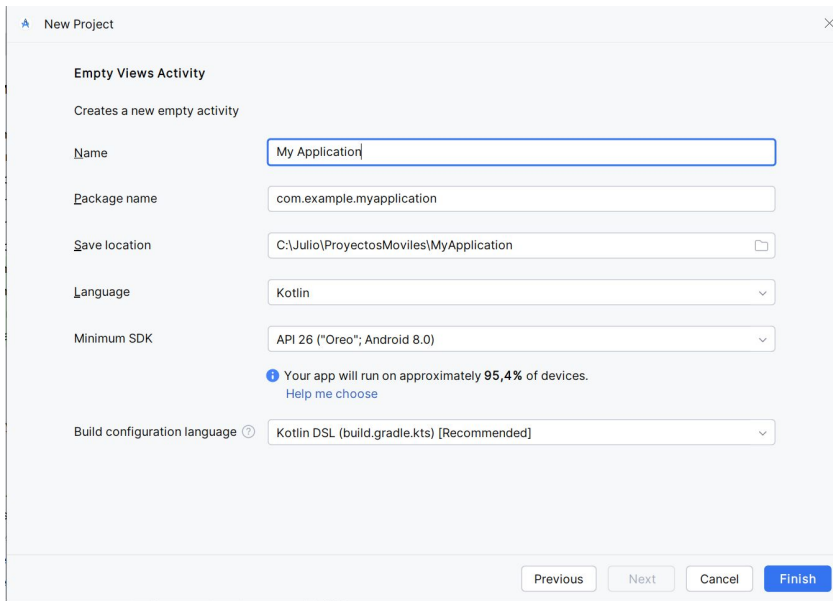
# Aplicación Android "Hola mundo"





# Configuración del proyecto "Hola mundo"

Terminaremos de crear el proyecto configurando algunos detalles sobre su nombre, el paquete “padre” donde se van a alojar las clases u otros paquetes, el lenguaje con el que vamos a programar la app, la ubicación y qué versión de API utiliza.



The screenshot shows the 'New Project' dialog box in Android Studio. 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**: A section header.
- Creates a new empty activity**: A description of the project type.
- Name**: A text input field containing 'My Application'.
- Package name**: A text input field containing 'com.example.myapplication'.
- Save location**: A text input field containing 'C:\Julio\ProyectosMoviles\MyApplication' with a folder icon on the right.
- Language**: A dropdown menu set to 'Kotlin'.
- Minimum SDK**: A dropdown menu set to 'API 26 ("Oreo"; Android 8.0)'.
- Build configuration language**: A dropdown menu set to 'Kotlin DSL (build.gradle.kts) [Recommended]'.

Below the 'Minimum SDK' field, there is a blue information icon and the text: 'Your app will run on approximately 95,4% of devices. [Help me choose](#)'.

At the bottom right, there are four buttons: 'Previous', 'Next', 'Cancel', and 'Finish'.

# Configuración del proyecto "Hola mundo"

La elección de la versión de API debe buscar ser relativamente actual y llegar a un número de usuarios. Android Studio te ayuda a elegir ("Help me choose").

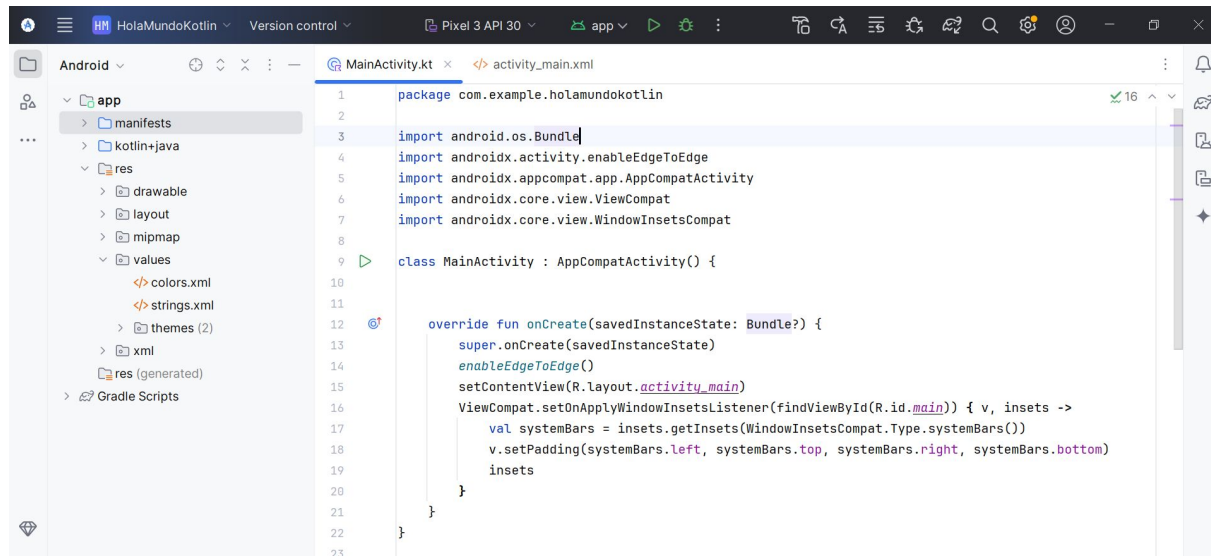
Además a la hora de hacer las pruebas necesitaremos configurar un emulador de un terminal móvil con un SO Android. Nosotros podremos instalar en un SO cuya versión de API sea la 30 una app que se haya desarrollado con un API versión 26 por ejemplo, pero NO al revés. Si elegimos para desarrollar el API versión 30 o 31 deberemos utilizar un emulador con SO mínimo versión 30 o 31.

Yo, para mis aplicaciones, voy a elegir generalmente versiones 6 u 8 del API.

Android Platform/API Version Distribution		
ANDROID PLATFORM VERSION	API LEVEL	CUMULATIVE DISTRIBUTION
4.1 Jelly Bean	16	
4.2 Jelly Bean	17	99,9%
4.3 Jelly Bean	18	99,7%
4.4 KitKat	19	99,7%
5.0 Lollipop	21	98,8%
5.1 Lollipop	22	98,4%
6.0 Marshmallow	23	96,2%
7.0 Nougat	24	92,7%
7.1 Nougat	25	90,4%
8.0 Oreo	26	88,2%
8.1 Oreo	27	85,2%
9.0 Pie	28	77,3%
10. Q	29	62,8%
11. R	30	40,5%
12. S	31	13,5%

# Archivos de un proyecto de Android Studio

Cada vez que comienzas un proyecto nuevo, Android Studio crea la estructura de carpetas y archivos necesarios para esa aplicación. Echemos un vistazo a los diferentes archivos involucrados en un proyecto de aplicación de Android





# Archivos de un proyecto de Android Studio

## La carpeta manifests

La carpeta manifests contiene el archivo `AndroidManifest.xml`. El archivo manifiesto describe información esencial sobre tu aplicación (encoding, versión de Android soportada, icono,...). Por ahora no será necesario modificar este fichero, pero más adelante si tendremos que modificar parte de la información que contiene.

## La carpeta kotlin + java

Esta carpeta contiene los archivos del código fuente. El archivo `MainActivity.kt` contiene el código fuente para la actividad principal de la aplicación. Esta carpeta es el sitio más natural para aquellas clases que generemos.

Dentro hay varios paquetes (con el nombre que indicamos al abrir el proyecto) y a su vez podemos generar otros paquetes para organizar nuestras clases



# Archivos de un proyecto de Android Studio

## La carpeta res

- ❑ Dibujables (drawable): este es un repositorio general para todos los gráficos que pueden ser dibujados en la pantalla, por ejemplo las imágenes.
- ❑ Diseños (layouts): los diseños son archivos XML que definen la arquitectura de la interfaz de usuario en una actividad o en un componente de una interfaz de usuario. Por ejemplo, en nuestra aplicación, el archivo `activity_main.xml` corresponde a la actividad principal.
- ❑ Valores (values): contiene el color, estilo y los archivos XML de cadenas para la aplicación.
  - ❑ `strings.xml`: proporciona cadenas de texto para tu aplicación, (por ejemplo el nombre de la aplicación). La idea de este archivo es que todos los textos que vayas a usar en tu app se incluyan dentro de este fichero y los referencias. Yo no lo haré en los proyectos, pero tenedlo en cuenta a la hora de hacer vuestros desarrollos.

Del apartado de scripts Gradle iremos viendo su uso a lo largo del curso. Es el motor de compilación de dependencias (similar a Maven)



## Archivos de un proyecto de Android Studio

De forma general cada pantalla, o más propiamente actividad hablando de móviles, de una aplicación de móvil corresponde con 2 ficheros:

- Su layout o archivo de diseño.
- Su clase (.kt o .java) con la lógica de la pantalla o actividad.

En el archivo de diseño o layout “pintaremos” aquellos controles o widgets que necesitemos indicando su posición y tamaño.

Por su parte, en la clase kotlin asociada a la actividad será donde se programe la lógica la lógica de la pantalla.

# Archivos de un proyecto de Android Studio - Layout

El layout es un fichero xml que Android Studio nos muestra en modo “código”, “diseño” o mixto.

En este layout será donde debemos incluir los controles de nuestra pantalla y colocarlos. Android nos ofrece diferentes opciones, entre las que destacan:



Diseño horizontal, vertical,  
modo tablet,...

Modo  
nocturno

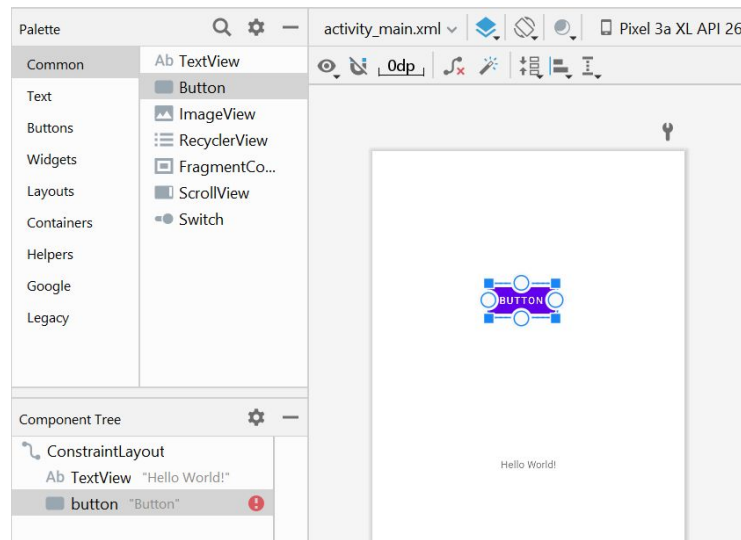
Diferentes tipos de  
dispositivos

De esta forma podremos asegurarnos que el diseño que hagamos encaja en los diferentes dispositivos



# Archivos de un proyecto de Android Studio - Layout

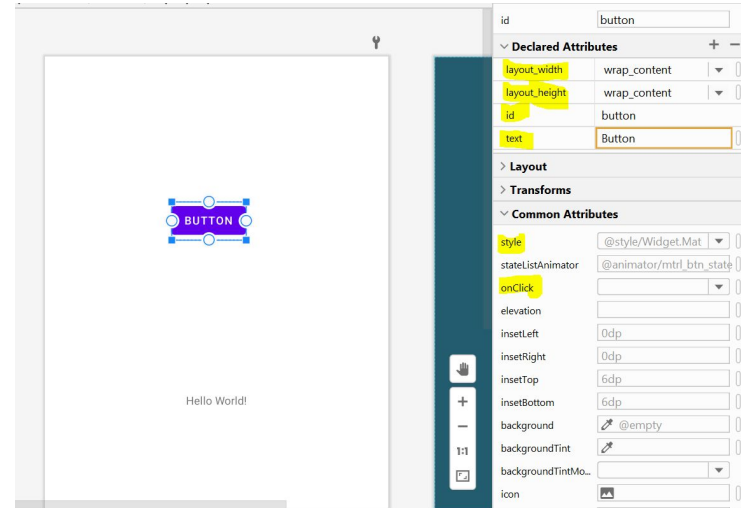
Desde la paleta podemos añadir diferentes controles o widgets. Se añaden con un simple drag&drop y se colocan en el lugar del layout que queramos.



# Archivos de un proyecto de Android Studio - Layout

Para cada control añadido, en la parte derecha tenemos un montón de propiedades configurables (distintas según el control que sea). Algunas de las principales serán:

- id
- Anchura y altura
- Estilo
- Onclick
- ...



Common

Text

Buttons

Widgets

Layouts

Containers

Helpers

Google

Legacy

Ab TextView

Button

ImageView

RecyclerView

FragmentCo...

ScrollView

Switch

Component Tree

ConstraintLayout

Ab TextView "Hello World!"

bt1 "HOLA PMDM"

Button

bt1

id

bt1

Declared Attributes

layout\_width

wrap\_content

layout\_height

wrap\_content

id

bt1

text

HOLA PMDM

Layout

Transforms

Common Attributes

style

@style/Widget.Mat

ateListAnimator

@animator/mtrl\_btn\_state

onClick

levation

setLeft

0dp

Show Baseline

Clear Constraints of Selection

Constrain 

parent top

parent bottom

parent start

parent end

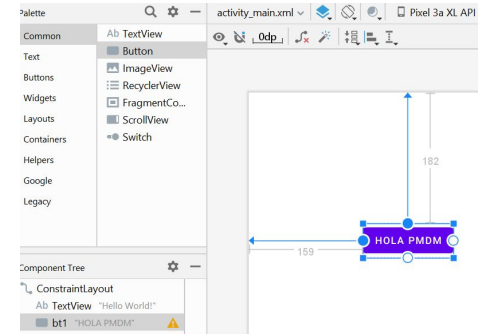
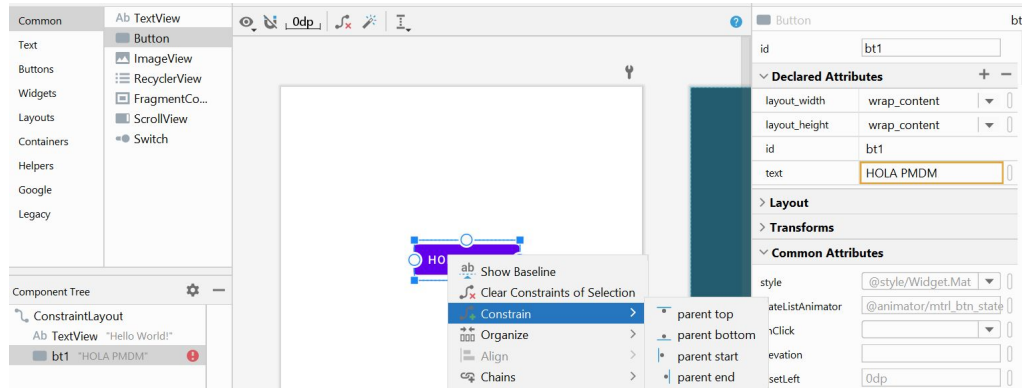
Organize

Align

Chains

# Archivos de un proyecto de Android Studio - Layout

Los controles añadidos una vez dimensionados y colocados hay que “fijarlos” al Layout. Fijarlos es indicar su posición respecto al eje de las X y de las Y para que el “aviso” que tenemos el control desaparezca:



Las “constrains” son anclas bien al eje de la X o bien al de la Y. Debes fijarlas y si quieres mover un control, borras sus constrains y cuando esté bien colocado, las vuelves añadir.

# Archivos de un proyecto de Android Studio - Layout

Vamos a generarnos un proyecto de HolaMundo.

En el Layout vamos a hacer lo siguiente:

- En el textview que viene por defecto, ponerle nombre (se sugiere tv1), borrar sus contrains, subirlo a la parte alta de la actividad, hacerlo más grande y cambiar el tamaño del texto (propiedad textsize) y su color (propiedad textcolor).
- Hay que “fijar el control” a los ejes X e Y
- Añadimos un botón. Le llamaremos bt1 y pondremos de texto “HOLA PMDM”. También necesitamos fijarlo a los ejes





## Archivos de un proyecto de Android Studio - Clase kotlin

Con cada actividad se genera una clase Java asociada. En el caso de la principal, la clase que se llama "MainActivity" y extiende de la clase AppCompatActivity. Esta clase que hemos importado anteriormente dispone de las definiciones necesarias para cargar una actividad.

Por otro lado se incorpora una sobreescripción del método "onCreate". Todas las actividades (activities) deben llevar el método "onCreate", el cual hará que inicie la actividad. Se debe pasar como parámetro un objeto de la clase Bundle, el cual permitirá inicializar el estado de la actividad.

Lo primero que tendremos que hacer en este método será llamar al método onCreate de la clase padre pasando el objeto Bundle anterior.

A continuación, a través de la línea de código: setContentView(R.Layout.activity\_main) se enlaza la parte lógica con la parte gráfica. El archivo XML que va a mostrarse cuando se ejecute la clase "MainActivity" será el archivo XML llamado dentro del registro de recursos de Android Studio llamado "activity\_main".



## Archivos de un proyecto de Android Studio - Clase kotlin

Cada control o widget tiene su espejo en una clase java que permite manejar este tipo de controles. Nuestro HolaMundo va a tener la siguiente lógica, y es que al pulsar el botón de HOLA PMDM queremos que en el TextView se lea HOLA ANDROID STUDIO.

Para ello debemos sobrescribir el mensaje que actualmente hay en el TextView tv1. Para eso tenemos que:

- Crear un objeto de tipo TextView
- Asignar o castear a este objeto el control visual

# Archivos de un proyecto de Android Studio - Clase kotlin

```
class MainActivity : AppCompatActivity() {
```

```
lateinit var miTexto: TextView
```

```
override fun onCreate(savedInstanceState: Bundle?) {
```

```
    super.onCreate(savedInstanceState)
```

```
    enableEdgeToEdge()
```

```
    setContentView(R.layout.activity_main)
```

```
    ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main)) { v, insets ->
```

```
        val systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars())
```

```
        v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom)
```

```
        insets
```

```
    }
```

```
    miTexto = findViewById<TextView>(R.id.tv_1)
```

```
}
```

```
}
```

Creamos un objeto de tipo TextView

Asignamos a mi objeto kotlin el control sobre el textview que hay en mi layout

**findViewById.** Método usado en Android cuya función es recorrer el árbol de vistas en busca del recurso que tiene la identificación aportada.



# Archivos de un proyecto de Android Studio - Clase kotlin

```
✓ class MainActivity : AppCompatActivity() {  
  
lateinit var miTexto: TextView  
✓ override fun onCreate(savedInstanceState: Bundle?) {  
    super.onCreate(savedInstanceState)  
    enableEdgeToEdge()  
    setContentView(R.layout.activity_main)  
    ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main)) { v, insets ->  
        ✓ val systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars())  
        v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom)  
        insets  
    }  
    miTexto = findViewById<TextView>(R.id.tv_1)  
}  
fun clickBoton(v: View) {miTexto.text = "HOLA CLASE"}
```

Creamos la función “clickBoton” dentro de la clase. Este método OBLIGATORIAMENTE debe recibir un objeto de tipo View

Dentro del método asignamos en el control TextView el texto que queremos

## Clase R

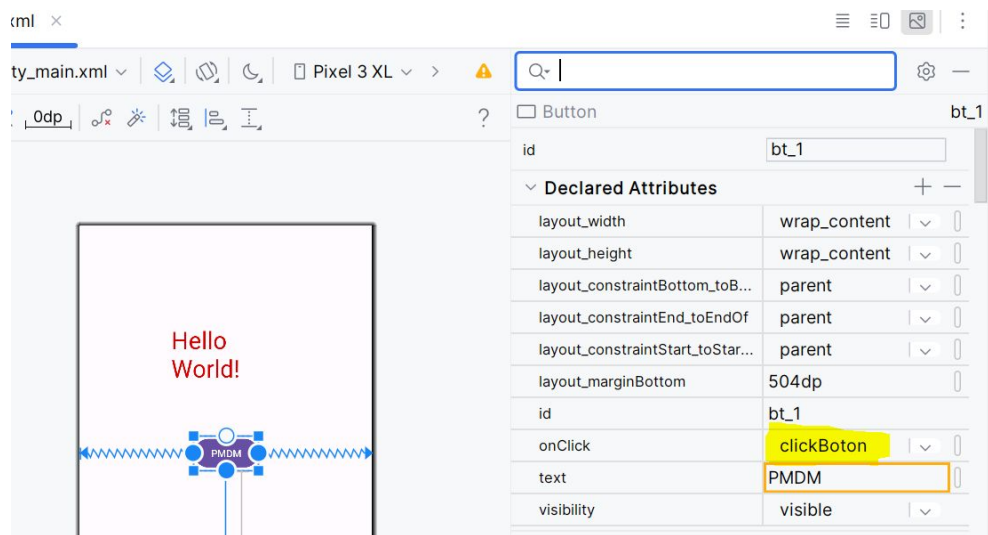
Se trata de una clase que contiene variables estáticas en las que se identifica cada tipo de recurso. Android lee el fichero XML, carga todas las estructuras solicitadas en memoria y mantiene el fichero R como referencia directa a los recursos cargados. Por tanto, cada atributo tiene una dirección de memoria asociada referenciada a un recurso en específico. En la figura 2.3, *mensaje* contiene la cadena “Hola Mundo”, que está almacenado en la dirección de memoria 0x7f0d0028.



**Figura 2.3**  
Organización de la clase R.

# Archivos de un proyecto de Android Studio - Clase Kotlin

Ahora debemos ir de nuevo al Layout y seleccionar el botón. Vamos a sus propiedades y buscamos la propiedad onClick y le indicamos que al “clicar” el botón lo que queremos es que llame al método “saludos”





# Probar en Android Studio - Emuladores

Para probar una aplicación con Android Studio tenemos tres opciones

- Conectar mediante USB o Bluetooth un dispositivo Android y [configurarlo para poder hacer pruebas](#).
- **Generar un emulador del dispositivo con el que vamos a probar. Nosotros utilizaremos esta.**
- [Probar con otras aplicaciones que son emuladores](#) (para los cual hay que compilar el código, hacer una apk e instalarla en ese emulador).

Dentro del menú de Tools / Device Manager se configuran los emuladores. Es bastante intuitivo, pero puedes encontrar ayuda en la página: <https://developer.android.com/studio/run/managing-avds>

Cada emulador requiere de bastante espacio (depende del SO, puede que varios Gbs). Recordad que debe haber cierta sincronización entre la versión de API del emulador y la versión sobre la que deseas desarrollar tus apps. En esta clase vamos a elegir SIEMPRE la versión 26 del API para hacer los ejercicios.

# Compilación

Para compilar y además ejecutar tu app, bastará con hacer clic en el botón Run (situado en la barra de herramientas, o bien, a través de menú Run-->Run 'app'. Con el martillo ejecutar)



situado en la barra de herramientas, o bien, a través de menú Run-->Run 'app'. Con el martillo ejecutar)

Android Studio compilará la app con Gradle, y a continuación, solicitará que seleccionemos un destino de implementación (un emulador o un dispositivo conectado) y, luego, cargará la app en dicho destino. Podremos personalizar parte de este comportamiento predeterminado (por ejemplo, la selección de un destino de implementación automático).



Nota: Además, puedes ejecutar tu app en el modo de depuración. Para ello, haz clic en Debug.

La ejecución de tu app en el modo de depuración te permite configurar puntos de interrupción en el código, examinar variables y evaluar expresiones en el tiempo de ejecución, y también ejecutar herramientas de depuración.

Para obtener más información, consulta [Depuración de tu app.](#)



## Pestañas de Android Studio

Las pestañas aparecen y desaparecen al pincharlas. Veamos las que más utilizaremos

La pestaña de “Logcat” que nos va a mostrar en tiempo de ejecución los errores o mensajes que nuestra aplicación va generando.

La pestaña de “Built” que nos va a mostrar los errores de compilación.

Además puedes cambiar configuraciones de Android Studio (como ponerlo en oscuro por ejemplo) dentro de la pantalla de preferencias (Settings) que se accede desde el menú de File.



# Hola mundo Android Studio

Una vez tengamos configurado el emulador, para probar nuestra app simplemente debemos ir a la opción de run. Si no hay emulador abierto, Android nos lo abrirá, compilará nuestra aplicación y la instalará para que podamos probarla

