Unidad 1: Introducción al desarrollo de aplicaciones móviles

Primera aplicación móvil.

Curso 2021/2022

IES Virgen de la Paloma

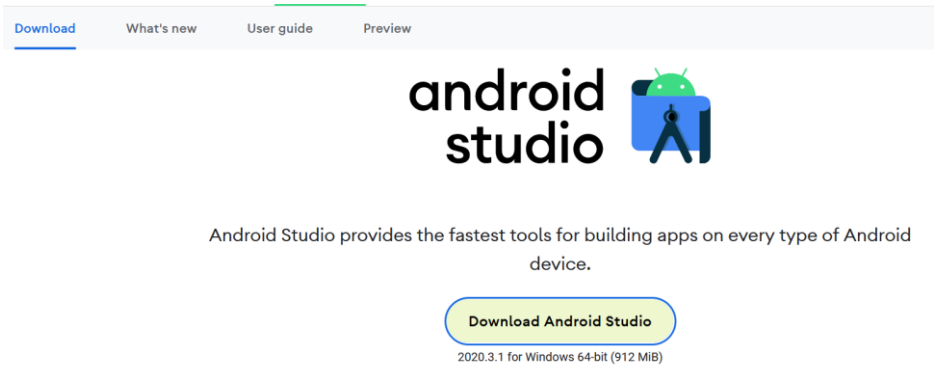
Mª Carmen Díaz González

Índice

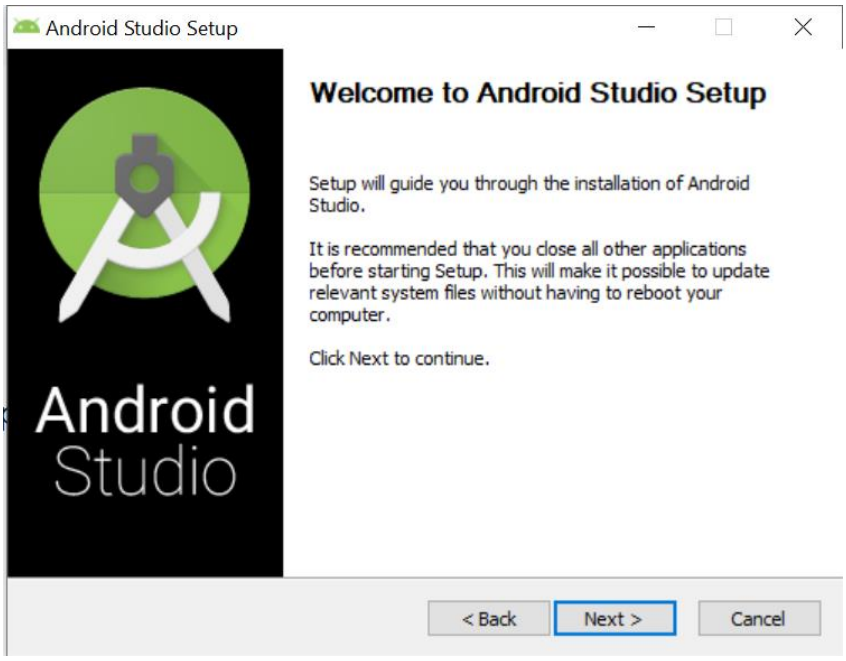
1. Instalación Android Studio

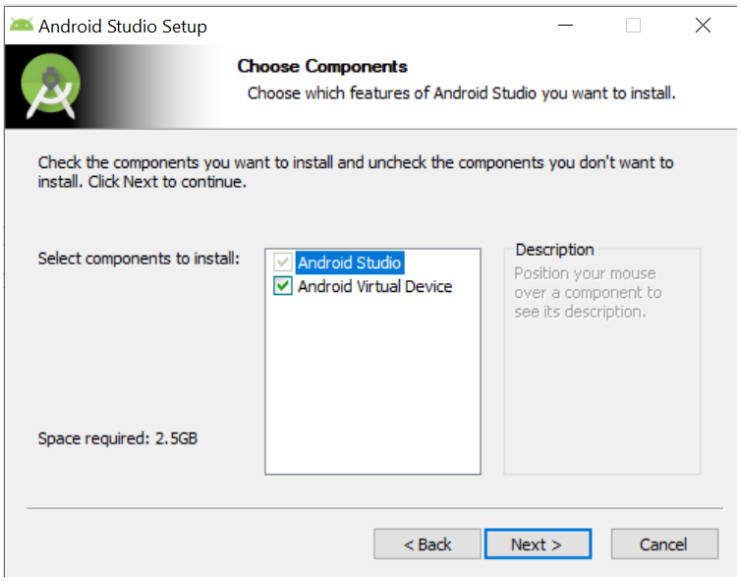
Podemos descargar el entorno de desarrollo Android Studio del siguiente enlace:

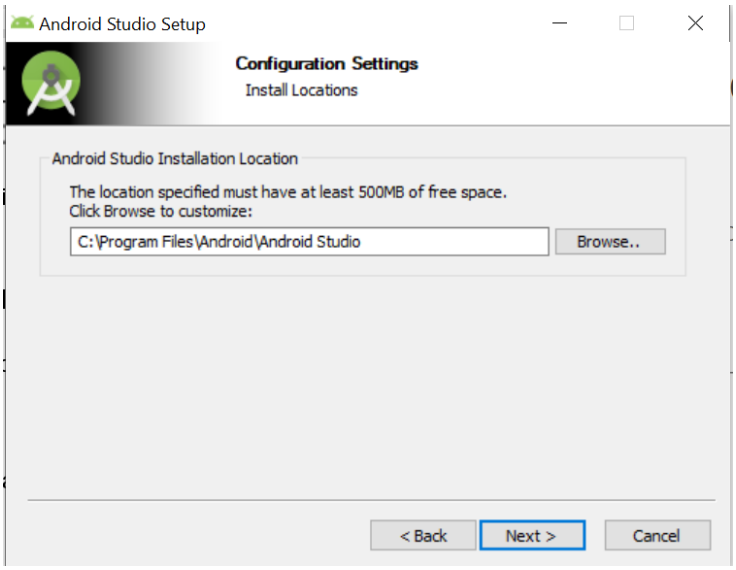
https://developer.android.com/studio#downloads

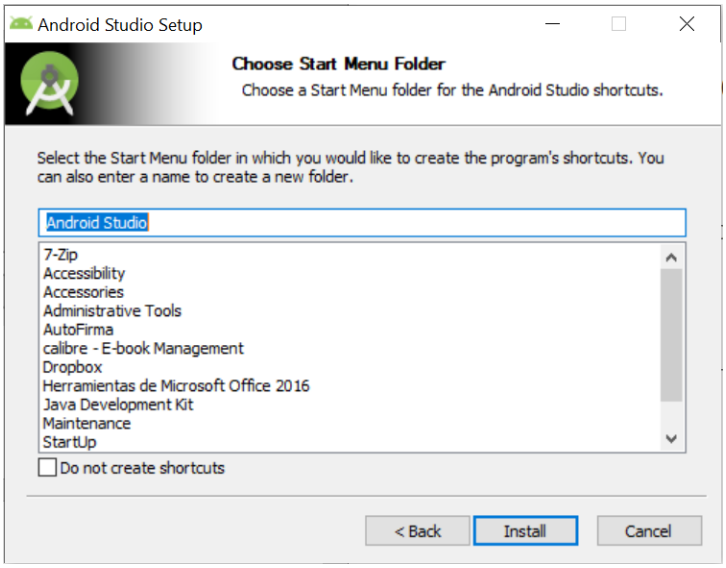


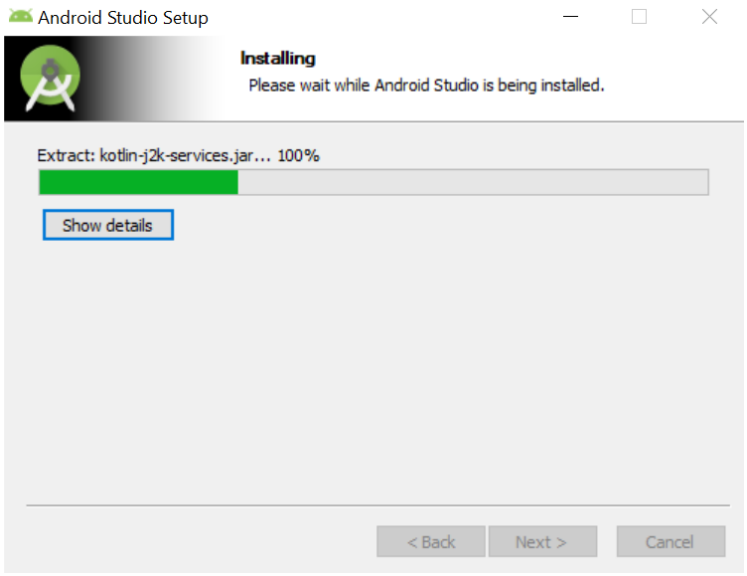
Una vez tenemos descargado el archivo .exe, lo ejecutamos y continuamos la instalación estándar con la ayuda del asistente.

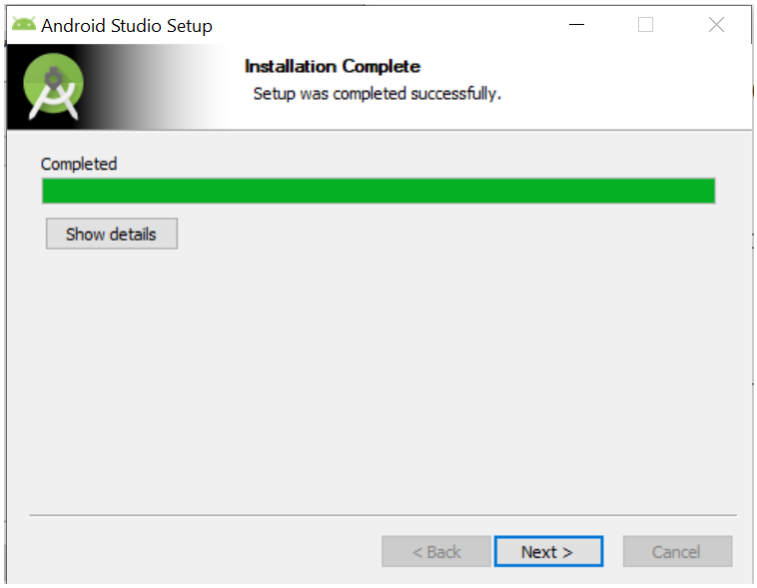


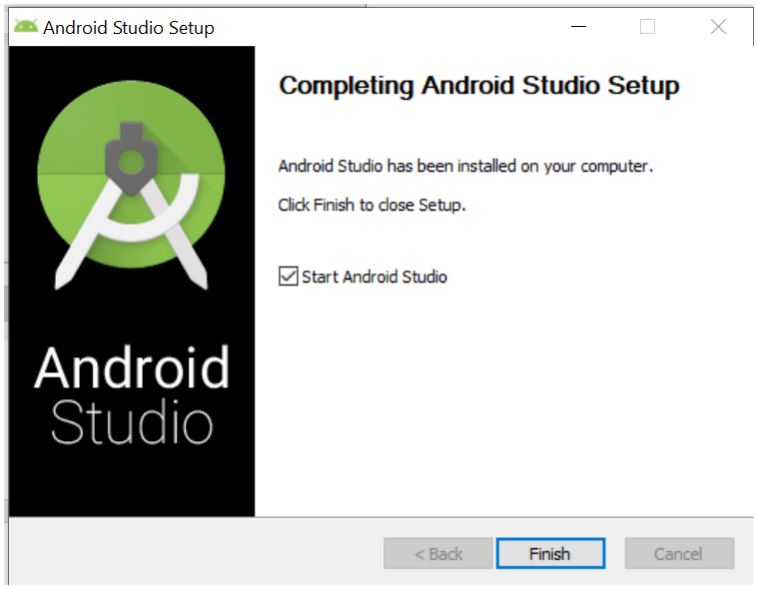

Android Studio Setup
Choose Components
Choose which features of Android Studio you want to install.
Check the components you want to install and uncheck the components you don't want to install. Click Next to continue.
Select components to install:
Android Studio
Android Virtual Device
Description
Position your mouse over a component to see its description.
Space required: 2.5GB
< Back
Next >
Cancel


Android Studio Setup
Configuration Settings
Install Locations
Android Studio Installation Location
The location specified must have at least 500MB of free space.
Click Browse to customize:
C:\Program Files\Android\Android Studio
Browse..
< Back
Next >
Cancel

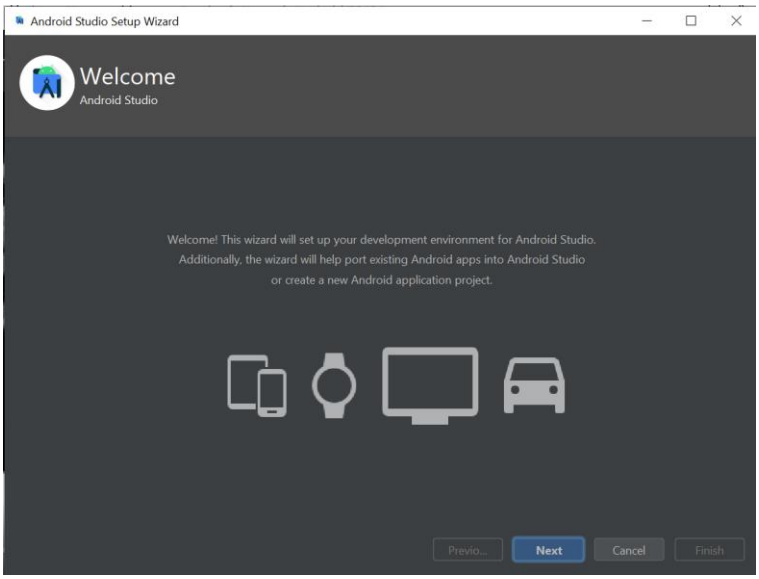

Android Studio Setup
Choose Start Menu Folder
Choose a Start Menu folder for the Android Studio shortcuts.
Select the Start Menu folder in which you would like to create the program's shortcuts. You can also enter a name to create a new folder.
Android Studio
7-Zip
Accessibility
Accessories
Administrative Tools
AutoFirma
calibre - E-book Management
Dropbox
Herramientas de Microsoft Office 2016
Java Development Kit
Maintenance
StartUp
Do not create shortcuts
< Back
Install
Cancel

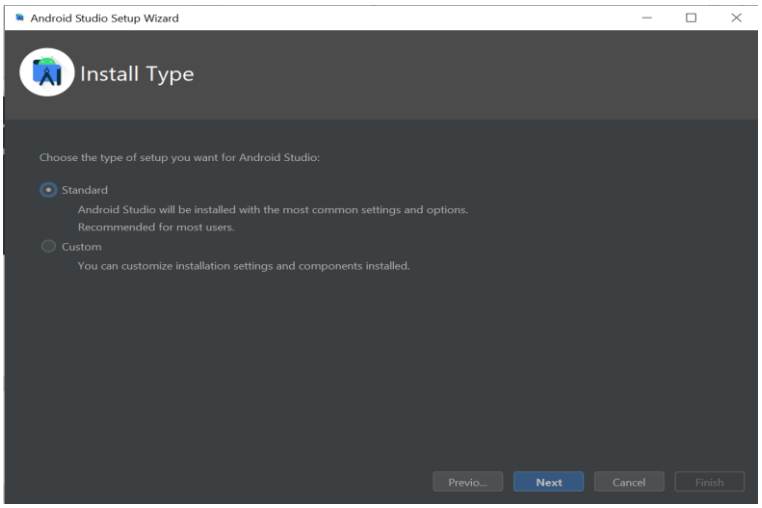

Android Studio Setup
Installing
Please wait while Android Studio is being installed.
Extract: kotlin-j2k-services.jar... 100%
Show details
< Back
Next >
Cancel


Android Studio Setup
Installation Complete
Setup was completed successfully.
Completed
Show details
< Back
Next >
Cancel

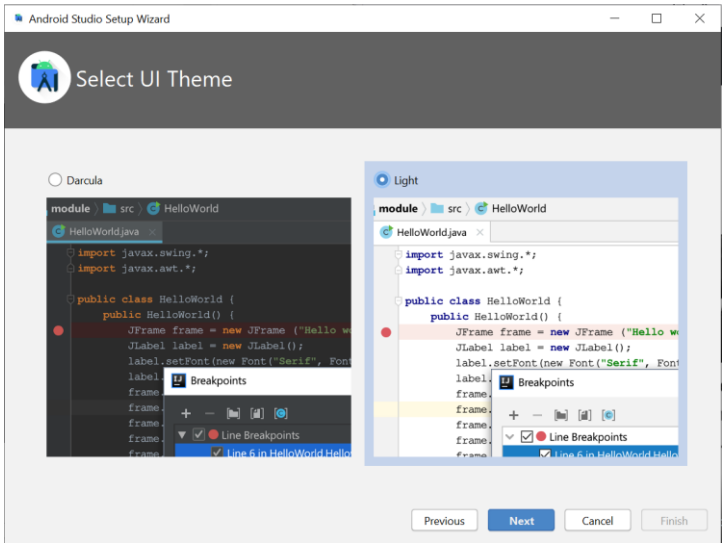

Android Studio Setup
Completing Android Studio Setup
Android Studio has been installed on your computer.
Click Finish to close Setup.
Start Android Studio
Android
Studio
< Back
Finish
Cancel

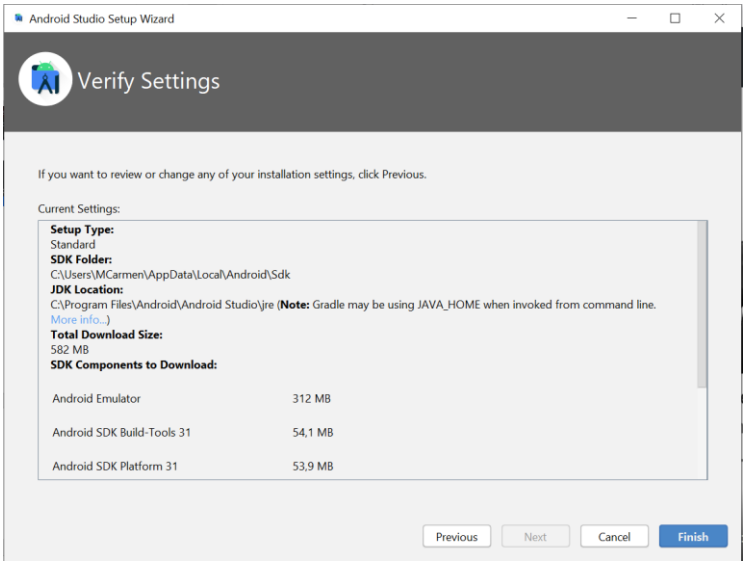
Comenzamos a Configurar el entorno:

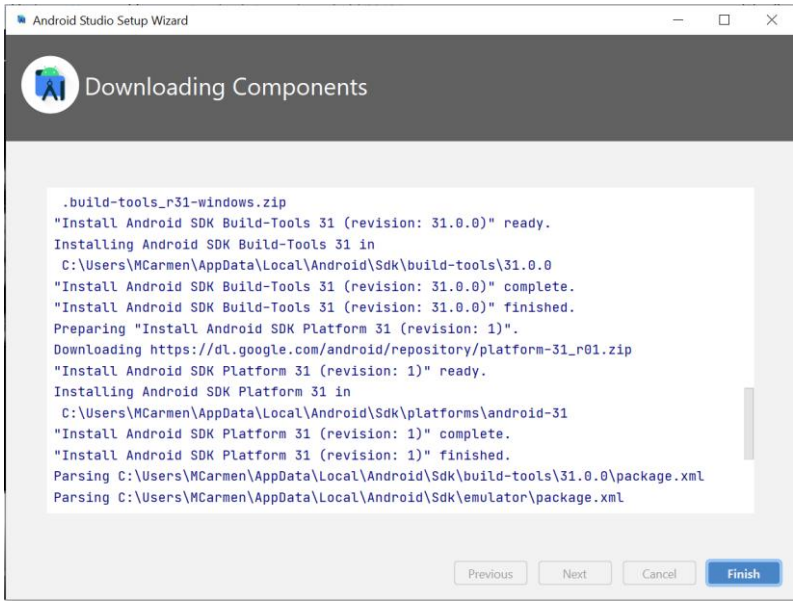




Se recomienda utilizar el tema Light para que se vea bien al proyectar en clase.




Android Studio Setup Wizard
Verify Settings
If you want to review or change any of your installation settings, click Previous.
Current Settings:
Setup Type:
Standard
SDK Folder:
C:\Users\MCarmen\AppData\Local\Android\Sdk
JDK Location:
C:\Program Files\Android\Android Studio\jre (Note: Gradle may be using JAVA_HOME when invoked from command line.
More info...)
Total Download Size:
582 MB
SDK Components to Download:
Android Emulator 312 MB
Android SDK Build-Tools 31 54,1 MB
Android SDK Platform 31 53,9 MB
Previous
Next
Cancel
Finish


Android Studio Setup Wizard
Downloading Components
.build-tools_r31-windows.zip
"Install Android SDK Build-Tools 31 (revision: 31.0.0)" ready.
Installing Android SDK Build-Tools 31 in
C:\Users\MCarmen\AppData\Local\Android\Sdk\build-tools\31.0.0
"Install Android SDK Build-Tools 31 (revision: 31.0.0)" complete.
"Install Android SDK Build-Tools 31 (revision: 31.0.0)" finished.
Preparing "Install Android SDK Platform 31 (revision: 1)".
Downloading https://dl.google.com/android/repository/platform-31_r01.zip
"Install Android SDK Platform 31 (revision: 1)" ready.
Installing Android SDK Platform 31 in
C:\Users\MCarmen\AppData\Local\Android\Sdk\platforms\android-31
"Install Android SDK Platform 31 (revision: 1)" complete.
"Install Android SDK Platform 31 (revision: 1)" finished.
Parsing C:\Users\MCarmen\AppData\Local\Android\Sdk\build-tools\31.0.0\package.xml
Parsing C:\Users\MCarmen\AppData\Local\Android\Sdk\emulator\package.xml
Previous
Next
Cancel
Finish

2. Componentes de una aplicación Android

Los componentes principales que pueden formar parte de una aplicación Android son:

2.1. Activity

Las actividades (activity) representan el componente principal de la interfaz gráfica de una aplicación Android. Se puede pensar en una actividad como el elemento análogo a una ventana o pantalla en cualquier otro lenguaje visual.

2.2. View

Las vistas (view) son los componentes básicos con los que se construye la interfaz gráfica de la aplicación, análogo por ejemplo a los controles de Java o .NET. De inicio, Android pone a nuestra disposición una gran cantidad de controles básicos, como cuadros de texto, botones, listas desplegables o imágenes, aunque también existe la posibilidad de extender la funcionalidad de estos controles básicos o crear nuestros propios controles personalizados.

2.3. Fragment

Los fragmentos (fragment) se pueden entender como secciones o partes (habitualmente reutilizables) de la interfaz de usuario de una aplicación. De esta forma, una actividad podría contener varios fragmentos para formar la interfaz completa de la aplicación, y adicionalmente estos fragmentos se podrían reutilizar en distintas actividades o partes de la aplicación. No es obligatorio utilizar fragmentos en una aplicación, pero sí nos serán de mucha ayuda en ciertas ocasiones, por ejemplo, para adaptar la interfaz de nuestra aplicación a distintos dispositivos, tamaños de pantalla, orientación, etc.

2.4. Service

Los servicios (service) son componentes sin interfaz gráfica que se ejecutan en segundo plano. Conceptualmente, son similares a los servicios presentes en cualquier otro sistema operativo. Los servicios pueden realizar cualquier tipo de acción, por ejemplo, actualizar datos, lanzar notificaciones, o incluso mostrar elementos visuales (p.ej. actividades) si se necesita en algún momento la interacción con del usuario.

2.5. Content Provider

Un proveedor de contenidos (content provider) es el mecanismo que se ha definido en Android para compartir datos entre aplicaciones. Mediante estos componentes es posible compartir determinados datos de nuestra aplicación sin mostrar detalles sobre su almacenamiento interno, su estructura, o su implementación. De la misma forma, nuestra aplicación podrá acceder a los datos de otra a través de los content provider que ésta última haya definido.

2.6. Broadcast Receiver

Un broadcast receiver es un componente destinado a detectar y reaccionar ante determinados mensajes o eventos globales generados por el sistema (por ejemplo: “Batería baja”, “SMS recibido”, “Tarjeta SD insertada”, …) o por otras aplicaciones (cualquier aplicación puede generar mensajes (intents, en terminología Android) de tipo broadcast, es decir, no dirigidos a una aplicación concreta sino a cualquiera que quiera escucharlo).

2.7. Widget

Los widgets son elementos visuales, normalmente interactivos, que pueden mostrarse en la pantalla principal (home screen) del dispositivo Android y recibir actualizaciones periódicas.

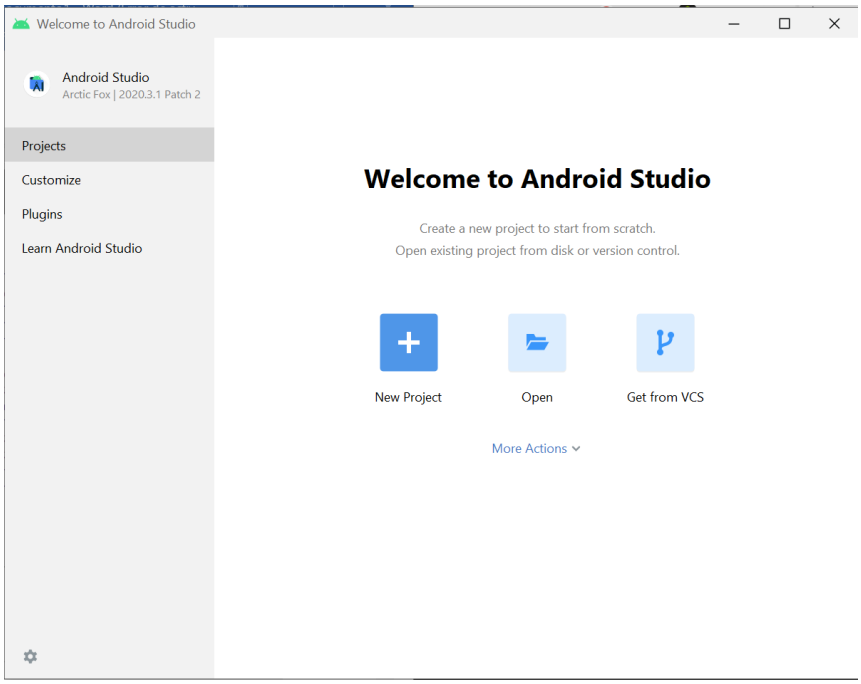
Permiten mostrar información de la aplicación al usuario directamente sobre la pantalla principal.

2.8. Intent

Un intent es el elemento básico de comunicación entre los distintos componentes Android que hemos descrito anteriormente. Se pueden entender como los mensajes o peticiones que son enviados entre los distintos componentes de una aplicación o entre distintas aplicaciones. Mediante un intent se puede mostrar una actividad desde cualquier otra, iniciar un servicio, enviar un mensaje broadcast, iniciar otra aplicación, etc.

3. Creando el primer proyecto

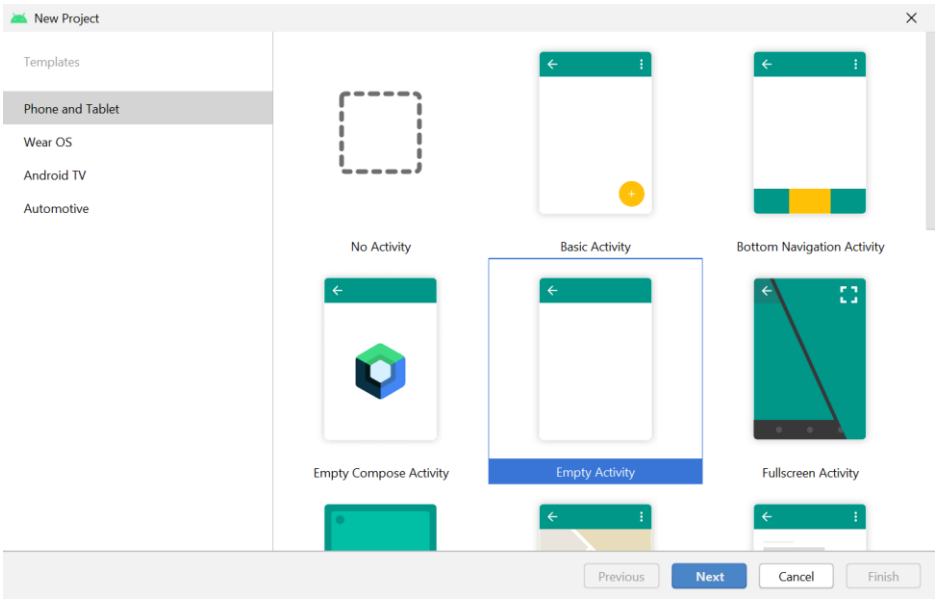
Para crear un nuevo proyecto ejecutaremos Android Studio y desde la pantalla de bienvenida pulsaremos la opción «New Project» para iniciar el asistente de creación de un nuevo proyecto.



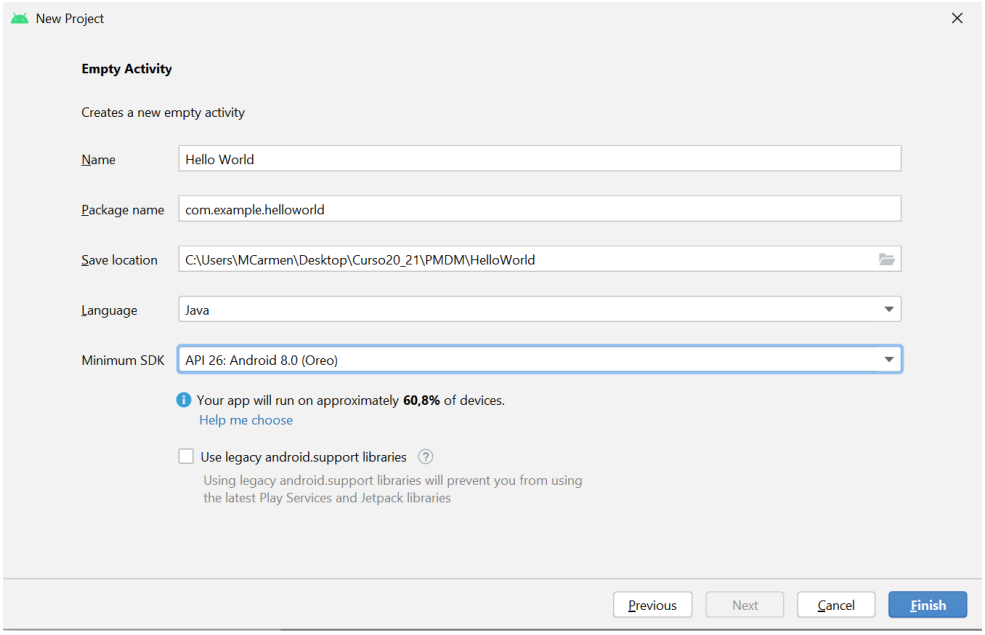
Si ya habíamos abierto anteriormente Android Studio es posible que se abra directamente la aplicación principal en vez de la pantalla de bienvenida. En ese caso accederemos al menú «File / New project…» para crear el nuevo proyecto.

El asistente de creación del proyecto nos guiará por las distintas opciones de creación y configuración de un nuevo proyecto Android.

En la primera pantalla del asistente elegiremos el tipo de actividad principal de la aplicación. Entenderemos por ahora que una actividad es una “ventana” o “pantalla” de la aplicación.

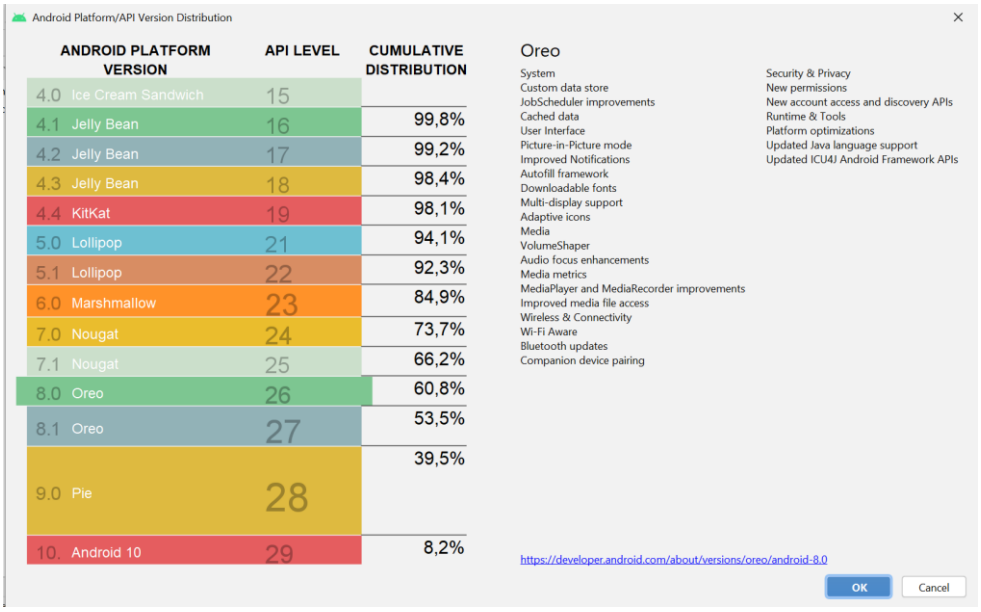


En la siguiente pantalla indicaremos, por este orden, el nombre de la aplicación, el paquete java para nuestras clases, y la ruta donde crear el proyecto. Para el segundo de los datos suele utilizarse un valor del tipo domino.invertido.proyecto. En mi caso utilizaré por tanto «com.example.helloworld». En tu caso puedes utilizar cualquier otro valor. Adicionalmente tendremos que indicar el lenguaje que utilizaremos para desarrollar (a partir de ahora utilizaré siempre Java) y la API mínima (es decir, la versión mínima de Android) que soportará la aplicación. Para soportar el máximo de dispositivos que sea posible, se debe establecer a la versión más baja posible. Sin embargo, cuanto más baja es la versión menos características de android modernas puede utilizar la app, como norma general, nos centraremos en Android 8.0 como versión mínima (API 26). Por último, el check de «Use legacy android.support libraries» lo dejaremos deshabilitado.

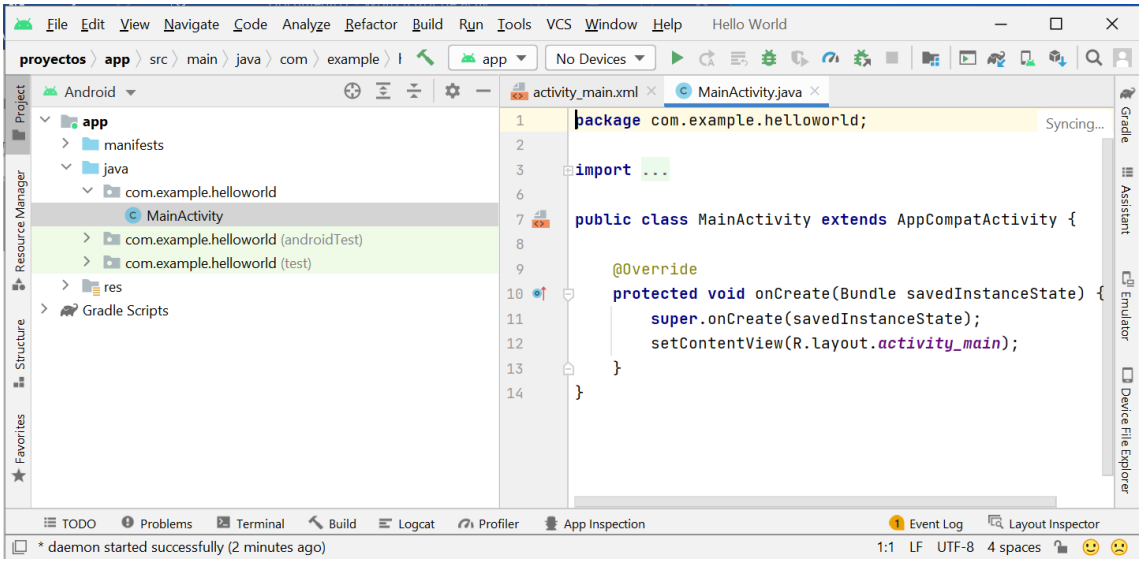


La versión mínima que seleccionemos en la pantalla anterior implicará que nuestra aplicación se pueda ejecutar en más o menos dispositivos. De esta forma, cuanto menor sea ésta, a más dispositivos podrá llegar nuestra aplicación, pero más complicado será conseguir que se

ejecute correctamente en todas las versiones de Android. Para hacernos una idea del número de dispositivos que cubrimos con cada versión podemos pulsar sobre el enlace «Help me choose», que mostrará el porcentaje de dispositivos que ejecutan actualmente cada versión de Android. Como información adicional, si pulsamos sobre cada versión de Android en esta pantalla podremos ver una lista de las novedades introducidas por dicha versión

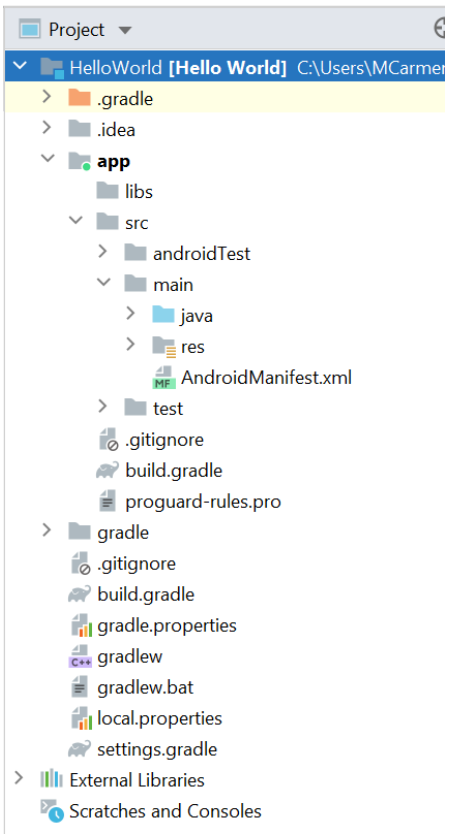


Una vez configurado todo pulsamos el botón Finish y Android Studio creará por nosotros toda la estructura del proyecto y los elementos indispensables que debe contener. Si todo va bien aparecerá la pantalla principal de Android Studio con el nuevo proyecto creado.



En la parte izquierda, podemos observar todos los elementos creados inicialmente para el nuevo proyecto Android, sin embargo, por defecto los vemos de una forma un tanto peculiar que inicialmente puede llevarnos a confusión. Para entender mejor la estructura del proyecto vamos a cambiar momentáneamente la forma en la que Android Studio nos la muestra. Para ello, pulsaremos sobre la lista desplegable situada en la parte superior izquierda, y cambiaremos la vista de proyecto al modo «Project» (en cualquier momento podremos volver al modo «Android» inicial).

Tras hacer esto, la estructura del proyecto cambia un poco de aspecto y pasa a ser como se observa en la siguiente imagen:



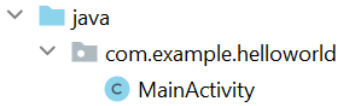
En los siguientes apartados describiremos los elementos principales de esta estructura.

Lo primero que debemos distinguir son los conceptos de proyecto y módulo. La entidad proyecto es única, y engloba a todos los demás elementos. Dentro de un proyecto podemos incluir varios módulos, que pueden representar aplicaciones distintas, versiones diferentes de una misma aplicación, o distintos componentes de un sistema (aplicación móvil, aplicación servidor, librerías, …). En la mayoría de los casos, trabajaremos con un proyecto que contendrá un sólo módulo correspondiente a nuestra aplicación principal. Por ejemplo, en este caso que estamos creando tenemos el proyecto «HelloWorld» que contiene un solo módulo «app» que contendrá todo el software de la aplicación de ejemplo.

A continuación, describiremos los contenidos principales de nuestro módulo principal.

Carpeta /app/src/main/java

Esta carpeta contendrá todo el código fuente de la aplicación, clases auxiliares, etc. Inicialmente, Android Studio creará por nosotros el código básico de la pantalla (actividad o activity) principal de la aplicación, que por defecto se llamará MainActivity, y siempre bajo la estructura del paquete java definido durante la creación del proyecto. En este fichero programaremos el comportamiento de esta Activity



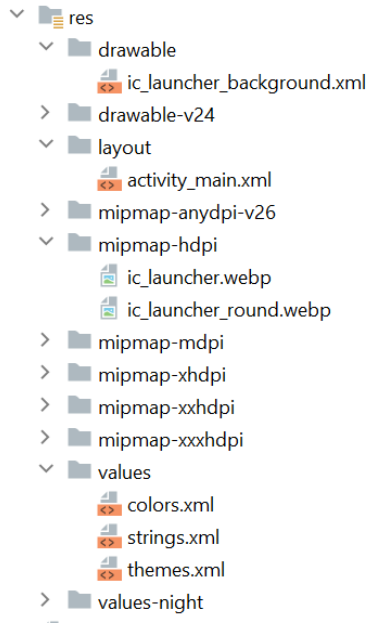
Carpeta /app/src/main/res/

Contiene todos los ficheros de recursos necesarios para el proyecto: imágenes, layouts, cadenas de texto, etc. Los diferentes tipos de recursos se pueden distribuir entre las siguientes subcarpetas:

Carpeta	Descripción
/res/drawable/	Contiene las imágenes y otros elementos gráficos usados por la aplicación. Para poder definir diferentes recursos dependiendo de la resolución y densidad de la pantalla del dispositivo se suele dividir en varias subcarpetas: * /drawable (recursos independientes de la densidad) * /drawable-ldpi (densidad baja) * /drawable-mdpi (densidad media) * /drawable-hdpi (densidad alta) * /drawable-xhdpi (densidad muy alta) * /drawable-xxhdpi (densidad muy muy alta :)
/res/mipmap/	Contiene los iconos de lanzamiento de la aplicación (el icono que aparecerá en el menú de aplicaciones del dispositivo) para las distintas densidades de pantalla existentes. Al igual que en el caso de las carpetas /drawable, se dividirá en varias subcarpetas dependiendo de la densidad de pantalla: * /mipmap-mdpi * /mipmap-hdpi * /mipmap-xhdpi * …
/res/layout/	Contiene los ficheros de definición XML de las diferentes pantallas de la interfaz gráfica. Para definir distintos *layouts* dependiendo de la orientación del dispositivo se puede dividir también en subcarpetas: * /layout (vertical) * /layout-land (horizontal)
/res/anim/ /res/animator/	Contienen la definición de las animaciones utilizadas por la aplicación.
/res/color/	Contiene ficheros XML de definición de listas de colores según estado.
/res/menu/	Contiene la definición XML de los menús de la aplicación.
/res/xml/	Contiene otros ficheros XML de datos utilizados por la aplicación.
/res/raw/	Contiene recursos adicionales, normalmente en formato distinto a XML, que no se incluyan en el resto de carpetas de recursos.
/res/values/	Contiene otros ficheros XML de recursos de la aplicación, como, por ejemplo: *strings.xml*, donde se definen todos los textos que se muestren en nuestra app *styles.xml* donde se define el tema de nuestra app, *colors.xml,* define los colores de la app

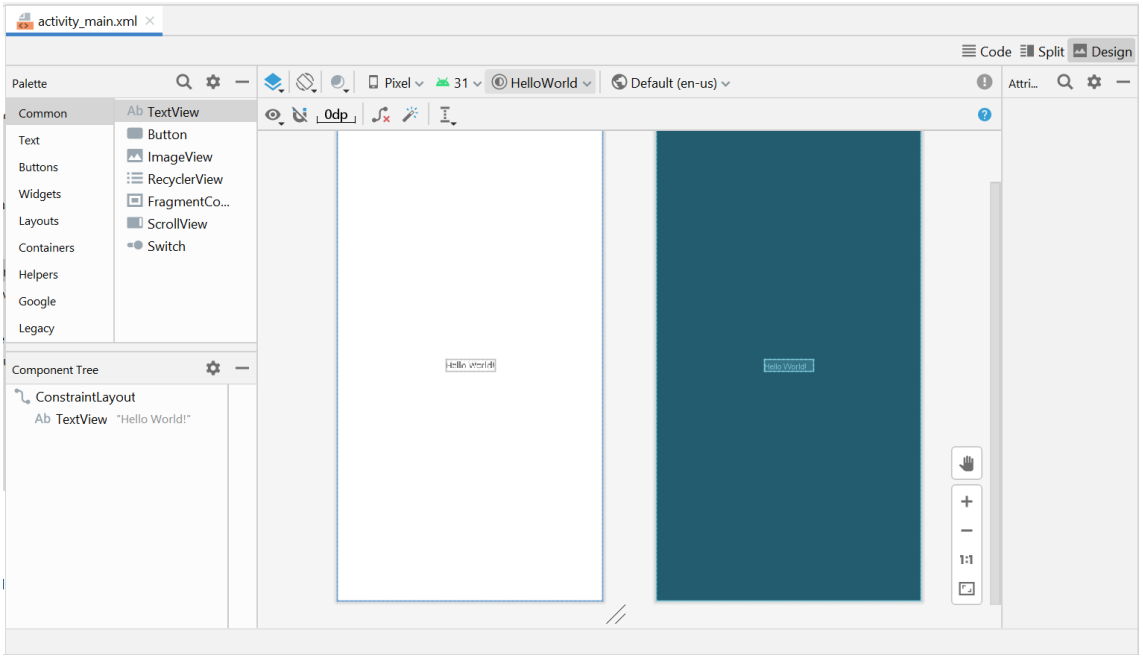
No todas estas carpetas tienen por qué aparecer en cada proyecto Android, tan sólo las que se necesiten. Iremos viendo durante el curso qué tipo de elementos se pueden incluir en cada una de ellas y cómo se utilizan.

Como ejemplo, para un proyecto nuevo Android como el que hemos creado, tendremos por defecto los siguientes recursos para la aplicación:

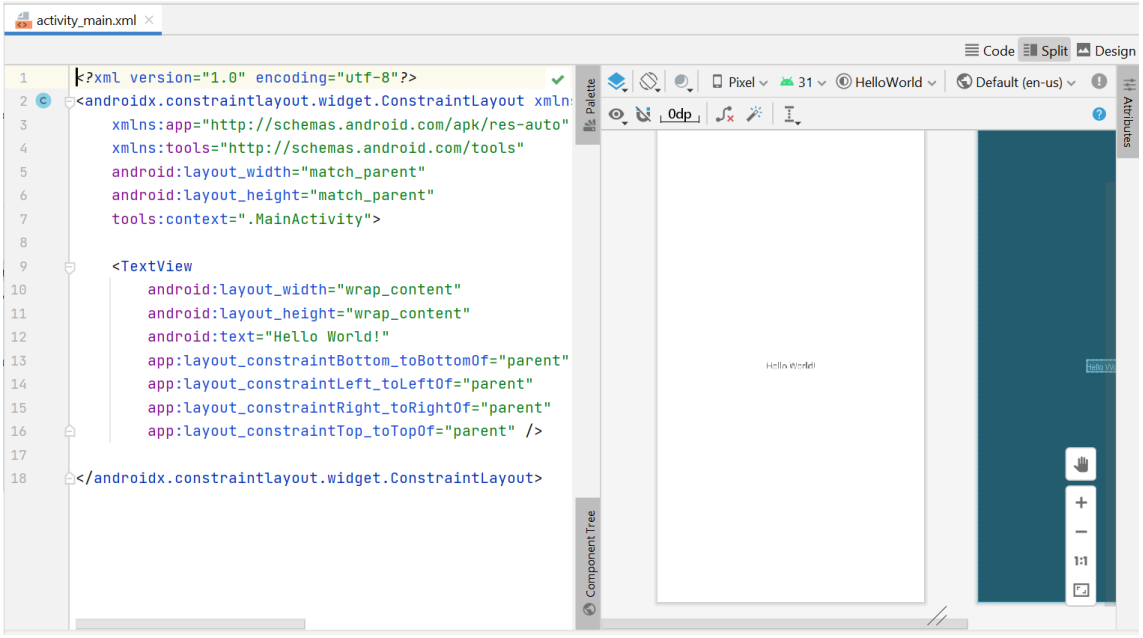


Como se puede observar, existen algunas carpetas en cuyo nombre se incluye un sufijo adicional, como por ejemplo «drawable-v24«. Éstos, y otros sufijos, se emplean para definir recursos independientes para determinados dispositivos según sus características. De esta forma, por ejemplo, los recursos incluidos en la carpeta «drawable-v24» se aplicarían tan sólo a dispositivos cuya versión de Android sea la 7.0 (API 24) o superior, o por ejemplo los incluidos en una carpeta llamada “values-w820dp” se aplicarían sólo a pantallas con más de 820dp de ancho. Al igual que estos sufijos «-w» y «–v» existen otros muchos para referirse a otras características del terminal, puede consultarse la lista completa en la documentación oficial del Android (https://developer.android.com/guide/topics/resources/providing-resources.html).

Entre los recursos creados por defecto cabe destacar los layouts, en nuestro caso sólo tendremos por ahora el llamado “activity_main.xml”, que contienen la definición de la interfaz gráfica de la pantalla principal de la aplicación. Si hacemos doble clic sobre este fichero Android Studio nos mostrará esta interfaz en su editor gráfico, y como podremos comprobar, en principio contiene tan sólo una etiqueta de texto con el mensaje “Hello World”.



Pulsando sobre los tres botones de la esquina superior derecha podemos alternar entre el editor gráfico (tipo arrastrar-y-soltar), mostrado en la imagen anterior, el editor de código XML, o una vista compartida que permita visualizar ambas cosas de forma simultánea, como en la imagen siguiente:



Fichero /app/src/main/AndroidManifest.xml

Contiene la definición en XML de muchos de los aspectos principales de la aplicación, como por ejemplo su identificación (nombre, icono, …), sus componentes (pantallas, servicios, …), o los permisos necesarios para su ejecución. Veremos más adelante más detalles de este fichero.

Fichero /app/build.gradle

Contiene información necesaria para la compilación del proyecto.

En un proyecto pueden existir varios ficheros build.gradle, para definir determinados parámetros a distintos niveles. Por ejemplo, en nuestro proyecto podemos ver que existe un fichero build.gradle a nivel de proyecto, y otro a nivel de módulo dentro de la carpeta /app. El primero de ellos definirá parámetros globales a todos los módulos del proyecto, y el segundo sólo tendrá efecto para cada módulo en particular.

En el fichero build.gradle (module: app), es donde se encuentran las dependencias de construcción de la aplicación, incluyendo los ajustes defaultConfig:

- applicationId es el nombre de paquete completo para la aplicación, que se especificó en el asistente de Nuevo Proyecto.
- minSdk es la versión del SDK mínimo especificado durante el asistente Nuevo proyecto. Es la versión más antigua del SDK de Android que soporta la aplicación.
- targetSdk indica la versión más alta de Android con la que se ha probado la aplicación.

Carpeta /app/libs

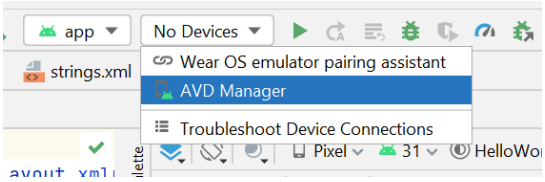
Puede contener las librerías externas que utilice nuestra aplicación. Normalmente no incluiremos directamente aquí ninguna librería, sino que haremos referencia a ellas en el fichero build.gradle descrito en el punto anterior, de forma que entren en el proceso de compilación de nuestra aplicación.

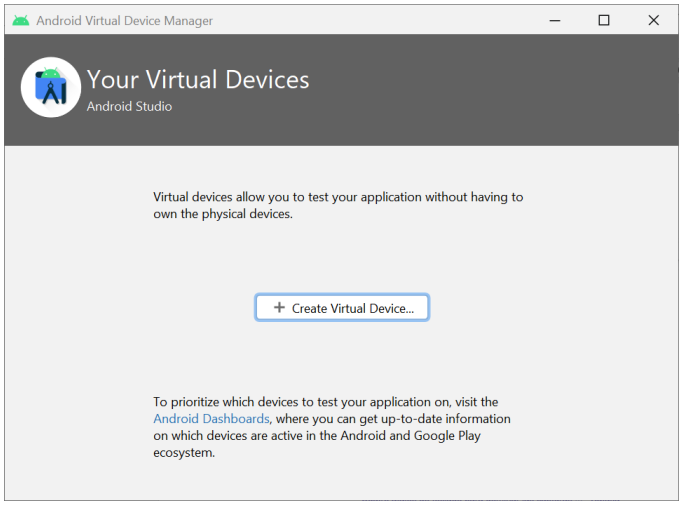
4. Configuración de un emulador

Un dispositivo virtual de Android (AVD) es una configuración que define las características de un teléfono o una tablet Android, o de un dispositivo Wear OS, Android TV o Automotive OS, que deseas simular en Android Emulator. El Administrador de AVD es una interfaz que puedes iniciar desde Android Studio y te permite crear y administrar los AVD.

Para abrir el Administrador de AVD, realiza una de las siguientes acciones:

- Selecciona Tools > AVD Manager.
- O en la barra de herramientas, haz clic en AVD Manager Ícono del Administrador de AVD.





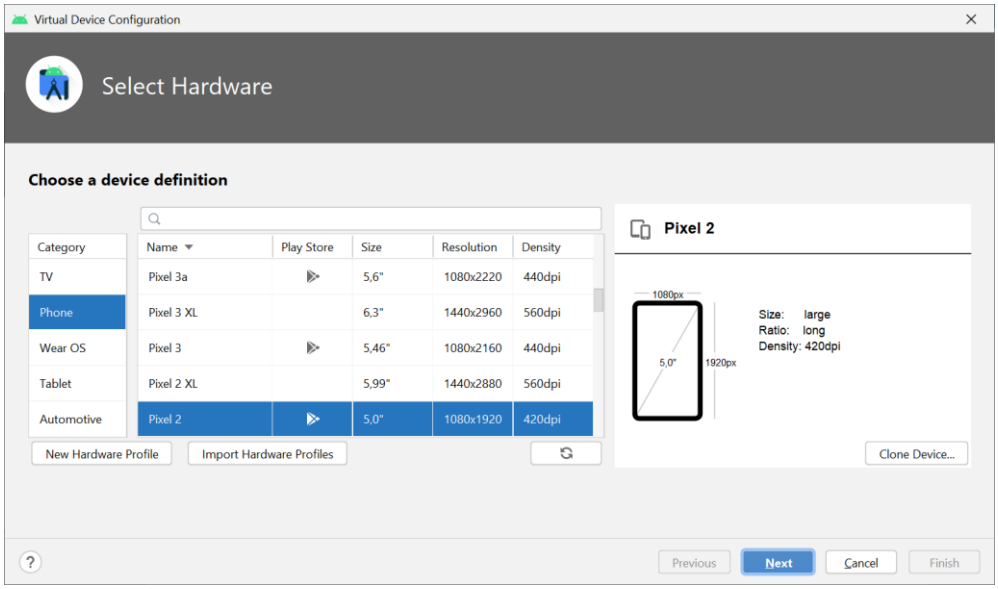
Deberemos seleccionar «Create Virtual Device…» para crear un nuevo dispositivo.

Esto iniciará el asistente de creación de un AVD. En la primera pantalla tendremos que seleccionar el tipo de dispositivo que queremos crear: teléfono, tablet, smartwatch (Wear OS), Android TV, o Android Auto. Una vez seleccionado el tipo de dispositivo, podemos seleccionar las características concretas del dispositivo (tamaño, resolución, y densidad de píxeles de su pantalla, memoria, cámaras, …). Para ellos podremos elegir entre una lista predefinida de modelos reales de dispositivo, algunos modelos genéricos, o bien definir nuestro propio dispositivo a medida pulsando el botón «New Hardware Profile».

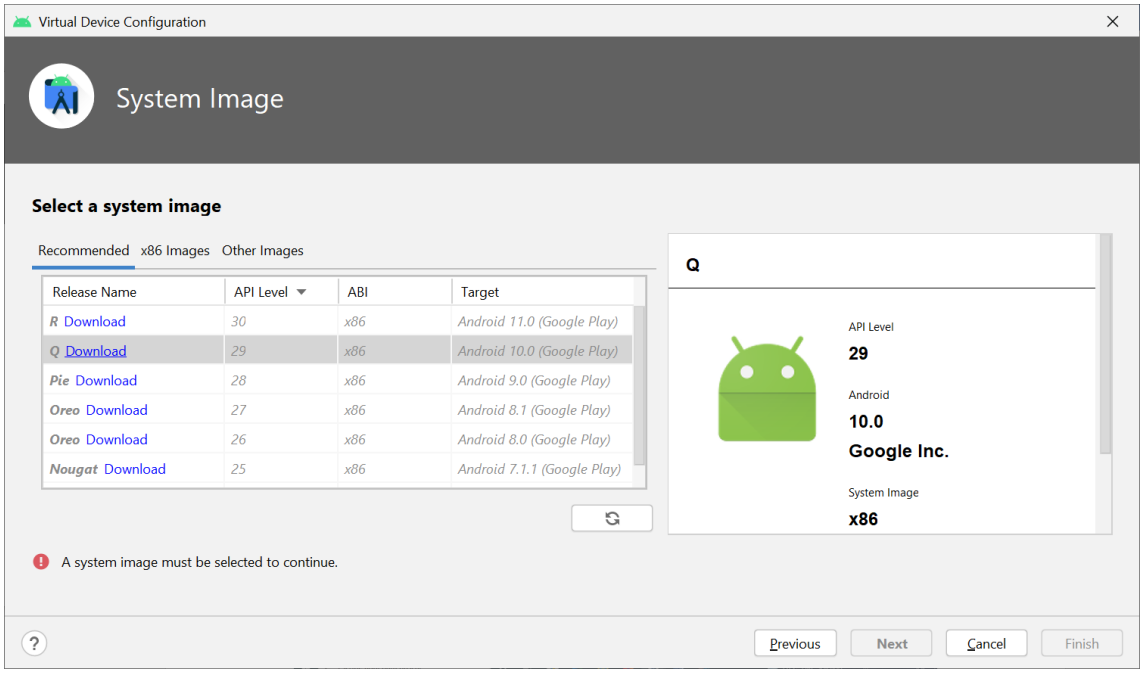
El Administrador de AVD incluye algunos perfiles de hardware previamente cargados, como los de los dispositivos Pixel, que puedes definir o personalizar cuando sea necesario.

Ten en cuenta que solo algunos perfiles de hardware incluyen Play Store por indicación. Esto indica que esos perfiles son totalmente compatibles con CTS (Compatibility Test Suite, es una suite de pruebas gratuita de nivel comercial) y pueden usar imágenes del sistema que incluyan la app de Play Store.

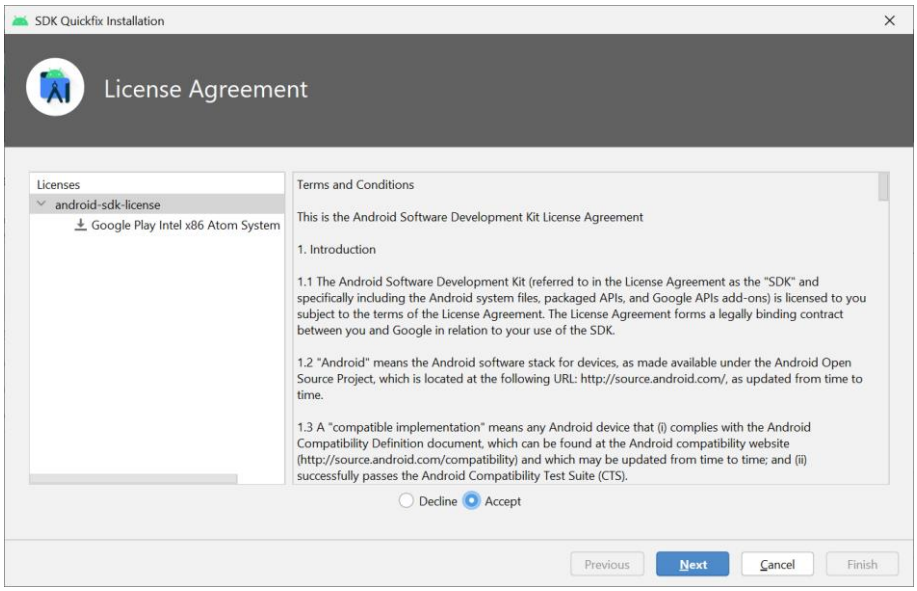
Vamos a seleccionar la siguiente configuración: “Phone” y “Pixel 2”

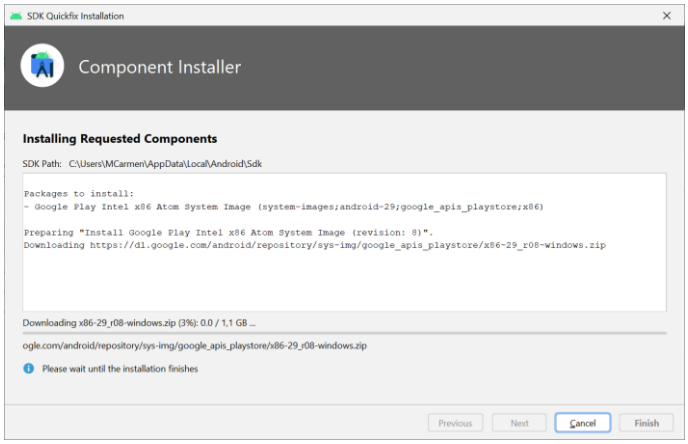


Pulsaremos el botón Next para continuar con el proceso. En la siguiente pantalla seleccionaremos la versión de Android que utilizará el AVD. Aparecerán directamente disponibles las que instalamos desde el SDK Manager al preparar el entorno de desarrollo, aunque tenemos la posibilidad de descargar e instalar nuevas versiones desde esta misma pantalla. En mi caso particular utilizaré la imagen x86 de Android 10 (API 29).

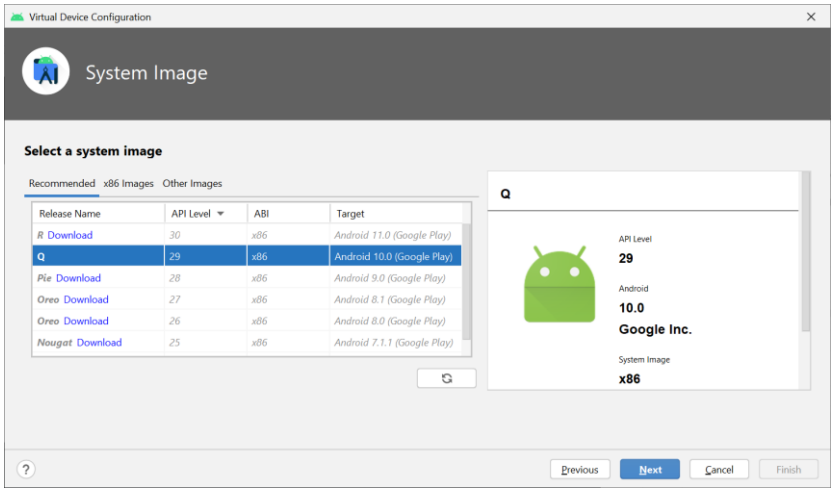


Si no tenemos la versión descargada, deberemos darle a “Download”, y aparecerá un nuevo asistente, donde debemos aceptar los términos y condiciones y esperar que finalice la descarga.

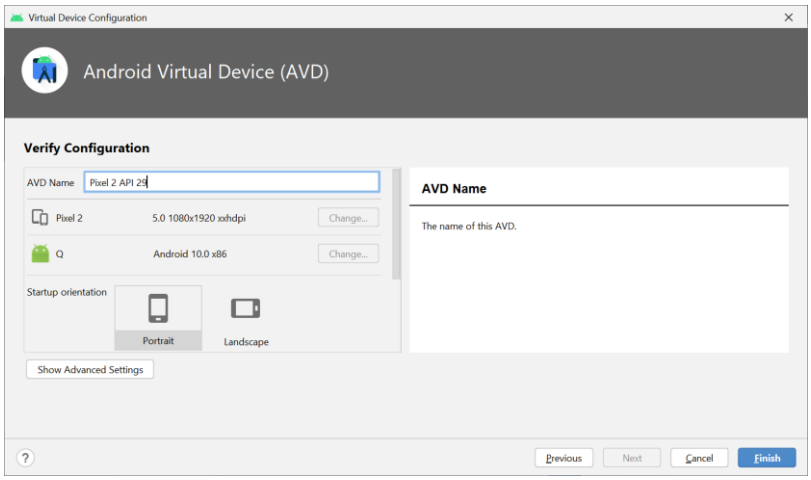




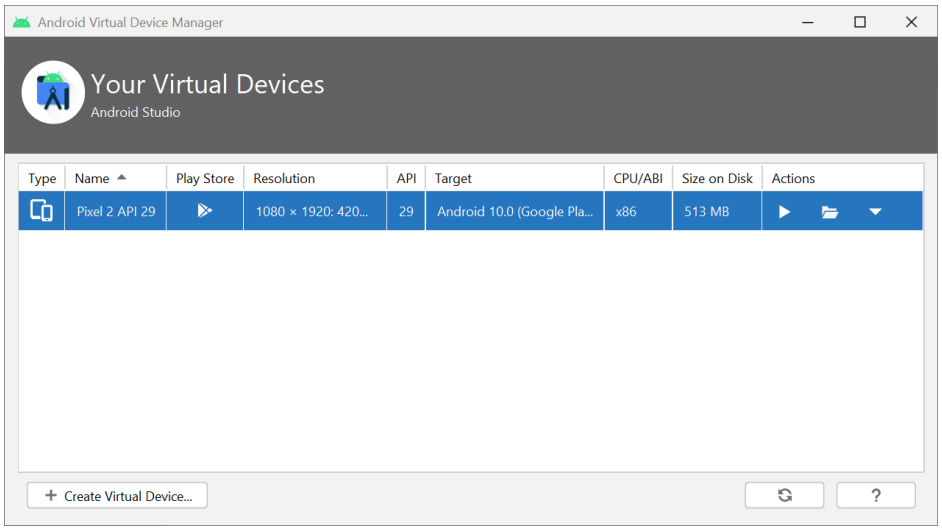
Una vez se ha descargado ya podemos seleccionar la versión que queremos y continuar.



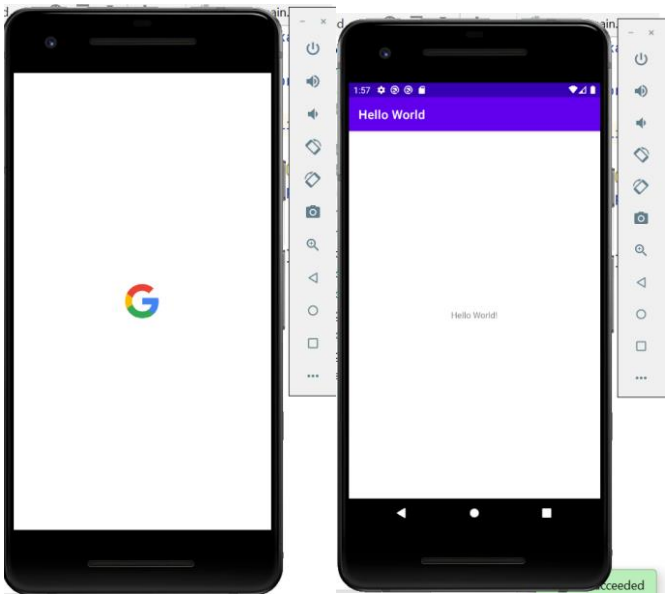
En el siguiente paso del asistente podremos configurar algunas características más del AVD, como por ejemplo la cantidad de memoria que tendrá disponible, si simulará tener cámara frontal y/o trasera, … Recomiendo pulsar el botón «Show Advanced Settings» de la parte inferior para revisar todas las opciones disponibles. Por el momento vamos a dejar todas las opciones por defecto.



Tras pulsar el botón Finish tendremos ya configurado nuestro nuevo AVD, por lo que podremos comenzar a probar nuestras aplicaciones sobre él



Podemos seleccionar nuestro nuevo AVD en la lista desplegable de la barra de herramientas superior y pulsar el botón o menú de ejecutar. Si todo va bien nuestra aplicación se iniciará en el nuevo AVD que hemos creado.



5. Aplicación para contar clicks

Vamos a modificar la aplicación base, para realizar una que tenga un botón que se podrá pulsar, y en el texto aparezca el número de clicks que se han realizado.

5.1. Modificación aspecto

Para ello lo primero que debemos hacer es modificar el layout.

En los ficheros XML de la carpeta res/layout se define la disposición de widgets de las pantallas.

Por el momento, en la pantalla activity_main.xml solamente hay un TextView que muestra el texto Hello world.

Como se ha explicado previamente, se puede modificar con el modo gráfico o el de código.

Comenzamos las modificaciones:

TextView

Para cambiar el texto que se muestra en un TextView, hay que modificar el atributo android:text.

Cambia el valor de este atributo al texto “Has clicado 0 veces”

```
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Has clicado 0 veces"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintTop_toTopOf="parent" />
```

Button

Añadimos un botón que al ser pulsado aumentará el contador. Para ello deberemos escribir el siguiente código:

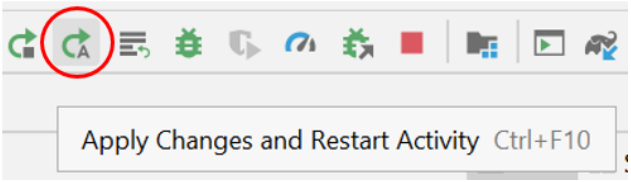
```
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Click aquí!" />
```

Al usar un entorno de desarrollo, se puede observar que nos va sugiriendo opciones, y va completando.

Los atributos layout_width y layout_height, especifican el ancho y alto del elemento.

- El valor "wrap_content" indica que el ancho/alto se debe adaptar al contenido del elemento. En el caso del Button y el TextView, al texto que contienen.
- El valor "match_parent" indica que el ancho/alto debe ser el mismo que el del elemento padre.
  También se puede establecer un ancho/alto de tamaño fijo en píxeles.

Podemos ver los cambios que hemos realizado pulsando sobre el botón aplicar cambios y reiniciar la actividad.



El nuevo diseño se verá de la siguiente forma en el emulador:

ConstraintLayout

En el diseño actual del layout podemos ver que la disposición del botón no es la más adecuada:

Quizá quedaría mejor si el botón estuviera dispuesto debajo del TextView...

Para disponer los elementos en un ConstraintLayout hay que definir las constraints en cada elemento.

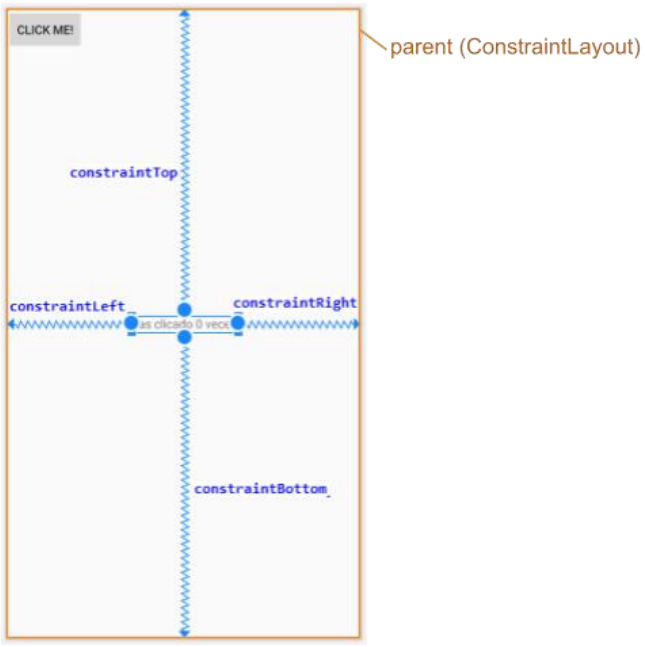
Podemos imaginar las constraints como muelles que tiran de un elemento en las cuatro direcciones (arriba, abajo, izquierda y derecha). El elemento queda posicionado en equilibrio entre las fuerzas de los muelles que se han definido.

En el caso del TextView vemos que ya hay definidas cuatro *constraints*:

```
app:layout_constraintBottom_toBottomOf="parent"
app:layout_constraintLeft_toLeftOf="parent"
app:layout_constraintRight_toRightOf="parent"
app:layout_constraintTop_toTopOf="parent" />
```

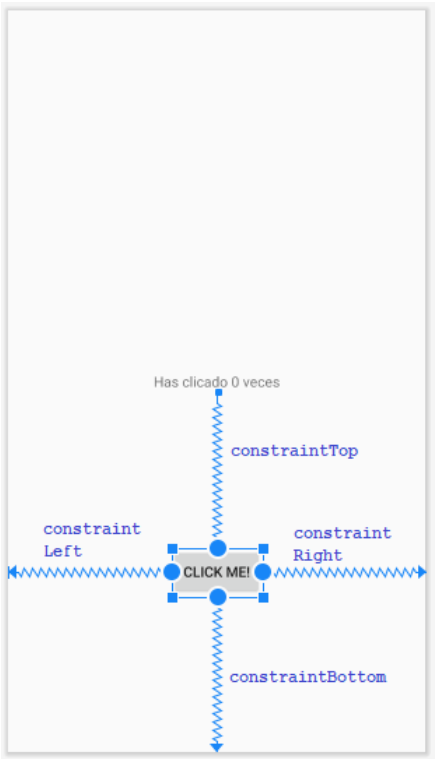
En cada constraint se indica hacia dónde debe tirar el "muelle". Podemos ver que:

- La constraintBottom tira del TextView hacia la parte inferior (toBottomOf) de su parent (El elemento padre del TextView es el ConstraintLayout).
- La constraintLeft tira del TextView hacia la parte izquierda (toLeftOf) de su parent.
- La constraintRight tira del TextView hacia la parte derecha (toRightOf) de su parent.
- La constraintTop tira del TextView hacia la parte superior (toTopOf) de su parent.



Como vemos, el TextView queda en equilibrio vertical entre las constraints Top y Bottom, y horizontal entre Left y Right, y de esta manera queda centrado dentro del ConstraintLayout.

Ahora, definiremos las constraints del Button para posicionarlo de esta manera:



Las constraint de los lados y la de abajo, vemos que son fáciles, sería definirlas igual que las del textview, sin embargo, la superior, constraintTop, tira del botón hacia la parte inferior del TextView.

Para hacer referencia al elemento padre hemos visto que podemos usar parent. Pero ¿cómo hacemos referencia al TextView? Para ello hay que asignarle un identificador al TextView.

Para asignar un identificador a un elemento hay que usar el atributo **android:id**, y en el valor hay que poner **@+id/** antes del identificador que queramos asignarle.

Para asignar el identificador contadorDeClics al TextView, le añadiremos este atributo:

```
<TextView
    android:id="@+id/contadorDeClics"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Has clicado 0 veces"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintTop_toTopOf="parent" />
```

Ahora ya podemos añadir la constraintTop al botón y hacer que tire de él hacia la parte inferior (toBottomOf) del TextView:

```
<Button
    android:id="@+id/botonContador"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Click aquí!"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintTop_toTopOf="@+id/contadorDeClics"/>
```

Aprovechamos para poner id al botón.

Si volvemos a probarlo, veremos que la disposición ya es más adecuada, pero por ahora el botón no hace nada, ni el textView cambia de valor, para ello tenemos que añadir el comportamiento.

5.2. Modificación Comportamiento

Para agregar un comportamiento al botón debemos realizarlo en el código Java. Escribiremos el código en el fichero MainActivity.java

```java
package com.example.helloworld;

import ...

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
}
```

En este fichero hay definida la clase MainActivity, y en ella el método onCreate().

Este método onCreate() es el que el Sistema Android ejecuta en primer lugar cuando se inicia nuestra App.

En este método lo primero que debes observar es la llamada setContentView(R.layout.activity_main). Esta llamada establece el layout activity_main.xml, es decir, lo que se mostrará en pantalla es lo que hemos puesto en este fichero de layout (el TextView y el Button).

Una vez visto esto vayamos al código necesario para nuestro contador. Necesitaremos:

- Una variable int que vaya almacenando el contador
- Programar el Button para que, cada vez que se haga click, augmente el contador y actualice el TextView

Creamos la variable para almacenar el número de clicks:

```java
public class MainActivity extends AppCompatActivity {

    int contador;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
}
```

Para programar el comportamiento del Button, y luego actualizar el texto del TextView, necesitaremos tener una referencia a estos dos elementos. Para esto utilizaremos findViewById.

Definimos una variable para cada elemento y llamaremos al método findViewById() para vincular estas variables con los elementos del layout correspondientes. Al método findViewById() hay que pasarle el identificador que hayamos definido en el fichero XML.

```
public class MainActivity extends AppCompatActivity {

    int contador;
    TextView contadorDeClics;
    Button botonClick;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        contadorDeClics = findViewById(R.id.contadorDeClics);
        botonClick = findViewById(R.id.botonContador);
    }

}
```

Las llamadas a findViewById() las hemos hecho después de establecer el layout activity_main.xml. El método findViewById() buscará los elementos en este fichero.

Una vez tenemos vinculados el Button y el TextView, el siguiente paso es programar el botón para que responda al CLICK (aumentando el contador y mostrando el texto en el TextView).

Para que el botón responda al evento CLICK, hay que hacer una llamada al método setOnClickListener() y pasarle un objeto de clase View.OnClickListener. En el método onClick() de este objeto programaremos lo que queremos que se haga cuando se pulse el botón:

```
botonClick.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        //Aumentar el contador
        //Mostrar el contador
    }
});
```

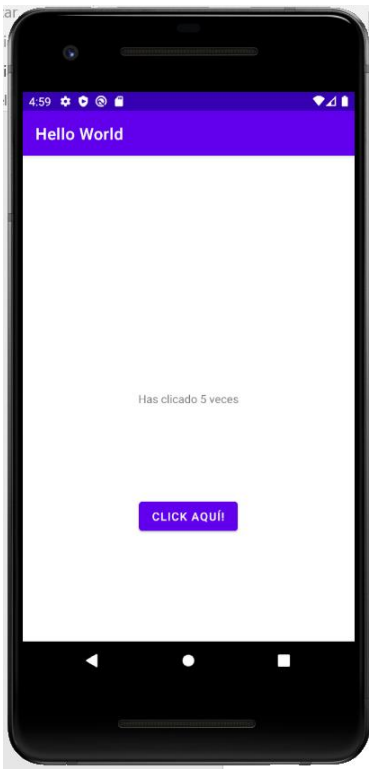
Ahora, dentro del método onClick() del *listener*, solo hay que incrementar el contador y cambiar el texto del TextView. Para cambiar el texto hacemos una llamada al método setText() del TextView.

```
//Aumentar el contador
contador++;
//Mostrar el contador
contadorDeClics.setText("Has clicado " + contador + " veces");
```

Finalmente, el código de la aplicación queda así:

```java
public class MainActivity extends AppCompatActivity {

    int contador;
    TextView contadorDeClics;
    Button botonClick;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        contadorDeClics = findViewById(R.id.contadorDeClics);
        botonClick = findViewById(R.id.botonContador);

        botonClick.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                //Aumentar el contador
                contador++;
                //Mostrar el contador
                contadorDeClics.setText("Has clicado " + contador + " veces");
            }
        });
    }
}
```

Si clicamos 5 veces se mostrará algo como lo siguiente:



6. Ejercicio 1

Vamos a realizar el siguiente ejemplo, en el que tendremos los siguientes elementos:

- TextView que muestra la frase “Introduzca nombre:”
- EditText que permite introducir un texto
- Button que recogerá el texto introducido en el EditText, y lo mostrará en el siguiente TextView
- TextView que muestra la frase “El nombre es XXX” donde XXX es el nombre introducido.