

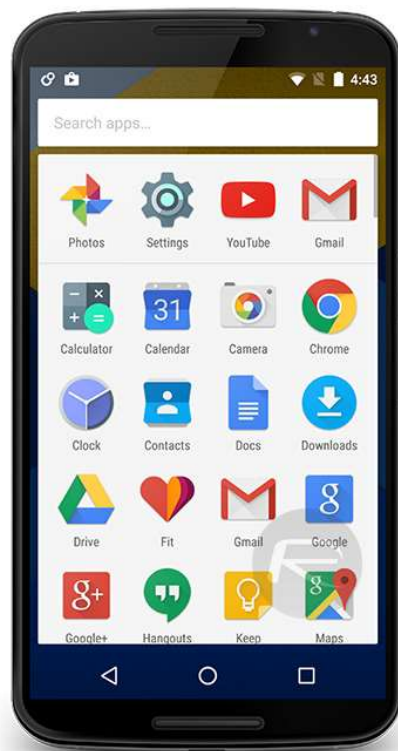
## **UNIDAD DE TRABAJO 1 - ANDROID EL SISTEMA OPERATIVO.**

### **INTRODUCCIÓN.**

1.1	¿Qué es Android?.....	2
1.2	Programación para Android.....	6
1.3	No tengo un teléfono Android, ¿Puedo completar este curso? .....	7
1.4	¿Qué necesito saber?.....	8
1.5	¿Se desarrolla igual para un equipo de escritorio (Desktop) que para un dispositivo Android?8	
1.6	Prerrequisitos... ¿Por dónde empezamos? .....	9
1.7	Primer Contacto: Instalando el Android Studio .....	10
1.8.	Nuestro primer proyecto .....	20
1.9.	Primer contacto con el código. ¿Dónde está el java? .....	24
1.10.	Programando sin escribir líneas de código .....	27
1.11.	Introduciendo un poco de código .....	29
1.12.	Probando, probando... ..	35
1.13.	Entendiendo un poco más el código.....	36
1.14.	Otros emuladores .....	36

## **1.1 ¿Qué es Android?**

Android es un Sistema Operativo de última generación basado en Linux y creado por Google para sus dispositivos. Parte de su éxito radica en su interfaz de usuario basada en la tecnología DMI (Direct Manipulation Interface), un tipo de interacción hombre-ordenador que representa objetos gráficos de interés en pantalla de manera rápida, reversible y de acciones incrementales y que proporcionan feedback, es decir, proporcionan en todo momento un resultado que el usuario puede interpretar fácilmente. Dicho de otro, utiliza metáforas del mundo real para utilizar los objetos representados en pantalla, que ayudan de manera muy fácil al usuario a interpretar lo que tiene en pantalla, cómo utilizarlo y los resultados obtenidos. Acciones como el swiping (pasar la mano por la pantalla) o tapping (presionar ligeramente para seleccionar un objeto) son muy naturales para los usuarios y hasta los niños más pequeños de manera intuitiva pueden utilizar los dispositivos con Android.



Android está diseñado principalmente para dispositivos con pantallas táctiles, desde Smartphones y Tablets hasta las Smarts Tv o televisiones inteligentes o los Smarts Watchs. También se usa en cámaras digitales y otros muchos dispositivos electrónicos.

# DESARROLLO DE APLICACIONES MULTIPLATAFORMA

## T1. ANDROID. EL SISTEMA OPERATIVO



Android está programado en C/C++ y tiene licencia *open source*, aunque muchos de los dispositivos que usan Android funcionan con una combinación de software libre y propietario.

La primera versión de Android se publicó en 2008, a finales del 2015 se publicó la versión 6 (Marshmallow) y a mediados del 2016 se publicó la versión N. Hay versiones compiladas para plataformas ARM, MIPS y x86.

Cada versión de Android tiene un nombre en clave y un nivel de API. Este nivel de API especifica un conjunto de librerías que se utilizan para desarrollar una aplicación. Por ejemplo, la versión 4.4 tiene el nombre en clave de Kitkat, y su nivel de API es el nivel 19.

Una de las mayores complicaciones que encontrarás cuando programes para Android es intentar dar soporte al mayor número posible de APIs. Cuando desarrolles, deberás especificar una versión mínima de API para la que funcionará tu aplicación *android:minSdkVersion* y una versión máxima *android:maxSdkVersion*. Estos dos parámetros definirán el rango de versiones para las que tu App funciona, y por tanto, los dispositivos sobre los que va a funcionar:

Versiones de Android:

Versión	Nivel de API	Nombre	Algunas de las novedades
1.0	1	Apple Pie	Primera versión comercial
1.1	2	Banana Bread	Resolvió los problemas de la 1.0
1.5	3	Cupcake	Núcleo de Linux 2.6.27
1.6	4	Donut	Incluye la Galería, y el motor Text-to-speech

**DESARROLLO DE APLICACIONES MULTIPLATAFORMA**  
**T1. ANDROID. EL SISTEMA OPERATIVO**

2.0/2.1	5/7	Eclair	GUI renovada, calendario y Google Maps renovado. Fondos de pantalla animados
2.2.x	8	Froyo	Soporte para pantallas de HD y optimización de memoria y rendimiento
2.3.x	9/10	Gingerbread	Actualizaciones variadas del diseño de la interfaz, mejoras en audios y gráficos, soporte de video y voz con Google Talk, mejora en el software de la cámara y eficiencia de la batería
3.x	11/13	Honeycomb	Añadida la ActionBar y la barra de sistema, teclado rediseñado, mejoras en el HTTPS, posibilidad de acceso a tarjetas SD. Multitarea simplificada, soporte para procesadores multinúcleo.
4.0.x	14/15	Ice Cream Sandwich	Numerosas optimizaciones y corrección de errores. Carpetas con Drag & Drop, captura de pantalla integrada, cámara mejorada, corrector ortográfico del teclado mejorado, mejoras en gráficos y bases de datos.
4.1	16	Jelly Bean	Interfaz de usuario remodelada con triple buffer y 60 fps. Mejoras en la redimensión de widgets. Inclusión de la barra de notificaciones y gestos.
4.2	17	Jelly Bean (Gummy Bear)	Soporte multiusuario, foto esfera y acceso rápido a la barra de notificaciones
4.3	18	Jelly Bean	Soporte de Bluetooth de baja energía, Open GL 3.0 y mejoras en la seguridad. Autocompletar en el teclado de marcación. Nueva interfaz para la cámara.
4.4	19	KitKat	Solucionados numerosos errores, diseño renovado para el marcador de teléfono, aplicación de contactos, MMS y otros elementos de la IU. Optimizado para sistemas con poca RAM y procesador
5.0	21	Lollipop	Nuevo diseño de la interfaz de usuario (Material Design). Nuevas formas de controlar las notificaciones, ahorro de batería mejorado. Mejoras en la multitarea y soporte para 64 bit.
5.1.	22	Lollipop MR1	Nuevo soporte para teléfonos con doble SIM, posibilidad de ocultar zona Wifi y notificaciones emergentes y otras mejoras de estabilidad y rendimiento
6	23	Marshmallow	Administrador de permisos y notificaciones sensibles al contexto (Now on tap), API para Voz y huellas dactilares.
7	24	Nougat	Compatibilidad con ventanas múltiples, actualizaciones de plantillas, notificaciones agrupadas, varias

## DESARROLLO DE APLICACIONES MULTIPLATAFORMA

### T1. ANDROID. EL SISTEMA OPERATIVO

			optimizaciones
8	26	Oreo	Nuevas Notificaciones, PIP, Marco Autofill, Fuentes XML, WebView API, etc.
9	28	Pie	Inteligencia artificial y batería adaptativa, soporte cámaras externas, mejoras en notificaciones y seguridad..
10	29	Android 10	Modo oscuro, 5G, Focus Mode, Live Caption

**ACTIVIDAD:** Es muy buena idea que te registres en el canal de Youtube de los desarrolladores de Android (<https://www.youtube.com/user/androiddevelopers>) para estar al tanto de las novedades que ocurren en el mundo "androide". En especial, puede buscar uno de los videos más vistos llamado Android Demo, donde se presentan los teléfonos móviles que funcionan con Android.

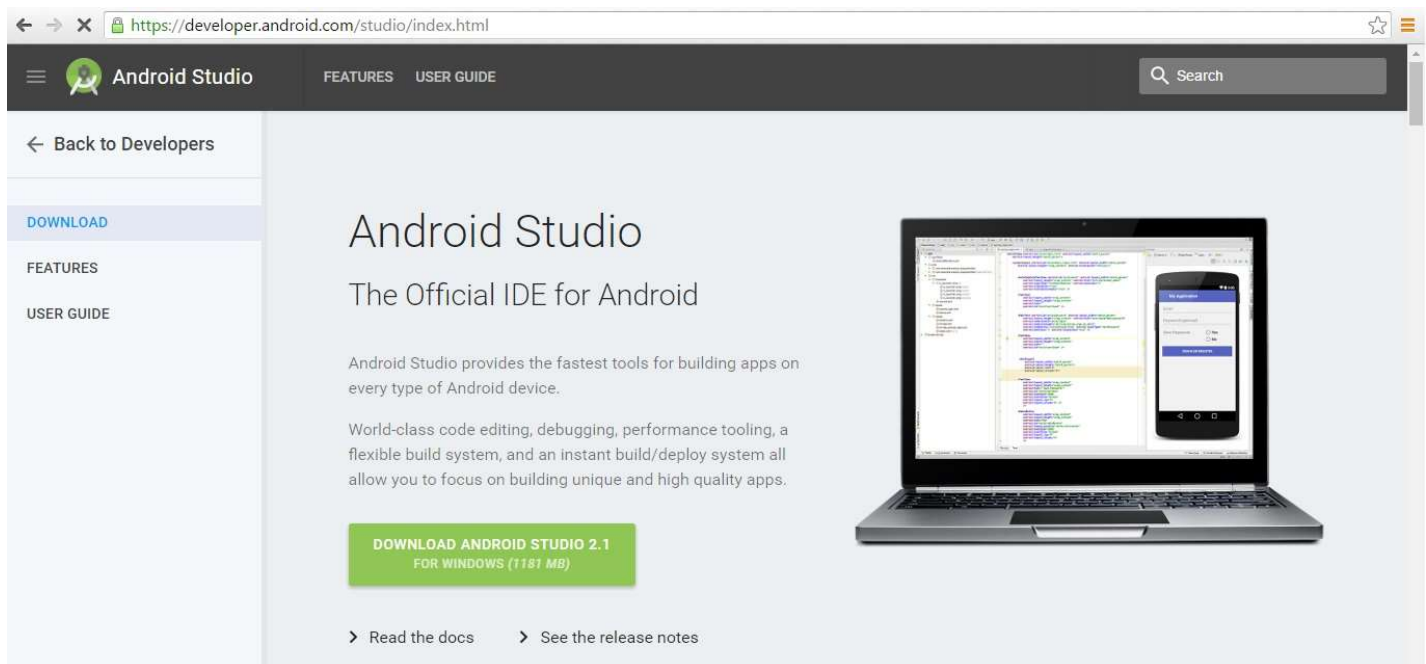


## 1.2 Programación para Android

En este curso utilizaremos como recursos para programar un paquete de herramientas llamado Android Studio. Android Studio incorpora un entorno integrado de desarrollo (IDE – Integrated Development Environment) propio muy potente y plenamente documentado.



En la actualidad es la única herramienta mantenida por Google para el desarrollo de aplicaciones en Android y en el momento de escribir este texto la última versión estable es Android Studio 2.1. Millones de desarrolladores utilizan ya esta plataforma para programar aplicaciones para dispositivos móviles a través de este entorno integrado de desarrollo. Prácticamente la totalidad de las aplicaciones más descargadas del Play Store de Google están ya desarrolladas con este IDE. Aplicaciones para Android como Whatsapp, Line o Spotify han utilizado este IDE como entorno de programación.



Android Studio incorpora los siguientes componentes:

- El entorno integrado de desarrollo (propriadamente llamado Android Studio). Integra **IntelliJ** como una potente característica que ayuda al programador a completar el código rápidamente y ser más productivo.
- Un sistema para construcción de aplicaciones basado en Gradle. Gradle es una herramienta que recopila todos los elementos de tu App (archivos java, xml, recursos, etc) y genera un

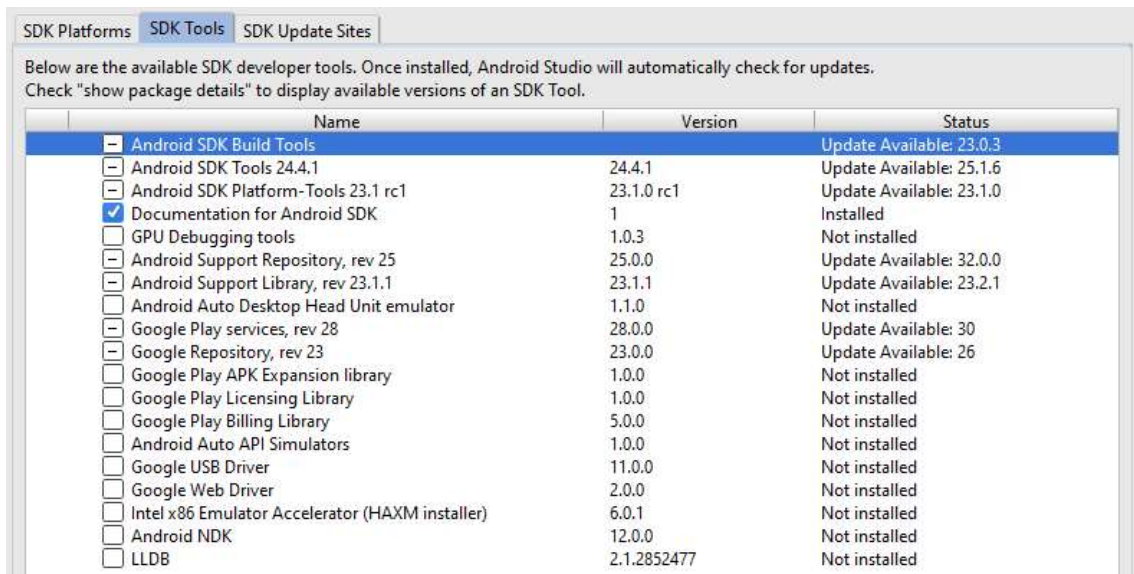


## DESARROLLO DE APLICACIONES MULTIPLATAFORMA

### T1. ANDROID. EL SISTEMA OPERATIVO

archivo APK. Un archivo APK es el archivo que contiene el formato usado para la instalación de una aplicación en un dispositivo Android.

- Android SDK: Es un componente que incluye un conjunto de herramientas de desarrollo y de depuración de programas.



	Name	Version	Status
<input checked="" type="checkbox"/>	Android SDK Build Tools		Update Available: 23.0.3
<input type="checkbox"/>	Android SDK Tools 24.4.1	24.4.1	Update Available: 25.1.6
<input type="checkbox"/>	Android SDK Platform-Tools 23.1 rc1	23.1.0 rc1	Update Available: 23.1.0
<input checked="" type="checkbox"/>	Documentation for Android SDK	1	Installed
<input type="checkbox"/>	GPU Debugging tools	1.0.3	Not installed
<input type="checkbox"/>	Android Support Repository, rev 25	25.0.0	Update Available: 32.0.0
<input type="checkbox"/>	Android Support Library, rev 23.1.1	23.1.1	Update Available: 23.2.1
<input type="checkbox"/>	Android Auto Desktop Head Unit emulator	1.1.0	Not installed
<input type="checkbox"/>	Google Play services, rev 28	28.0.0	Update Available: 30
<input type="checkbox"/>	Google Repository, rev 23	23.0.0	Update Available: 26
<input type="checkbox"/>	Google Play APK Expansion library	1.0.0	Not installed
<input type="checkbox"/>	Google Play Licensing Library	1.0.0	Not installed
<input type="checkbox"/>	Google Play Billing Library	5.0.0	Not installed
<input type="checkbox"/>	Android Auto API Simulators	1.0.0	Not installed
<input type="checkbox"/>	Google USB Driver	11.0.0	Not installed
<input type="checkbox"/>	Google Web Driver	2.0.0	Not installed
<input type="checkbox"/>	Intel x86 Emulator Accelerator (HAXM installer)	6.0.1	Not installed
<input type="checkbox"/>	Android NDK	12.0.0	Not installed
<input type="checkbox"/>	LLDB	2.1.2852477	Not installed

- Emulador de Android: Un conjunto de dispositivos virtuales para probar tus aplicaciones sin necesidad de tener un dispositivo físico.

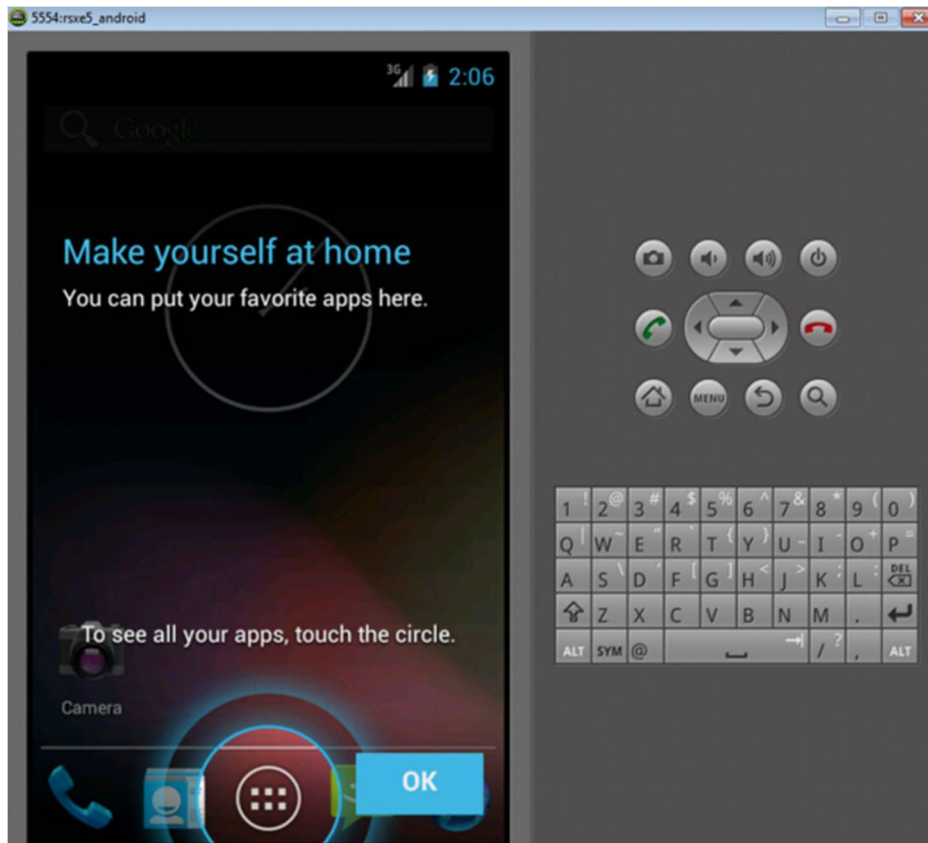
ACTIVIDAD: Mientras sigues leyendo, pon a descargar el Android Studio. Te hará falta en unos momentos. ¡Ah! Y si no tienes la última versión del JDK de Java, descárgalo también.

### 1.3 No tengo un teléfono Android, ¿Puedo completar este curso?

Por supuesto que sí.



No necesitas gastarte un solo centavo en comprar un dispositivo para poder probar los programas que aprendas a hacer aquí. Utilizarás un emulador proporcionado por el SDK de android que funciona con una imagen del software de Android. Este emulador monta una imagen del sistema operativo, es decir, el mismo software que lleva cualquier dispositivo android, se monta en una máquina virtual formando el entorno de emulación. Sobre este entorno, se puede ejecutar prácticamente cualquier programa que hayas realizado (a excepción de algunas funcionalidades):



## 1.4 ¿Qué necesito saber?

- Necesitar tener conocimientos previos de programación. Y más concretamente de Java. Necesitas saber conceptos como bucles, arrays, funciones, etc.
- Un poco de diseño de aplicaciones y entornos de desarrollo. Aunque no es estrictamente necesario, siempre te ayudará a entender conceptos y a crear aplicaciones profesionales.
- Un poco de bases de datos. Aunque no es fundamental, parte del temario cubre el tratamiento de información con SQLite, y viene bien entender un poco cómo funcionan los sistemas gestores de bases de datos.
- Manejo básico de sistemas operativos y redes de ordenadores.

En resumen, eso es lo que necesitas saber, programar en java y manejar un IDE. El resto son añadidos que te vendrán bien.

## 1.5 ¿Se desarrolla igual para un equipo de escritorio (Desktop) que para un dispositivo Android?

Pues va a ser que no.

Tienes que tener en cuenta unas cuantas reglas:

- Un dispositivo android generalmente es un dispositivo portable y pequeño, con capacidad de procesamiento mucho más limitada, además:
  - Las pantallas son pequeñas



# DESARROLLO DE APLICACIONES MULTIPLATAFORMA

## T1. ANDROID. EL SISTEMA OPERATIVO

- Los teclados si existen, son pequeños
- Los dispositivos con los que se señalan los objetos son torpes, molestos e imprecisos, por ejemplo, unos dedos gordos y pantallas LCD “multitouch”
- Los dispositivos tienen conexión a redes de datos (4G, Internet) pero esa conectividad es limitada o de ancho de banda pequeño, e incluso a veces intermitente.

Por tanto, tienes que aceptar en tu mente de programador el concepto de que un Smartphone no es un ordenador, es un teléfono, y que una tablet no es un ordenador, es una tablet.

No hay nada que peor siente a un usuario que una “App” convierta a su teléfono móvil en un “NO TELÉFONO”, es decir, crear un App que le tome el control de tantos recursos que bloquee el acceso a otras aplicaciones y por ejemplo, no le permita coger el teléfono cuando está recibiendo una llamada.

Por tanto, tienes que tener en cuenta todos estos aspectos cuando desarrolles en Android.

## 1.6 Prerrequisitos... ¿Por dónde empezamos?

Desde Android Studio 2.2. el entorno de desarrollo se instala junto el open JDK, por lo que no es necesario instalar el JDK de Java.

Si instalas una versión anterior, tendrás que descargar la última versión de JDK e instálalo de la página siguiente:

<http://www.oracle.com/technetwork/java/javase/downloads/index.html>

(Si cuando leas esto, el enlace ya no es válido, busca en google Java JDK y descárgalo).



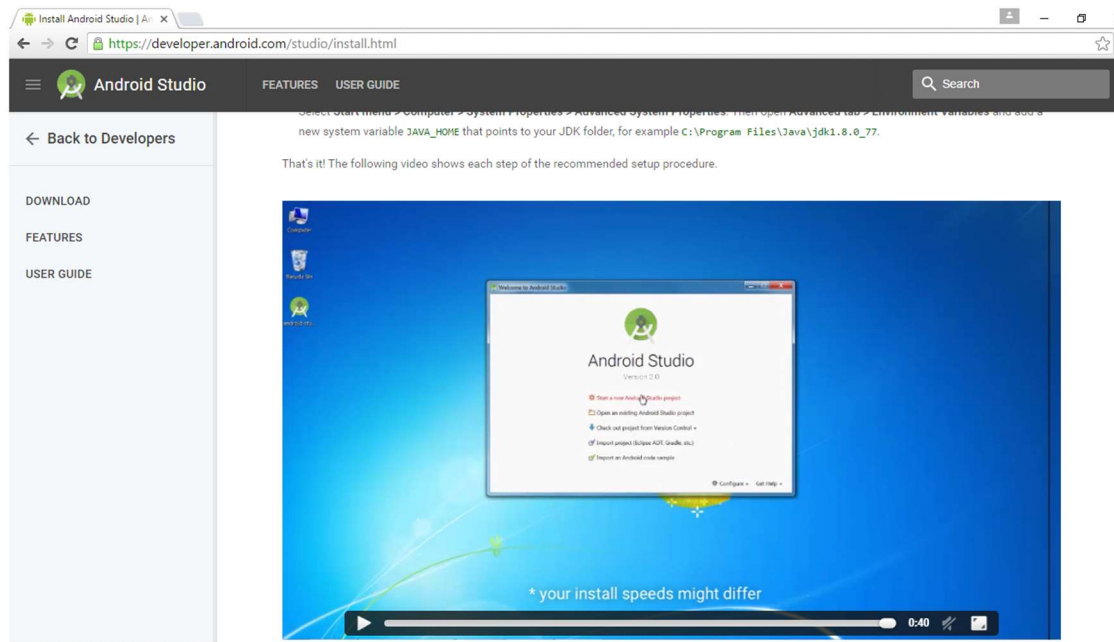
1. ¿Has descargado ya el Android Studio? Si no, vete a la página <https://developer.android.com/studio/index.html> y descárgalo. Ocupa alrededor de 1GB. Tardará un buen rato, así que es buen momento para que te tomes un café y reflexiones sobre todo lo que has leído hasta ahora.

## 1.7 Primer Contacto: Instalando el Android Studio

Puedes descargar Android Studio desde <https://developer.android.com/studio>

En la página de documentación de google tienes un video describiendo el proceso de instalación. Verás que es muy sencillo:

<https://storage.googleapis.com/androiddevelopers/videos/studio-install-windows.mp4>

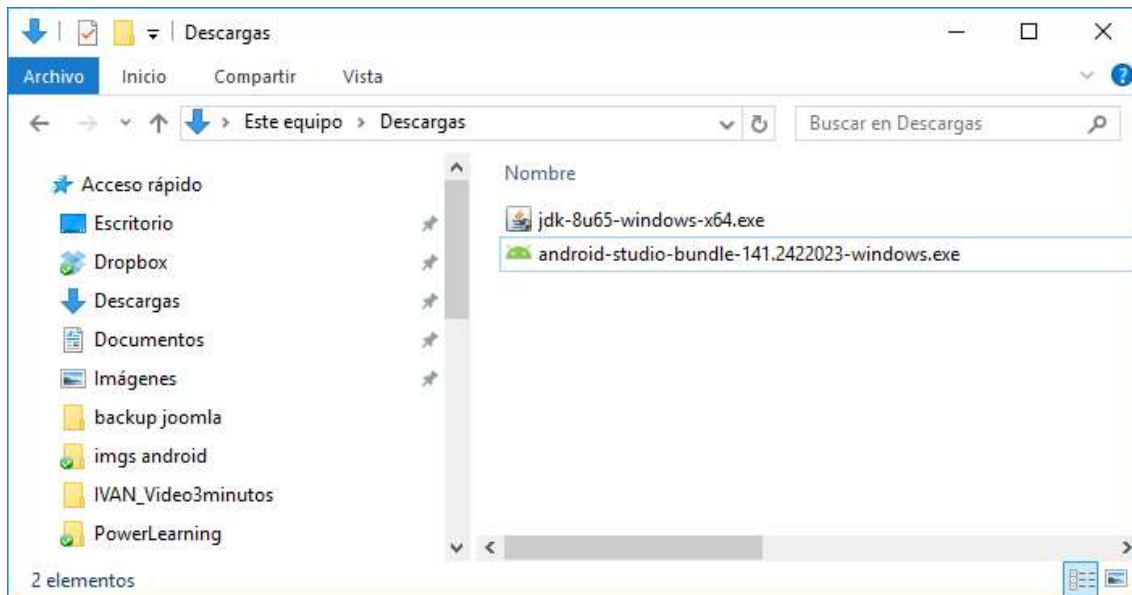


A continuación, te resumimos la instalación en varios pasos:

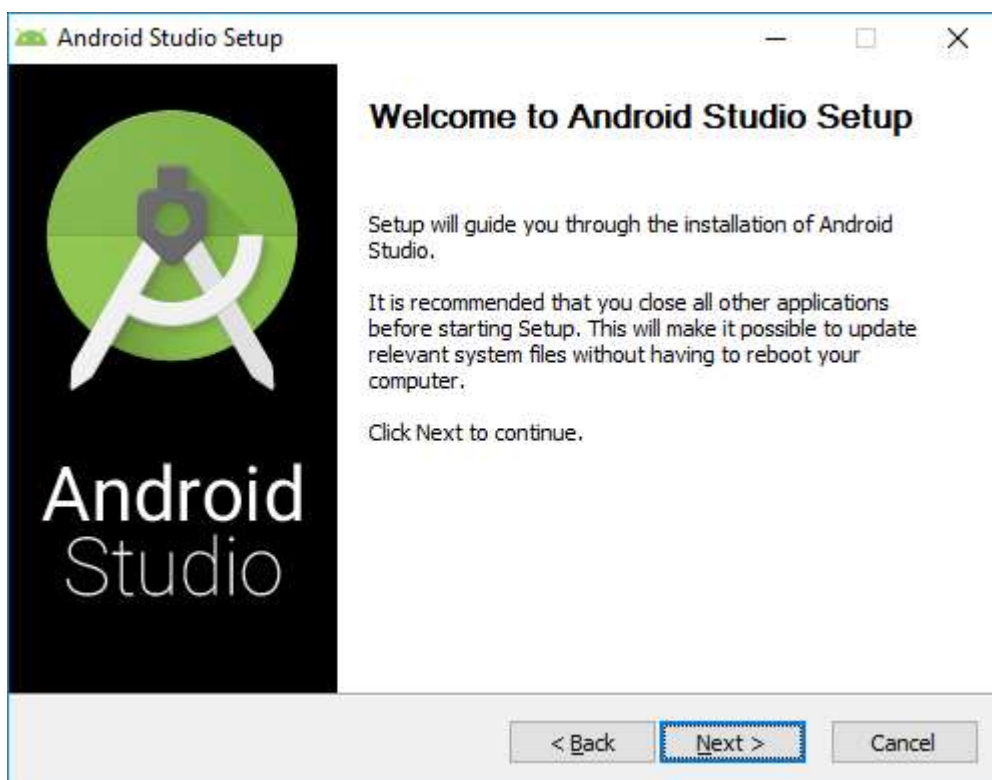
Abre la carpeta donde hayas descargado el Android Studio y ejecuta el archivo .exe para comenzar la instalación:

## DESARROLLO DE APLICACIONES MULTIPLATAFORMA

### T1. ANDROID. EL SISTEMA OPERATIVO



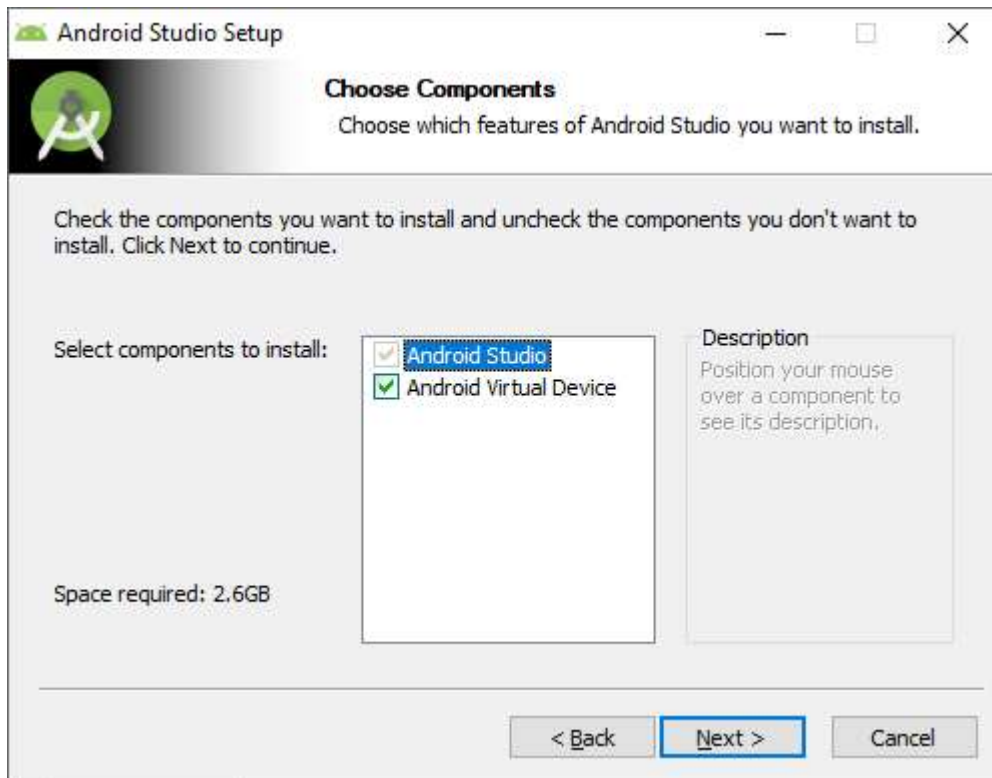
Sigues los pasos de la instalación básica, con el asistente que te guiará mediante un proceso fácil de tipo “clic-siguiente-clic-siguiente....”:



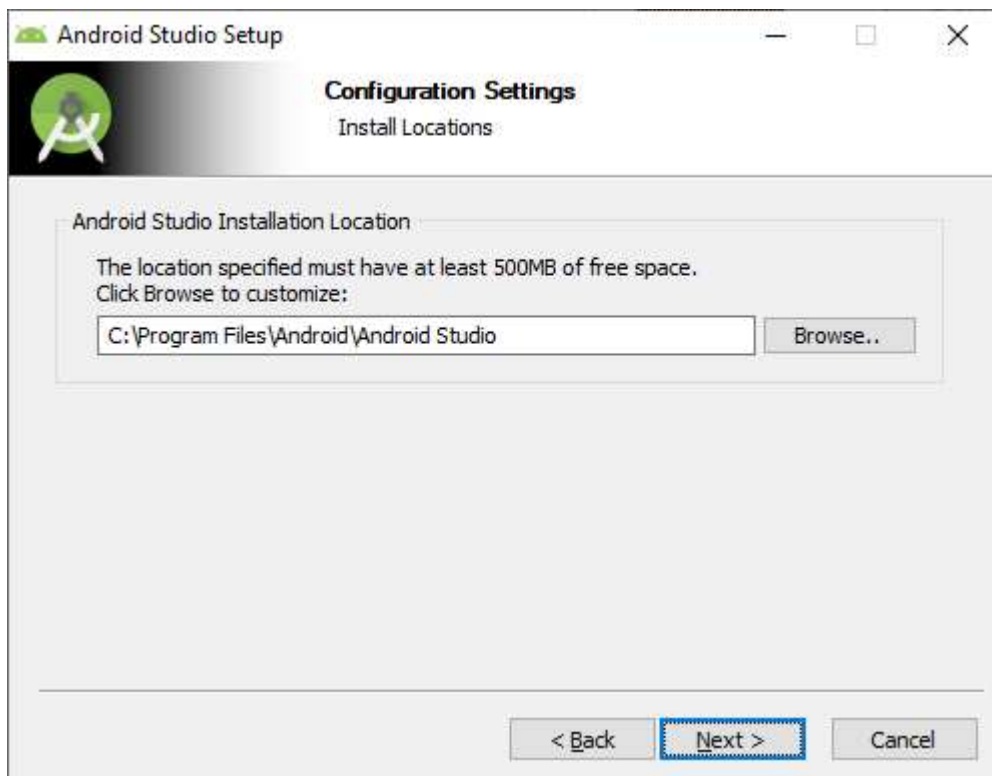
Pulsa en Next y selecciona todos los componentes a instalar. Vuelve a pulsar en next un par de veces más:

## DESARROLLO DE APLICACIONES MULTIPLATAFORMA

### T1. ANDROID. EL SISTEMA OPERATIVO



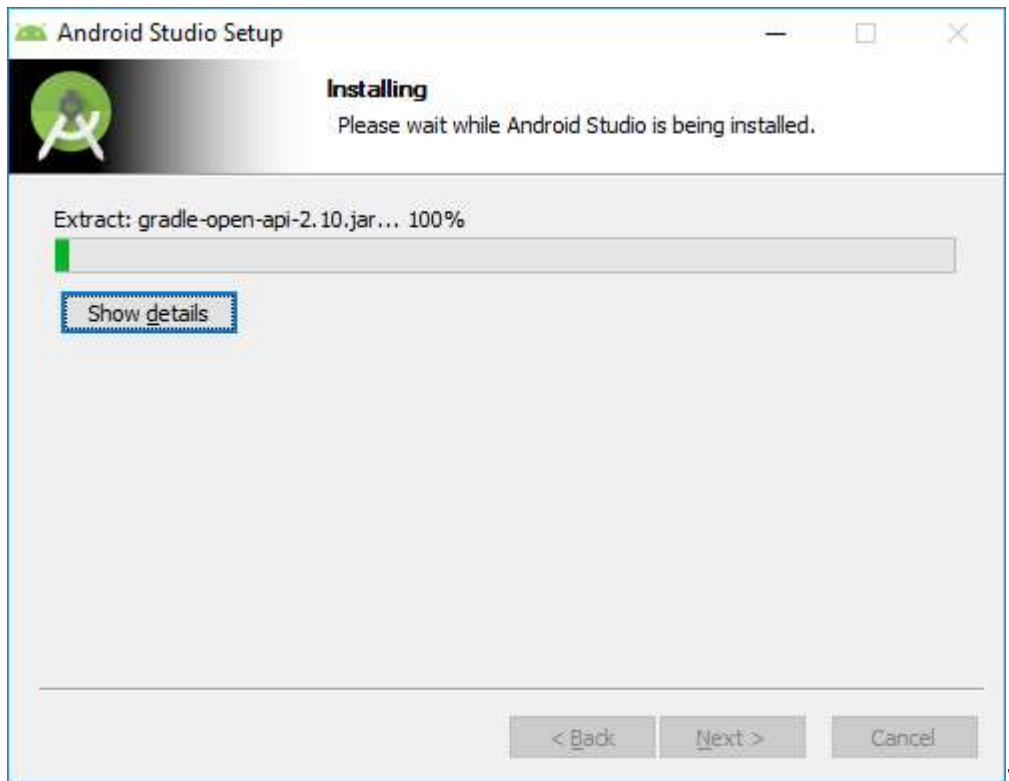
Selecciona la ubicación donde instalarás el entorno:



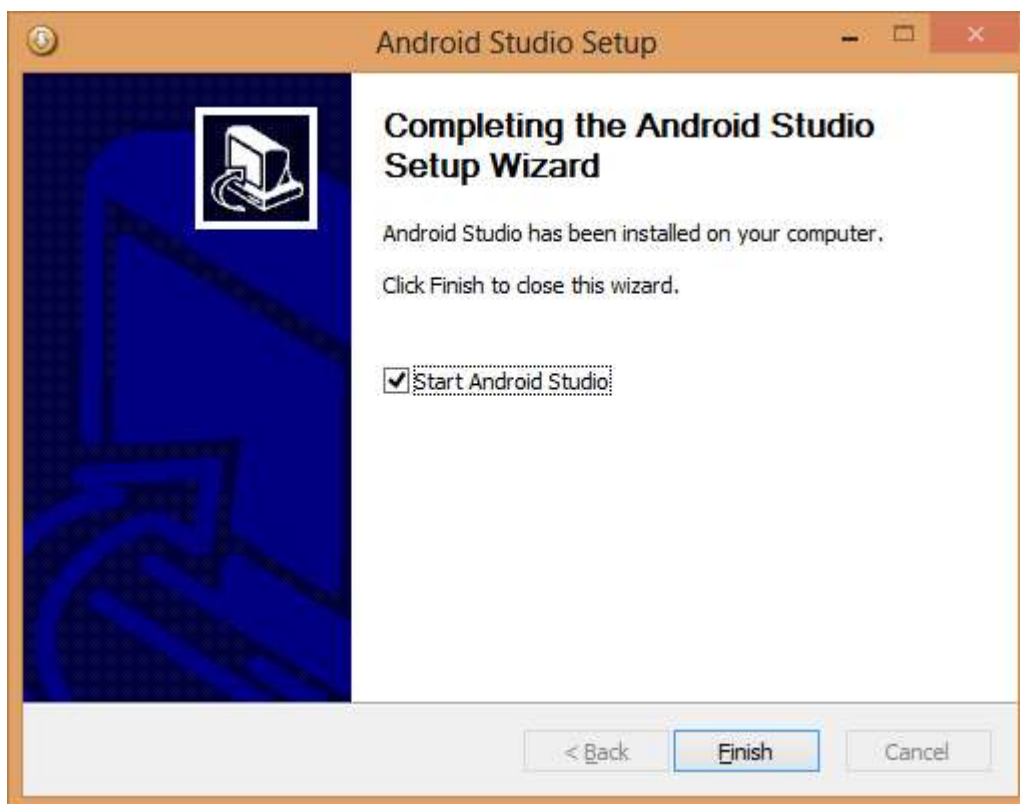
Pulsa otro par de veces en "Next" y comenzará la instalación:

## DESARROLLO DE APLICACIONES MULTIPLATAFORMA

### T1. ANDROID. EL SISTEMA OPERATIVO



Después de esperar un buen rato, ha concluido la instalación y podemos empezar a configurar el entorno.

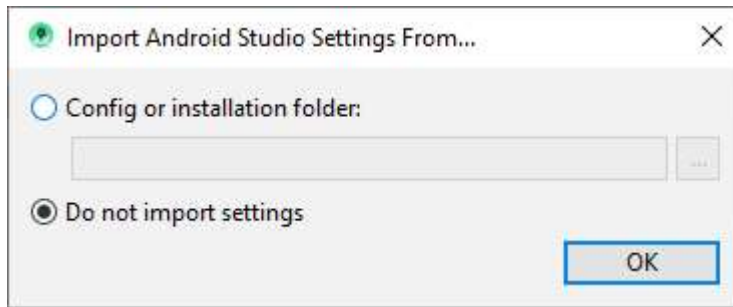


A continuación arrancamos Android Studio. Si es la primera vez que lo instalamos hay que indicarlo en la siguiente pantalla, si tuviéramos versiones anteriores de Android Studio, existe la posibilidad

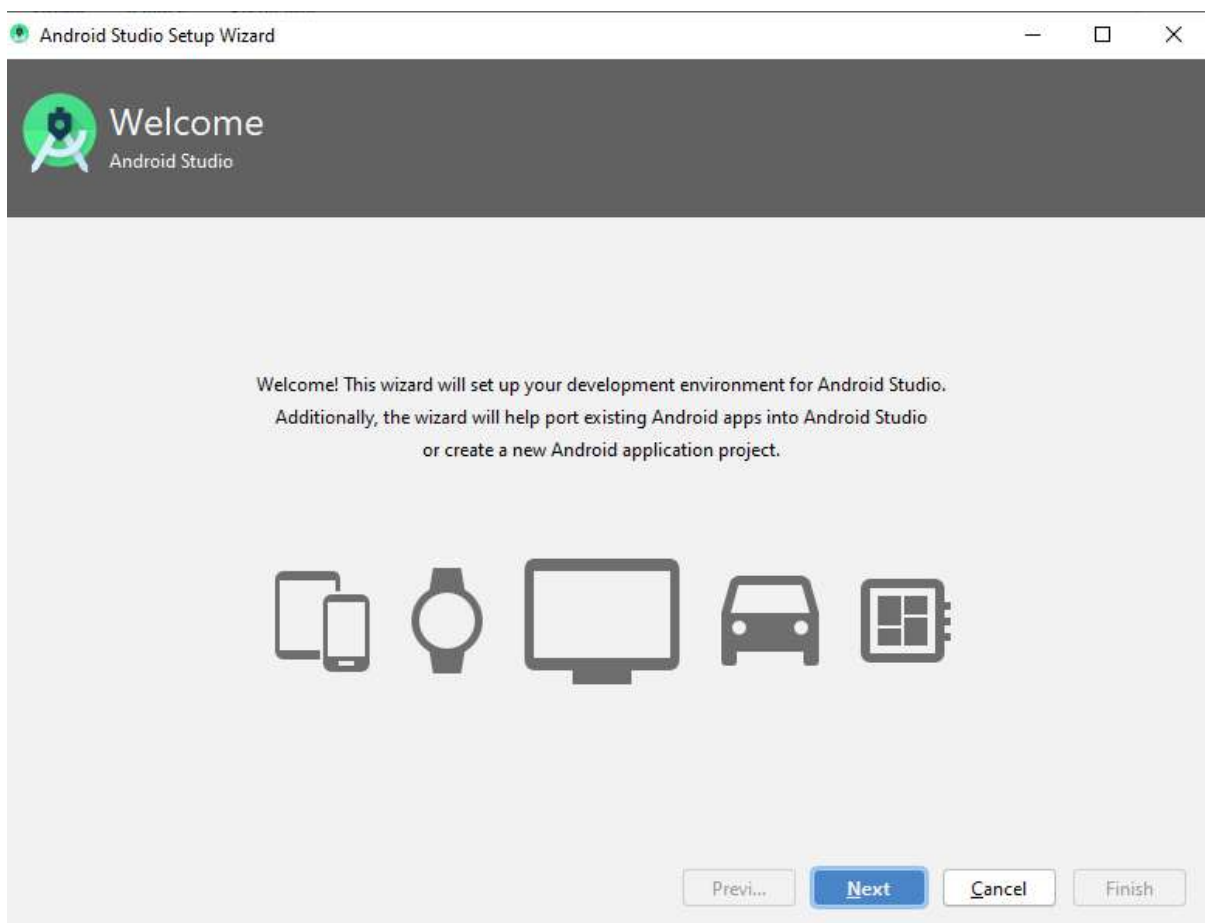
## DESARROLLO DE APLICACIONES MULTIPLATAFORMA

### T1. ANDROID. EL SISTEMA OPERATIVO

de importar las preferencias que hubiéramos configurado en versiones anteriores. En nuestro caso, indicamos que no queremos importar nada y pulsamos ok:



Para terminar, el instalador lanzará un asistente “Wizard” para configurar el entorno integrado de desarrollo. Pulsamos en “Next” para continuar:

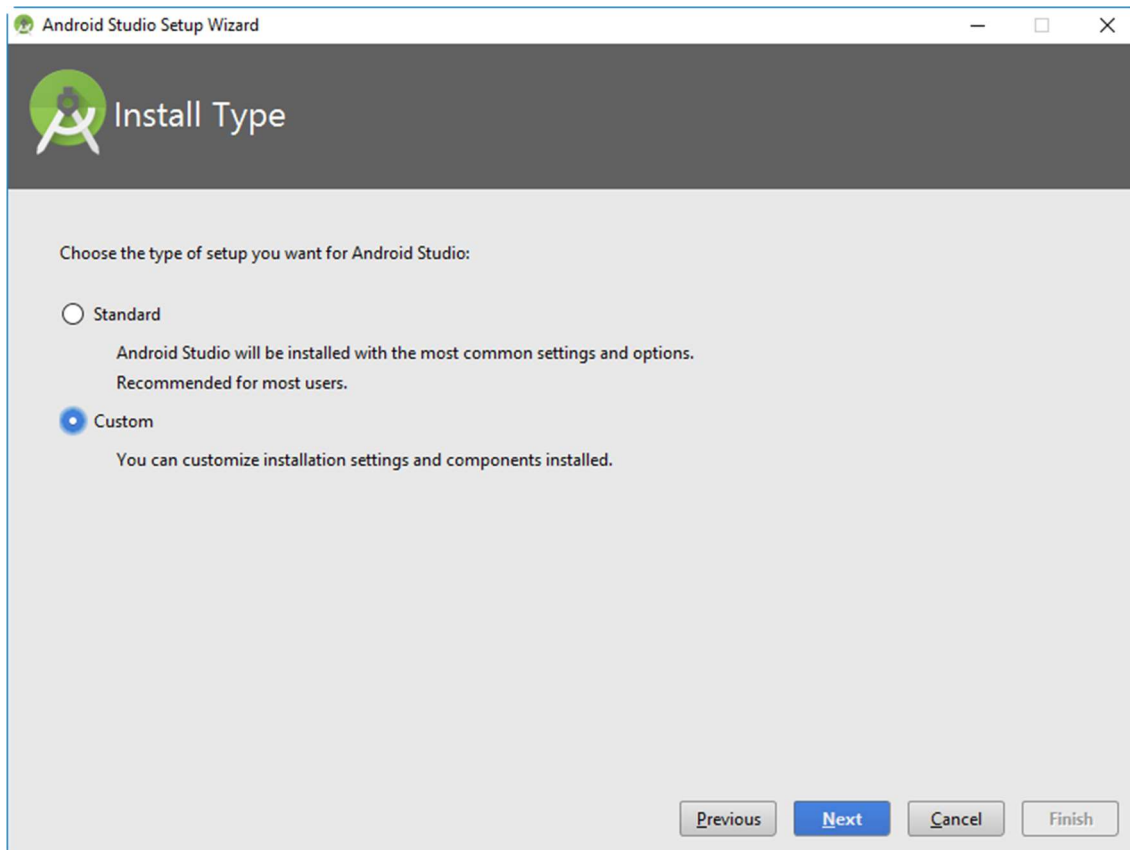


Nos permitirá a continuación seleccionar entre instalación estándar o personalizada. Seleccionamos “Custom” y pulsamos “Next”.

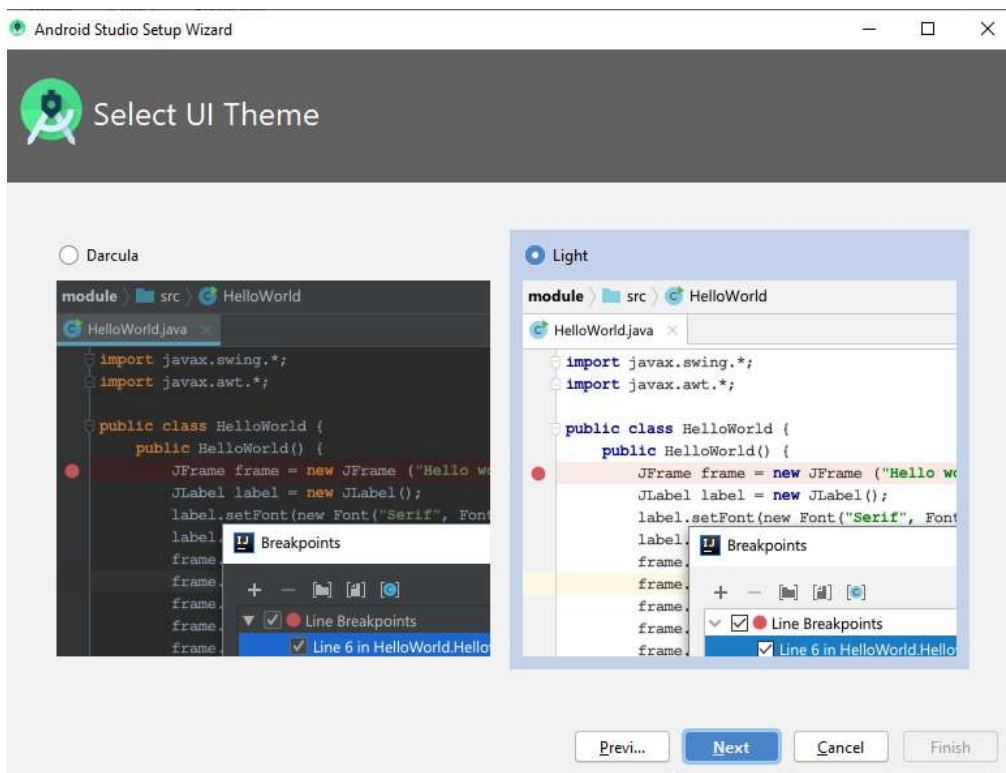


# DESARROLLO DE APLICACIONES MULTIPLATAFORMA

## T1. ANDROID. EL SISTEMA OPERATIVO



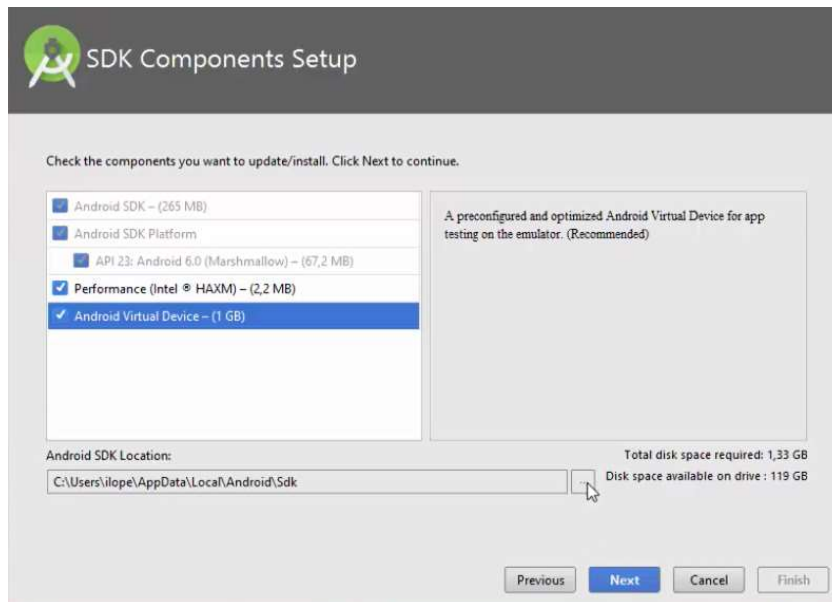
Al permitir la personalización del entorno el configurador de Android Studio nos permitirá seleccionar entre dos temas distintos para utilizar, el **IntelliJ** de IDEA o **Darcula**. Seleccionamos IntelliJ y pulsamos “Next”.



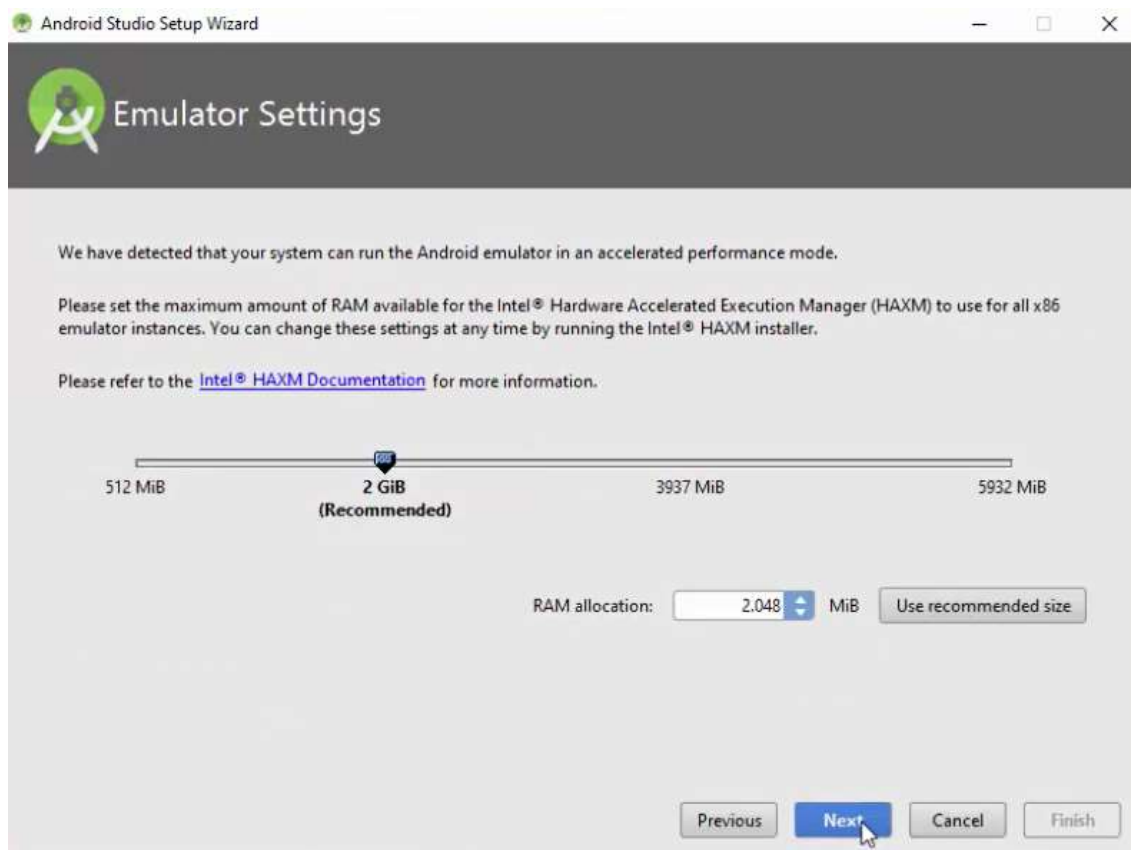
# DESARROLLO DE APLICACIONES MULTIPLATAFORMA

## T1. ANDROID. EL SISTEMA OPERATIVO

La siguiente opción nos dará la posibilidad de instalar el SDK de Android Studio junto con su colección de APIs, herramientas y utilidades para depurar, compilar y ejecutar tus aplicaciones. También permitirá seleccionar la instalación de un emulador de Android (Android Virtual Device) y las opciones de aceleración de hardware para mejorar el rendimiento en procesadores Intel (HAXM). Si tienes un procesar Intel y marcas esta opción, el emulador se ejecutará mucho más rápido.



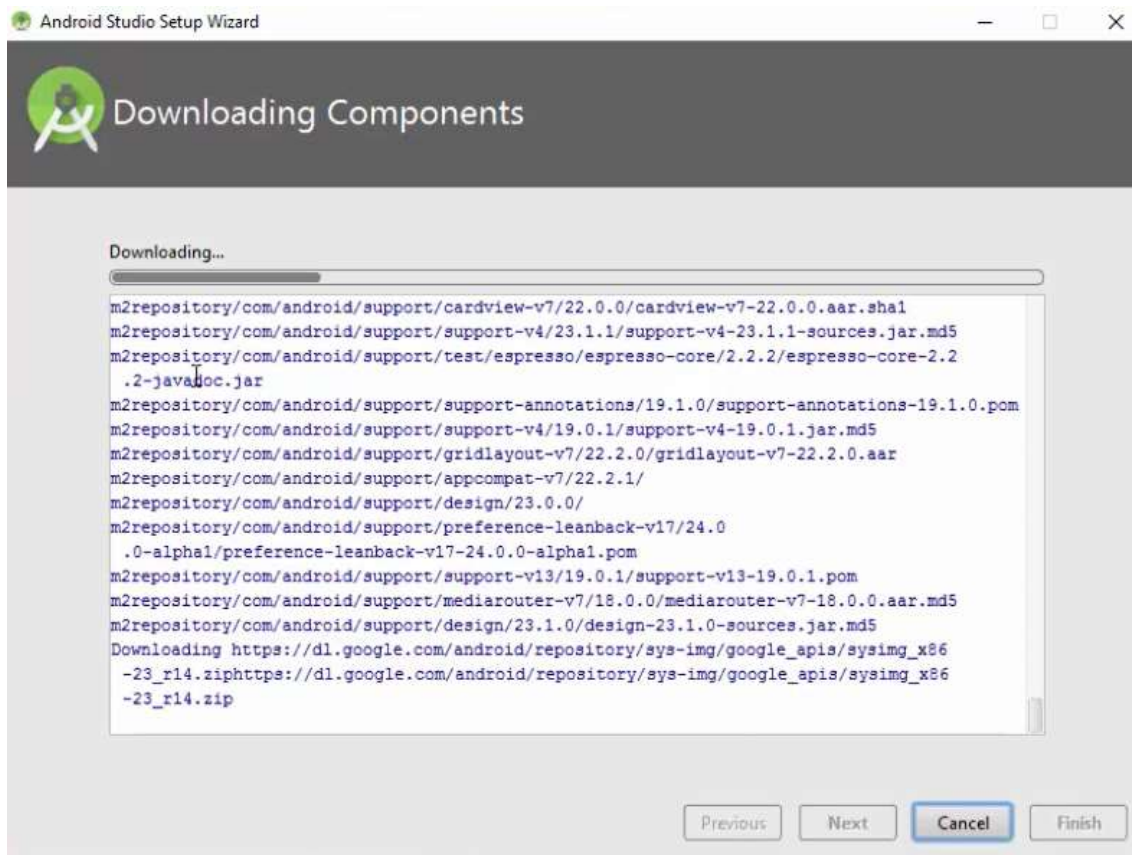
Si seleccionaste la opción de instalar el Dispositivo Virtual y has instalado HAXM, el asistente te preguntará cuánta memoria quieres reservar para el sistema de aceleración (2 Gb recomendado):



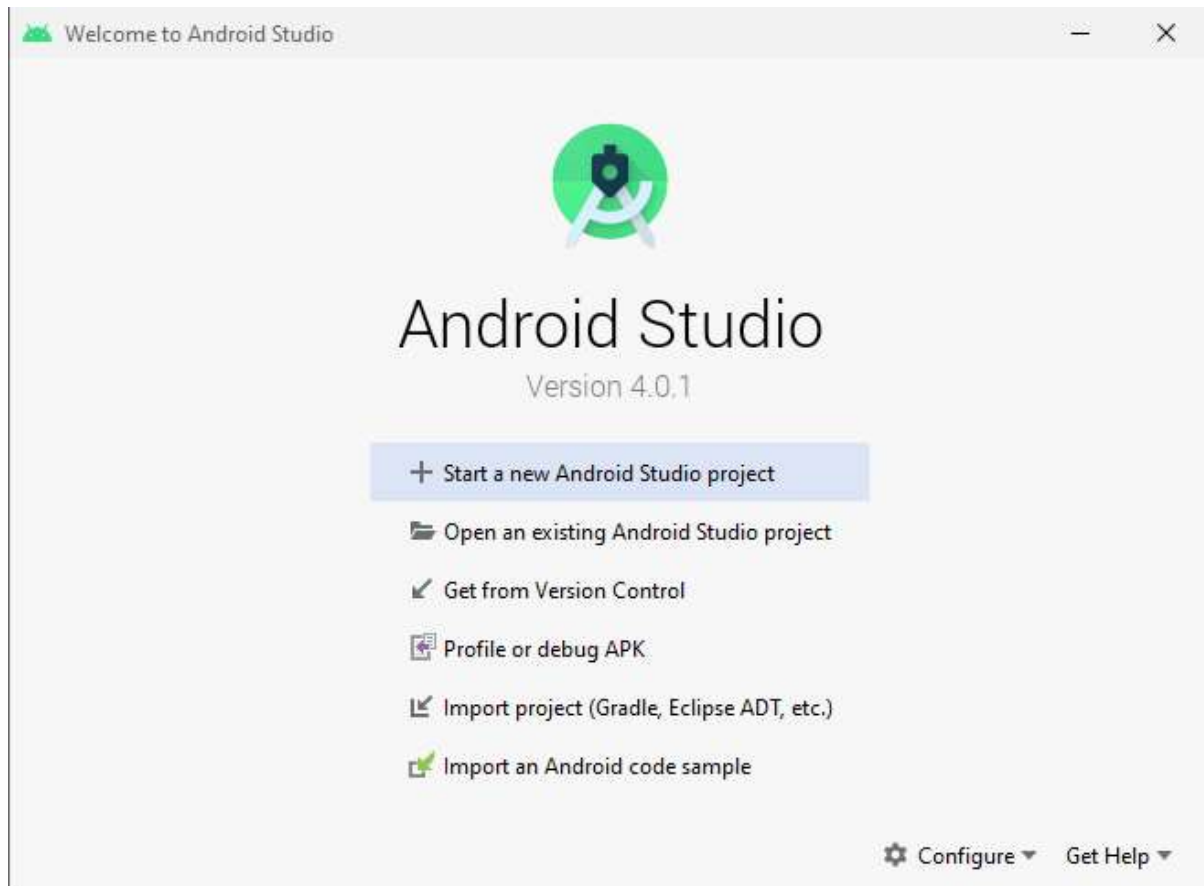
## DESARROLLO DE APLICACIONES MULTIPLATAFORMA

### T1. ANDROID. EL SISTEMA OPERATIVO

Finalmente, el asistente comenzará a descargar los componentes que necesita para terminar la instalación:



Cuando termine la instalación, se lanzará la pantalla de bienvenida:



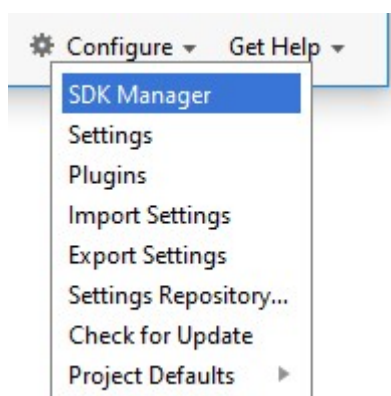
Desde esta pantalla podemos crear un nuevo proyecto, abrir uno que existe o importarlo (incluso de versiones anteriores de Android Studio o Eclipse).

También es posible obtener un proyecto desde repositorios de software estilo GitHub o CSV. Si no conoces GitHub quizá te interese saber qué es leyendo el siguiente artículo:

<http://conociendogithub.readthedocs.io/en/latest/data/introduccion/>



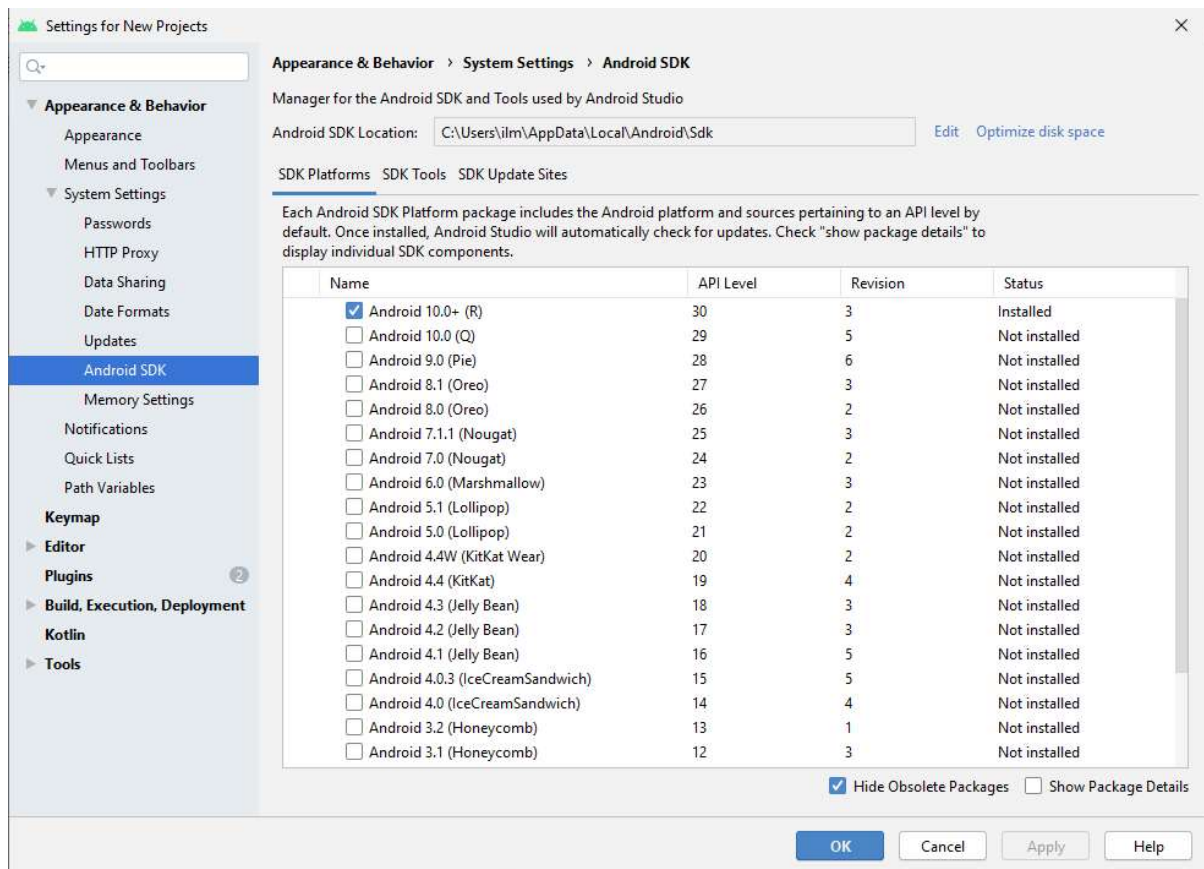
Fíjate además en el botón de configuración en la esquina inferior derecha. Desde este botón tienes acceso a muchas opciones de configuración, fíjate en alguna de ellas, por ejemplo, pulsa en “Configure” y selecciona SDK Manager:



# DESARROLLO DE APLICACIONES MULTIPLATAFORMA

## T1. ANDROID. EL SISTEMA OPERATIVO

El SDK Manager permite descargar los componentes que nos van a hacer falta para poder compilar los proyectos, probarlos y depurarlos. Pulsa en SDK Manager y verás la siguiente pantalla:



Verás que hay un paquete instalado, correspondiente a la última versión estable de Android (en el momento de escribir este texto Android 10.0) y otros que es posible que en algún momento necesitamos instalar si quieres dirigir tu aplicación a alguna versión de Android en concreto. Por un lado verás las SDK platforms, es decir las imágenes del sistema Android que contienen aparte del sistema operativo, iconos, apps, sonidos, configuraciones y un montón de características más y las herramientas del SDK "SDK Tools", por ejemplo, podríamos instalar los **Google Play Services** para incorporar a tu aplicaciones funciones como Maps o Google Cloud Messaging.

Mira el anuncio que Google lanzó para publicitar las Google Play Services:

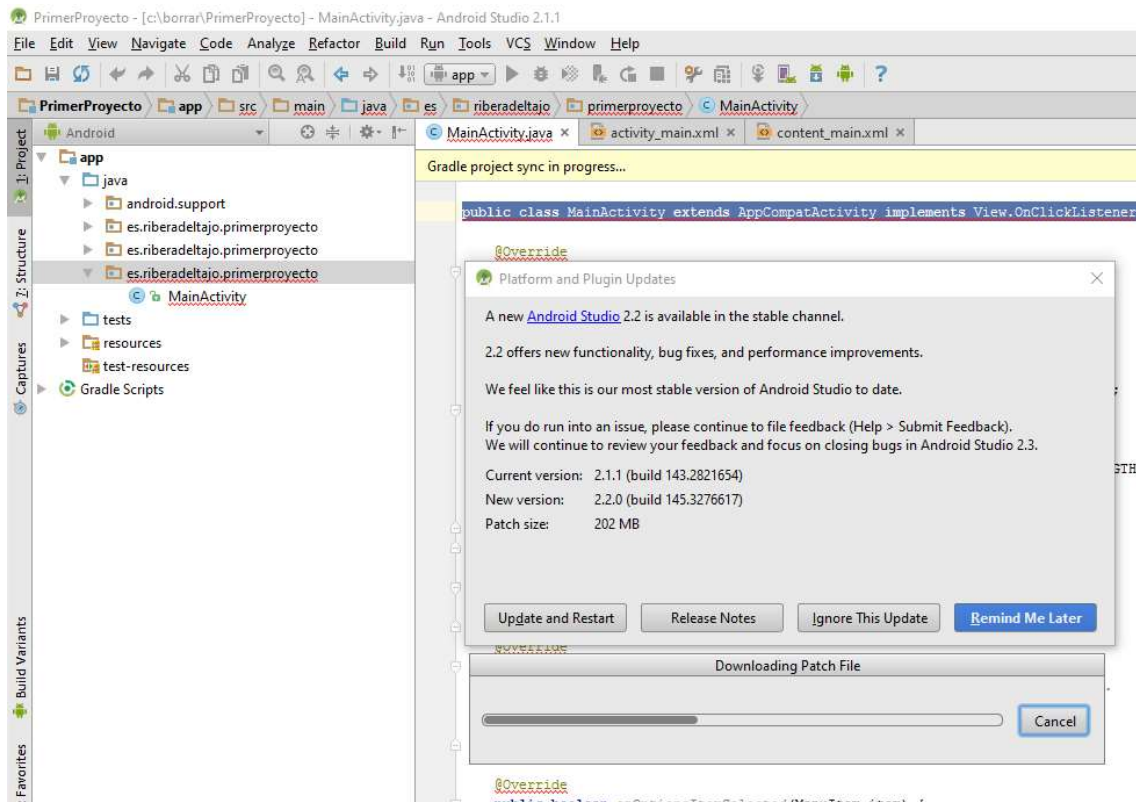
<https://www.youtube.com/watch?v=M3Udfu6qidk>

Otra de las ventajas de Android Studio es que te informará automáticamente de las actualizaciones disponibles, tanto del entorno de desarrollo como de las plataformas disponibles.




# DESARROLLO DE APLICACIONES MULTIPLATAFORMA

## T1. ANDROID. EL SISTEMA OPERATIVO

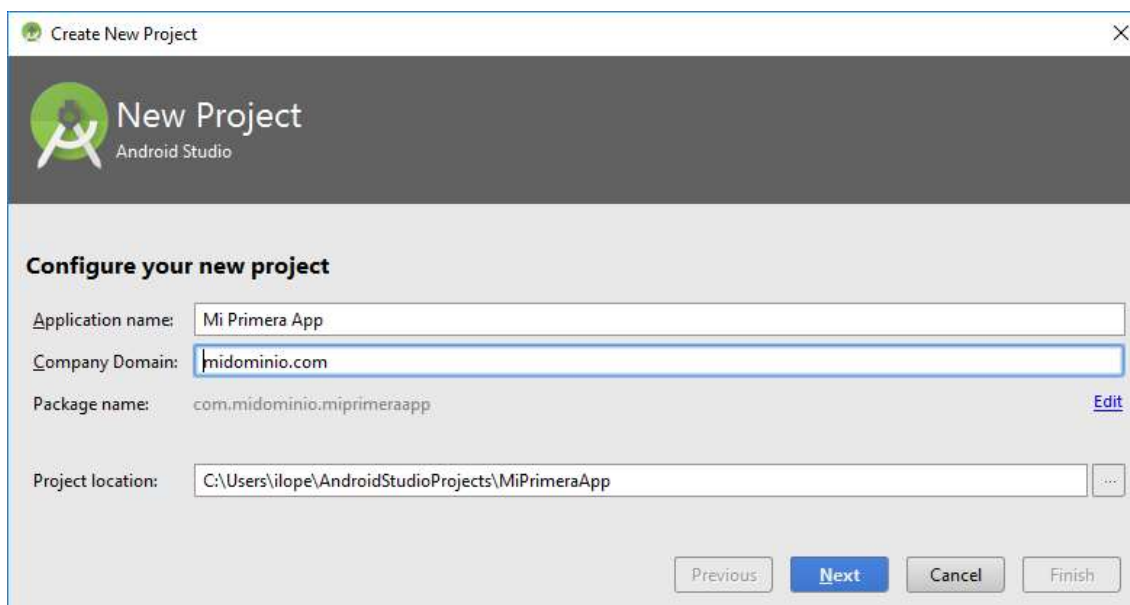


### 1.8. Nuestro primer proyecto

Para calentar y conocer el entorno un poco más, vamos a realizar nuestro primer proyecto, para ello, en la pantalla de bienvenida de Android Studio, pulsamos en “Start a new Adnroid Studio Project”:

 Start a new Android Studio project

A continuación, ponemos el nombre de la aplicación (incluyendo el dominio de la compañía) y la ubicación donde irá almacenado en nuestro disco duro:

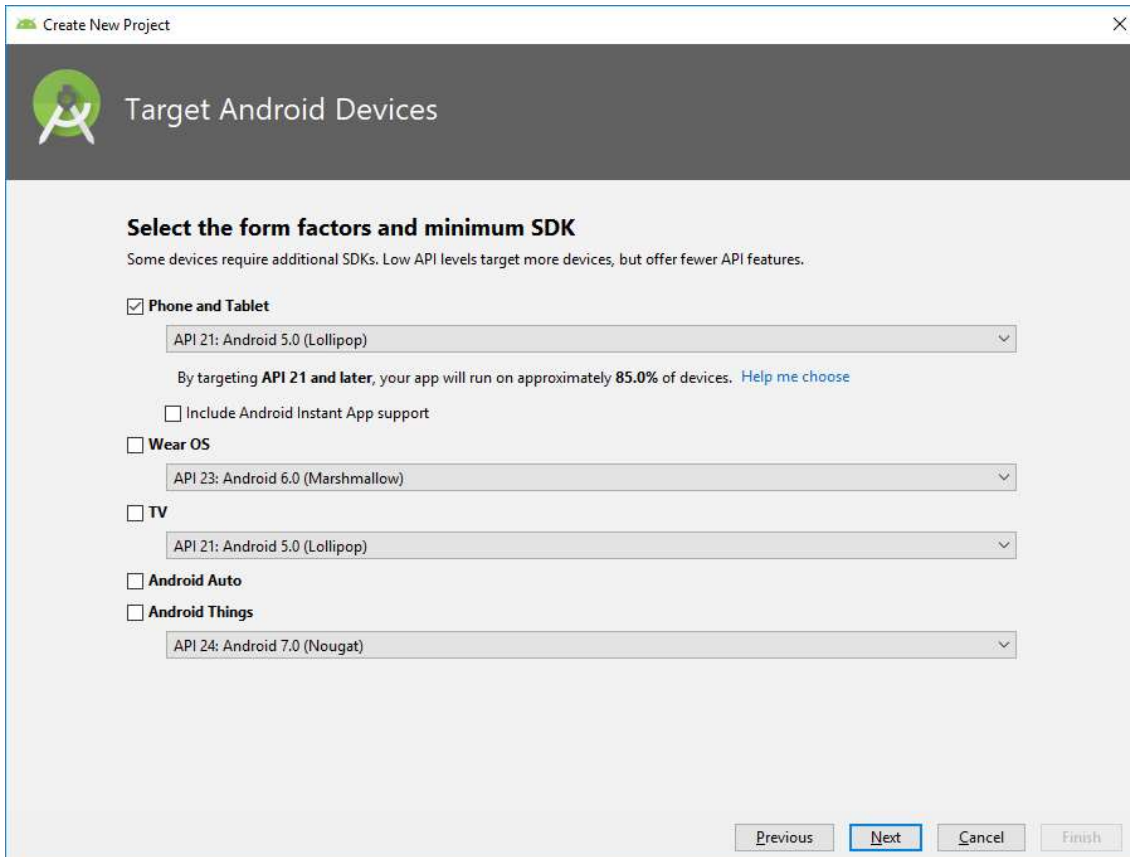




## DESARROLLO DE APLICACIONES MULTIPLATAFORMA

### T1. ANDROID. EL SISTEMA OPERATIVO

Pulsamos en “Next” y seleccionamos el tipo de plataforma para los que será compilada nuestra aplicación y el tipo de dispositivo (Teléfono o tables, pequeños dispositivos, tv, etc). Hay que escoger el mínimo SDK con el que funcionará nuestra app. Esto dependerá de los requisitos de nuestra aplicación, pero, para empezar, puedes escoger una que es bastante compatible con todos los móviles y tables que hay, por ejemplo, la API 21, que se puede ejecutar en el 85% de los dispositivos:



Create New Project

### Target Android Devices

**Select the form factors and minimum SDK**  
Some devices require additional SDKs. Low API levels target more devices, but offer fewer API features.

☒ **Phone and Tablet**  
API 21: Android 5.0 (Lollipop) ▼  
By targeting **API 21 and later**, your app will run on approximately **85.0%** of devices. [Help me choose](#)  
☐ Include Android Instant App support

☐ **Wear OS**  
API 23: Android 6.0 (Marshmallow) ▼

☐ **TV**  
API 21: Android 5.0 (Lollipop) ▼

☐ **Android Auto**

☐ **Android Things**  
API 24: Android 7.0 (Nougat) ▼

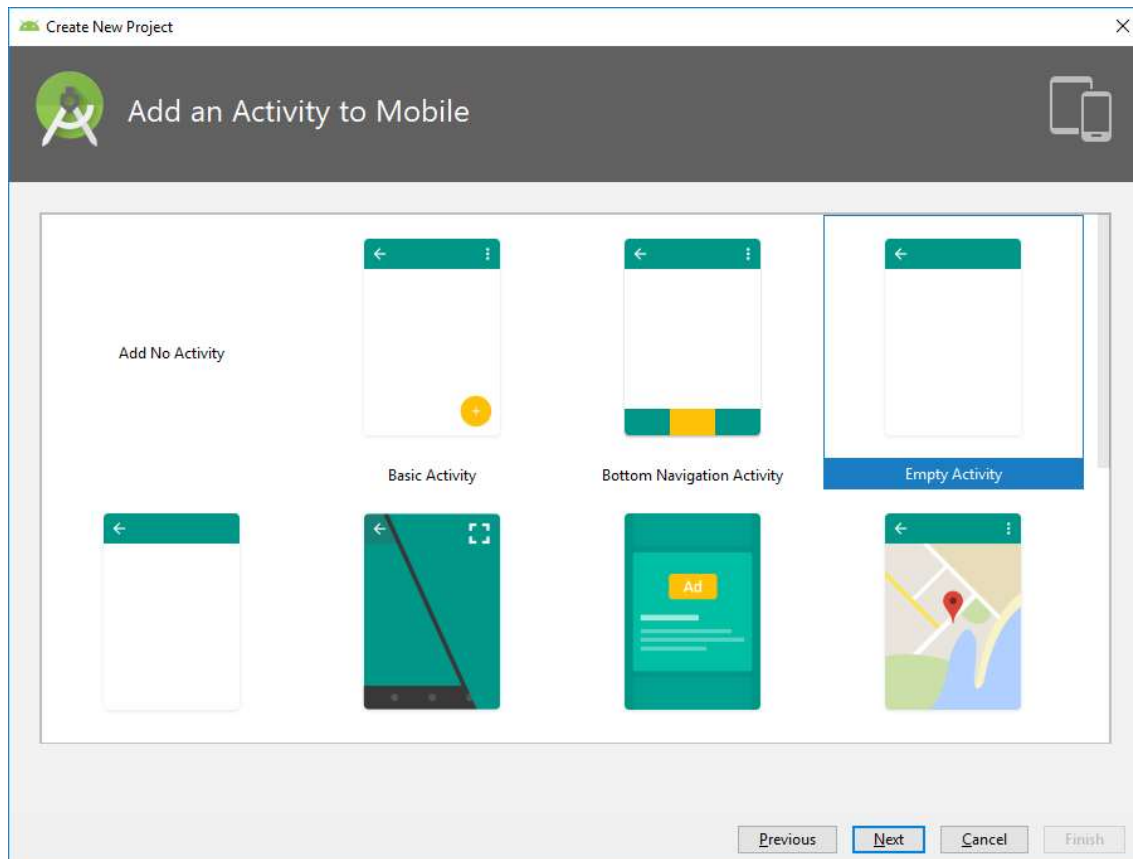
Previous Next Cancel Finish

Pulsas en next, y verás que aparece otra pantalla pidiendo que selecciones el tipo de actividad. Aquí nos detenemos un momento, debemos primero saber qué es una actividad:

Una actividad es nuestro programa en sí mismo, contiene la interfaz de usuario de nuestra App, pero vamos a investigar un poco más sobre el concepto de actividad:

# DESARROLLO DE APLICACIONES MULTIPLATAFORMA

## T1. ANDROID. EL SISTEMA OPERATIVO



Cuando no sabemos o no conocemos un concepto (y con la nueva tecnología es totalmente normal que nos bombardeen constantemente con nueva terminología), debemos buscar en una fuente fiable la definición de ese concepto que no sabemos. Por ejemplo, si buscamos en la documentación de Android el concepto de actividad, en inglés, activity, obtenemos:

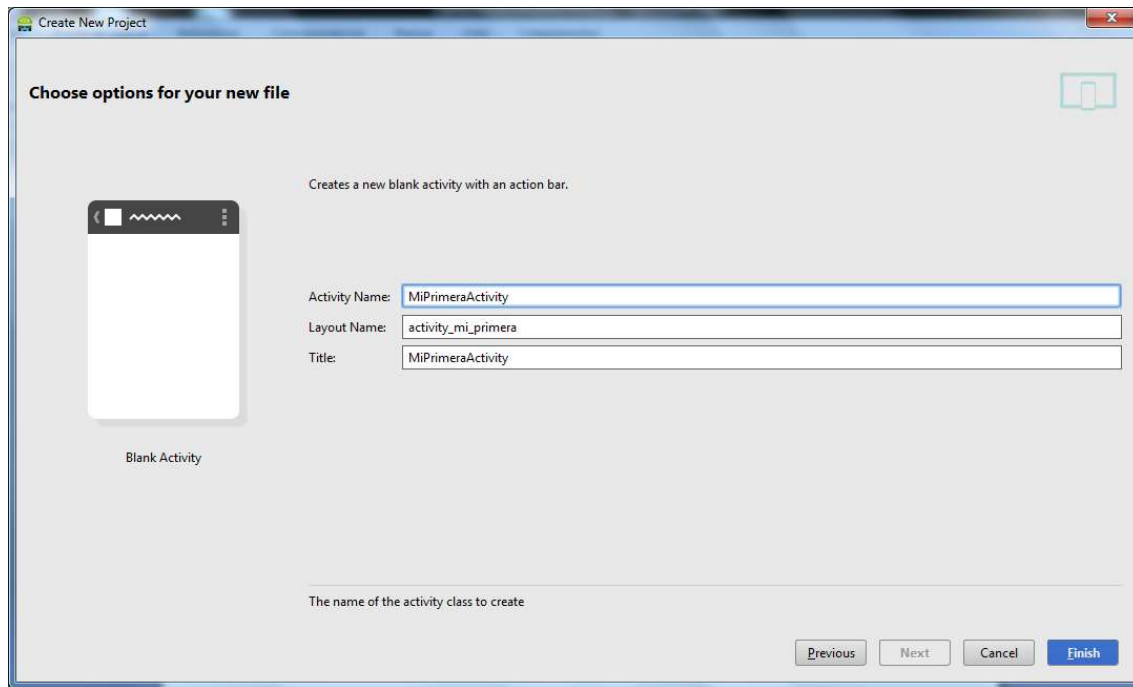
*An activity is a single, focused thing that the user can do. Almost all activities interact with the user, so the Activity class takes care of creating a window for you in which you can place your UI with `setContentView(View)`.*

Traducido vagamente sería algo así como “Una cosa simple y concreta que el usuario puede hacer y que contiene la interfaz de usuario”. Dicho de otro modo y teniendo en cuenta la definición de *Activity*, y puesto que los móviles están preparados para ejecutar muchas apps pequeñas a la vez, se puede afirmar que una actividad es un programa pequeño y ligero, controlada por Android y sometida a las normas de funcionamiento de Android. De esta manera, evitaremos convertir el dispositivo móvil en un ordenador común.

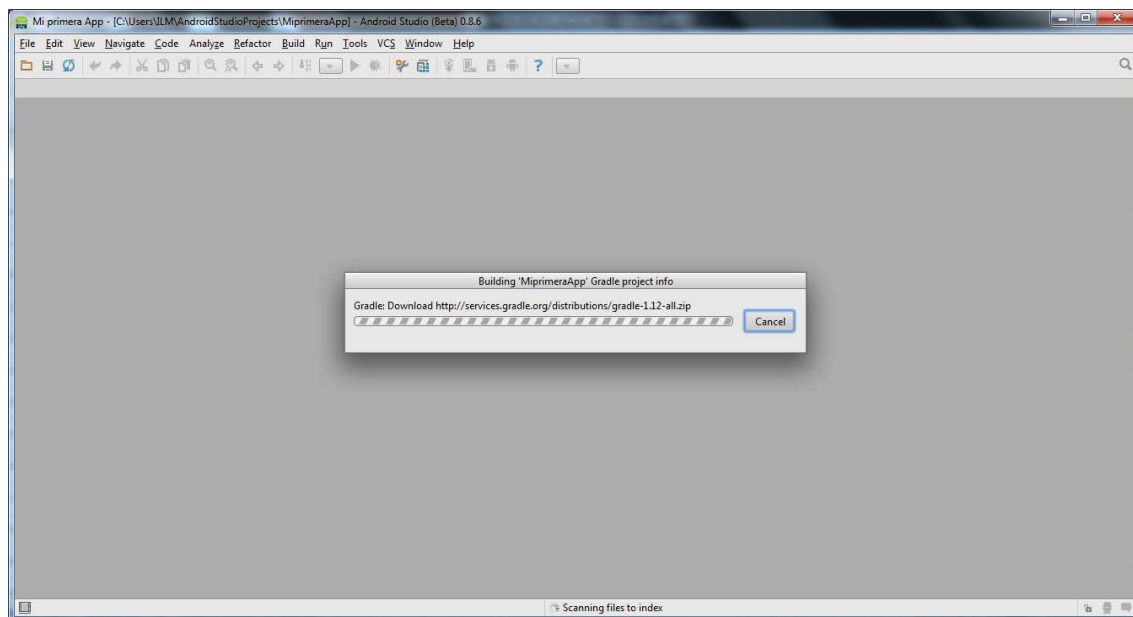
De momento, escogeremos una actividad en blanco “Empty Activity” y, aunque hay varios tipos, de momento trabajaremos con actividades en blanco. Pulsa en Next y elige el nombre de tu actividad y pulsa en Finish:

# DESARROLLO DE APLICACIONES MULTIPLATAFORMA

## T1. ANDROID. EL SISTEMA OPERATIVO



Comenzará el proceso de creación del proyecto y configuración del entorno para que comiences a programar:



Una vez creado el proyecto, Android Studio nos da la bienvenida amablemente con consejos sobre cómo utilizarlo:

# DESARROLLO DE APLICACIONES MULTIPLATAFORMA

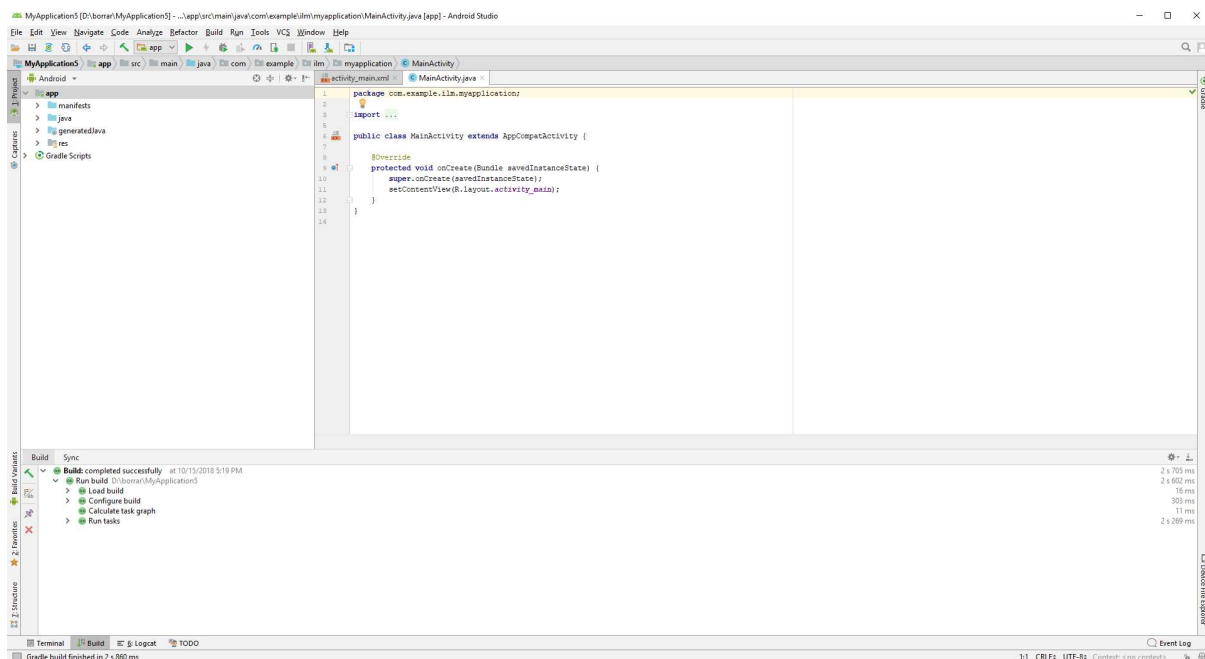
## T1. ANDROID. EL SISTEMA OPERATIVO



Es importante leerse estos consejos porque a la larga facilitan el aprendizaje del uso de la herramienta y pueden proporcionar trucos para efectuar operaciones que a priori pueden parecer no triviales.

### 1.9. Primer contacto con el código. ¿Dónde está el java?

Nada más abrir el primer proyecto, aparece esta pantalla:



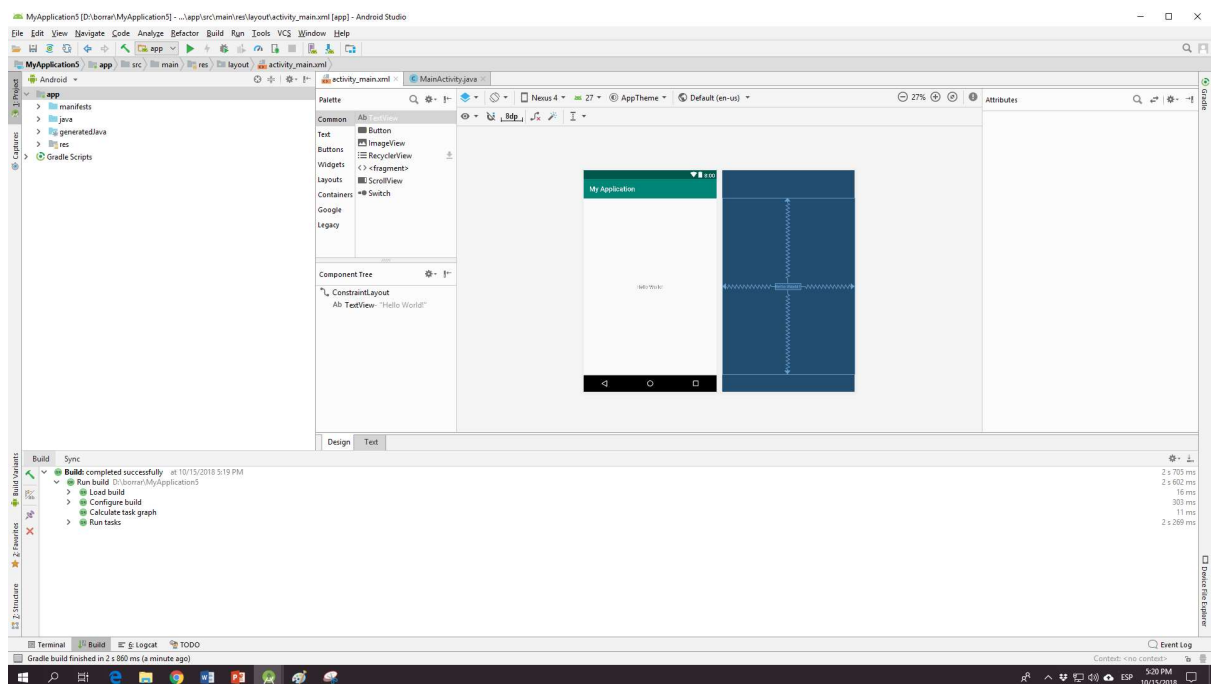
La pantalla aparece dividida en tres partes:

- El explorador de proyectos, a la izquierda, donde aparecen todas las carpetas y archivos que componen el proyecto. Presta especial atención a la carpeta src, donde están todos los archivos que modificarás para dar vida a tu App. Las otras carpetas, en principio, hasta que no veamos cosas más avanzadas no nos interesan, pero conviene saber qué contienen
  - o build: contienen los archivos binarios resultado del proceso de compilación (tanto generados como intermedios)

# DESARROLLO DE APLICACIONES MULTIPLATAFORMA

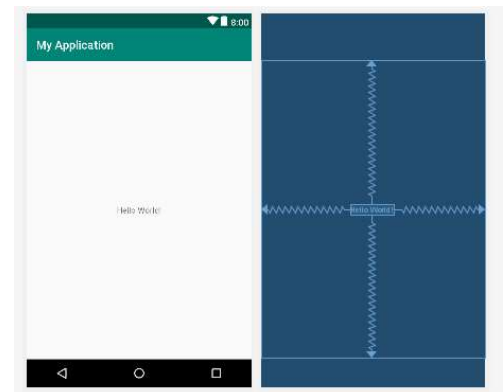
## T1. ANDROID. EL SISTEMA OPERATIVO

- libs: Inicialmente vacía, contendrá referencias a librerías de código programadas por nosotros.
  - gradle: Gradle es el plugin que utiliza android studio para compilar, construir y depurar tus programas.
  - Un archivo importante: El AndroidManifest.xml, que es un archivo XML que contiene toda la descripción de la aplicación que estamos creando y qué componentes (servicios, actividades, imágenes, etc) están incluidas.
- El panel de archivos de código, en el centro, inicialmente aparece con una ventana con código Java autogenerado. Si pulsa en la pestaña “activity\_main.xml”, mostrará un archivo XML autogenerado por Android Studio. Sí, has leído bien, XML. La interfaz gráfica de tu app se puede y se recomienda definir con definiciones en XML. Si te fijas en los *tabs*, por un lado tienes el archivo XML y por otro lado un archivo Java. Pues en XML se declaran todos los componentes y en el archivo java se programan los comportamientos.



- A la derecha encuentras la vista previa de cómo quedará tu App en el dispositivo android. Si modificas el archivo XML verás cómo cambia. Puedes experimentar a cambiar alguna cadena de texto para ver cómo cambia el diseño. Por ejemplo, el fragmento de código:

```
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Hello World!"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintTop_toTopOf="parent" />
```

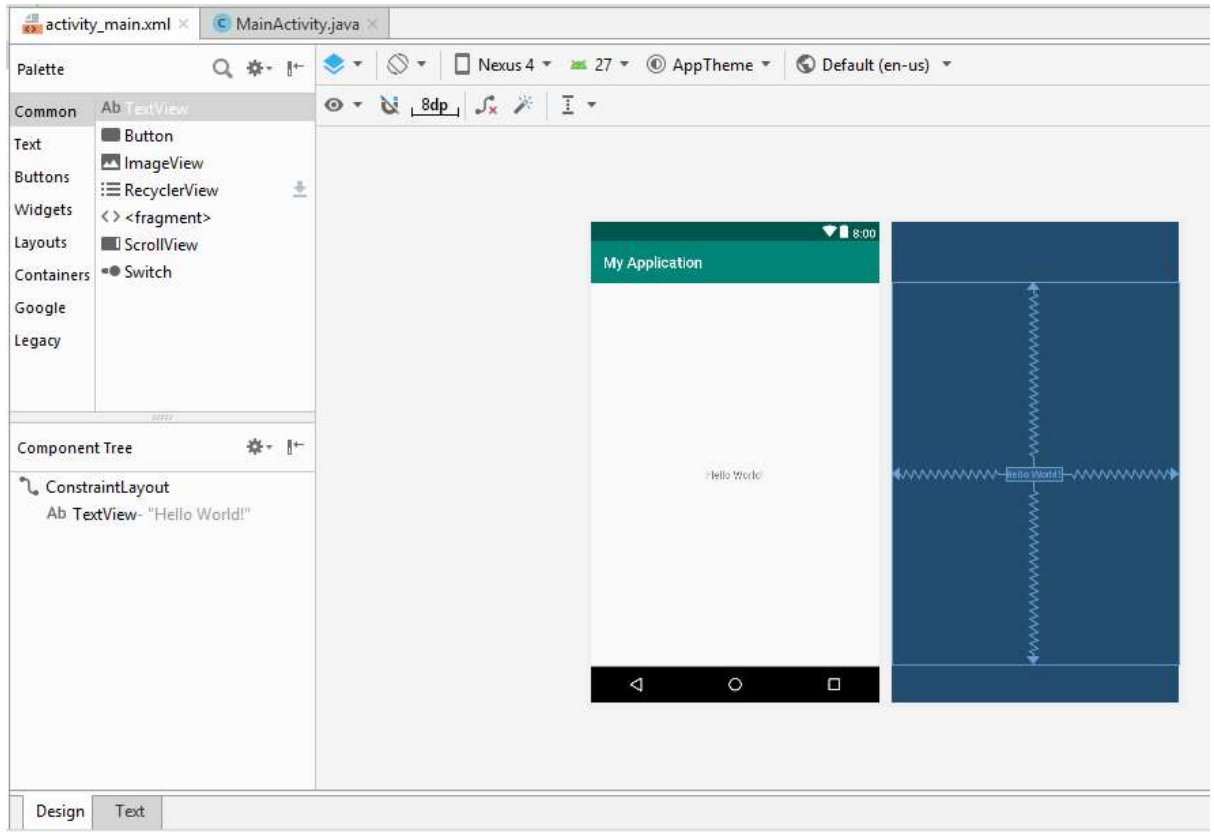


## DESARROLLO DE APLICACIONES MULTIPLATAFORMA

### T1. ANDROID. EL SISTEMA OPERATIVO

corresponde a un campo de texto que se presenta en pantalla y que contiene la cadena de texto “Hello World”. Si pruebas a cambiar la cadena de caracteres por “Elearning rocks!”, verás el efecto en el emulador.

Con esta vista tendrás una visión de todos los componentes o *Widgets*, que puedes ir insertando para configurar tu interfaz gráfica.



- Por último, si seleccionas el archivo con extensión java “MiPrimeraActivity”, verás código que te sonará conocido:

```
activity_main.xml x MainActivity.java x
1 package com.example.ilm.myapplication;
2
3 import ...
4
5
6 public class MainActivity extends AppCompatActivity {
7
8     @Override
9     protected void onCreate(Bundle savedInstanceState) {
10         super.onCreate(savedInstanceState);
11         setContentView(R.layout.activity_main);
12     }
13 }
14
```

Éste es el código Java que se genera automáticamente cuando creas el proyecto.

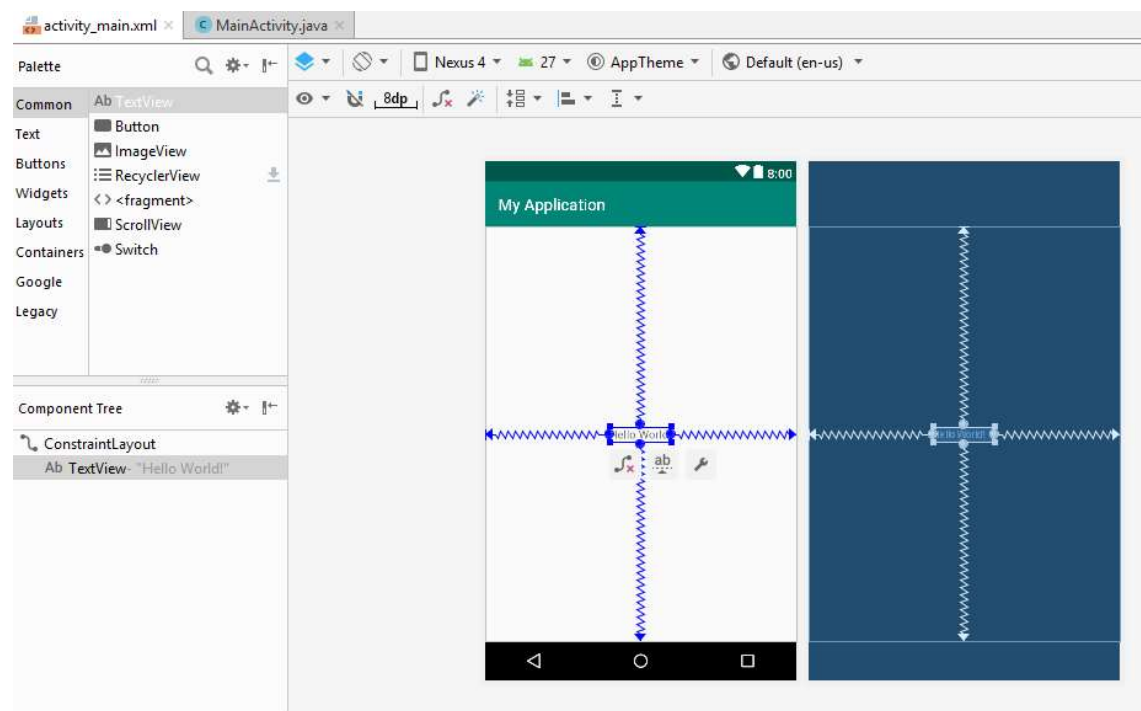


También te habrás dado cuenta de que ha desaparecido la pantalla de vista previa y que sólo está disponible cuando modificas el archivo XML. Aunque es perfectamente posible cambiar la interfaz de usuario a través de código fuente, sólo verás los cambios en la interfaz reflejados en la vista previa cuando cambies el archivo XML.

## 1.10. Programando sin escribir líneas de código

Una de las características del desarrollo de Android es que se puede llegar a diseñar muchas cosas sin apenas tocar código, de hecho, ya habrás comprobado la cantidad de código en XML y Java que genera Android Studio con tan sólo trastear un poco con los menús.

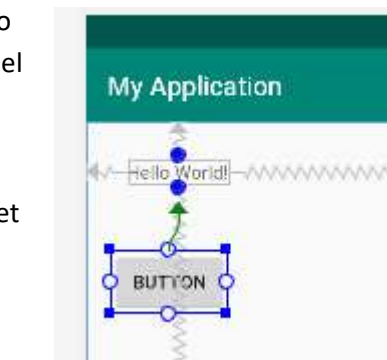
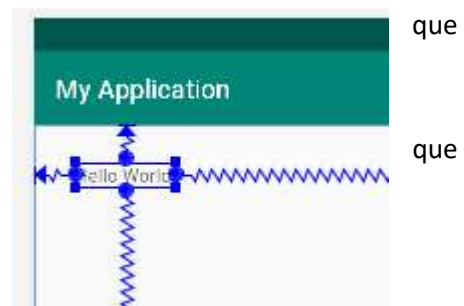
Para ilustrarlo, haz la siguiente prueba:



- En vista diseño, selecciona el widget de tipo TextView está ubicado justo en el centro de la pantalla y arrástralo hasta la esquina inferior izquierda. A continuación, añade un widget de tipo botón a la pantalla del móvil muestra la vista previa y sitúalo justo debajo del TextView que viene por defecto.

A continuación, pulsa sobre el círculo que está en la parte superior del botón y arrastra la flecha hacia el círculo de la parte inferior del texto. De esta manera indicas, que el botón se debe ubicar “justo debajo” del texto.

En las últimas versiones de Android, el contenedor donde insertas los widgets se llama ConstraintLayout. Cada widget que depositas ahí dentro, debe tener una restricción



## DESARROLLO DE APLICACIONES MULTIPLATAFORMA

### T1. ANDROID. EL SISTEMA OPERATIVO

(Constraint) que indique cómo se dispone el widget en relación al resto de componentes del contenedor.

De esta manera, para conectar un componente con otro, puedes pulsar sobre los circulitos que aparecen a los lados de los componentes y unirlos con otros componentes. Además puedes redimensionarlo arrastrando las esquinas y alinearlos a las diferentes partes de la pantalla. Pruébalo, ¡es muy fácil!

Puedes aprender más sobre ConstraintLayout, viendo los vídeos que te hemos dejado en la plataforma. Aunque estén en inglés, verás que es muy sencillo:

**extra**

-  [Video sobre el uso constraint Layout](#)
-  [Video sobre Constraint Layout \(GOOGLE\)](#)
-  [Lab sobre el ConstraintLayout \(by Google\)](#)

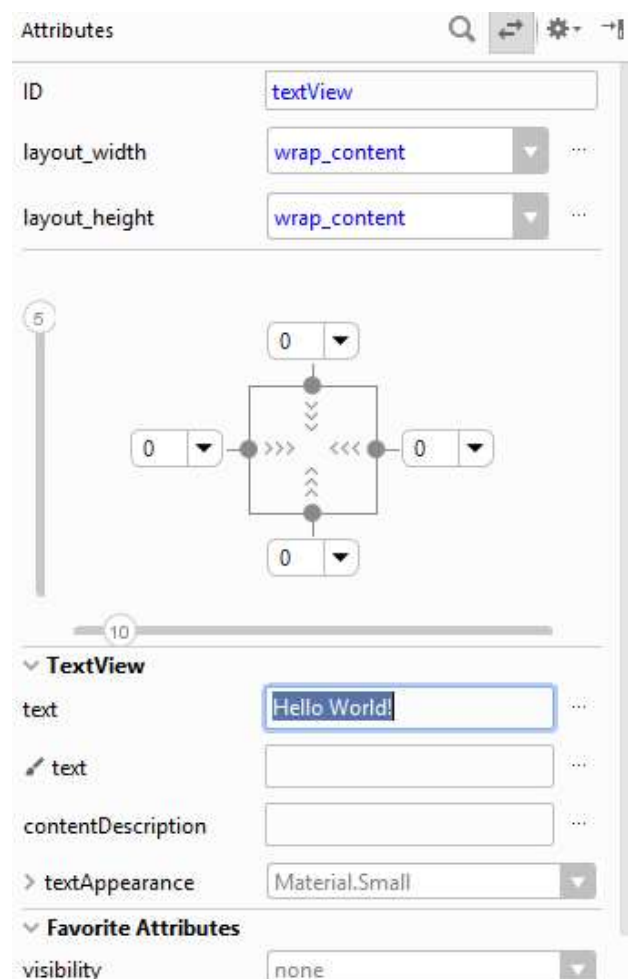
#### Evaluación

Si cambias a la vista de texto “Text”, verás que se ha añadido en XML el siguiente código:

```
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="New Button"
    android:id="@+id/button"
    ...
/>
```

Para cambiar los textos del botón y del campo de texto, puedes hacerlo editando el código pulsando dos veces sobre el propio botón y abriendo las propiedades.

Puedes ponerle en el campo *text* una cadena de caracteres con el nuevo texto del botón y el id lo usarás para referenciarlo después desde el código. Puedes ponerle, por ejemplo, “Pulsa aquí”.



## 1.11. Introduciendo un poco de código

Vamos a darle un poco de funcionalidad a nuestro primer proyecto. ¿Adivinas cuál? Pues claro, vamos a hacer que cuando pulsemos el botón, cambie el texto del TextView. Para empezar, ¿Te acuerdas de lo que era un `EventListener` o un `ActionListener`? Efectivamente, eran métodos que había que implementar para responder a los eventos que “escuchábamos” y así dar una funcionalidad a un componente. Eso sí, previamente había que registrarlo para que cuando el componente causara el evento, el programa ejecutara el método que responde al evento del componente. Por ejemplo, si te acuerdas de la asignatura de programación de primero, con los componentes de Swing, si queremos responder al evento acción de un botón debemos primero crear una clase (o aprovechar la que estuviéramos codificando) que implementara la interfaz `ActionListener`. Esta implementación nos obligaba a codificar un método llamado `actionPerformed` que respondía a la acción de pulsar en el botón (en inglés esto se llama método o función *Callback*). Y al componente había que registrarle el objeto que implementa `ActionListener` mediante el método `addActionListener`. Pues en Android, no es muy distinto, hay que implementar la interfaz `OnClickListener`, registrar el objeto que implementa la interfaz mediante el método `setOnClickListener` y programar un método llamado `onClick` que hace de *Callback*.

Lo primero de todo que debes saber es que para poder referencia en tu código a las clases de los componentes que has incluido en el XML y que van a ser parte de tu interfaz de usuario, debes importar las clases. Dentro del paquete `android`, subpaquete `widget` tienes las clases `TextView` y `Button`, que son las dos que has agregado en tu primer proyecto.

```
import android.widget.TextView;  
import android.widget.Button;
```

¿Cómo referencio en mi código java los componentes que he agregado mediante el código XML de la actividad?

Muy fácil, solo tienes que crear una referencia al objeto de la clase que quieras, por ejemplo botón (`Button`) y llamar a la función `findViewById( ... )`

```
Button miBoton;  
miBoton=(Button)findViewById(R.id.button);
```

Nota: A partir de android studio 3.0 no es necesario realizar un `Cast` a la clase destino cuando se usa el método `findViewById`, pudiendo hacer:  
`miBoton=findViewById(R.id.button);`

Desde aquí, puedo acceder a un sinfín de propiedades y métodos para programar mi botón como me apetezca.

Lo siguiente es saber dónde ubicar mi código. Si buscas una función `main`, que sepas que no la vas a encontrar. De hecho, no sólo no existe como tal, sino que cada actividad tiene un ciclo de vida, que va sucediendo llamadas a funciones `callback` según la actividad experimente una interacción por parte del usuario, por ejemplo, arrancar una actividad, abandonar una actividad, retomar una actividad. A continuación, puedes ver el gráfico extraído de la página de desarrolladores de Android, que ilustra perfectamente el ciclo de vida de una actividad y la transición de llamadas a funciones `callback` según van pasando por diferentes estados:

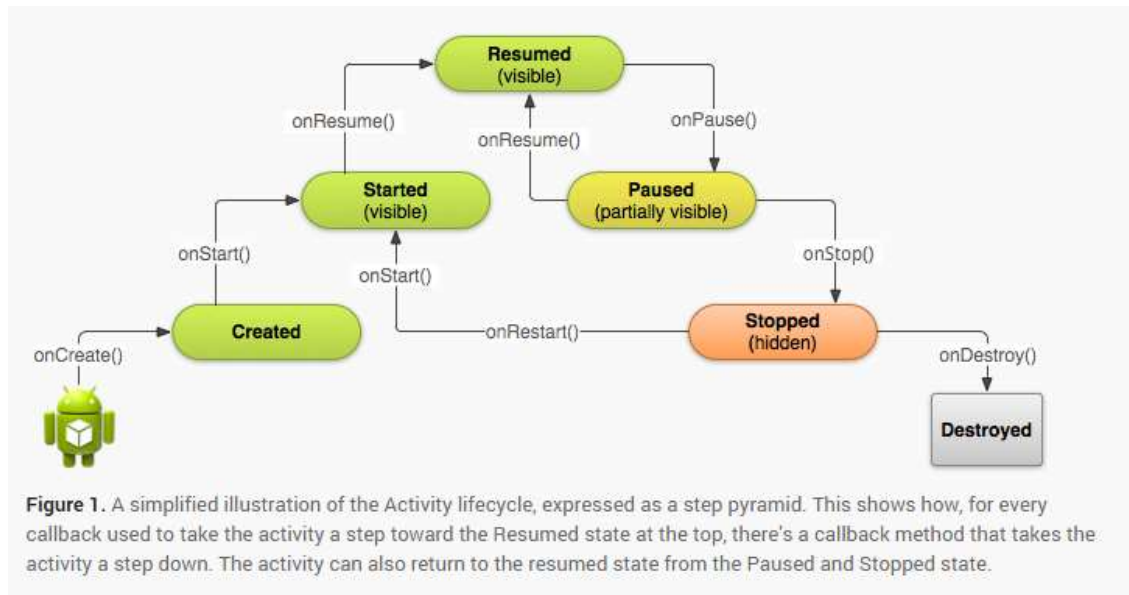


Imagen: Ciclo de vida de una actividad de developer.android. com

De esta manera, la arquitectura de actividad de Android aseguramos que nuestra aplicación será una app adaptada a un dispositivo móvil y no un programa típico para un ordenador de tipo Desktop, es decir, evitamos:

- Que la actividad se bloquee o deje de funcionar cuando el usuario recibe una llamada o cambia a otra app mientras está usando la tuya.
- No consume recursos valiosos del sistema cuando el usuario no está usándola activamente
- No se pierde el progreso del usuario si abandonan la app y luego vuelve a ella.
- No se bloquea cuando el usuario, por ejemplo, cambia la posición de la pantalla de vertical a horizontal..
- Etc...

No es necesario implementar todas las funciones callback, aunque conforme tus apps sean más completas y más complejas, seguro que acabas peleándote con todas y cada una de ellas.

De momento nos centraremos en la primera acción que el ciclo de vida ejecuta cuando el sistema operativo arranca la App que estás desarrollando. Esta función callback es “onCreate”, y si, si quieres puedes pensar en ella como en la función main, pero teniendo en cuenta las diferencias técnicas.

Pues manos a la obra, en primer lugar has de añadir la implementación de la clase `View.OnClickListener`, después añadir el código para acceder a los widgets (button y textView) que has agregado en el archivo XML y finalmente conseguir acceso a ellos. Después añade en el código de la función `onCreate` el código para poder referenciar a los componentes textView y button, y registra el listener “OnClick”. A continuación te señalamos las líneas de código que hemos añadido a “MiPrimeraActivity.java”:

# DESARROLLO DE APLICACIONES MULTIPLATAFORMA

## T1. ANDROID. EL SISTEMA OPERATIVO

```
public class MainActivity extends AppCompatActivity implements View.OnClickListener
{

    Button miBoton; //referencias a los widgets añadidos
    TextView miTexto;

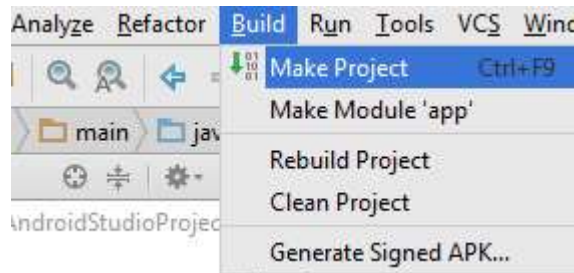
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        miBoton=(Button) findViewById(R.id.button);
        miBoton.setOnClickListener(this);

    }

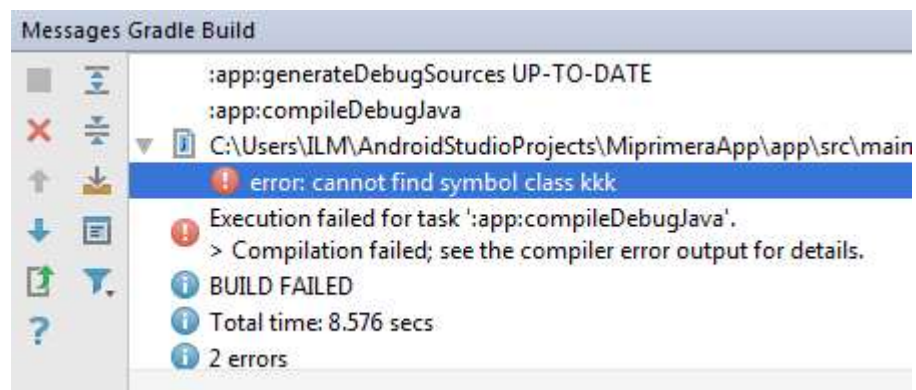
    @Override
    public void onClick(View view) {
        //responde al evento Click
        miTexto=(TextView) findViewById(R.id.textView);
        miTexto.setText("pulsado");
    }

}
```

Y a compilar...



Si tuvieras errores saldría algo del estilo:



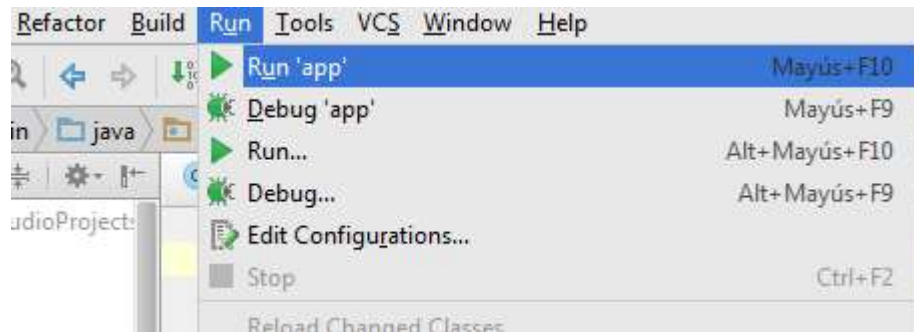
Y si has escrito exactamente lo que te hemos propuesto no tendrás errores, podrás ejecutar tranquilamente.

Ejecutamos...

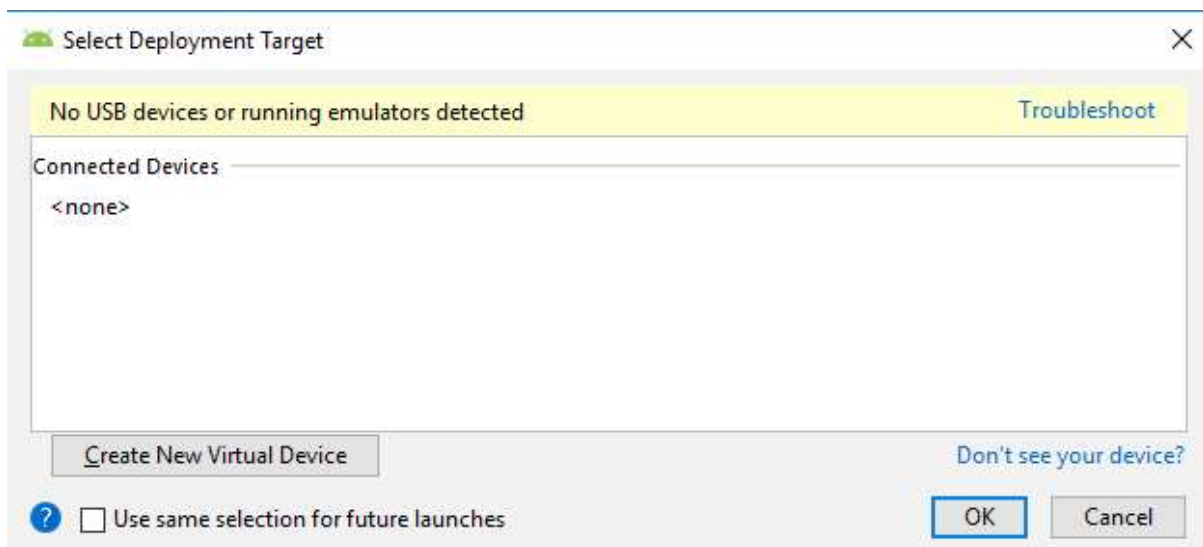


# DESARROLLO DE APLICACIONES MULTIPLATAFORMA

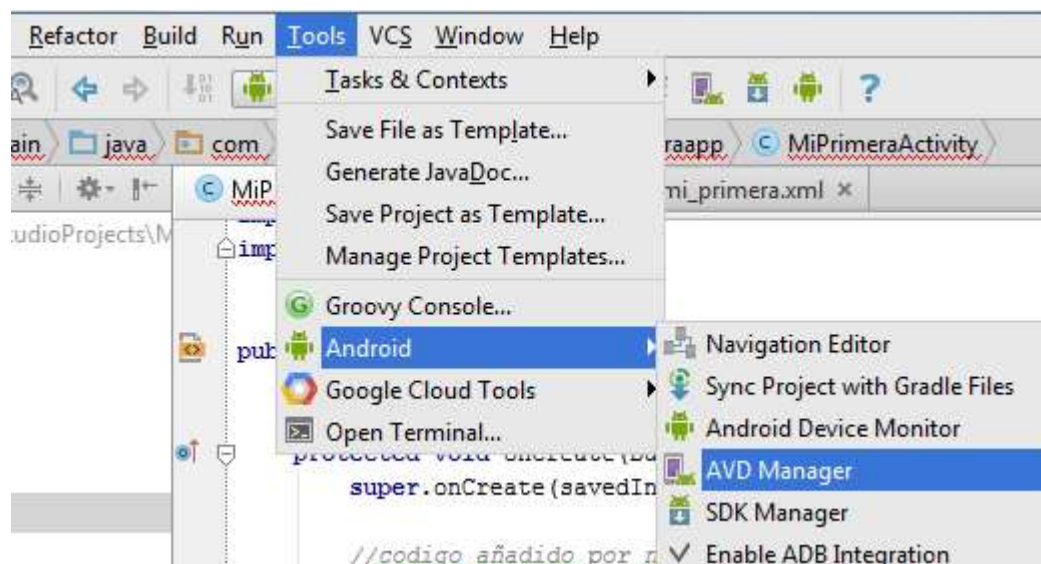
## T1. ANDROID. EL SISTEMA OPERATIVO



Y... ¡anda! ¡No tenemos ejecutando el emulador! No hay problema....



Es hora de crear el emulador, es decir el teléfono virtual donde poder ejecutar y depurar nuestros programas. Para crearlo, sacamos el Android Virtual Device Manager (AVD Manager):

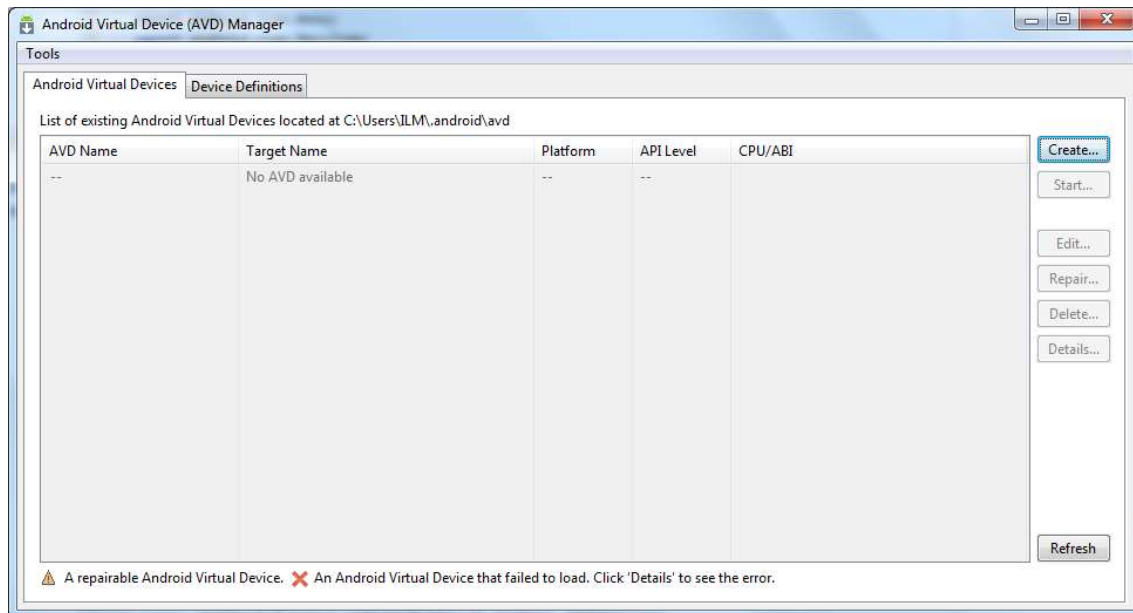


El AVD Manager es muy sencillo de usar:

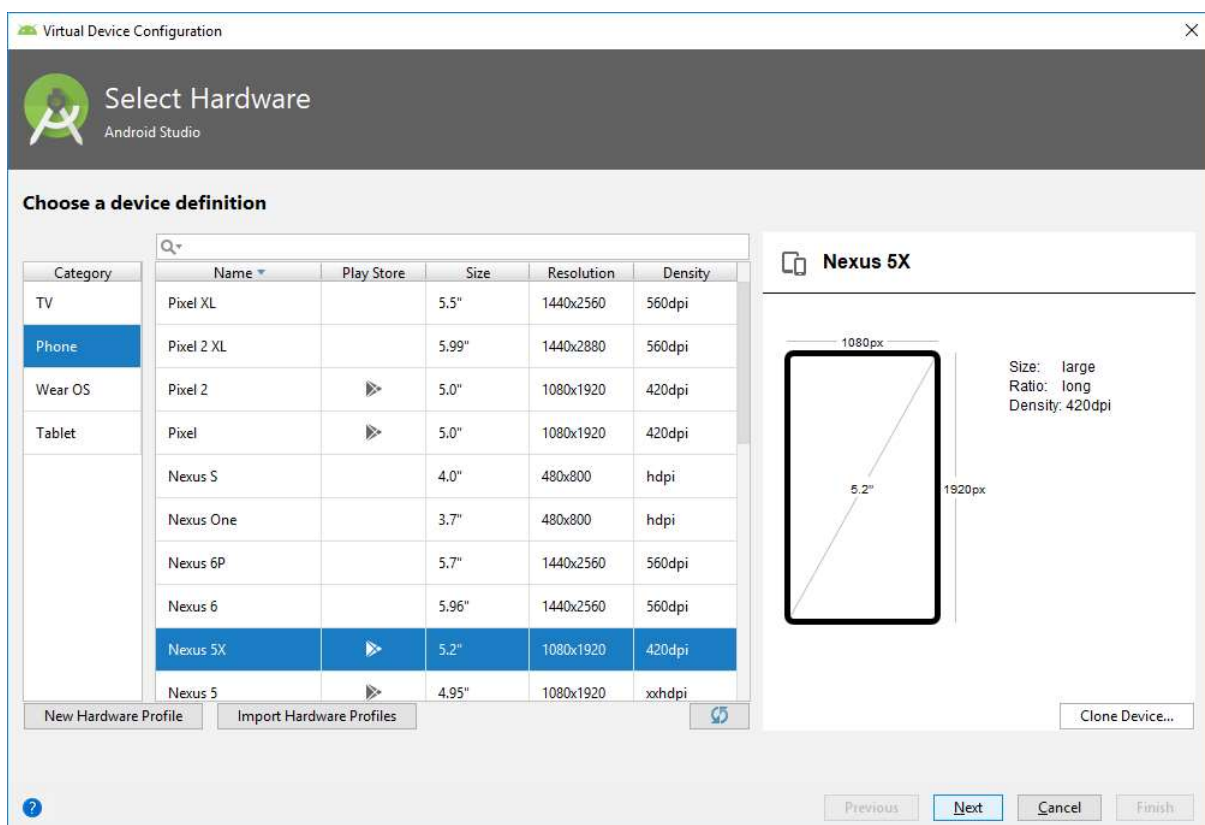


# DESARROLLO DE APLICACIONES MULTIPLATAFORMA

## T1. ANDROID. EL SISTEMA OPERATIVO



Podemos crear tantos dispositivos virtuales como queramos, de momento solo nos hace falta uno:

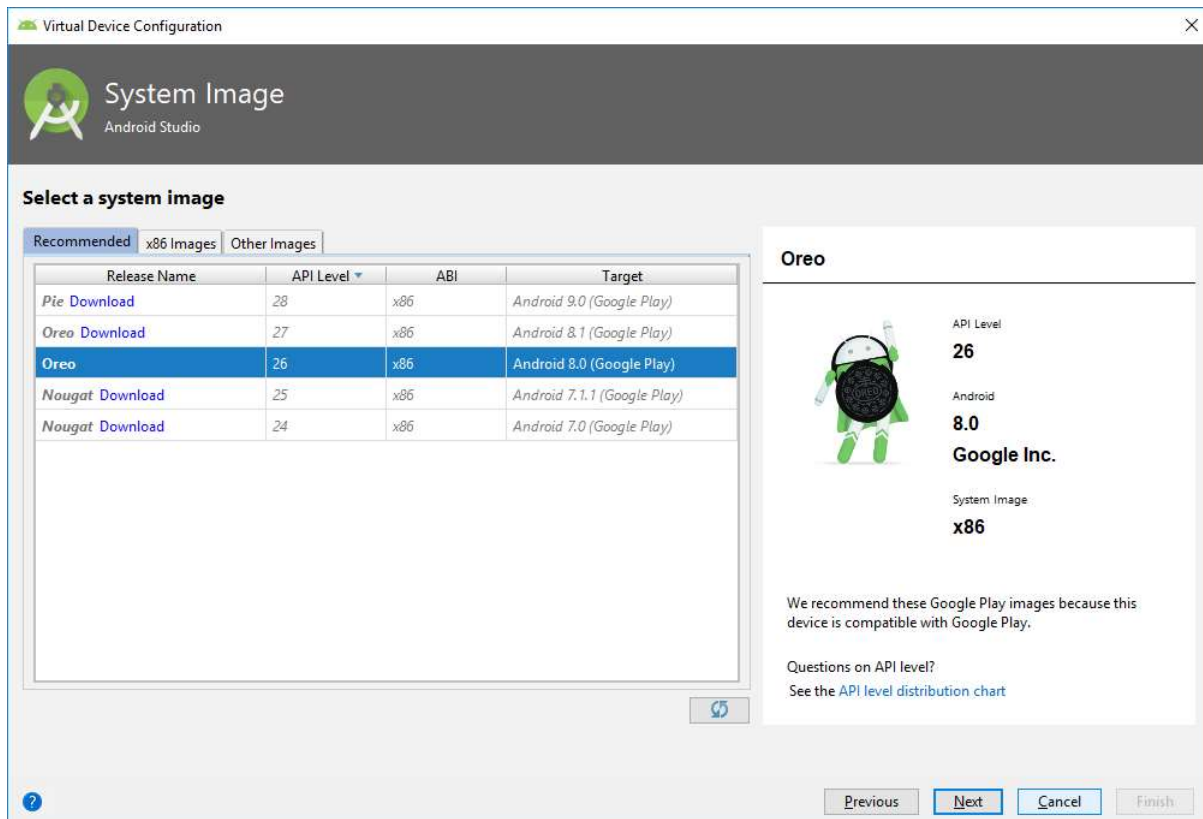


En el pantallazo verás que hemos creado, por ejemplo, un dispositivo virtual basado en un teléfono Nexus 5X, con pantalla grande "large".

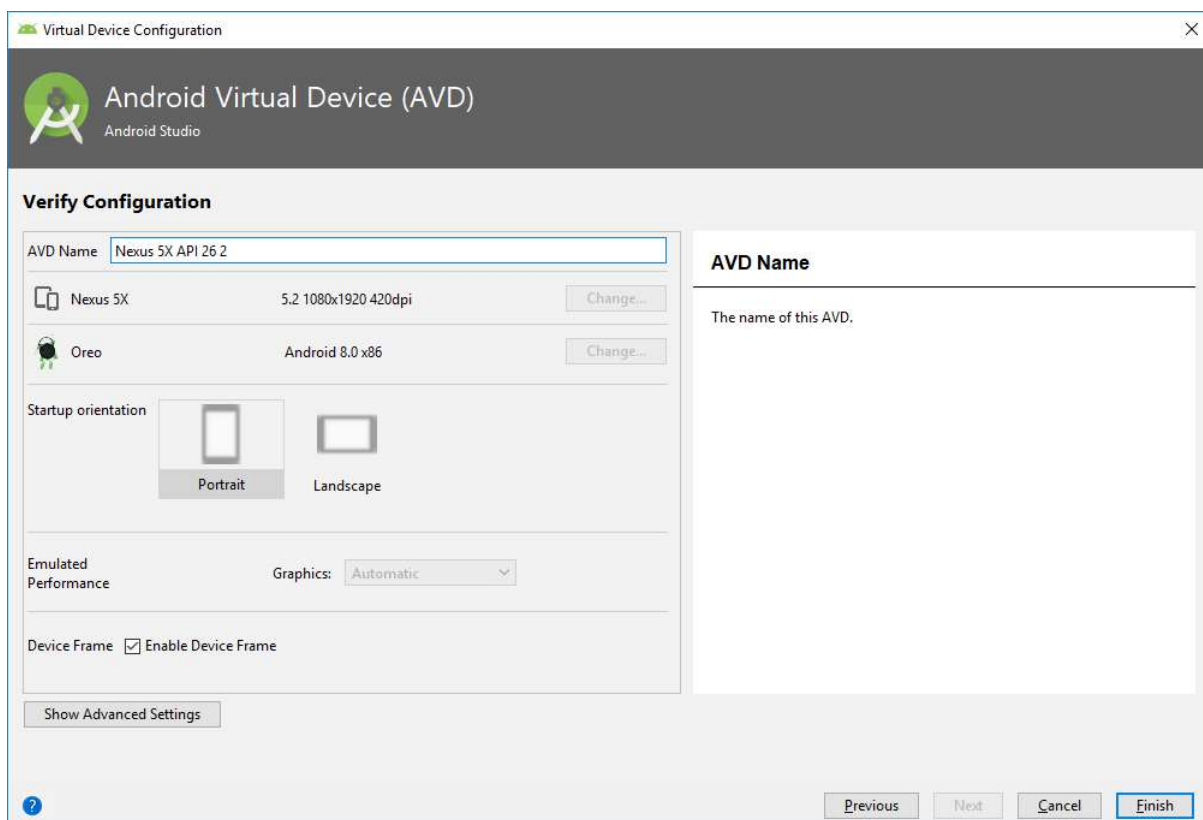
A continuación, selecciona la imagen del sistema operativo que quieras seleccionando la versión que le vas a instalar a tu dispositivo virtual:

# DESARROLLO DE APLICACIONES MULTIPLATAFORMA

## T1. ANDROID. EL SISTEMA OPERATIVO

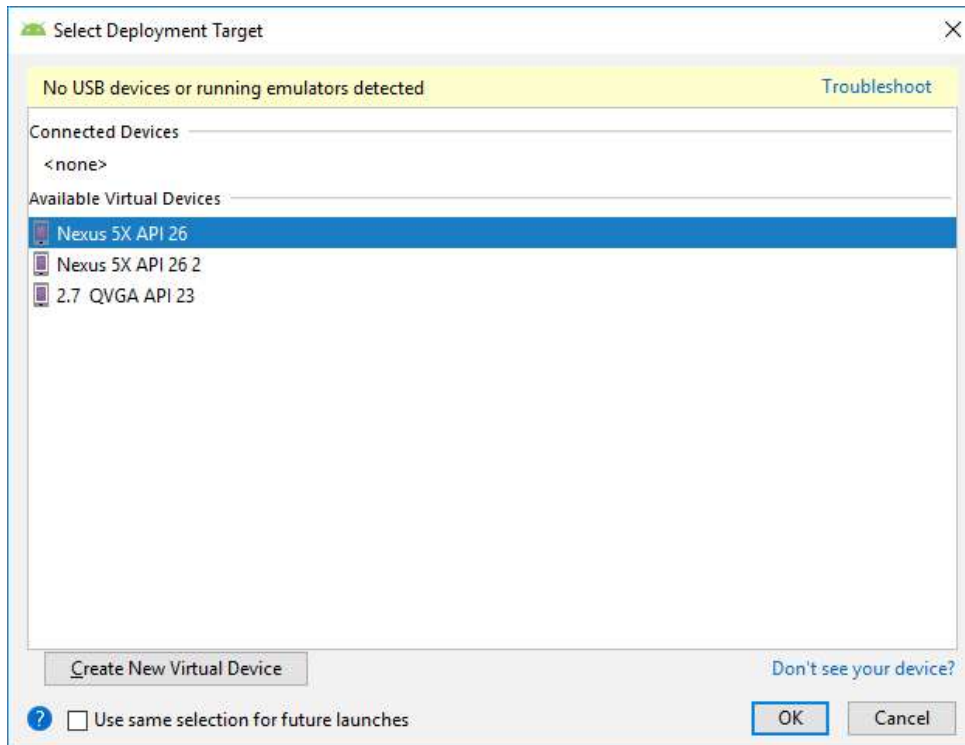


Finalmente, ponle un nombre y dale a Finalizar:

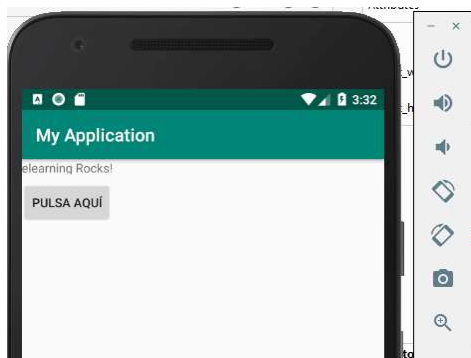


## 1.12. Probando, probando...

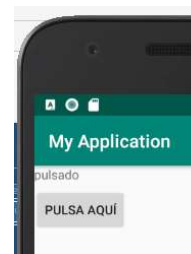
Ahora sí, compilado el código y creado el emulador, volvemos a lanzar la ejecución de la app y esta vez, podemos seleccionar el dispositivo creado.



Tardará un poco en arrancar, pero una vez arrancado no es necesario arrancarlo de nuevo entre ejecución y ejecución de tu app.



Después, al pulsar, ocurrirá lo que le hemos programado:



## 1.13. Entendiendo un poco más el código

Varios aspectos fundamentales deben quedarte claro desde este ejemplo:

A: La necesidad de conseguir una referencia a los widget de la interfaz de usuario.

```
miBoton=(Button)findViewById(R.id.button);  
miBoton.setOnClickListener(this);
```

La primera instrucción declara la referencia, la segunda, consigue el acceso al widget y a partir de ahí, ya podemos operar con el. Ten en cuenta que este código debe ser situado después de la instrucción setContentView(R.layout.activity\_mi\_primera); Si no lo haces, el resultado de la llamada a findViewById() será nulo y no podrás acceder al widget.

B: La necesidad de implementar la interfaz para responder mediante callback al evento del click:

```
public class MainActivity extends AppCompatActivity implements View.OnClickListener  
{  
...  
}
```

C: El registro de la función callback:

```
miBoton.setOnClickListener(this);
```

this es la referencia al objeto creado de la clase actual, que como implementa la función de callback OnClickListener, pues se puede pasar como parámetro.

D: La programación de la función OnClickListener():

```
@Override  
public void onClick(View view) {  
    //responde al evento Click  
    miTexto=(TextView)findViewById(R.id.textView);  
    miTexto.setText("pulsado");  
}
```

Consistente en obtener la referencia al objeto de texto y establecer el valor pulsado (método setText)

## 1.14. Otros emuladores

Otra opción muy extendida para aquellos con problemas de recursos a la hora de ejecutar un emulador o con procesadores no Intel o sin soporte a la virtualización, es el software de una empresa llamada Genymotion, que proporciona un emulador muy potente y rápido. Tiene una versión gratis, funciona a través de máquinas virtuales Virtual Box, y aunque exige registro, es muy completa:

# DESARROLLO DE APLICACIONES MULTIPLATAFORMA

## T1. ANDROID. EL SISTEMA OPERATIVO

