ASIGNATURA: Programación de Móviles.

TEMA: Creación de proyectos Android y Kotlin.

ACTIVIDAD DE ENSEÑANZA – APRENDIZAJE – EVALUACIÓN: Utilizar la herramienta Android Studio diseñar y construir app móviles nativas.

TIEMPO DE LA ACTIVIDAD DE E-A-E: 1 hora

TIEMPO DE LA GUIA DE APRENDIZAJE: 1 hora

ORIENTACIONES GENERALES: Seguir el siguiente guion para construir y lanzar una aplicación en Android.

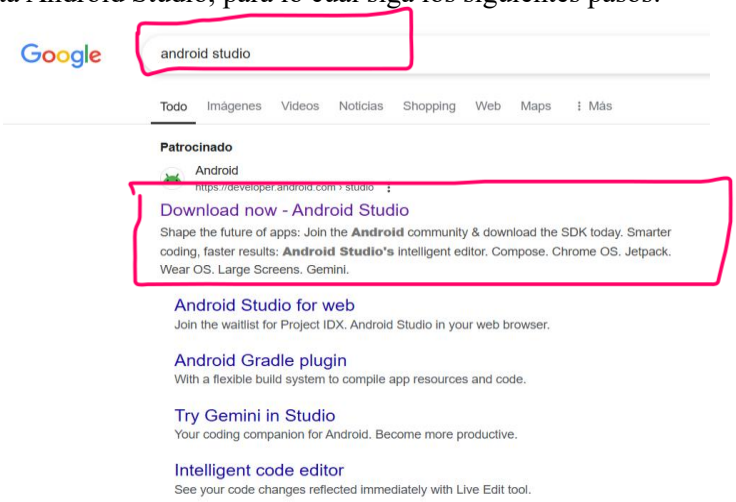
Conceptos

Entorno de desarrollo Android
En este apartado vamos a describir los pasos básicos para disponer en nuestro PC del entorno y las herramientas necesarias para comenzar a programar aplicaciones para la plataforma Android.

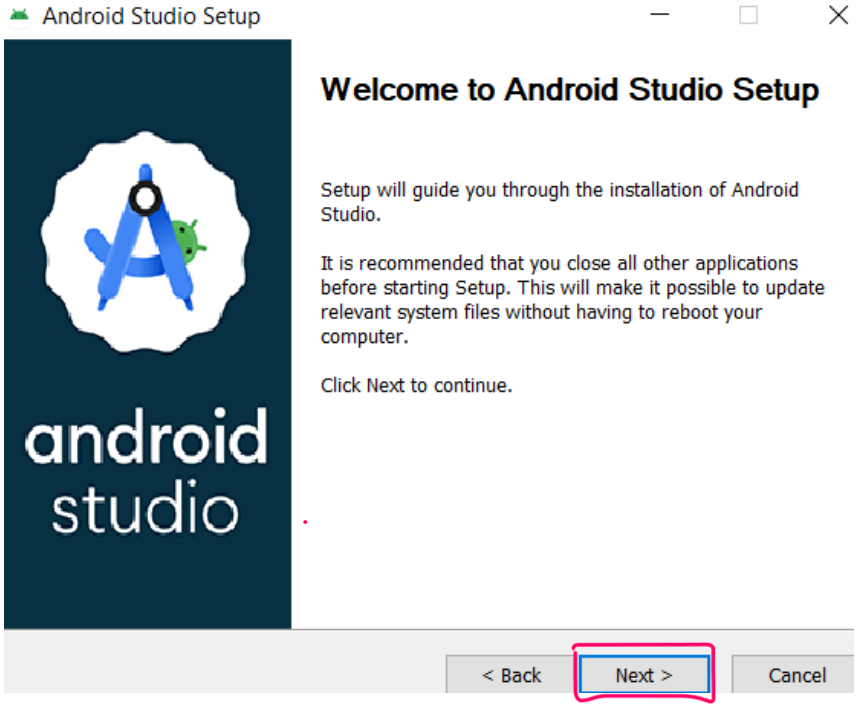
Utilizar un entorno de desarrollo nos facilita mucho la creación de programas. Esto es especialmente importante en Android dado que tendremos que utilizar una gran variedad de ficheros. Gracias a Android Studio, la creación y gestión de proyectos se realizará de forma muy rápida, acelerando los ciclos de desarrollo.

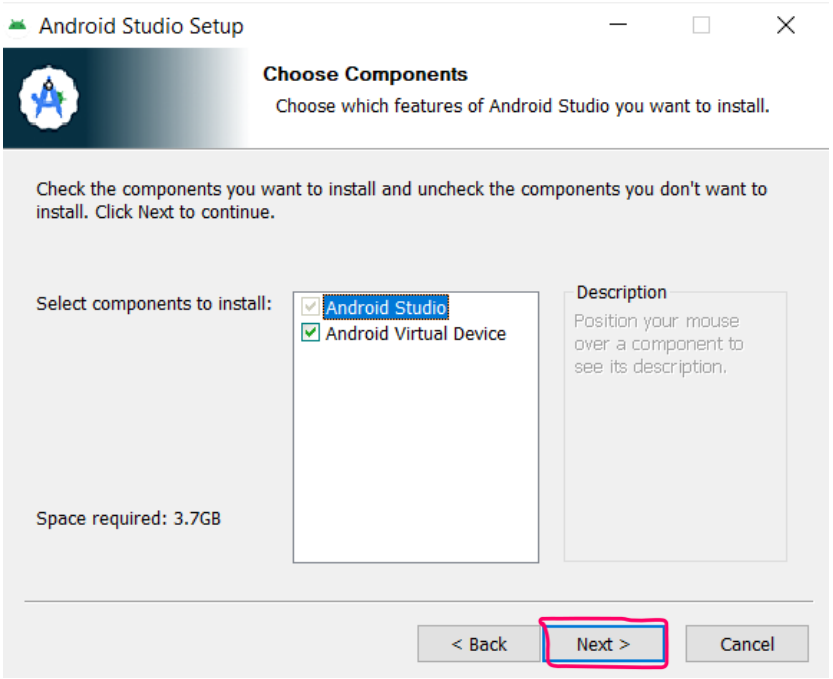
Ejercicio paso a paso.
Para crear un primer proyecto Android o en Kotlin, con Android Studio sigue los siguientes pasos:

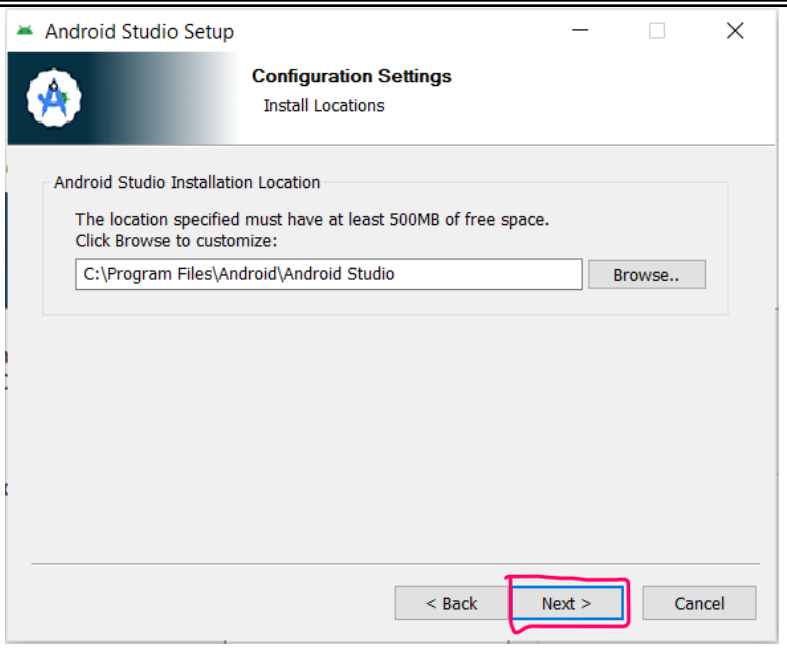
1. Descargue la herramienta Android Studio, para lo cual siga los siguientes pasos:

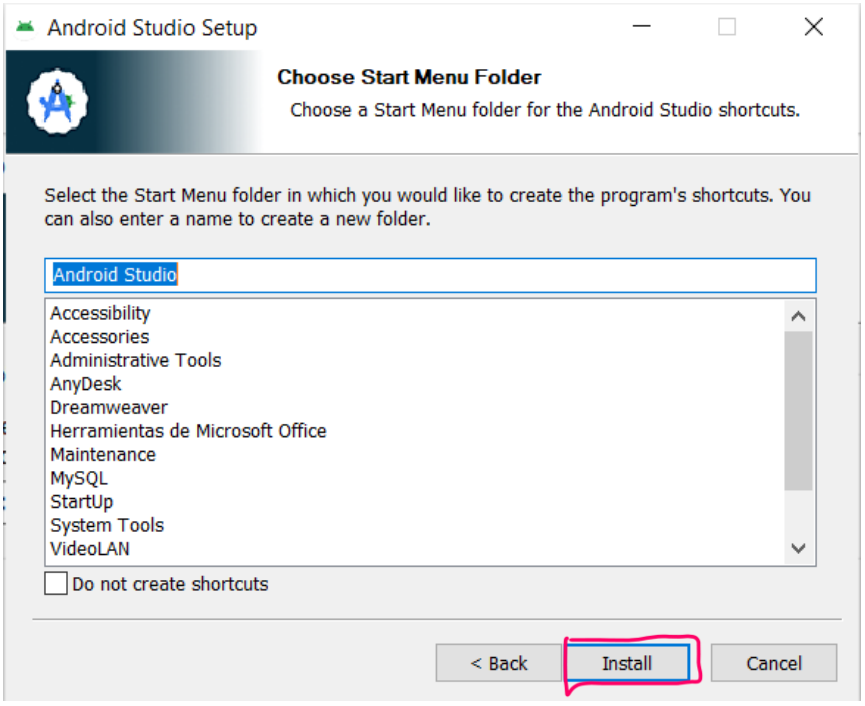


Des pues de haber descargado la herramienta, proceda con la instalación abriendo el archivo, posteriormente pulse en la opción **Next**

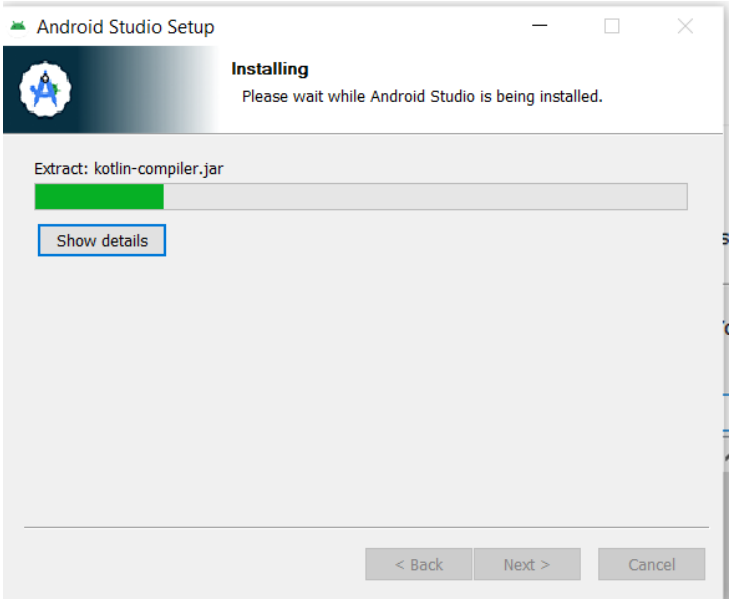


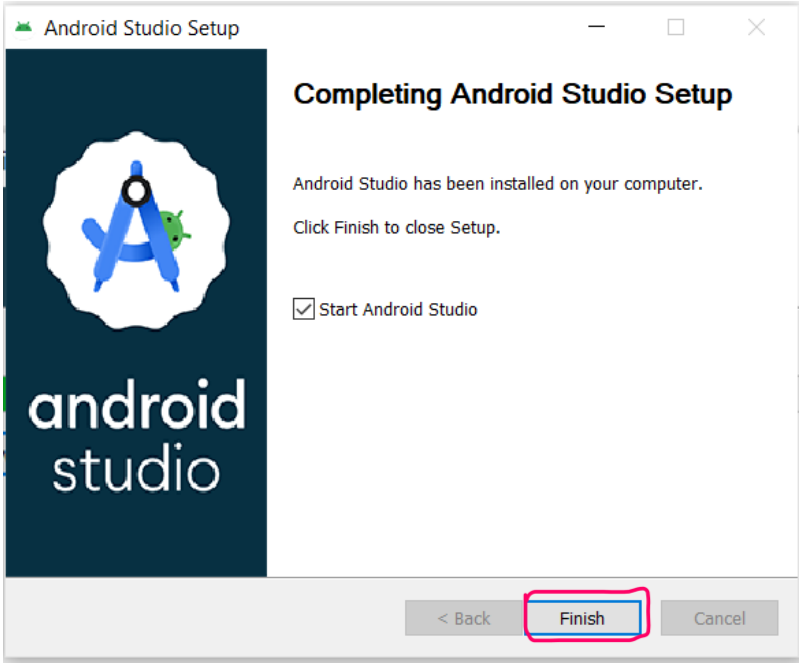



Android Studio Setup
Configuration Settings
Install Locations
Android Studio Installation Location
The location specified must have at least 500MB of free space.
Click Browse to customize:
C:\Program Files\Android\Android Studio
Browse..
< Back
Next >
Cancel


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
Accessibility
Accessories
Administrative Tools
AnyDesk
Dreamweaver
Herramientas de Microsoft Office
Maintenance
MySQL
StartUp
System Tools
VideoLAN
Do not create shortcuts
< Back
Install
Cancel

Proceso de instalación…

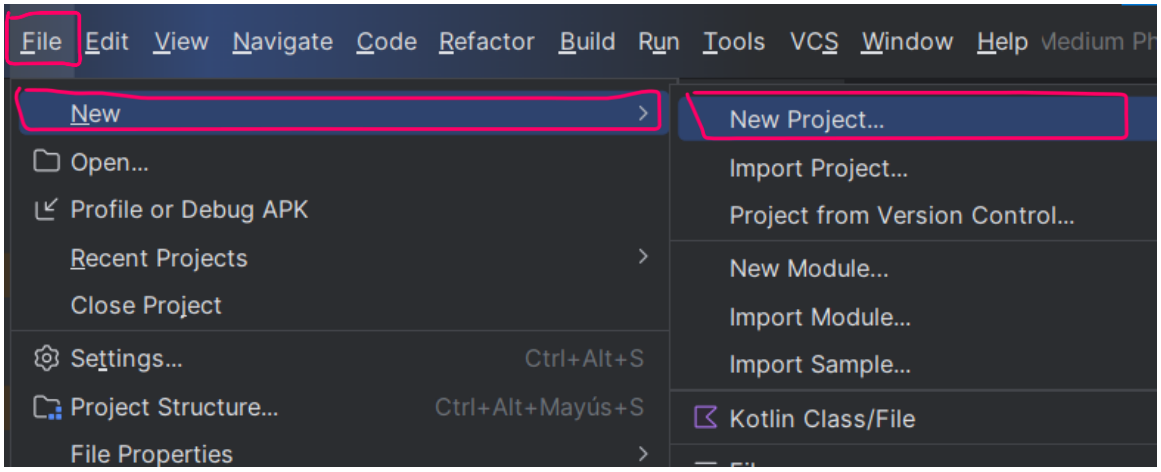




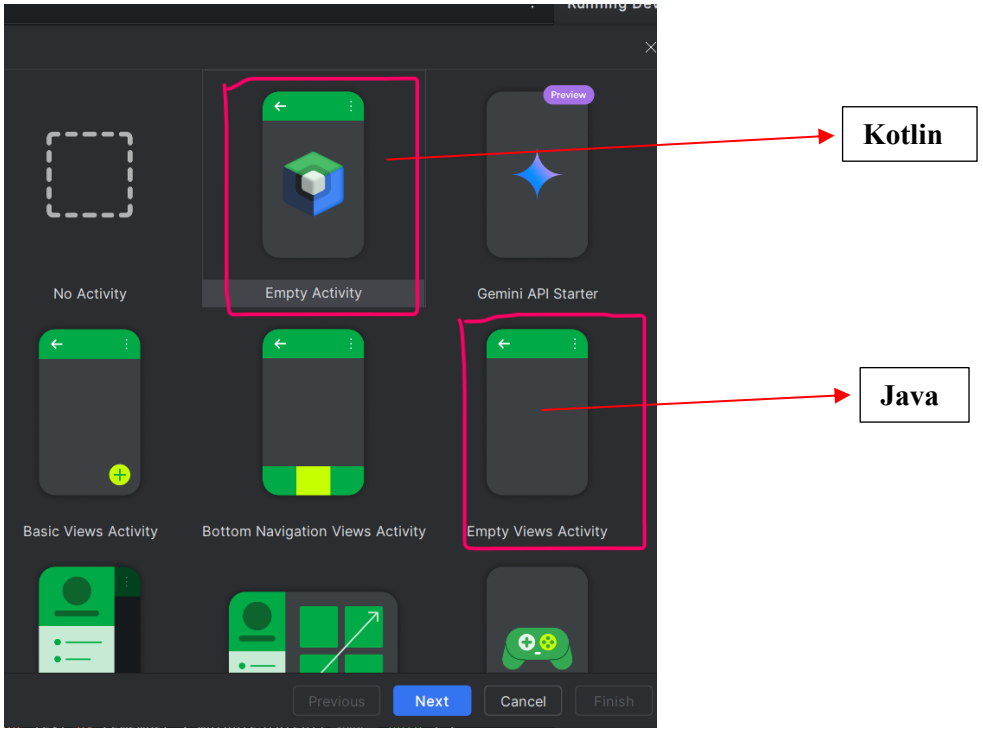
Por último, para finalizar la instalación pulse sobre el botón Finish.

Para crear un proyecto siga los siguientes pasos:

1. Selecciona File > New > New Project...

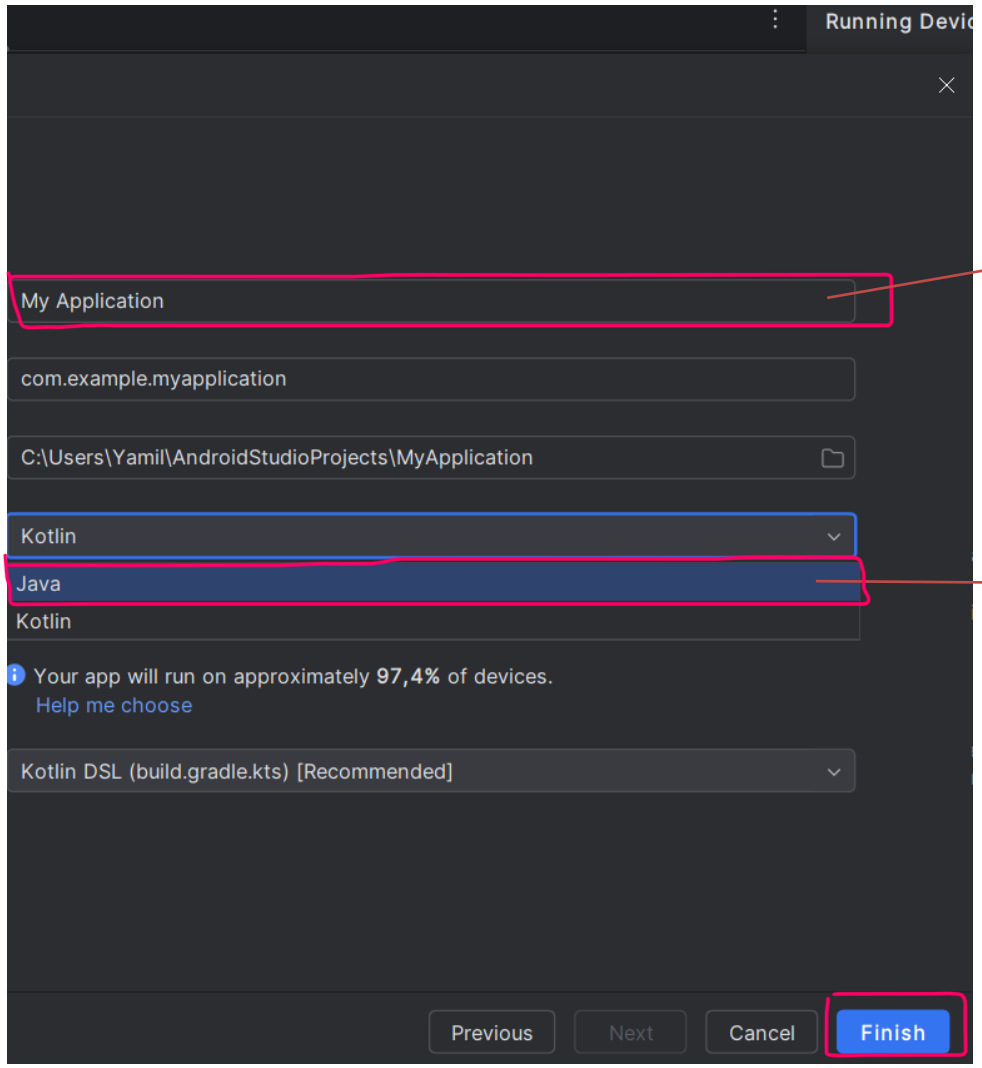


2.Luego seleccione una plantilla de acuerdo al tipo de lenguaje que se vaya a desarrollar **Java o Kotlin**

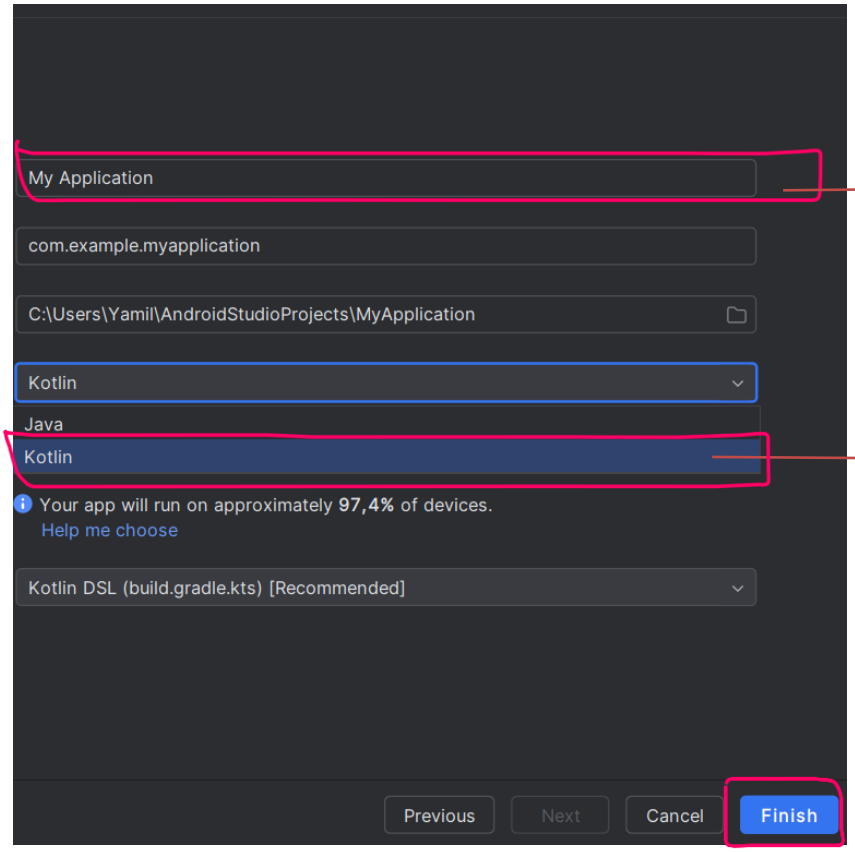


Profesor: Yamil Buenaños Palacios
ybuenano@usbbog.edu.co

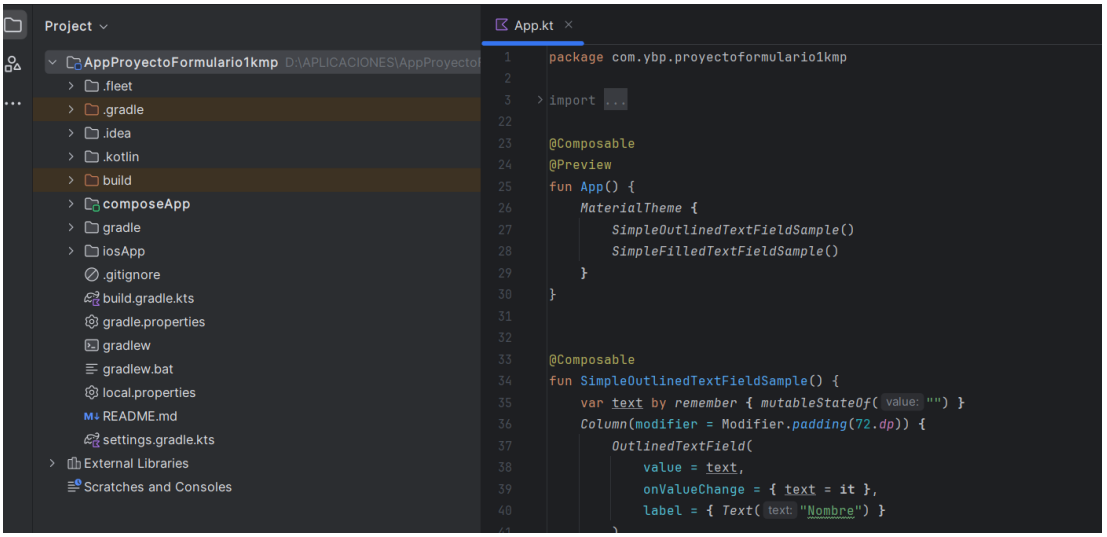
En caso de desarrollar en Java seleccione la plantilla y luego pulse sobre el botón **Next**, luego rellene los datos ingresando nombre del proyecto, luego seleccione la opción de java y por último pulse sobre el botón **Finish**.



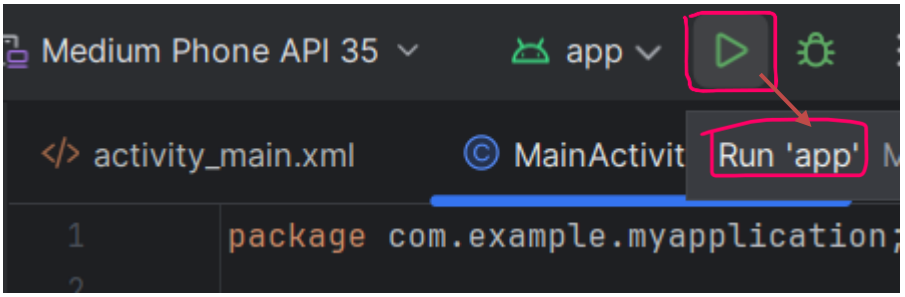
En caso que vaya a desarrollar con Kotlin repita los pasos anteriores y seleccione el lenguaje Kotlin



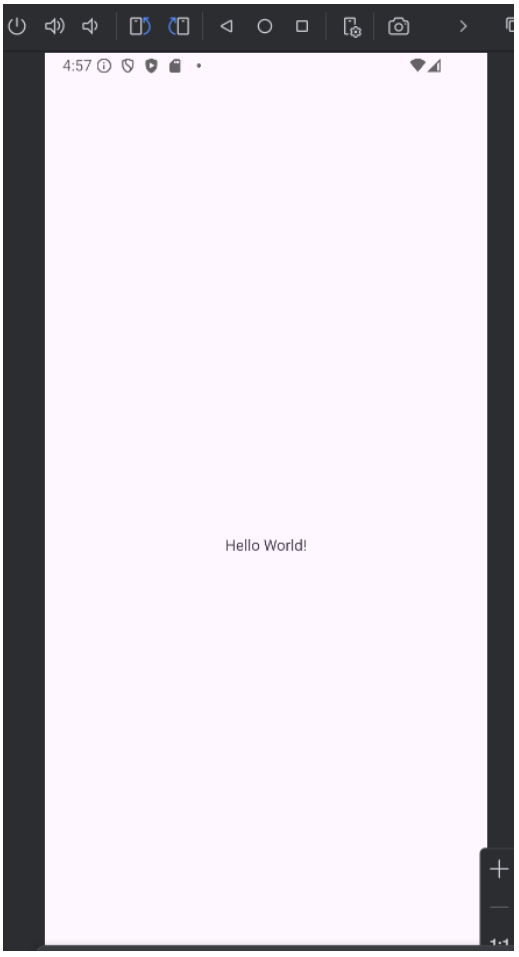
Proyecto creado tendrá la siguiente forma



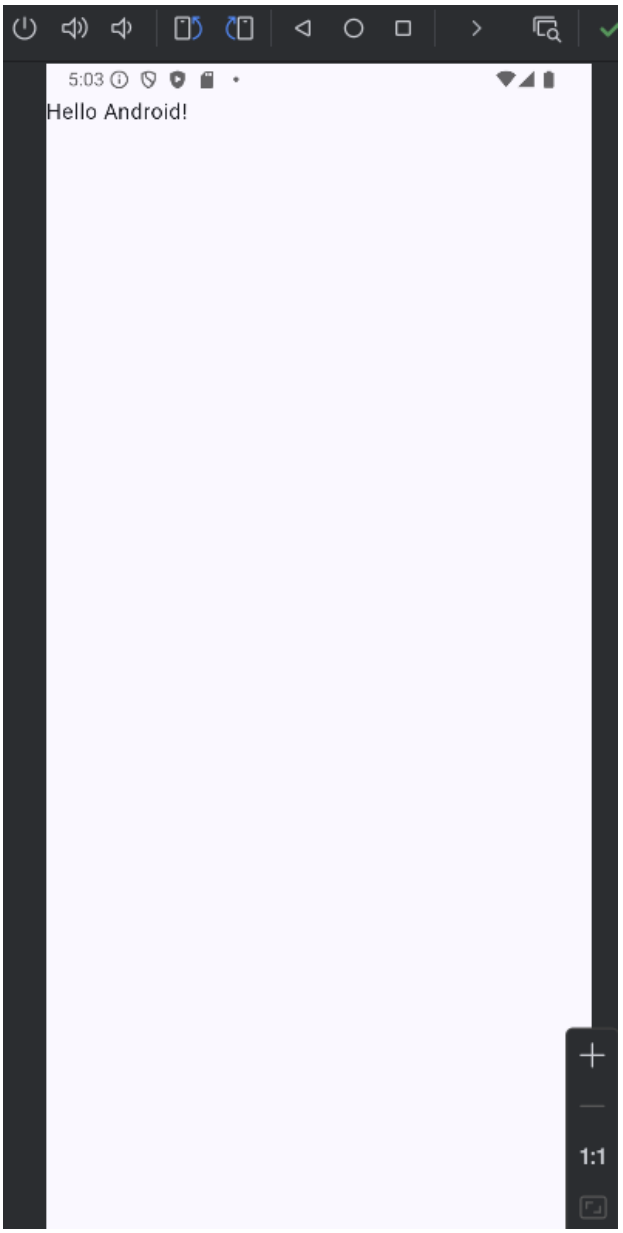
Para ejecutar el proyecto pulse sobre el símbolo Play



Emulador con Android



Emulador con Kotlin



Estructura de un programa en Android

AndroidManifest.xml: Este fichero describe la aplicación Android. Se define su nombre, paquete, icono, estilos, etc. Se indican las *actividades*, las *intenciones*, los *servicios* y los *proveedores de contenido* de la aplicación. También se declaran los permisos que requerirá la aplicación. Se indica la versión mínima de Android para poder ejecutarla, el paquete Java, la versión de la aplicación, etc.
java: Carpeta que contiene el código fuente de la aplicación. Como puedes observar los ficheros Java se almacenan en carpetas según el nombre de su paquete.
MainActivity: Clase Java con el código de la actividad inicial.
ApplicationTest: Clase Java pensada para insertar código de testeo de la aplicación utilizando el API JUnit.
res: Carpeta que contiene los recursos usados por la aplicación.
drawable: En esta carpeta se almacenan los ficheros de imágenes (JPG o PNG) y descriptores de imágenes en XML.
mipmap: Es una carpeta con la misma finalidad que *res/drawable*. La única diferencia es que, si ponemos los gráficos en mipmap, estos no son rescaldados para adaptarlos a la densidad gráfica del dispositivo donde se ejecuta la aplicación, sino que se buscará en las subcarpetas el gráfico con la densidad más parecida y se utilizará directamente. Es recomendable guardar los iconos de aplicación en esta carpeta [1]. En el proyecto se ha incluido el fichero *ic_launcher.png* que será utilizado como icono de la aplicación. Observa como este recurso se ha añadido en cuatro versiones diferentes. Como veremos en el siguiente capítulo, usaremos un sufijo especial si queremos tener varias versiones de un recurso, de forma que solo se cargue al cumplirse una determinada condición. Por ejemplo: (hdpi) significa que solo ha de cargar los recursos contenidos en esta carpeta cuando el dispositivo donde se instala la aplicación tiene una densidad gráfica alta (180- dpi); (mdpi) se utilizará con densidad gráfica alta (180- dpi). Si pulsas sobre las diferentes versiones del recurso, observarás como se trata del mismo icono, pero con más o menos resolución, de forma que en función de la densidad gráfica del dispositivo se ocupe un tamaño similar en la pantalla. Véase la referencia sobre recursos alternativos en el anexo y para más detalles.
layout: Contiene ficheros XML con vistas de la aplicación. Las vistas nos permitirán configurar las diferentes pantallas que compondrán la interfaz de usuario de la aplicación. Se utiliza un formato similar al HTML usado para diseñar páginas web. Se tratarán en el siguiente capítulo.
menu:Ficheros XML con los menús de cada actividad. En el proyecto no hay ningún menú por lo que no se muestra esta carpeta.
values: También utilizaremos ficheros XML para indicar valores usados en la aplicación, de esta manera podremos cambiarlos desde estos ficheros sin necesidad de ir al código fuente. En *colors.xml* se definen los tres colores primarios de la aplicación. En *dimens.xml*, se ha definido el margen horizontal y vertical por defecto. Observa cómo hay dos ficheros, el usado por defecto y el etiquetado como (w820dp) que será utilizado en dispositivos con ancho superior a 820 dp, esto ocurrirá en tabletas. En el fichero *strings.xml*, tendrás que definir todas las cadenas de caracteres de tu aplicación. Creando recursos alternativos resultará muy sencillo traducir una aplicación a otro idioma. Finalmente, en *styles.xml*, podrás definir los estilos y temas de tu aplicación.
anim: Contiene ficheros XML con animaciones de vistas (Tween).
animator: Contiene ficheros XML con animaciones de propiedades.
xml: Otros ficheros XML requeridos por la aplicación.
raw: Ficheros adicionales que no se encuentran en formato XML.

Gradle Scripts: En esta carpeta se almacenan una serie de ficheros Gradle que permiten compilar y construir la aplicación. Observa cómo algunos hacen referencia al módulo app y el resto son para configurar todo el proyecto. El fichero más importante es *build.gradle (Module:app)* que es donde se configuran las opciones de compilación del módulo:

El primer parámetro que podemos configurar es compileSdkVersion que nos permite definir la versión del sdk con la que compilamos la aplicación. Las nuevas versiones no solo añaden funcionalidades al API, también añaden mejoras en los procesos. Por ejemplo, a partir de la versión 3.0 (API 11) solo se permite el acceso a Internet desde un hilo auxiliar[2]. **buildToolsVersion** define la versión de las herramientas de construcción. Interesa que estos valores correspondan con las últimas versiones disponibles. Puedes utilizar *Android SDK Manager* para descargarte las últimas versiones. **applicationId** ha de coincidir con el nombre del paquete Java creado para la aplicación. Se utiliza como identificador único de la aplicación, de forma que no se permite instalar una aplicación si ya existe otra con el mismo paquete. **minSdkVersion** especifica el nivel mínimo de API que requiere la aplicación. Es un parámetro de gran importancia, la aplicación no podrá ser instalada en dispositivos con versiones anteriores y solo podremos usar las funcionalidades del API hasta este nivel (con excepción de las librerías de compatibilidad). **targetSdkVersion** indica la versión más alta con la que se ha puesto a prueba la aplicación. Cuando salgan nuevas versiones del SDK tendrás que comprobar la aplicación con estas versiones y actualizar el valor. **versionCode** y **versionName** indica la versión de tu aplicación. Cada vez que publiques una nueva versión incrementa en uno el valor de **versionCode** y aumenta el valor de **versionName** según la importancia de la actualización. Si es una actualización menor el nuevo valor podría ser "1.1" y si es mayor "2.0".
Dentro de **buildTypes** se añaden otras configuraciones dependiendo del tipo de compilación que queramos (relase para distribución, debug para depuración, etc.). Los comandos que aparecen configuran la ofuscación de código. Para más información leer el capítulo «Ingeniería Inversa en Android» de El *Gran Libro de Android Avanzado*.
Un apartado importante es el de **dependencies**. En él has de indicar todas las librerías que han de ser incluidas en nuestro proyecto. Si necesitas usar alguna librería de compatibilidad adicional has de incluirla aquí.