Apuntes

Descargar estos apuntes

Tema 2 . Estructura de un proyecto Android Studio

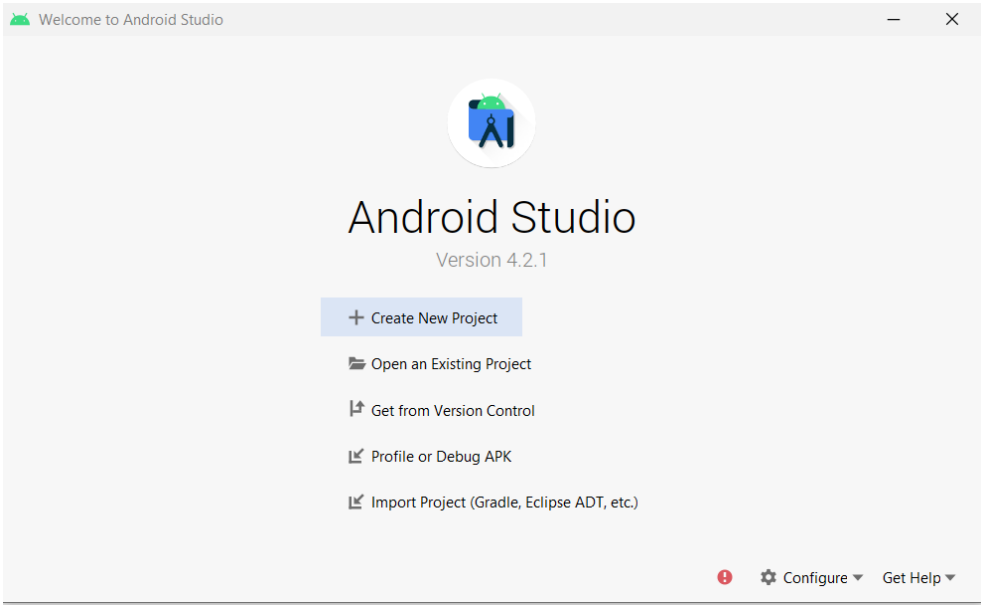
Índice

Conocimientos iniciales
Estructura de un proyecto Android Studio
¿Qué es el archivo androidmanifest.xml?
Uso recursos en Android FALTA

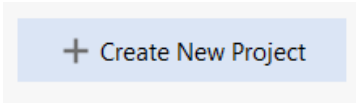
Indroducción

Creando el primer proyecto de Android

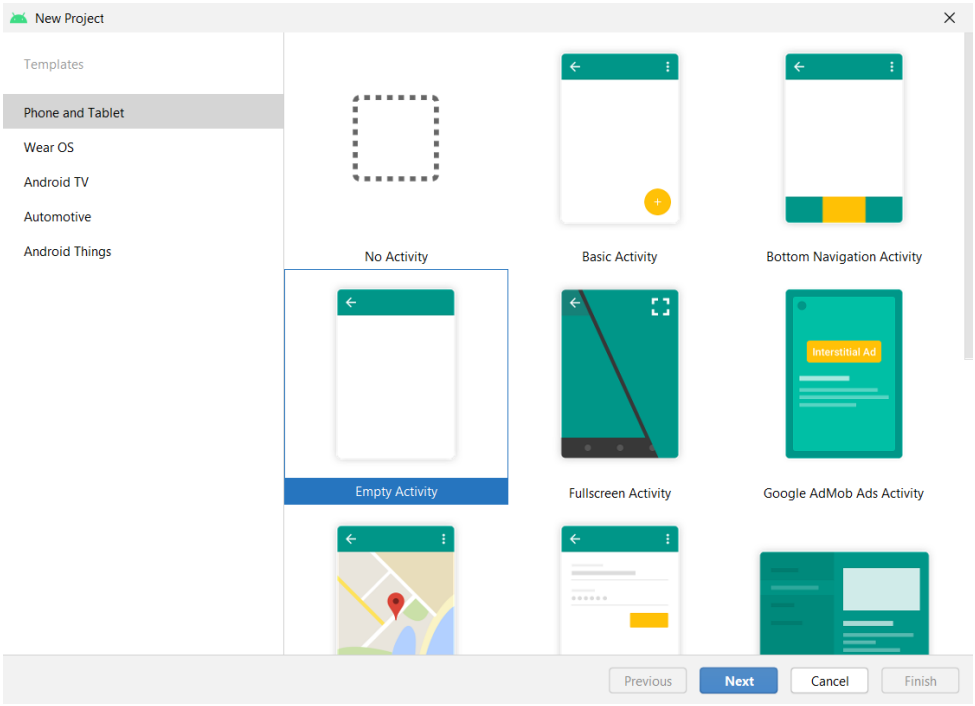
Vamos a realizar nuestro primer ejemplo en Android, nos servirá básicamente para conocer el entorno Android Studio y las carpetas de las que consta un proyecto Android creadas con este entorno. Una vez instalada la aplicación y si no se ha creado un proyecto con anterioridad, al abrir el entorno nos encontraremos con una ventana como la que sigue:



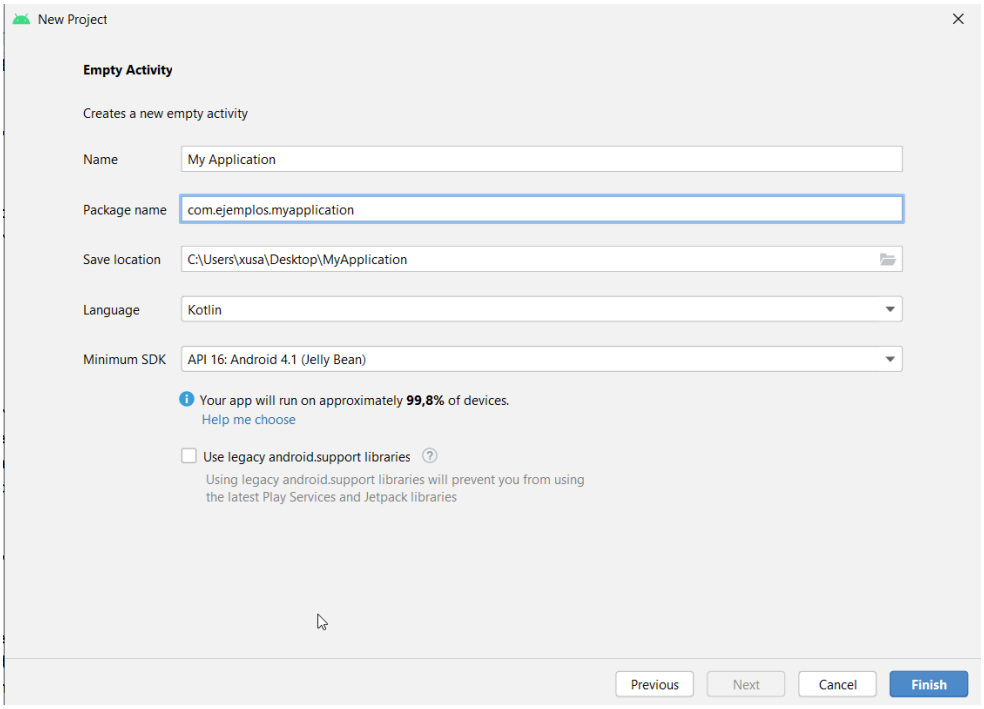
Como podemos ver, esta ventana nos permite crear un proyecto nuevo, abrir uno existente o realizar otras opciones, pulsaremos en **Create a New Project**.



Deberemos seleccionar el tipo de actividad que se creará como inicio de nuestro proyecto, la opción que necesitaremos a lo largo del curso es una **Empty Activity**, que nos creará una actividad vacía.



A partir de aquí deberemos seleccionar el nombre de paquete que vamos a dar a nuestra aplicación, la localización y el nombre de la aplicación, además también deberemos seleccionar el Lenguaje con el que queremos trabajas (Java o Kotlin). Es mejor no editar el nombre de paquete, se generará solo al modificar el nombre de dominio y el nombre de aplicación. En **Save location** podemos seleccionar donde queremos dejar nuestra aplicación.



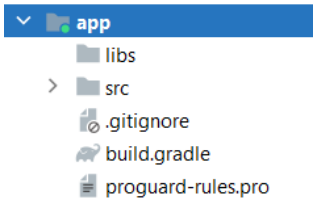
La última opción permitirá seleccionar el **SDK mínimo** con el que queráis que corra vuestra aplicación. El máximo no hace falta seleccionarlo porque el proyecto cogerá el SDK más actualizado que tengas instalado. Junto al selector del sdk, podéis ver una explicación y un enlace en el que se pueden ver las estadísticas actualizadas del porcentaje de móviles que podrán correr vuestra aplicación, según el mínimo SDK seleccionado.

A partir de aquí deberemos esperar un poco hasta que se genere nuestro proyecto y aparezca una ventana parecida a la siguiente:

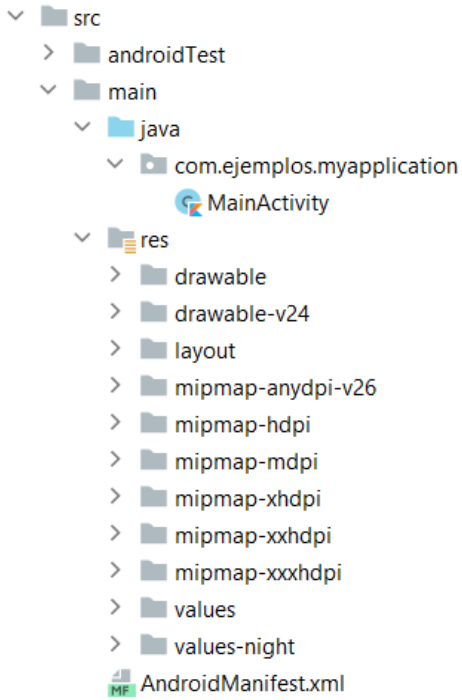
Carpetas de un proyecto en Android Studio

Una vez llegados aquí, se habrá abierto el árbol de carpetas de la aplicación a la izquierda (aunque esta ventana es movible. Por si os resulta más útil en otro lugar de la ventana de Android Studio, sabed que podéis moverla pulsando y arrastrando la pestaña con nombre **1: Project** a cualquier parte. También podéis mover el resto de las ventanas que están abiertas.
En el árbol de directorios tenéis varias formas de listar las carpetas y archivos de la aplicación. Las dos principales son **Android, Project, y Package**, pero hay otras que podéis usar, por ejemplo: **Problems** mostrará sólo los archivos que tienen errores.

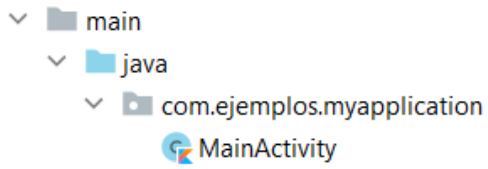
La **carpeta app** es la que contiene todo lo relacionado con tu proyecto, es la carpeta que nos interesa por el momento y donde incluiremos los archivos necesarios para que nuestra aplicación sea empaquetada. Si despliegas su contenido veras tres carpetas: build, libs y src. Por ahora ignoraremos los demás archivos.



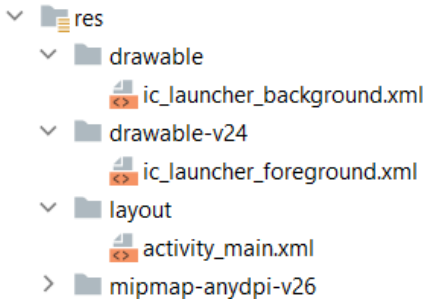
Normalmente la mayor parte del tiempo y fuerzas las usaremos en la carpeta **src**(Source). Dentro de ella se ubica la carpeta main, la cual contiene todos los archivos fuente Java para nuestra aplicación. La carpeta **res** (Resources) que contiene los recursos del proyecto (iconos, sonido, diseños, etc.) y el archivo **AndroidManifest.xml** , más adelante profundizaremos en él.



La Activity (o Activitys) que hemos creado se encuentran en `Nombre de vuestra App->src->main->java->Paquete de vuestra App`.



Algunos de los otros archivos imprescindibles de toda aplicación Android, son **los recursos** que se encuentran en la carpeta `Nombre de vuestra App->src->main->res`. Una vez dentro podéis ver, entre otras, las carpetas `layout`, `drawable` y `mipmap`.



Recursos en Android

Los recursos son archivos o datos externos que soportan el comportamiento de nuestra aplicación Android por ejemplo imágenes, strings, colores, estilos, etc.
Formalmente en un proyecto, estos elementos se encuentran en la carpeta **res** nombrada anteriormente. Allí se encuentran subdirectorios que agrupan los diferentes tipos de recursos.
La idea del uso de recursos es dividir el código de tu app para mantener independencia. Todo con el fin de agregar variaciones de los archivos para adaptar la aplicación a diferentes tipos de pantallas, idiomas, versiones, dimensiones, configuraciones de orientación, etc. Por ejemplo: *no es lo mismo el espacio de un teléfono móvil a una tableta. Sin embargo se pueden crear variaciones de los recursos para que la aplicación se adapte a la densidad de pantalla de cada uno.*
Los grupos de recursos se dividen en subdirectorios. Cada uno de ellos contiene archivos que cumplen una función específica dentro de la aplicación. Debes respetar esta estructura de documentos para no tener problemas en la ejecución. En la siguiente Tabla de Recursos, podemos ver un resumen de los más usados.

Nombre del subdirectorio	Contenido

Nombre del subdirectorio	Contenido
drawable	Recursos gráficos que puedan ser proyectados en pantalla. Generalmente encontrarás archivos de imagen como .png, .jpg o .gif, sin embargo es posible usar otros como: archivos nine-patch, listas de estado, drawables con múltiples niveles, drawables con figuras 2D definidas en XML, y muchas más.
mipmap	Contiene el o los iconos de la aplicación para evitar distorsión entre varias densidades de pantalla.
layout	Archivos XML que contienen definiciones de la interfaz de usuario.
menu	Archivos XML que establecen las características para los menús usados en la interfaz. Normalmente contienen definiciones sobre los ítems albergados en un menú y las agrupaciones entre ellos.
values	Archivos XML que contienen datos simples como enteros, strings, booleanos, colores.

Recursos Alternativos En Android

Un recurso alternativo es una variación de un recurso, que se ajusta a una característica de configuración en el dispositivo móvil donde se ejecuta la aplicación. Esto se logra a través de una tabla de calificadores creada por Google que estandariza todas las configuraciones posibles que se puedan presentar.
Un calificador es un mecanismo gramatical que especifica el propósito de un recurso. Su sintaxis es la siguiente: `<nombre_recurso>-<calificador>`

Por ejemplo: ***mipmap-hdpi*** *contendrá la variación del recurso ic_launcher.png para dispositivos con densidad de pantalla de 240dpi. Esta condición es especificada por el calificador hdpi*.

> ✋ todos los archivos que correspondan al mismo recursos deberán de tener el mismo nombre, se identificarán gracias al calificador añadido al identificador de carpeta.

Si hay que indicar más de un calificador, se separan por guiones y el orden de preferencia es el que se sigue en la tabla, el primer calificador será el de más preferencia. Por ejemplo: *dispositivos en inglés de Estados Unidos y en orientación horizontal, tendrán un nombre de recurso alternativo en el drawable, de la siguiente manera* ***drawable-en-rUS-land***.

> ✋ Una vez que guardes los recursos alternativos en directorios denominados con estos calificadores, Android aplicará automáticamente los recursos en tu aplicación de acuerdo con

> la configuración del dispositivo actual

La carpeta Layout

En la carpeta layout encontrarás los archivos que contienen las vistas de las actividades o fragments. Al crear el proyecto contiene el archivo `activiy_main.xml`. Este archivo representa el diseño de la interfaz de mi actividad principal.

Construir la interfaz a través de nodos XML es mucho más sencillo que la creación a través de código Java. Adicionalmente Android Studio nos ha dotado de un panel de diseño estilo **Drag and Drop**, lo cual es una bendición para los desarrolladores, ya que facilita la creación de la interfaz de usuario.

Este archivo de diseño comienza con un nodo raíz llamado `<ConstraintLayout>`. Un Layout es el contenedor principal que define el orden y secuencia en que se organizarán los widgets en nuestra actividad. Existen varios tipos de Layouts, como por ejemplo el `LinearLayout, GridLayout, FrameLayout, ConstraintLayout, etc`.

Android Studio crea por defecto un ConstraintLayout porque permite crear un grupo de componentes con ubicaciones relativas y además permite todo el control a través del Editor de Diseño. Si abres el archivo `activiy_main.xml` de tu aplicación verás algo como esto:

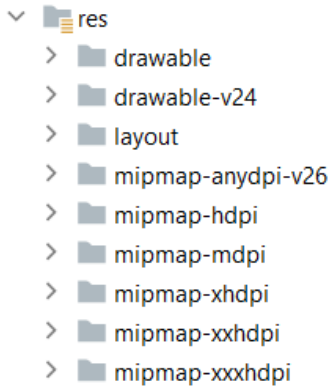
```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
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

    <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" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

Más adelante profundizaremos sobre los elementos de diseño tipo Layouts.

Las carpetas Mipmap y Drawable

Como ya has visto, hay una serie de carpetas que comienzan por el cualificador mipmap y drawable. Estas carpetas se relacionan directamente con el tipo de densidad de pantalla donde se

ejecutará nuestra aplicación.



La mayoría de dispositivos móviles actuales tienen uno de estos 4 tipos de densidades:

- **Mediun Dots per Inch(mdpi)**: Este tipo de pantallas tienen una densidad de 160 puntos por pulgada.
- **High Dots per Inch(hdpi)**: En esta clasificación encontraremos teléfonos cuya resolución es de 240 puntos por pulgada.
- **Extra high dots per inch(xhdpi)**: Resoluciones mayores a 340 puntos por pulgada
- **Extra Extra high dots per inch(xxhdpi)**: Rangos de resoluciones mayores a 480 puntos por pulgada.
- **Extra Extra high dots per inch(xxhdpi)**: Rangos de resoluciones mayores a 640 puntos por pulgada.

Miremos una pequeña imagen ilustrativa:

La imagen muestra algunos ejemplos de dispositivos móviles cuya densidad se encuentra dentro de los rangos establecidos. Al final podemos ver una categoría extra llamada xxxhdpi. Esta denominación describe a la densidad de televisores de última generación que ejecutan Android. Android Studio 3 crea un icono adaptativo para su aplicación que solo está disponible en SDK 26 y posteriores. Estos iconos adaptativos se incluyen en la carpeta `mipmap-anydpi-v26`, son archivos

.xml que se adaptan a la tecnología del dispositivo. Por ejemplo: *un ícono de selector adaptable se puede mostrar con una forma circular en un dispositivo OEM y con un cuadrado con esquinas redondeadas en otro*.
La carpeta **drawable** sirve para colocar todas las imágenes que vaya a utilizar nuestra aplicación, siguiendo el mismo esquema de sufijos que mipmap. Mientras que en la carpeta mipmap únicamente colocaremos el icono de la app. Como podemos observar, cuando creamos un proyecto en Android Studio, por defecto, ya nos coloca ahí el icono con el nombre de ic_launcher.

La carpeta values

Dentro de la carpeta **values** se crean por defecto tres archivos `colors, strings, themes`, vamos a comentar los dos primeros y dejas el de themes para más adelante, ya que necesita una explicación mas detallada.
Dentro de un archivo de colores **(colors.xml)** los elementos deben ser declarados con la etiqueta `<color>`. Se debe asignar un nombre único con el atributo name y su contenido es un color en forma hexadecimal que empieza por el símbolo numeral '#'. Se puede acceder a la paleta de colores al pulsar sobre cualquiera de los cuadros de color de la izquierda.

```xml
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="purple_200">#FFBB86FC</color>
    <color name="purple_500">#FF6200EE</color>
    <color name="purple_700">#FF3700B3</color>
    <color name="teal_200">#FF03DAC5</color>
    <color name="teal_700">#FF018786</color>
    <color name="black">#FF000000</color>
    <color name="white">#FFFFFFFF</color>
</resources>
```

Uno de los recursos más relevantes es el archivo strings.xml que se encuentra dentro de la subcarpeta values. Este fichero almacena todas las cadenas que se muestran en los widgets (controles, formas, botones, vistas, etc.) de nuestras actividades.
Por ejemplo, si tuvieses un botón cuyo título es “Presiona aquí”, es recomendable incluir dicha cadena en tu archivo strings.xml.