INDICE

1.1. Introducción

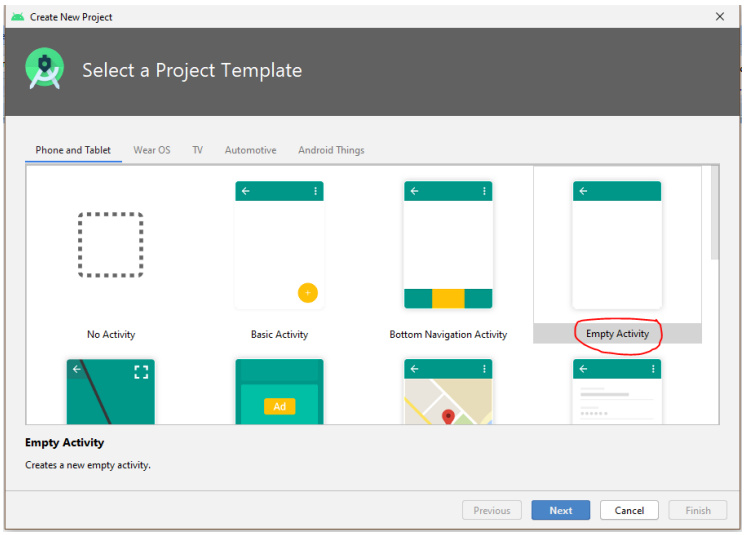
- Empezaremos por crear un proyecto que servirá de base para desarrollar el curso.
- Si el estudiante quiere puede crear un proyecto diferente para cada ejercicio o para cada unidad ...

1.2. Crear un proyecto base

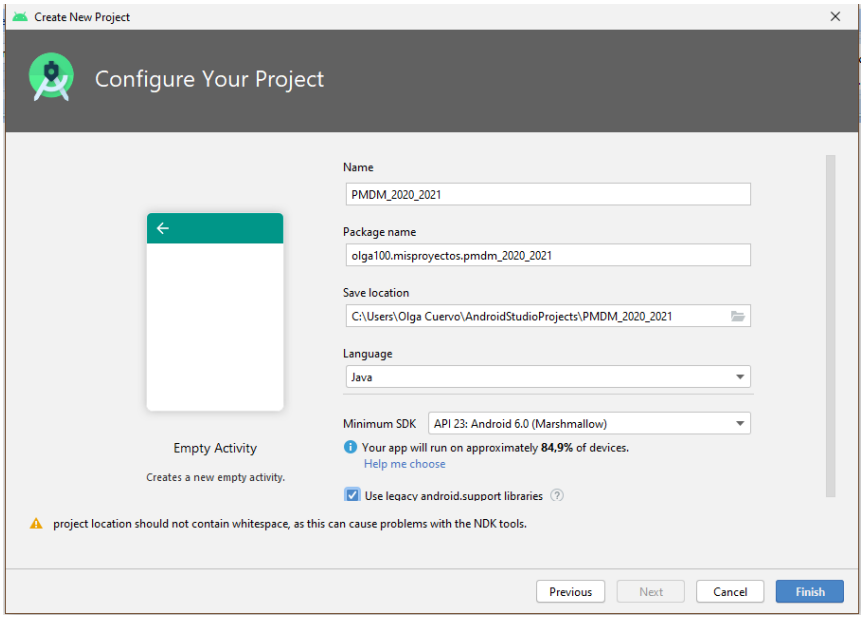
- En cada unidad crearemos un paquete diferente dentro del mismo proyecto.

  En cada paquete pondremos las diferentes actividades creadas.

- Crea un nuevo proyecto:
  - Indicamos que desarrollaremos una aplicación móvil.
  - Elegimos uno de los patrones. En el ejemplo será una pantalla en blanco.

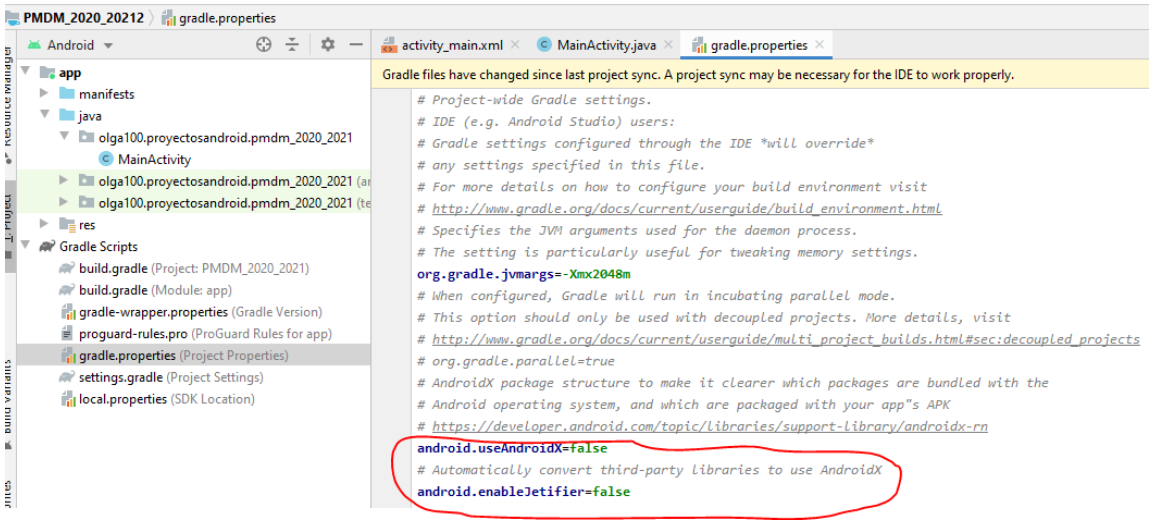


- Nombre de la aplicación: **PMDM_2020_2021**.
- Nombre del paquete: **olga100.proyectosandroid.pmdm_2020_2021**
- Carpeta donde se guarda el proyecto: Podemos dejar lo que trae por defecto.
- Idioma: **Java**
- Nivel mínimo de API: 23
- Marcaremos la opción **androidx** (*Use legacy android.support libraries*)



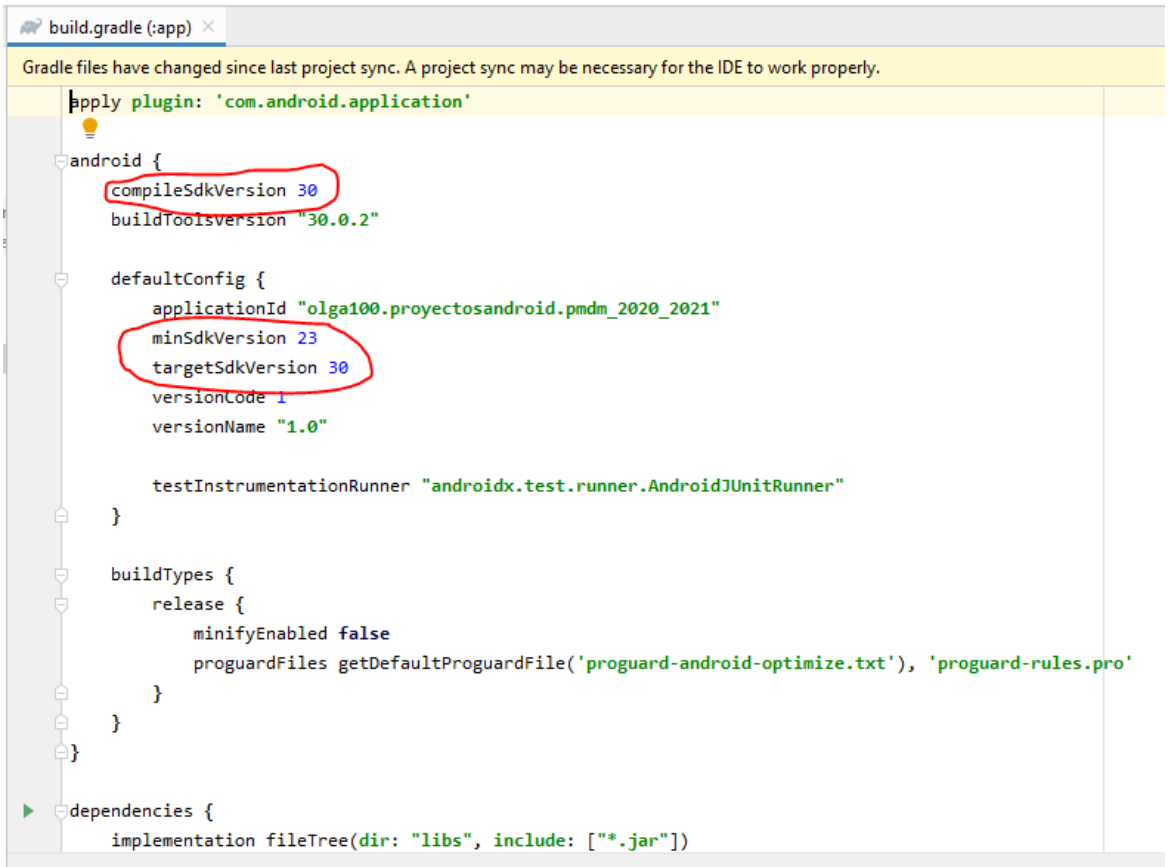
Pulsar el botón **Finalizar**.

NOTA: Si desea desmarcar la última opción (*androix*) y no le permite eliminarla, una vez creado el proyecto, debe ir al archivo **gradle.properties** y cambiar las propiedades *'enableJetifier'* y *'useAndroidX'* a FALSE. Una vez cambiado, tendrá que hacer clic en el enlace "sincronizar ahora" en la parte superior derecha.



- Datos del archivo **build.gradle** a nivel de módulo (puede ser que tengáis otras versiones dependiendo de la versión de Android Studio instalada):
  - MinSDK: 23: Es lo que indicamos cuando creamos el proyecto.
  - Target_SDK: 30
  - Compile_SDK: 30
  - Build Tools: 30 (o la que hay en el momento de instalar el Android Studio)

Nota: Explicaremos el significado en un apartado posterior.

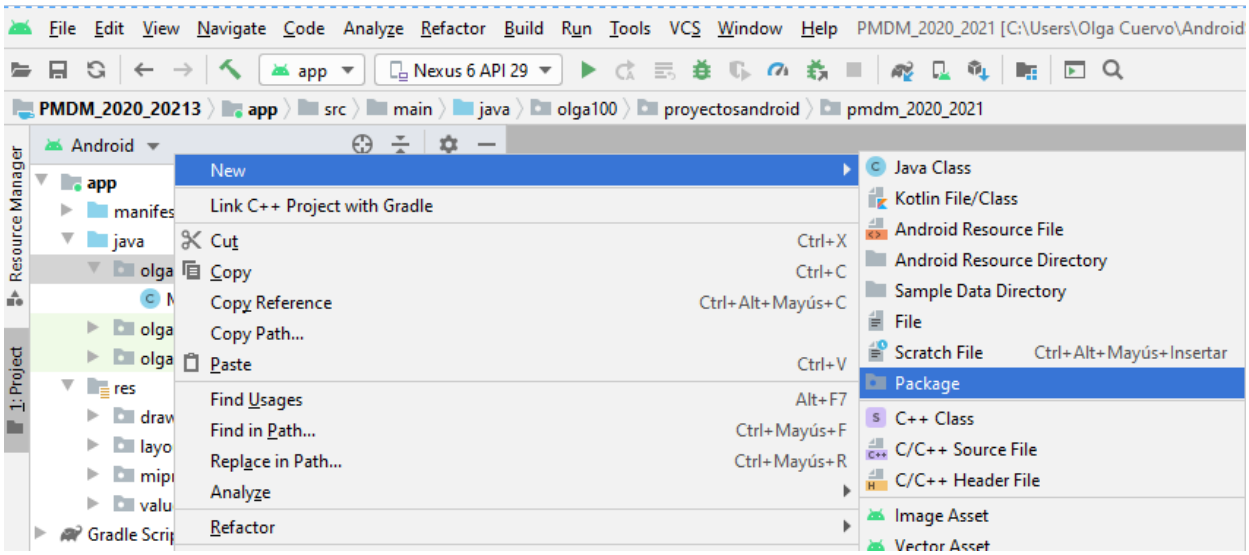


- A lo largo del curso, se crearán diferentes paquetes (vienen como subdirectorios).

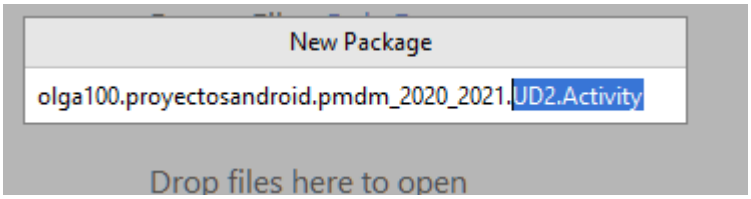
  Veamos un ejemplo de cómo crear un nuevo paquete llamado '**UD2.Activity**' (es un ejemplo)

  Dejar al menos una actividad en el paquete inicial, porque si no gráficamente, Android Studio la hace desaparecer.

Creando un nuevo paquete



Hacer clic derecho sobre el paquete en el que queremos crear un subpaquete.



Teclear el nombre del paquete a crear.

Ahora, al hacer clic derecho en el nuevo paquete, podemos crear nuevas actividades que se incluirán en él.

En cualquier momento podemos mover actividades de un paquete a otro (arrastrando con el ratón) pero al hacerlo aparecerá una ventana en la que tendremos que elegir la opción **Refactorizar**.

- Aunque no se indica en las distintas actividades creadas, cada vez que se cree una actividad tipo **launcher** modificaremos el archivo **AndroidManifiest.xml** para darle una etiqueta a cada una de ellas. Los datos de la etiqueta serán los mismos que el nombre de la actividad. Hacemos esto para poder identificar dentro del emulador / dispositivo si vamos a la pantalla donde ves todas las aplicaciones instaladas:

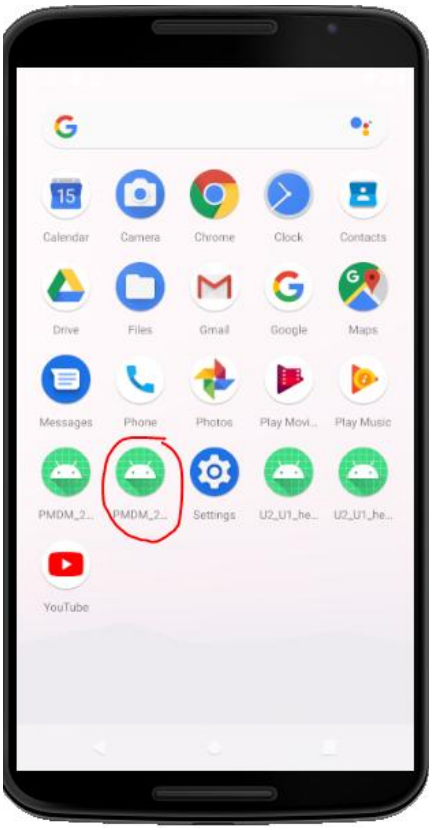
```
<activity android:name=".UD2.Activity.UD02_01_ConstraintLayout">
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />

        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>
```

En este caso, la actividad está en un paquete de nombre **UD2.Activity**, pero el nombre de la actividad es: **UD02_01_ConstraintLayout.**

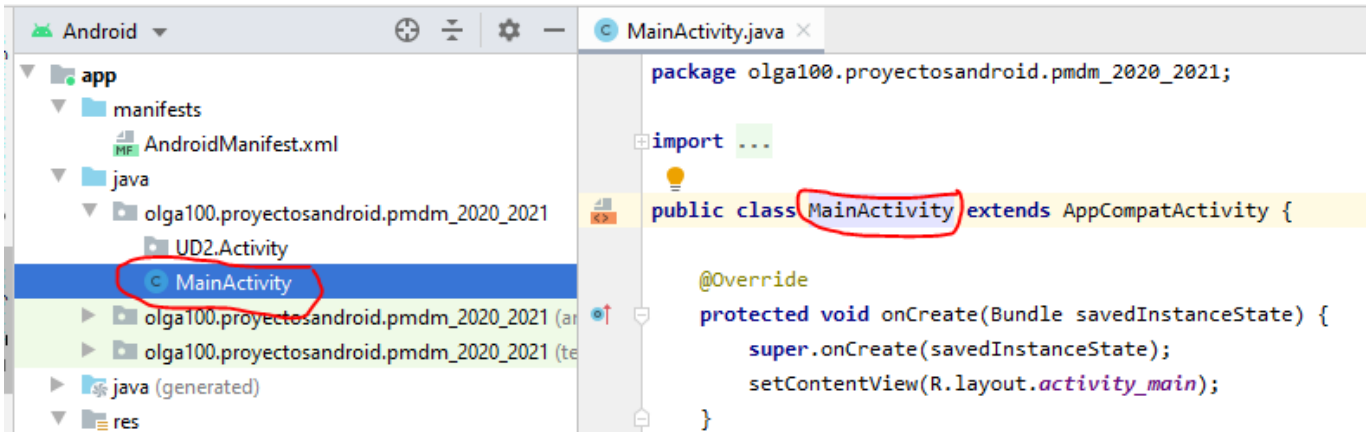
Si ejecutamos esta aplicación:

Arrastramos el ratón desde abajo para mostrar las aplicaciones instaladas en el emulador.

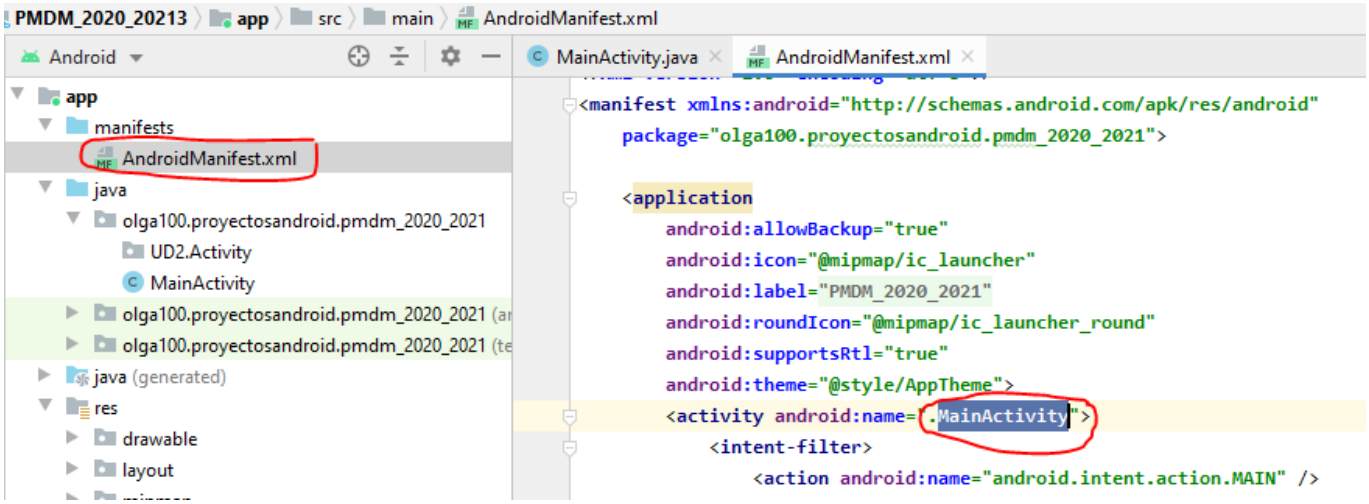


Aparece la aplicación instalada.

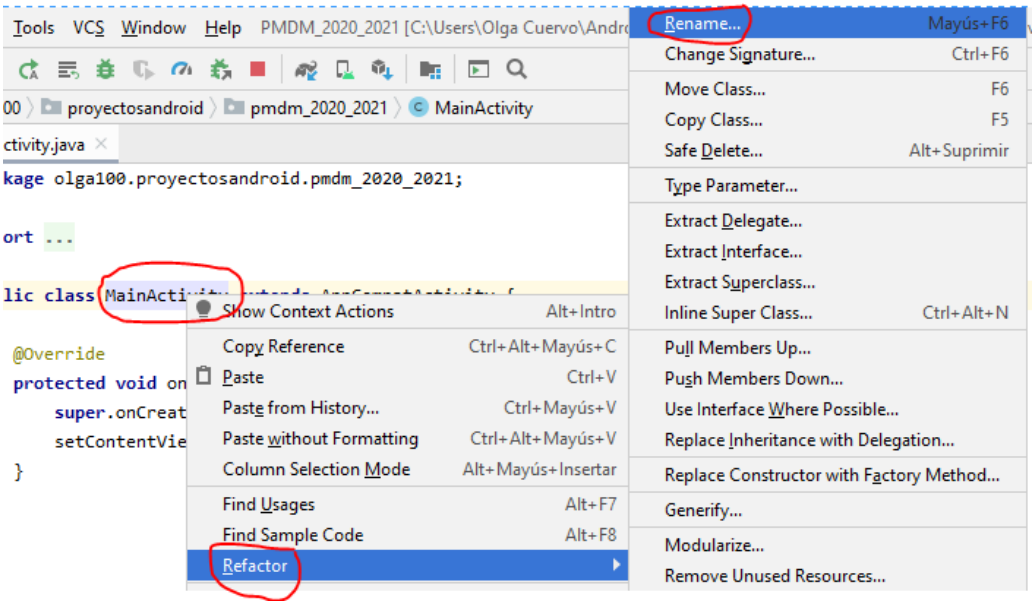
1.3. Nombre de las clases y layouts



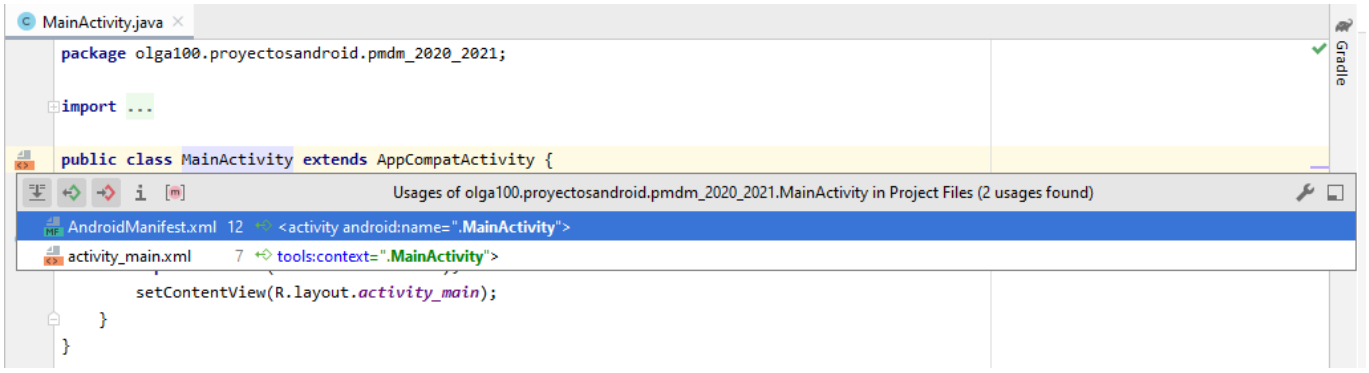
Como en Java, el nombre de la clase pública debe coincidir con el nombre físico del archivo. Como vemos, la clase se deriva de la clase *AppCompactActivity*, no de la clase Activity. AndroidStudio anterior me permitió señalar esto, pero ahora no es así. A lo largo de este curso, se verán ejemplos de código en los que las actividades se derivan de la clase Activity. Ignoraremos y haremos que todas las actividades se deriven de AppCompactActivity (recordar cambiarlo).



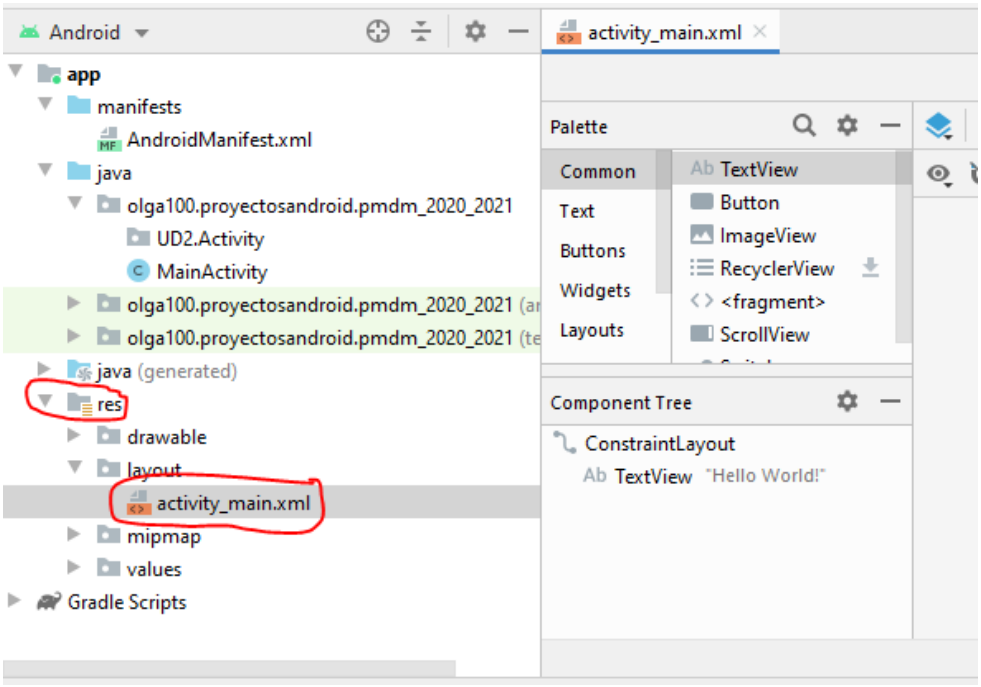
Si queremos cambiar el nombre no basta con cambiar el nombre de la clase como se usa en otros sitios, como *AndroidManifest.xml.*



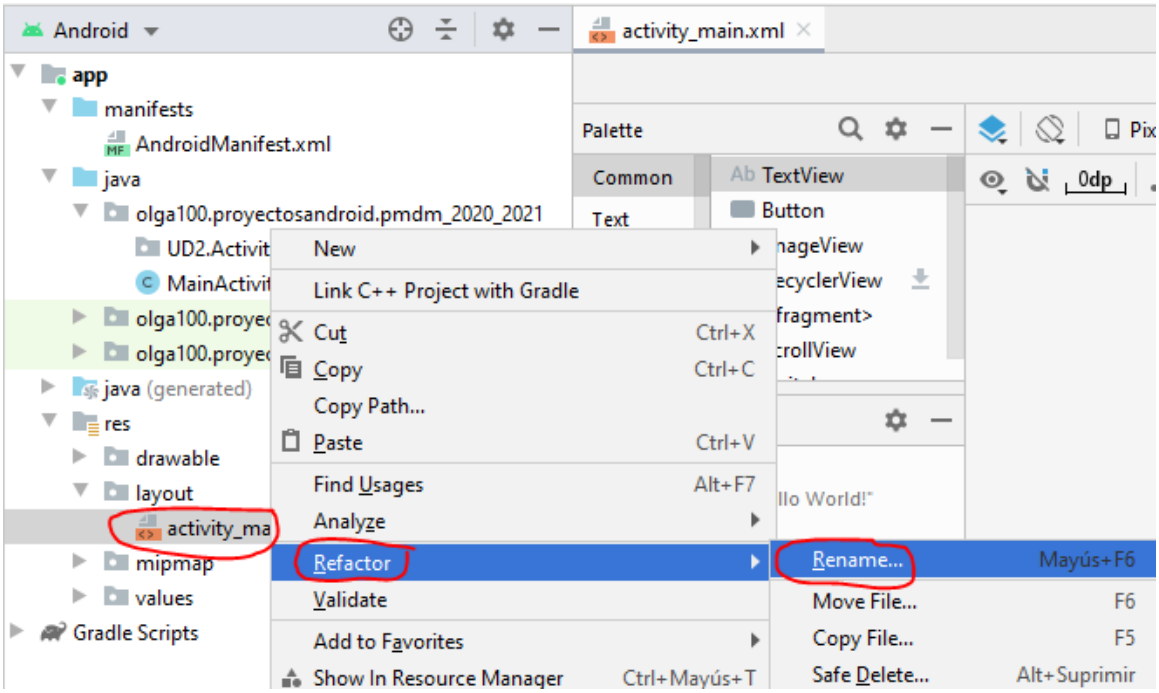
Haga clic derecho sobre lo que queremos cambiar y elija la opción **Refactor => Renombrar**.



El Layout es donde se encuentran los componentes gráficos de la activity siguiendo una distribución. Podemos acceder presionando la tecla Control y haciendo clic con el mouse sobre el nombre del diseño...



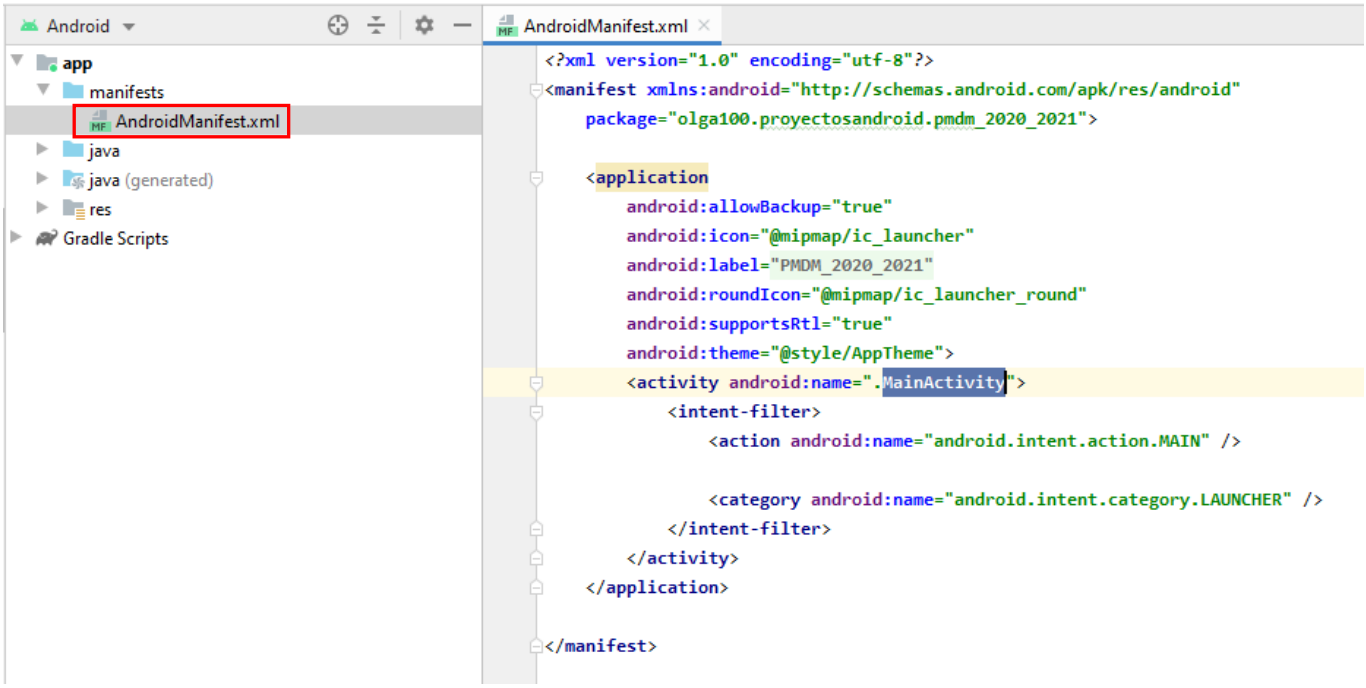
O podemos ir a la carpeta de **res => layout** y hacer doble clic en ella.



Para cambiar el nombre haremos lo mismo que en la clase.

1.4. Archivo AndroidManifest.xml

- Contiene la definición de las características más importantes de la aplicación: nombre, versión, icono, actividades, servicios, instrumentación, permisos de acceso a recursos, etc.



- En la imagen podemos ver entre otros datos:
  - Nombre del ícono a cargar (se refiere a un ícono ubicado en **/app/res/mipmaps/**
- Nombre de la aplicación que se define en una constante en el archivo **/res/values/strings.xml**. Es el nombre que aparece asociado al icono de la aplicación.
- Cómo se verá la aplicación (es similar a un archivo CSS HTML). Se encuentra en el archivo **/res/values/styles.xml**.
- Las actividades que componen el proyecto. Inicialmente solo tenemos una.

1.5. Identificar o minSDK, targetSDK e buildSDK

- Anteriormente en Eclipse, dentro del archivo **AndroidManifest.xml** también estaban las líneas que indicaban el targetSDK y el minSDK.
- Como se mencionó anteriormente, la API determinará la versión del sistema operativo en el que desarrollaremos la aplicación.

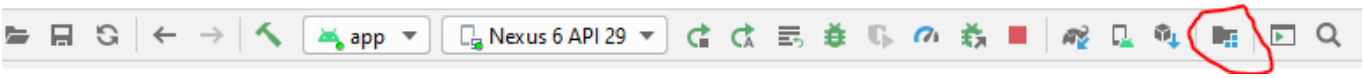
  Cada nueva versión incorpora nuevos componentes gráficos y nuevas funcionalidades, pero tenemos que mantener un equilibrio entre la versión a elegir y el mercado que queremos monopolizar.

- La API también determinará qué aplicaciones se pueden instalar en el dispositivo móvil, ya que una aplicación desarrollada para una versión superior no se puede instalar en un sistema operativo Android con una versión inferior.

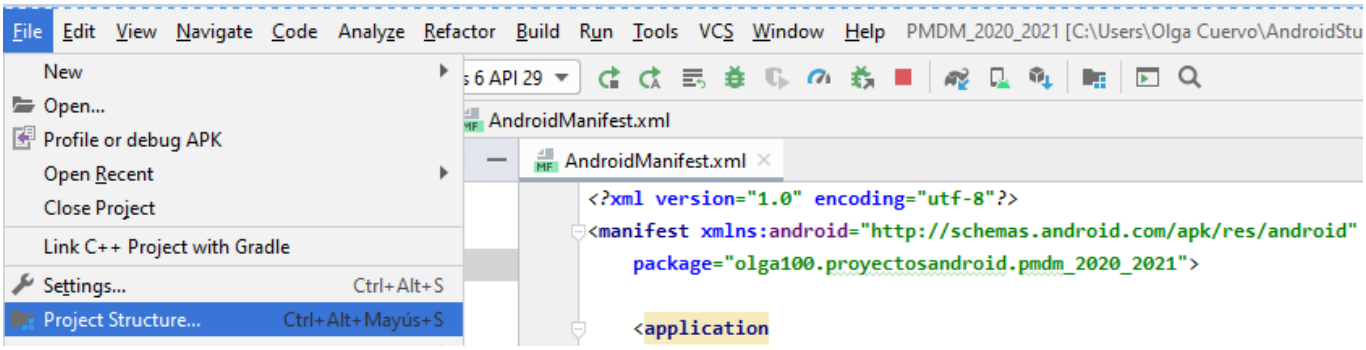
- Al determinar la API en un proyecto de Android debemos elegir lo siguiente:
  - **MIN SDK**: Será la API mínima (versión del SO Android) en la que se puede ejecutar nuestra aplicación. No se puede instalar en una versión anterior.
  - **TARGET SDK**: Indica en qué versión del sistema operativo Android se prueba nuestra aplicación y sabemos que funciona correctamente. Suele ser el más alto. Podría funcionar en una versión posterior o inferior (siempre que sea más grande que minSdk). El uso de un objetivo u otro puede afectar la apariencia de nuestra aplicación en tiempo de ejecución (por ejemplo, puede cambiar el tema o la apariencia predeterminados, según el objetivo)
  - **BUILD SDK o Compile SDK**: Indica la versión del SDK que usamos para compilar nuestra aplicación. Esto limitará los recursos que podemos usar en el desarrollo de la aplicación, porque si por ejemplo, tenemos una nueva característica en una API por ejemplo, en la API 22, y hemos marcado un BUILD_SDK de 19, no podemos usarla. El build_sdk siempre tendrá que ser mayor o igual que min_sdk.

Generalmente se sigue la regla: minSdkVersion (mínimo posible) <= targetSdkVersion == compileSdkVersion (SDK más alto posible)

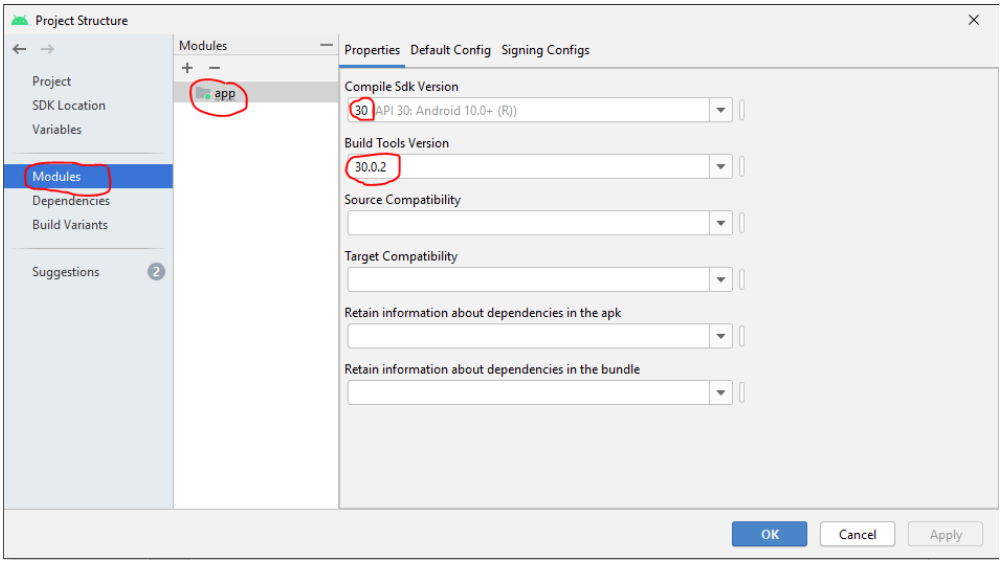
Especificando la API de un proyecto Android



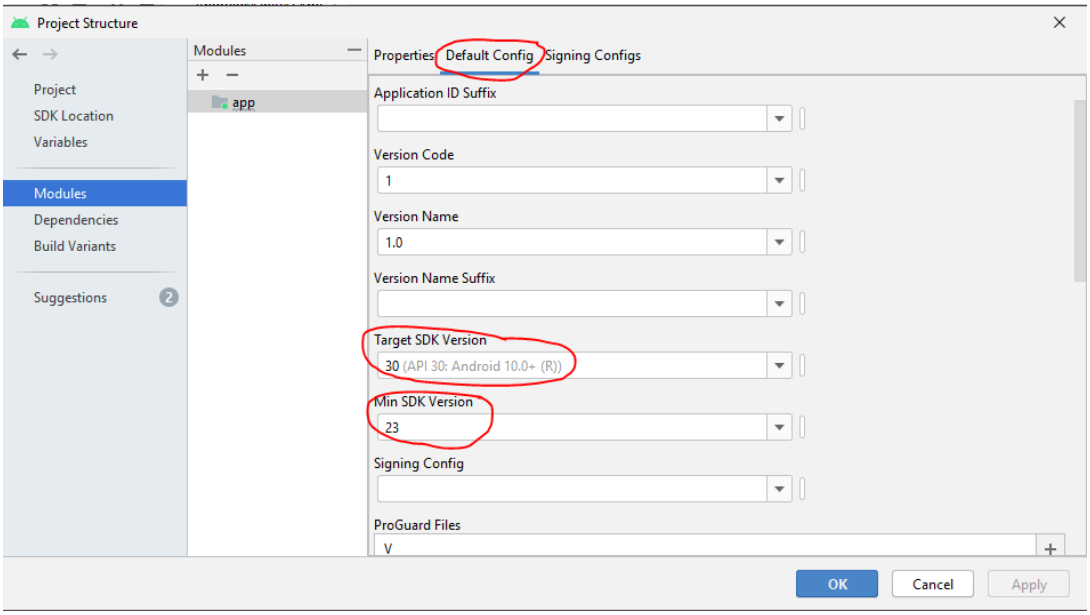
En la barra de tareas escogemos la opción **Project Structure**



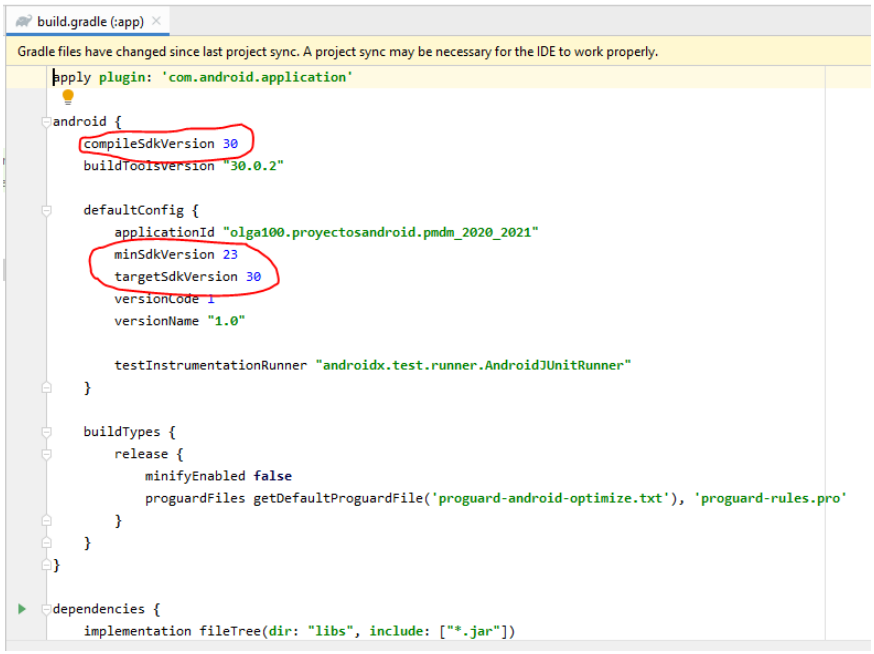
También podemos ir desde el menú principal o con la combinación de teclas indicada.



Debemos elegir el módulo **App** y en la pestaña **Properties** tenemos el BUILD SDK. Debemos asegurarnos de que también se seleccione un **build-tools**, generalmente la última versión.



En la pestaña **Default Config** tenemos a MIN SDK y TARGET SDK.



Estos cambios son reflejados en el archivo **build.gradle** a nivel de módulo.

Nota: Estos datos se pueden definir en el archivo AndroidManifiest.xml, pero si están en dos sitios, tendrá preferencia (sobrescribe) el que se escribe en el archivo **build.gradle**.