

INTRODUCCIÓN A ANDROID



Programación multimedia
y dispositivos móviles
2º DAM

ÍNDICE DE CONTENIDOS

1. Fundamentos de una aplicación móvil
2. Componentes de una aplicación
3. Instalación e inicio del IDE
4. Emulador / Dispositivo físico para probar aplicaciones
5. Elementos de un proyecto
6. Android Manifest
 - 6.1. Permisos
7. Recursos de una aplicación
 - 7.1. Cómo acceder a los recursos
 - 7.2. Uso de recursos
8. Clase R
9. Generar ejecutable

1. FUNDAMENTOS DE UNA APLICACIÓN MÓVIL

Cada aplicación es considerada propiedad de un usuario distinto. El sistema asigna un identificador de usuario diferente a cada aplicación.

Cada aplicación tiene su propio sistema de seguridad:

- Identificador de usuario diferente
- Permisos propios para cada usuario con máquinas virtuales propias
- Proceso de Linux propio
- **Principio de menor privilegio**

Las aplicaciones deben pedir permisos para acceder a datos, recursos...

1. FUNDAMENTOS DE UNA APLICACIÓN MÓVIL

El SDK (Software Development Kit) de Android compila el código, datos y recursos, incluyéndolo todo en un fichero APK.

APK: archivo ejecutable para Android.



2. COMPONENTES DE UNA APLICACIÓN

Los componentes de una aplicación son bloques de código mediante los cuales el sistema puede relacionarse con esta. Cada componente tiene una identidad propia y cumple un papel específico.

- **Activity:** componente principal de una aplicación puesto que se encarga de gestionar gran parte de las interacciones con el usuario.
- **Service:** son aplicaciones que corren en segundo plano para hacer operaciones de larga duración. No tienen interfaz de usuario.
- **Content Provider:** se ocupa de gestionar un conjunto de datos de una aplicación para compartir.
- **Broadcast Receiver:** utilidad que permite responder a anuncios de difusión del sistema.

3. INSTALACIÓN E INICIO DEL IDE

ANDROID STUDIO

- Google 2013
- IDE multiplataforma
- Descarga:

<https://developer.android.com/studio/>

3. INSTALACIÓN E INICIO DEL IDE

System requirements

Windows

- Microsoft® Windows® 7/8/10 (64-bit)
- 4 GB RAM minimum, 8 GB RAM recommended
- 2 GB of available disk space minimum,
4 GB Recommended (500 MB for IDE + 1.5 GB for Android SDK and emulator system image)
- 1280 x 800 minimum screen resolution

Mac

- Mac® OS X® 10.10 (Yosemite) or higher, up to 10.14 (macOS Mojave)
- 4 GB RAM minimum, 8 GB RAM recommended
- 2 GB of available disk space minimum,
4 GB Recommended (500 MB for IDE + 1.5 GB for Android SDK and emulator system image)
- 1280 x 800 minimum screen resolution

Linux

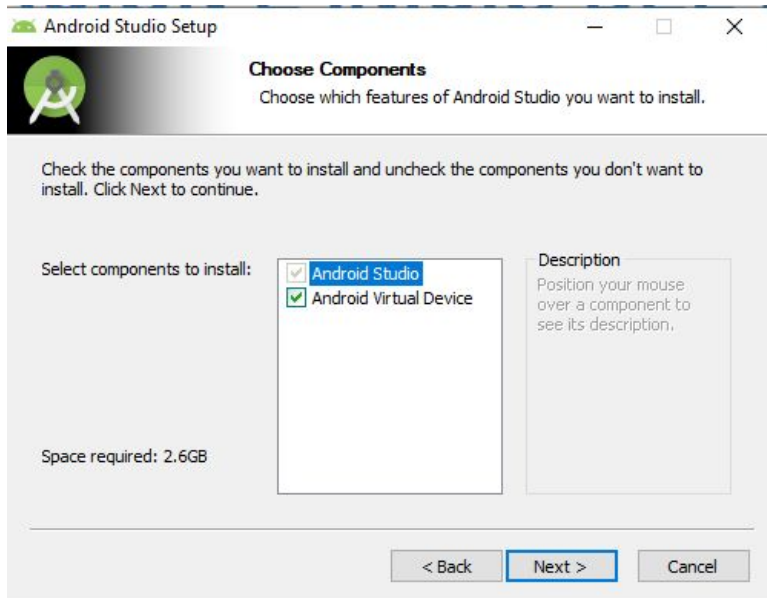
- GNOME or KDE desktop
Tested on gLinux based on Debian.
- 64-bit distribution capable of running 32-bit applications
- GNU C Library (glibc) 2.19 or later
- 4 GB RAM minimum, 8 GB RAM recommended
- 2 GB of available disk space minimum,
4 GB Recommended (500 MB for IDE + 1.5 GB for Android SDK and emulator system image)
- 1280 x 800 minimum screen resolution

Chrome OS

- 8 GB RAM or more recommended
- 4 GB of available disk space minimum
- 1280 x 800 minimum screen resolution
- Intel i5 or higher (U series or higher) recommended

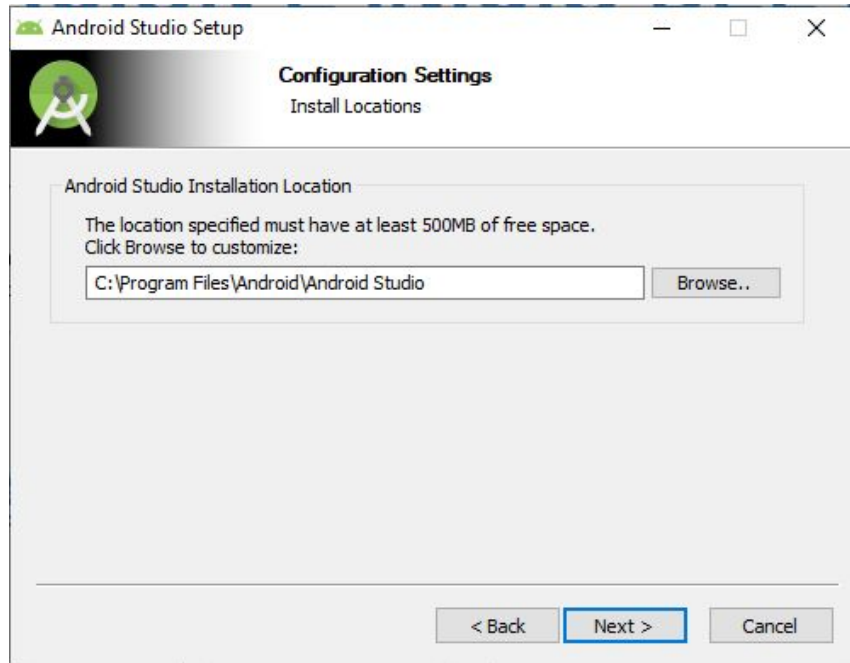
For more information on recommended devices as well as Android emulator support, visit chromeos.dev

3. INSTALACIÓN E INICIO DEL IDE



Android Virtual Device(AVD):
herramienta necesaria para probar las aplicaciones desde un emulador.

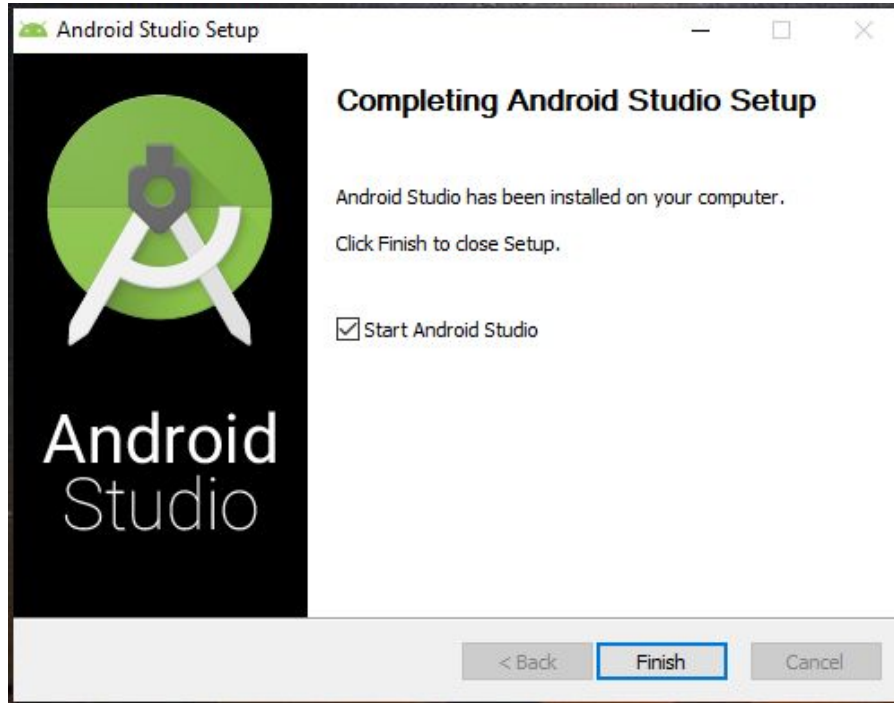
3. INSTALACIÓN E INICIO DEL IDE



Ruta de instalación:

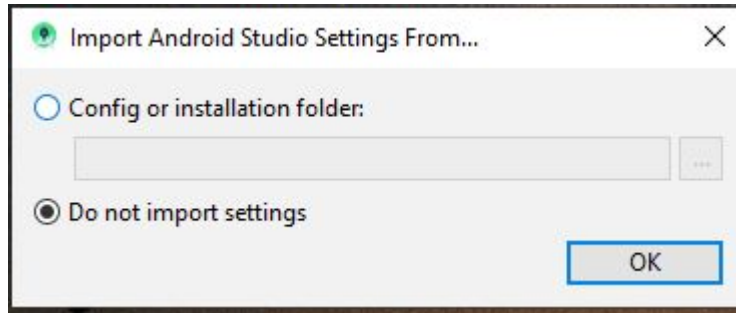
- No puede contener caracteres especiales.
- Fácil acceso
- Si es posible, se recomienda instalar en disco SSD. Los proyectos podrán estar en otro disco.

3. INSTALACIÓN E INICIO DEL IDE



3. INSTALACIÓN E INICIO DEL IDE

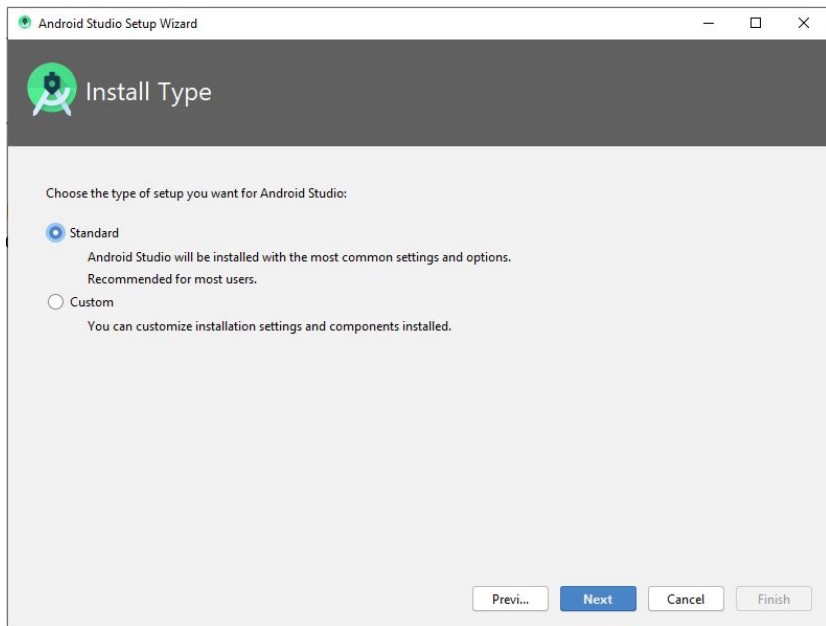
Una vez instalado, la primera vez que abrimos Android Studio nos indicará que podemos importar la configuración. En nuestro caso, al ser la primera vez, marcaremos la opción por defecto: No importar



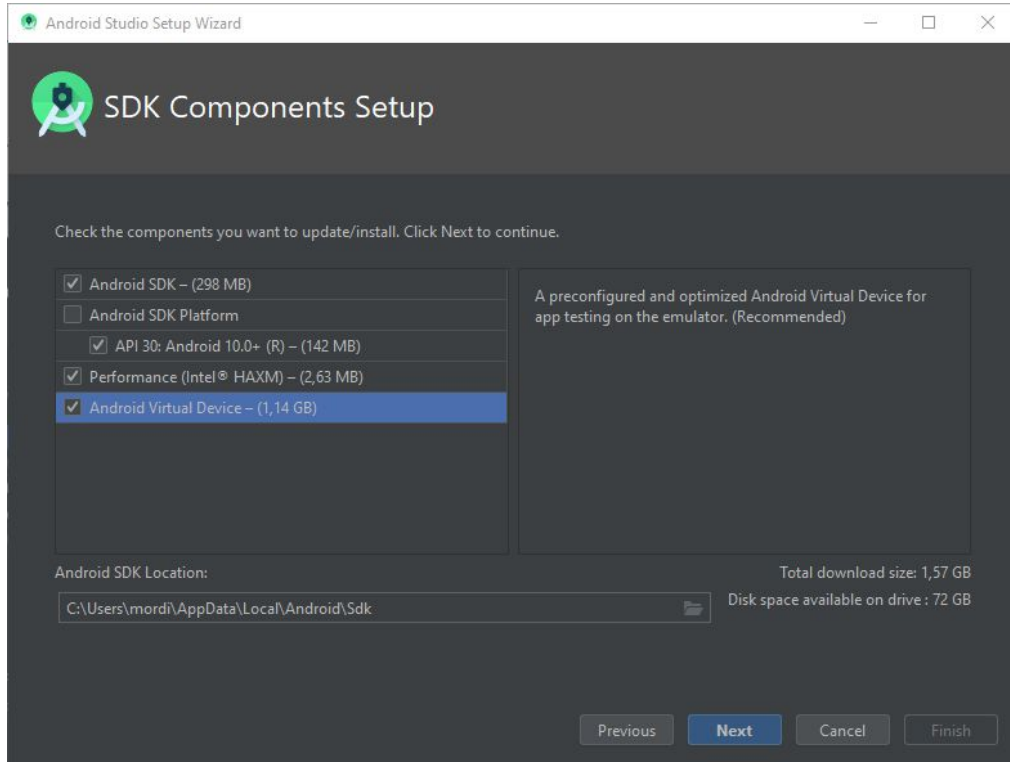
A la hora del reporte de fallos, elegid la opción que queráis.

3. INSTALACIÓN E INICIO DEL IDE

Podéis configurar la instalación si optáis por la opción Custom, como por ejemplo la ruta de instalación del JDK (1.5 GB) y una serie de características a instalar. **(Recomendado)**



3. INSTALACIÓN E INICIO DEL IDE



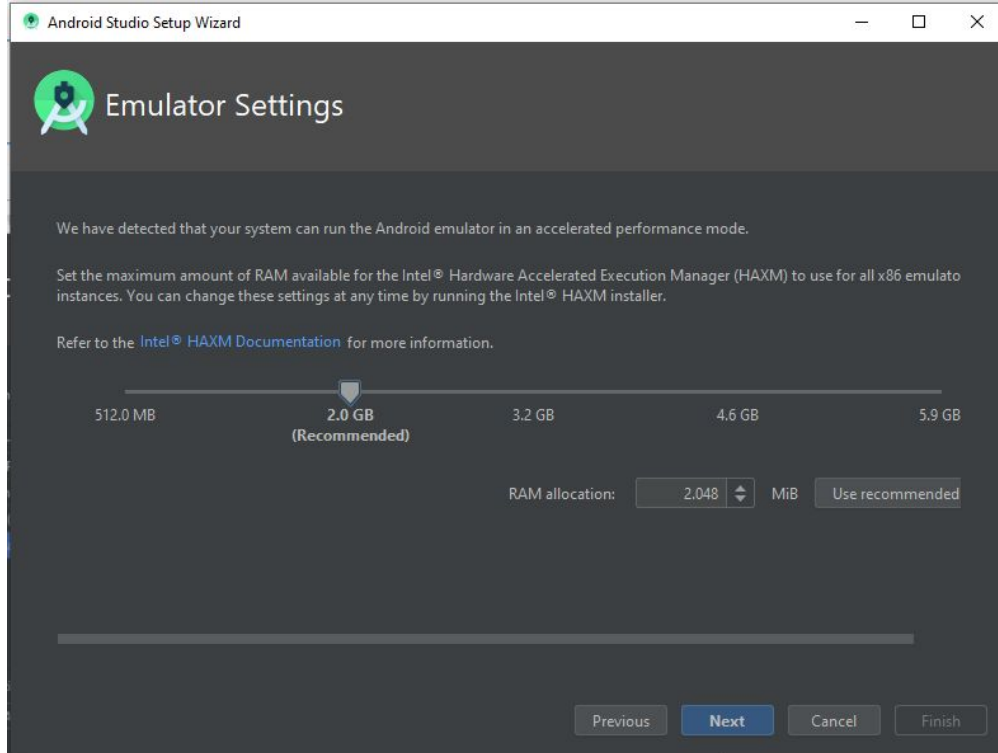
Android SDK: Es obligatoria su instalación para compilar cualquier aplicación.

Android SDK Platform: Nos permite instalar la última versión API disponible. Se puede instalar más tarde.

Intel HAXM: si el ordenador en el que se está trabajando lo permite es muy recomendable. Es un sistema de virtualización que ayudará a mejorar el rendimiento del emulador en los procesadores Intel.

Android Virtual Device: Imprescindible si queremos probar las aplicaciones en un emulador.

3. INSTALACIÓN E INICIO DEL IDE

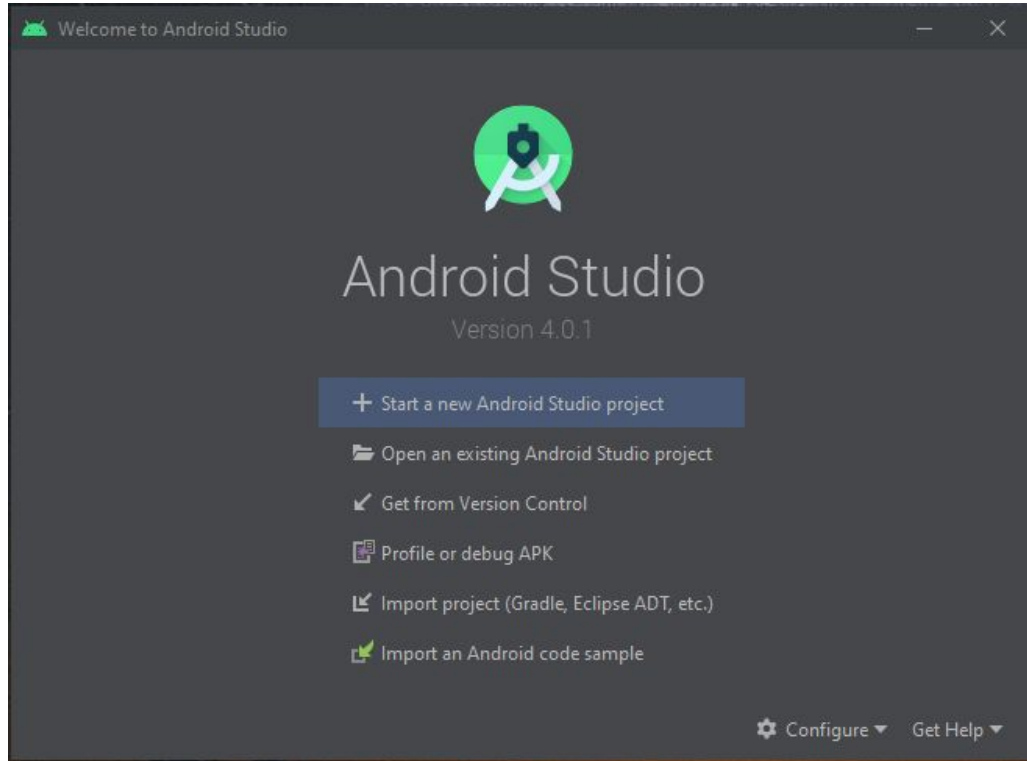


Indicar la máxima RAM que pueda utilizar el emulador. Cada uno tendrá unos datos según la RAM disponible en cada ordenador.

En este caso, ordenador de 8GB RAM, recomiendan 2GB, y es suficiente para que funcione con fluidez.

Por último, encontraréis un repaso de lo que se va a instalar. Tardará un rato...

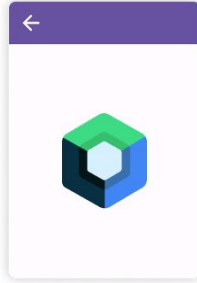
3. INSTALACIÓN E INICIO DEL IDE



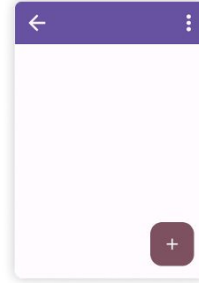
3. INSTALACIÓN E INICIO DEL IDE



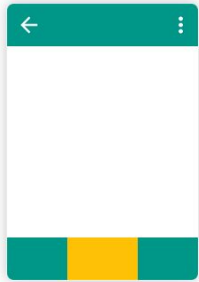
No Activity



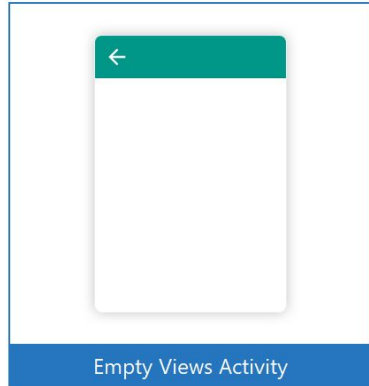
Empty Activity



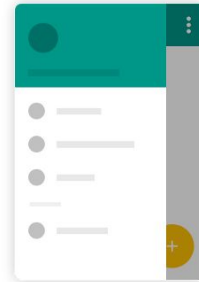
Basic Views Activity



Bottom Navigation Views Activity



Empty Views Activity

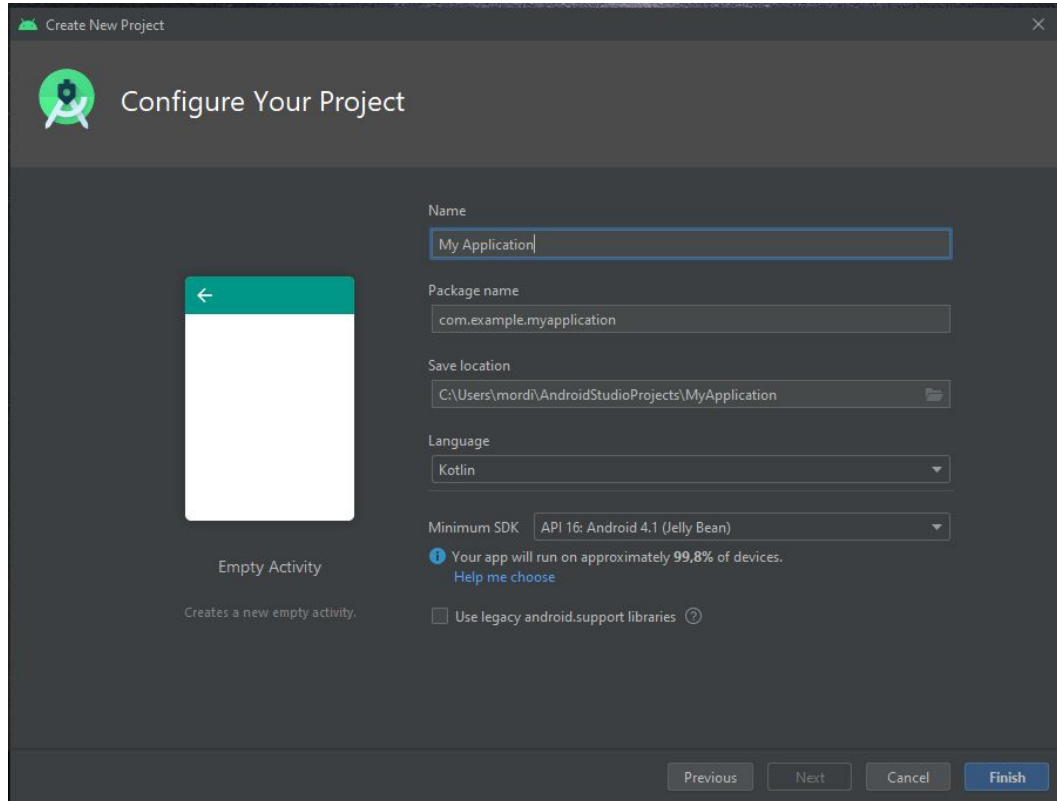


Navigation Drawer Views Activity

Elegir el tipo del dispositivo para la aplicación, y la plantilla de la actividad.

Para comenzar a utilizar Android se recomienda Empty Views Activity, que es el proyecto más simple.

3. INSTALACIÓN E INICIO DEL IDE



Nombre de la aplicación: No usar caracteres especiales. Después se podrá cambiar el nombre de la aplicación.

Nombre del paquete: Se puede dejar el que viene por defecto.

Ruta para guardar el proyecto: Puede ser diferente a la ruta que está instalado Android Studio.

Lenguaje: Java o Kotlin

SDK mínimo: API mínima para la que funcionará la aplicación.

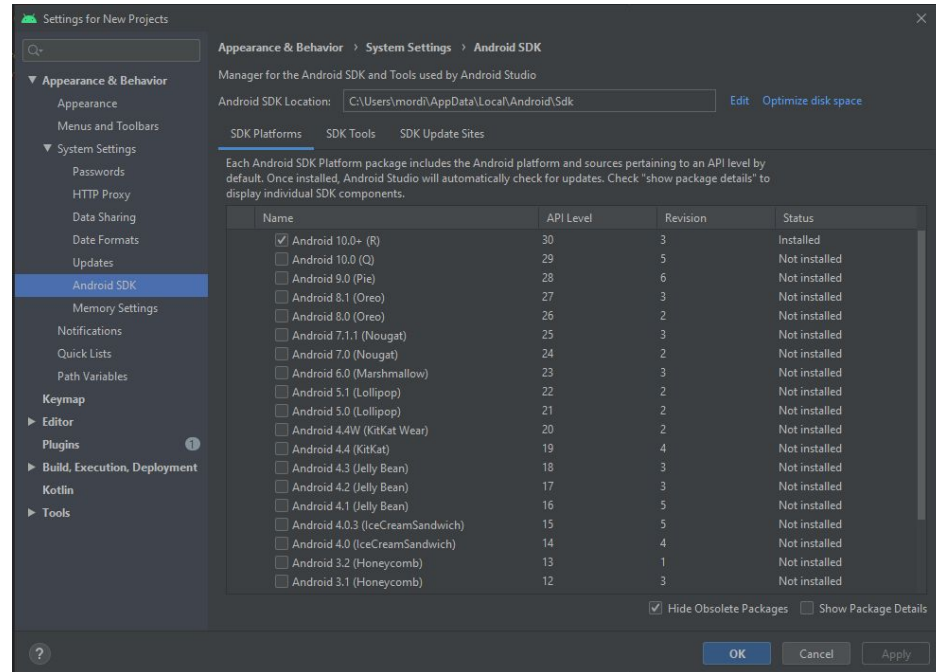
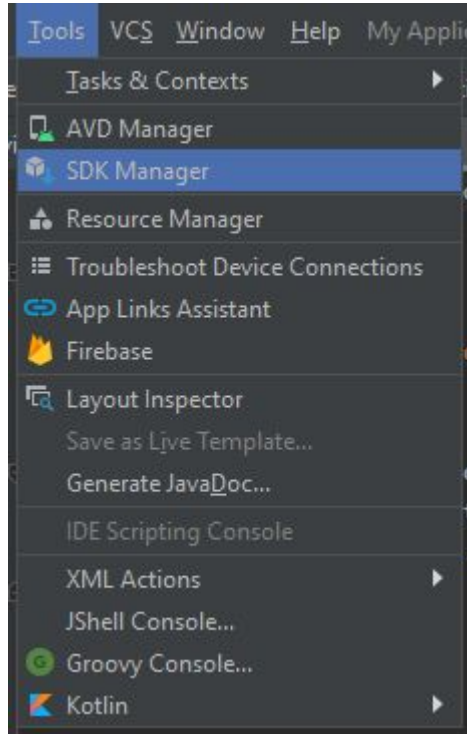
Use legacy android.support libraries: Si marcamos esta opción, no podemos utilizar las últimas bibliotecas de android, por lo que se recomienda no marcar la opción.

3. INSTALACIÓN E INICIO DEL IDE

Configurar SDK

SDK Platform:

Instalar las API necesarias para nuestro trabajo. Se recomienda tener instaladas todas las API permitidas en las aplicaciones que se van a realizar. Para ello, sólo hay que marcarlas y OK. Tardará un rato...



3. INSTALACIÓN E INICIO DEL IDE

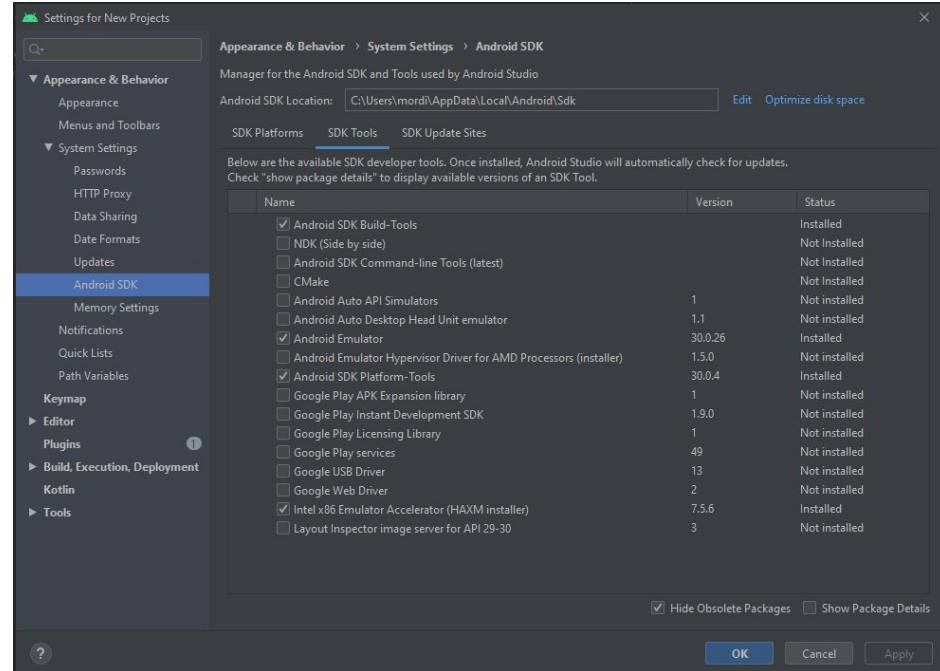
Configurar SDK

SDK Tools: Herramientas que necesitamos instalar para nuestro desarrollo de aplicaciones.

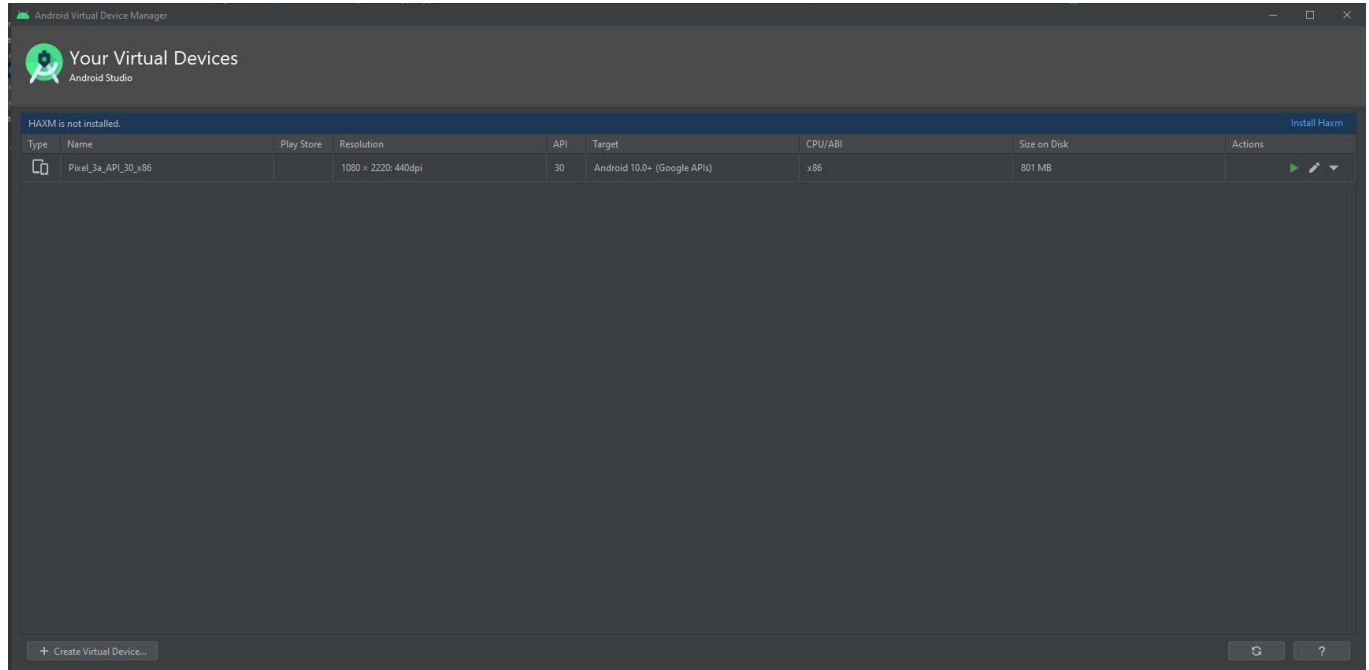
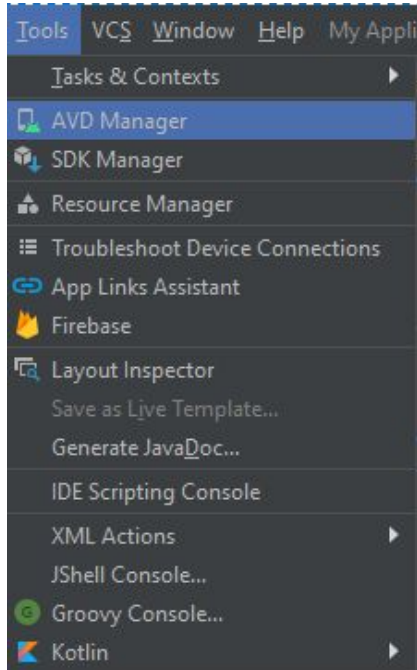
Las esenciales deberán estar marcadas por la configuración que hicimos al realizar la instalación del IDE.

Además se recomienda instalar Google Play Services y Google USB Driver.

Esto no debería tardar mucho...

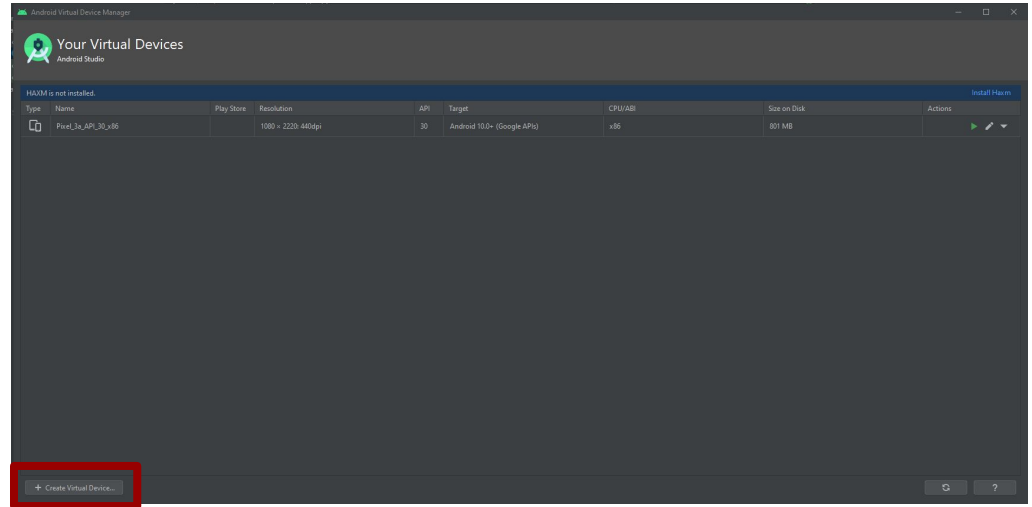


4. EMULADOR / DISPOSITIVO FÍSICO PARA PROBAR APLICACIONES



4. EMULADOR / DISPOSITIVO FÍSICO PARA PROBAR APLICACIONES

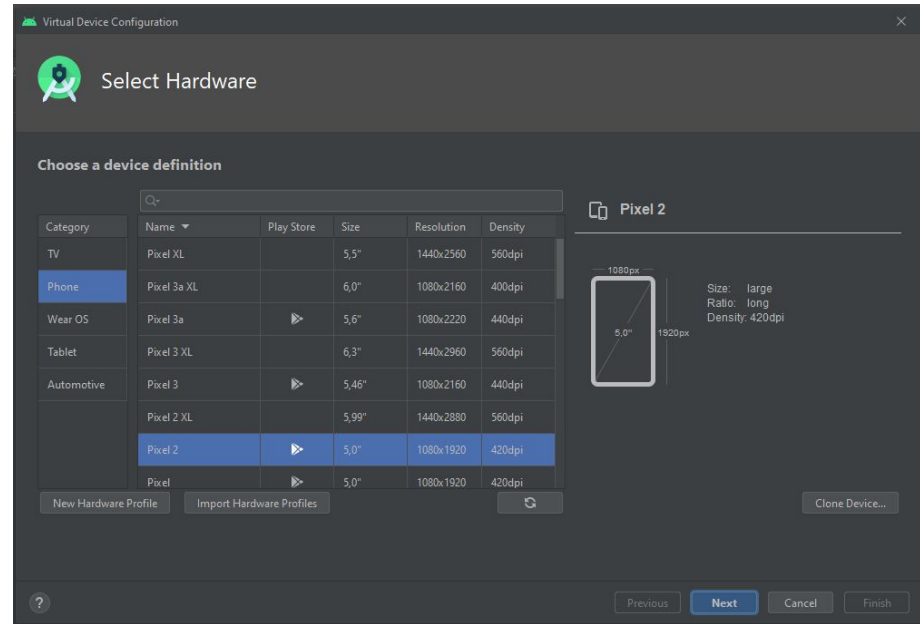
- Instalar HAXM
- Emuladores que están disponibles
- Crear un nuevo emulador



4. EMULADOR / DISPOSITIVO FÍSICO PARA PROBAR APLICACIONES

Categoría: tipo de dispositivo que queremos emular.

Modelo de dispositivo que queremos con sus características.



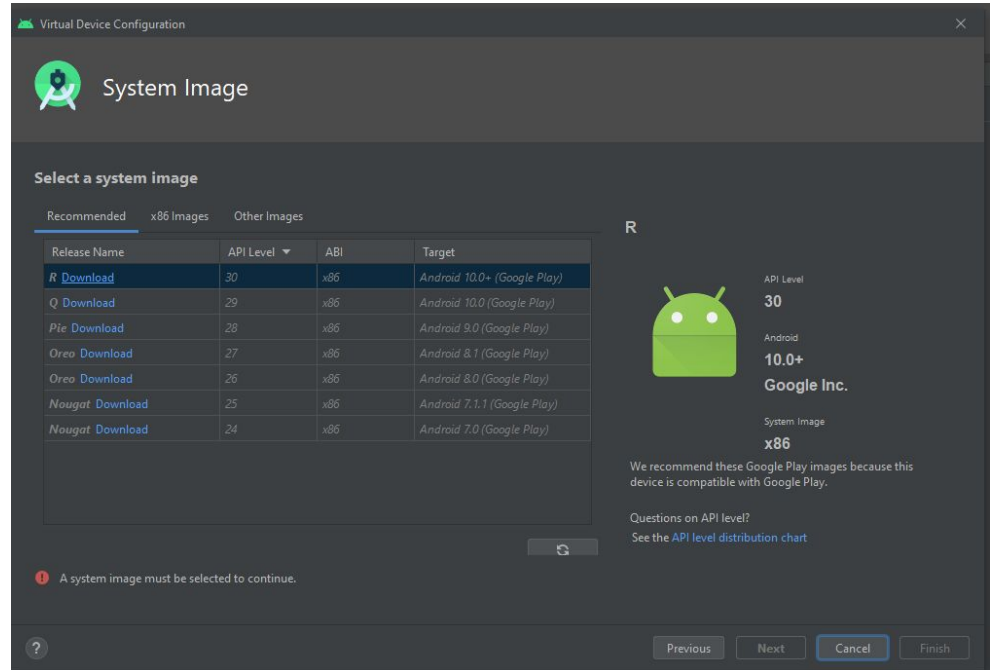
4. EMULADOR / DISPOSITIVO FÍSICO PARA PROBAR APLICACIONES

Seleccionar la versión Android que quieres que tenga el emulador.






Al ser el primer emulador, hay que descargarla.

Puedes dejar todas las imágenes descargadas.

Esto va a tardar...



4. EMULADOR / DISPOSITIVO FÍSICO PARA PROBAR APLICACIONES

Type	Name	Play Store	Resolution	API	Target	CPU/ABI	Size on Disk	Actions
	Pixel 3a API 30		1080 x 2220: 440dpi	30	Android 10.0+ (Google Play)	x86	8,6 GB	  

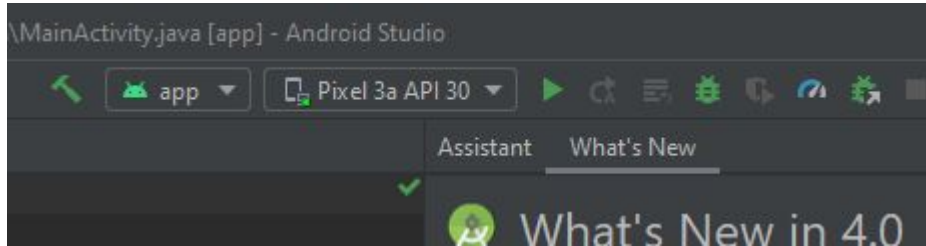
Iniciamos el emulador.

Se recomienda no abrir y cerrar el emulador por cada prueba que hagamos, sino dejarlo abierto durante todo nuestro trabajo.

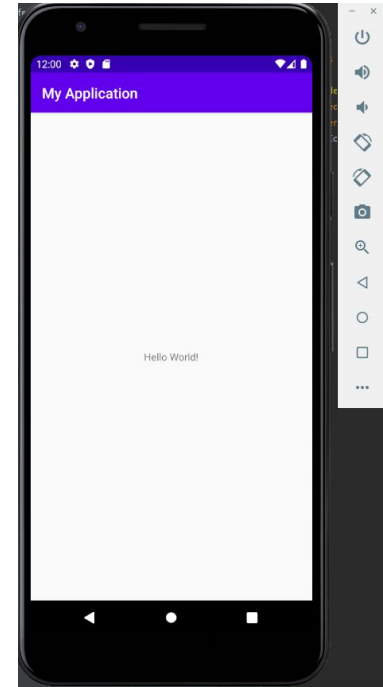
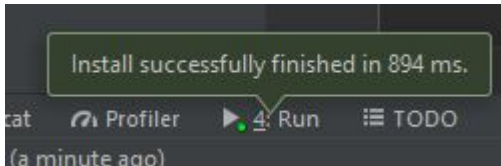


4. EMULADOR / DISPOSITIVO FÍSICO PARA PROBAR APLICACIONES

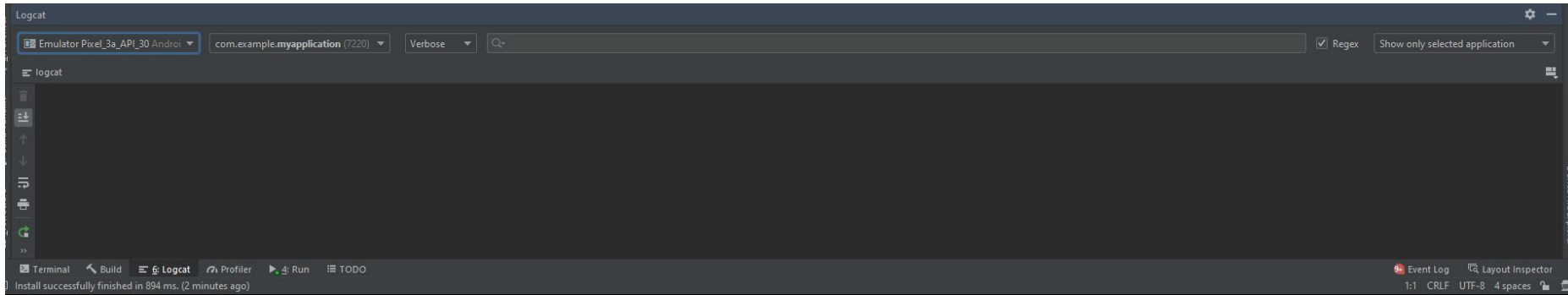
Una vez comprobado que el emulador está listo, vamos a probar nuestro proyecto.



En la parte superior de la derecha encontramos el botón Play para ejecutar la aplicación, y al lado el Debug.

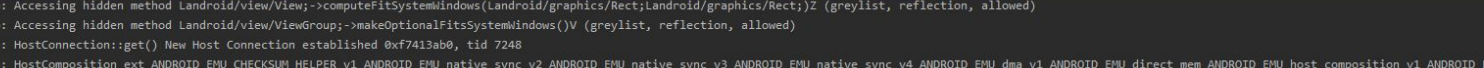


4. EMULADOR / DISPOSITIVO FÍSICO PARA PROBAR APLICACIONES



Logcat: Mensajes Log

4. EMULADOR / DISPOSITIVO FÍSICO PARA PROBAR APLICACIONES



The screenshot shows the Android Studio interface. The top toolbar includes icons for Run, Build, Logcat, Profiler, and TODO. The Logcat window is open, displaying a list of log messages. The messages include system-level logs such as 'W/e.myapplication: Accessing hidden method', 'D/HostConnection: HostConnection: get() New Host Connection established', 'D/OpenGLRenderer: Failed to choose config with EGL_SWAP_BEHAVIOR_PRESERVED, retrying without...', 'D/EGL_emulation: eglCreateContext: 0xf7225460: maj 2 min 0 rcv 2', 'D/EGL_emulation: eglMakeCurrent: 0xf7225460: ver 2 0 (tinfo 0xf7572870) (first time)', 'I/Gralloc4: mapper 4.x is not supported', and 'D/HostConnection: createUnique: call'. The bottom status bar shows 'Event Log' and 'Layout Inspector' tabs.

En la pestaña Run veremos el detalle del error en caso de fallo.

4. EMULADOR / DISPOSITIVO FÍSICO PARA PROBAR APLICACIONES



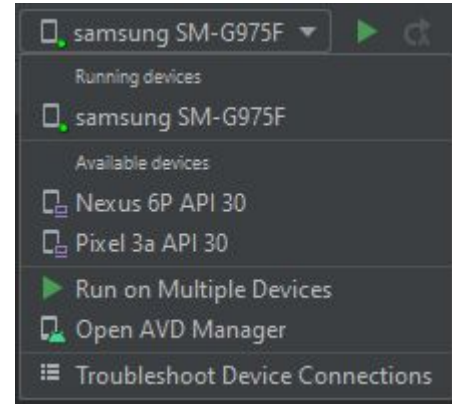
En caso de tener varios emuladores, se seleccionará uno antes de dar al Play.

Para utilizar un dispositivo físico para la depuración:

Dispositivo: En el dispositivo, abre la app de **Configuración**, selecciona **Opciones para desarrolladores** y luego habilita **Depuración por USB**.

Conecta el dispositivo por USB al ordenador, y acepta el mensaje en el dispositivo.

<https://developer.android.com/studio/debug/dev-options?hl=es>



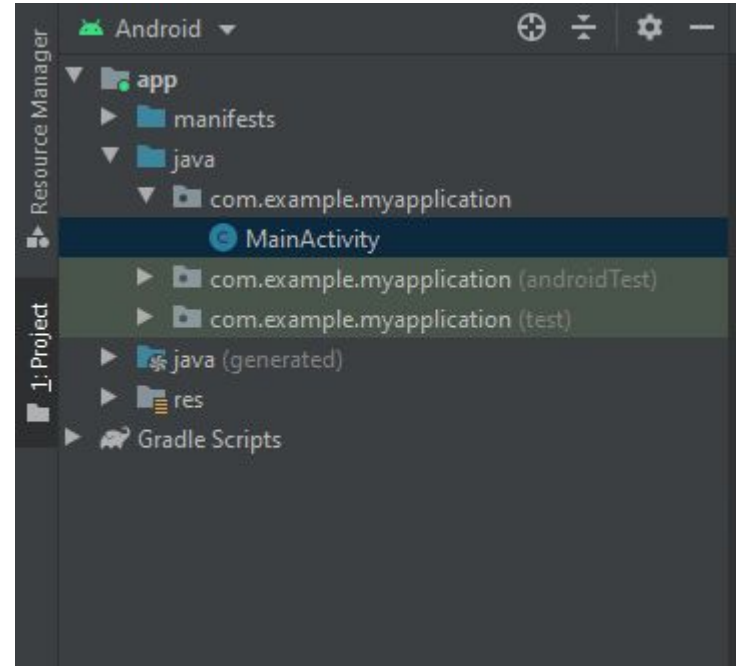
5. ELEMENTOS DE UN PROYECTO

En la parte izquierda de nuestro IDE encontraremos la estructura de un proyecto Android.

Dentro de la carpeta **app** encontramos 4 carpetas:

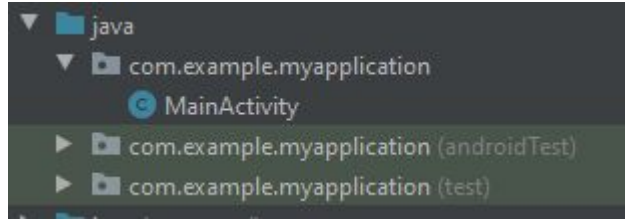
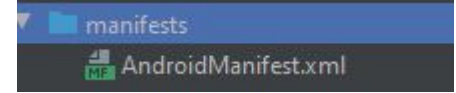
- manifests
- java
- java (generated)
- res

Y además, aparte de la carpeta app encontramos una carpeta llamada **Gradle Scripts**



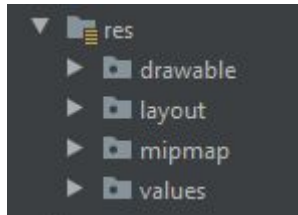
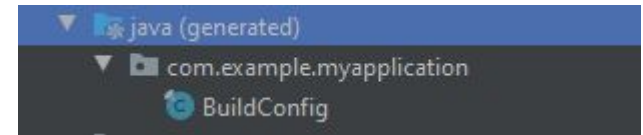
5. ELEMENTOS DE UN PROYECTO

Carpeta manifest: Contiene el archivo AndroidManifest.xml.



Carpeta java: Contiene los subpaquetes y clase Java que se hayan creado. Durante el desarrollo se trabajará en el paquete que no indica que se usa para pruebas.

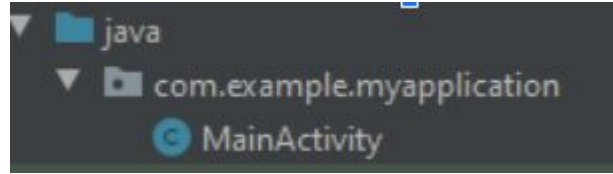
Carpeta java (generated): Contiene las clases a partir de las cuales se construirá el proyecto, incluidas las bibliotecas y la clase R.



Carpeta res: Carpeta de recursos.

5. ELEMENTOS DE UN PROYECTO

- Carpeta java

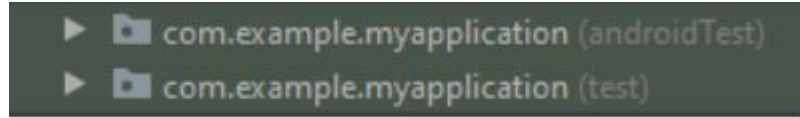


En esta carpeta trabajaremos con el modelo MVC (Modelo - Vista - Controlador). Esto quiere decir que tenemos 3 paquetes, y las distintas clases estarán en estos paquetes según su funcionalidad.

- modelo: Clases de definición de datos.
- vista: Actividades.
- controlador: Adaptadores, servicios, hilos...

5. ELEMENTOS DE UN PROYECTO

- Carpeta java



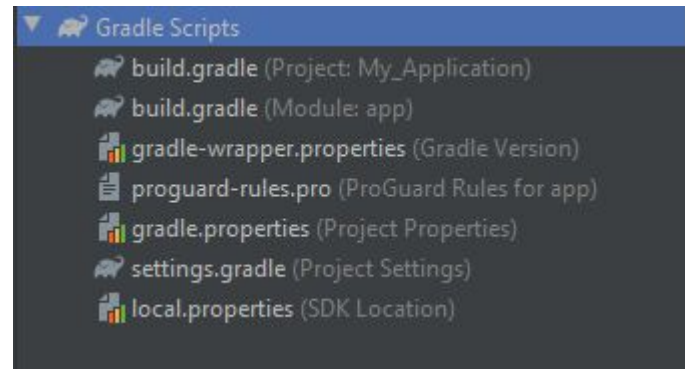
Carpeta para pruebas con JUnit.

5. ELEMENTOS DE UN PROYECTO

- **Gradle Scripts**


Contiene la información necesaria para la compilación del proyecto.

El contenido de esta carpeta se modificará en alguna ocasión, por ejemplo cuando incorporemos librerías externas o de terceras partes.



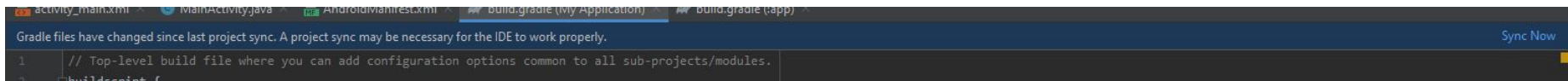
5. ELEMENTOS DE UN PROYECTO

- **Build.gradle (Project: NombreAplicación)**

 build.gradle (Project: My_Application)


```
dependencies {  
    classpath "com.android.tools.build:gradle:4.0.1"  
  
    // NOTE: Do not place your application dependencies here; they belong  
    // in the individual module build.gradle files  
}
```

Aquí se indica la versión de **Android Studio** con la que se está trabajando esta aplicación. Si cambiamos de ordenador, y tenemos otra versión, hay que cambiarlo manualmente y después, sincronizar.



5. ELEMENTOS DE UN PROYECTO

- Build.gradle (Module: app)

 build.gradle (Module: app)

```
apply plugin: 'com.android.application'

android {
    compileSdkVersion 30
    buildToolsVersion "30.0.2"

    defaultConfig {
        applicationId "com.example.myapplication"
        minSdkVersion 24
        targetSdkVersion 30
        versionCode 1
        versionName "1.0"

        testInstrumentationRunner "androidx.test.runner.AndroidJUnitRunner"
    }

    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android-optimize.txt'), 'proguard-rules.pro'
        }
    }
}
```

5. ELEMENTOS DE UN PROYECTO

- **Build.gradle (Module: app)**

compileSdkVersion: define la versión del SDK con la que compilamos la aplicación.

buildToolsVersion: define la versión de las herramientas de construcción.

minSdkVersion: define el mínimo API de Android que puede instalar la aplicación.

targetSdkVersion: indica la versión más alta con la que se ha puesto a prueba la aplicación.

5. ELEMENTOS DE UN PROYECTO

- **Build.gradle (Module: app)**

applicationId: el nombre del paquete Java de la aplicación.

versionCode: versión de la aplicación. Se incrementa cuando hay nueva versión.

versionName: versión de la aplicación. Actualización menores (1.1) y mayores (2.0)

5. ELEMENTOS DE UN PROYECTO

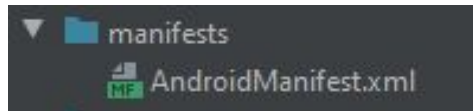
- **Build.gradle (Module: app)**

Librerías que se incluyen en el proyecto

```
dependencies {  
    implementation fileTree(dir: "libs", include: ["*.jar"])  
    implementation 'androidx.appcompat:appcompat:1.1.0'  
    implementation 'androidx.constraintlayout:constraintlayout:1.1.3'  
    testImplementation 'junit:junit:4.12'  
    androidTestImplementation 'androidx.test.ext:junit:1.1.1'  
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.2.0'  
}
```

6. ANDROID MANIFEST

- **AndroidManifest.xml**



Archivo de configuración de la aplicación. Proporciona información esencial sobre la aplicación al sistema Android, información que el sistema debe tener para poder ejecutar el código de la APP.

Para que el compilador conozca los diversos componentes de la aplicación es necesario que estén aquí declarados.

Componentes: activity, service, content provider, broadcast receiver.

Además se definen las siguientes características: nombre de la aplicación, paquete, icono, estilos, y por último los permisos.

6. ANDROID MANIFEST

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.myapplication">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="My Application"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

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

</manifest>
```

6. ANDROID MANIFEST

Etiquetas principales:

- manifest: Engloba a las demás etiquetas. Define el espacio de nombres, el nombre del paquete y atributos del mismo.

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"  
    package="com.example.myapplication">
```

- application: Contiene metadatos de la aplicación (título, icono, tema), y las etiquetas de los componentes.

```
<application  
    android:allowBackup="true"  
    android:icon="@mipmap/ic_launcher"  
    android:label="@string/app_name"  
    android:roundIcon="@mipmap/ic_launcher_round"  
    android:supportRtl="true"  
    android:theme="@style/AppTheme">  
    <activity android:name=".MainActivity">  
        <intent-filter>  
            <action android:name="android.intent.action.MAIN" />  
  
            <category android:name="android.intent.category.LAUNCHER" />  
        </intent-filter>  
    </activity>  
</application>
```

6. ANDROID MANIFEST

Cuando sale en gris un dato, quiere decir que está cogiendo una referencia a otro archivo.

```
android:label="My Application"
```

Al pulsar sobre el elemento, sale la referencia:

```
android:label="@string/app_name"
```

6.1. PERMISOS

Principio de menor privilegio

Los permisos para la aplicación son declarados por el programador en el archivo AndroidManifest.xml, de manera que los usuarios saben que va a hacer la aplicación.

Para ello se utiliza la etiqueta uses-permission dentro de la etiqueta **manifest**, y fuera de la etiqueta **application**.

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.myapplication">

    <uses-permission android:name="android.permission.INTERNET"/>

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
```

6.1. PERMISOS

Tipos de permisos:

- Permisos normales:

- Comunicaciones
- Conexión WIFI
- Bluetooth
- Consumo de batería
- Aplicaciones
- Configuración del sistema
- Audio
- Sincronización
- Ubicación
- Seguridad

- Permisos peligrosos:

- Almacenamiento externo
- Ubicación
- Teléfono
- Mensajes de texto
- Contactos
- Calendario
- Cámara
- Micrófono
- Sensores corporales

6.1. PERMISOS

A partir de Android 6.0 (Marshmallow) existe un nuevo administrador de permisos peligrosos, una nueva medida de seguridad que nos dará más control y privacidad, ya que podemos revisar los permisos de las aplicaciones de una forma sencilla e intuitiva.

Ajustes - Aplicaciones

Dentro de la aplicación, hay una opción de Permisos. Donde se pueden ver los permisos aceptados de la aplicación, y modificarlo.

Además, como desarrolladores podemos establecer diálogos para su aceptación cuando se instala la aplicación, y además debemos comprobar que los permisos han sido aceptados cuando se va a hacer uso de su función.

6.1. PERMISOS

Los permisos normales es necesario declararlos en AndroidManifest.xml pero el usuario no los tendrá que aceptar, ni el programador solicitar su uso.

Por ejemplo, internet.

<https://developer.android.com/training/permissions/requesting?hl=es-419>

7. RECURSOS DE UNA APLICACIÓN

Los recursos son los datos que utiliza una aplicación; imágenes, textos, estilos...

Las aplicaciones se ejecutarán en dispositivos con diferentes características y configuraciones. Por lo que hay que tener en cuenta los siguientes aspectos:

- Densidad de pantalla
- Idiomas del usuario
- Orientación
- Tabletas

Y habrá definir recursos para cada estilo de configuración.

7. RECURSOS DE UNA APLICACIÓN

- **Carpeta res**
 - Recursos de la aplicación. Diferentes subcarpetas:
 - **drawable**: recursos dibujables, ficheros de bitmaps o de imágenes “escalables”, formas y colores de objetos dibujados. Tipos de archivos aceptados: **.png, .9.png, .jpg y .gif**
 - **layout**: definidos como ficheros XML. Engloban los elementos visuales que definen la interfaz de la aplicación.
 - **mipmap**: contenido similar a drawable. Alberga elementos bitmap, aunque se suele utilizar específicamente para ubicar el icono de la aplicación.

7. RECURSOS DE UNA APLICACIÓN

- **Carpeta res**
 - Recursos de la aplicación. Diferentes subcarpetas:
 - **values:** de carácter genérico con ficheros XML que configuran diferentes aspectos de la aplicación, como es el color, dimensiones, cadenas de texto y estilos.
 - Strings.xml - cadenas utilizadas.
 - Colors.xml - definición de los tres colores primarios.
 - Carpeta themes - estilos y temas de la aplicación.
 - themes.xml - estilos para la aplicación.
 - themes.xml (night) - estilos para la aplicación cuando el dispositivo está en modo noche.

7. RECURSOS DE UNA APLICACIÓN

- **Carpeta res**
 - Aunque no se vean al crear el proyecto, también tenemos:
 - **menu**: ficheros xml donde se definen las diferentes opciones de los menús, submenús, barras de navegación incluidos en la aplicación.
 - **animator**: permite crear animaciones sencillas sobre uno o varios gráficos, como rotaciones, fading, movimiento y estiramiento. Cada animación se define en un fichero XML.
 - **xml**: almacena archivos XML utilizados por Android para dar soporte a tareas múltiples, como configurar búsquedas.

7. RECURSOS DE UNA APLICACIÓN

- **Carpeta res**
 - Aunque no se vean al crear el proyecto, también tenemos:
 - **raw**: archivos raw (multimedia). Aquí se guardan los archivos de audio/vídeo. El nombre de los archivos está muy limitado, no se aceptan mayúsculas ni caracteres especiales, por lo que nos permiten guardar este tipo de recursos en otra carpeta llamada **assets**.

7.1. CÓMO ACCEDER A LOS RECURSOS

Automáticamente, Android asigna un identificador a cada uno de los recursos de tu aplicación, por lo que quedan registrados en el archivo de la clase R tanto el tipo como el id.

De esta forma, tanto desde Java como desde XML podemos acceder a los recursos.

7.1. CÓMO ACCEDER A LOS RECURSOS

- Desde XML:

Escribir la dirección del archivo donde se encuentra seguido del nombre del recurso. Para ello, se antepone un @ a la carpeta, y se separa con /.

```
android:icon="@mipmap/ic_launcher"  
android:label="@string/app_name"  
android:roundIcon="@mipmap/ic_launcher_round"  
android:supportsRtl="true"  
android:theme="@style/AppTheme">
```

7.1. CÓMO ACCEDER A LOS RECURSOS

- Desde Java:

Mediante el método getResources podemos localizar el recurso como en XML, con la dirección del archivo y el nombre del recurso.

```
getResources().getString(R.string.app_name);  
getResources().getColor(R.color.colorAccent);  
getResources().getDrawable(R.drawable.ic_launcher_background);
```

Hay que tener cuidado, puesto que hay métodos que se han quedado obsoletos, y por tanto hay que investigar la nueva forma de uso.

```
ContextCompat.getColor( context: this, R.color.colorAccent);  
ContextCompat.getDrawable( context: this, R.drawable.ic_launcher_background);
```

Estos ejemplos, se han actualizado de manera que utilizan el contexto de la actividad para trabajar con los recursos.

7.1. CÓMO ACCEDER A LOS RECURSOS

Además de nuestros recursos, Android nos ofrece toda una gama de recursos para nuestro trabajo. Para acceder a ellos el mecanismo es el mismo, simplemente hay que indicar que se encuentran en una ruta propia de android.

XML:

```
android:textColor="@android:color/black"
```

Java:

```
ContextCompat.getColor( context: this, android.R.color.black);
```


7.2. USO DE RECURSOS

- **Archivo strings.xml**

El archivo strings.xml se encuentra en res/values. Contiene las cadenas utilizadas en la aplicación, y se deben indicar aquí los textos usados. Simplificará el trabajo si después queremos meter idiomas en nuestra aplicación.

```
<resources>
  <string name="app_name">Componentes</string>
</resources>
```

XML:

```
android:text="enviar"/>
```

Hardcoded string "enviar", should use @string resource [more...](#) (Ctrl+F1)

JAVA:

```
button.setText(R.string.cancelar);
```

KOTLIN:

```
button.setText(R.string.enviar)
```

7.2. USO DE RECURSOS

- **Archivo colors.xml**

El archivo colors.xml se encuentra en res/values. Contiene el código HTML de los diferentes colores a utilizar en la aplicación.

```
<resources>
    <color name="colorPrimary">#008577</color>
    <color name="colorPrimaryDark">#00574B</color>
    <color name="colorAccent">#D81B60</color>
</resources>
```

7.2. USO DE RECURSOS

- Archivo colors.xml

XML:

```
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@+id/button"
    android:text="@string/enviar"
    android:background="@color/colorRed"/>
```

JAVA:

```
button.setBackgroundColor(resources.getColor(R.color.colorRed, applicationContext.theme))
```

KOTLIN:

```
button.setBackgroundColor(resources.getColor(R.color.colorRed, getApplicationContext().getTheme()));
```

7.2. USO DE RECURSOS

- **Archivo themes.xml**

Un estilo es una colección de propiedades que definen el formato y la apariencia que tendrá una vista. Podemos especificar tamaño, márgenes, color, fuentes...

Un estilo se define en el fichero themes.xml que se encuentra en la carpeta res/values/themes, y se utiliza en cualquier layout que definamos.

```
<style name="EstiloBoton" parent="TextAppearance.AppCompat.Button">
    <item name="android:layout_height">wrap_content</item>
    <item name="android:layout_width">wrap_content</item>
    <item name="android:textColor">@color/colorRed</item>
    <item name="android:typeface">monospace</item>
</style>
```

7.2. USO DE RECURSOS

- **Archivo themes.xml**

Si vas a heredar de un estilo definido por ti, no es necesario utilizar el atributo parent. Puedes utilizar el mismo nombre de un estilo ya creado y completar el nombre con un punto más un sufijo.

```
<style name="EstiloBoton.Grande">
    <item name="android:textSize">24pt</item>
</style>
```

7.2. USO DE RECURSOS

- Archivo themes.xml

Definición de un elemento visual (Botón) sin estilos:

```
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@+id/button"
    android:text="enviar"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    android:textColor="@color/colorRed"
    android:typeface="monospace"
/>
```

7.2. USO DE RECURSOS

- Archivo themes.xml

Definición de un elemento visual (Botón) con estilos:

```
<Button
    style="@style/EstiloBoton"
    android:id="@+id/button"
    android:text="enviar"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintTop_toTopOf="parent"
/>
```

7.2. USO DE RECURSOS

- **Archivo themes.xml**

Un tema es un estilo aplicado a toda una actividad o aplicación, en lugar de a una vista individual. Cada elemento del estilo solo se aplicará a aquellos elementos donde sea posible.

Para aplicar un tema a toda una aplicación, edita el fichero AndroidManifest.xml y añade el parámetro android:theme en la etiqueta <application>

```
android:theme="@style/AppTheme">
```

Si queremos quitar el AppBar, hay que cambiar el theme.

```
android:theme="@style/Theme.AppCompat.NoActionBar">
```


7.2. USO DE RECURSOS

Dimensiones icono aplicación Android:

- mipmap-mdpi: 48×48
- mipmap-hdpi: 72×72
- mipmap-xhdpi: 96×96
- mipmap-xxhdpi: 144×144
- Mipmap-xxxhdpi: 512 x 512

8. CLASE R

Se trata de una clase que contiene variables estáticas en las que se identifica cada tipo de recurso. Android lee el fichero XML, carga todas las estructuras solicitadas en memoria y mantiene el fichero R como referencia a los recursos cargados. Cada atributo tiene una dirección de memoria asociada referenciada a un recurso específico.

Se pueden producir errores relativos a esta clase.

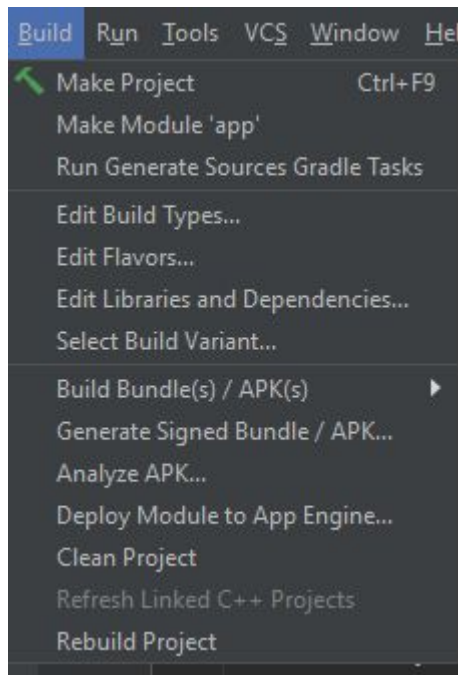
¡NUNCA HAY QUE MODIFICAR ESTA CLASE!

Posibles soluciones:

- Asegurarse de que no existan enlaces rotos.
- Asegurarse de que no haya errores de sintaxis en XML.

8. CLASE R

Si todo lo anterior parece estar correcto, entonces volveremos a compilar nuestro proyecto:

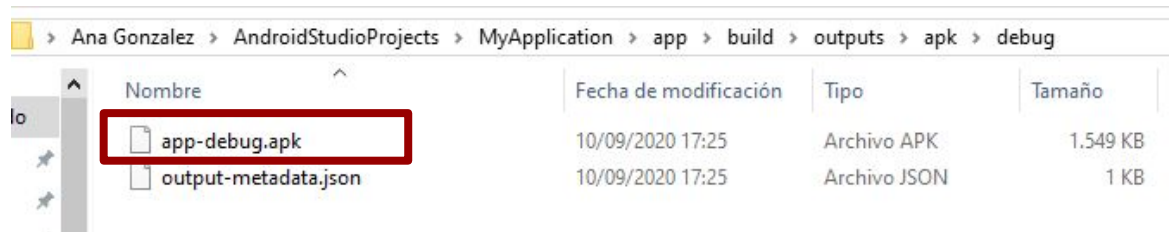
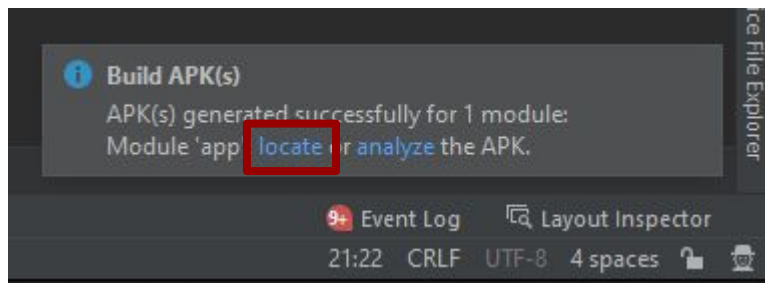
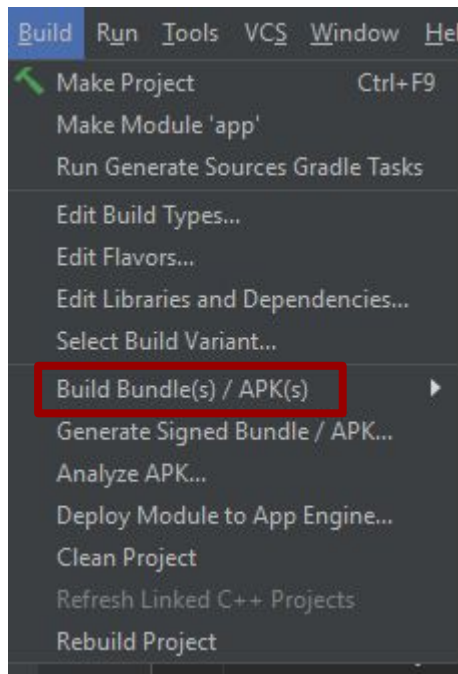


Podemos hacerlo de dos formas:

- Clean Project y después Make Project
- Rebuild Project

9. GENERAR EJECUTABLE

Para generar el APK:



8. DUDAS Y PREGUNTAS

