# Cómo instalar Android Studio



Puedes configurar Android Studio con unos pocos clics.

Primero, asegúrate de descargar la versión más reciente de Android Studio.

## Windows

Para instalar Android Studio en Windows, haz lo siguiente:

1. Accede a la página de **Android Developer** y selecciona la opción "**Descargar Android Studio**". Marca la casilla de "He leído y acepto los..." y pulsa en "**Descargar Android Studio para Windows**".
2. Tras su descarga, pulsa sobre el instalador. Pulsa Next y mantén las dos opciones marcadas seleccionadas. Vuelve a pulsar Next.
3. Elige la carpeta de destino de Android Studio, y pulsa Next. Si quieres crear una carpeta en el menú de inicio, pulsa Install. Si no, marca la opción "Do not create shortcuts", y pulsa Install.
4. Una vez se complete la instalación, vuelve a pulsar Next, y después Finish. Android Studio se iniciará automáticamente.
5. En el primer arranque del programa, te preguntará si quieres importar tus preferencias. Marca la casilla "Do not import settings" y pulsa OK.
6. Comenzará el proceso de **configuración de Android Studio**. Selecciona la configuración estándar (Standard), elige el tema del programa que más te guste y pulsa finish.
7. Cuando finalice el proceso de descarga e instalación de paquetes, se abrirá la **pantalla principal de Android Studio.** Para **comenzar un nuevo proyecto de Android Studio**, pulsa "Start a new Android Studio Project".
8. Elige un nombre para la aplicación, escribe el nombre de tu dominio web (si lo tienes) y pulsa Next. Selecciona la versión de Android sobre la que quieras trabajar, elige la actividad que se añadirá al teléfono, el nombre de dicha actividad y de su disposición.
9. ¡Ya puedes comenzar a **trabajar en tu aplicación**!

Se ve así. En el siguiente video, se muestra cada paso del procedimiento de configuración cuando se usa la descarga de `.exe` recomendada.

Link del video:

https://www.youtube.com/watch?v=As2o73dsw2E

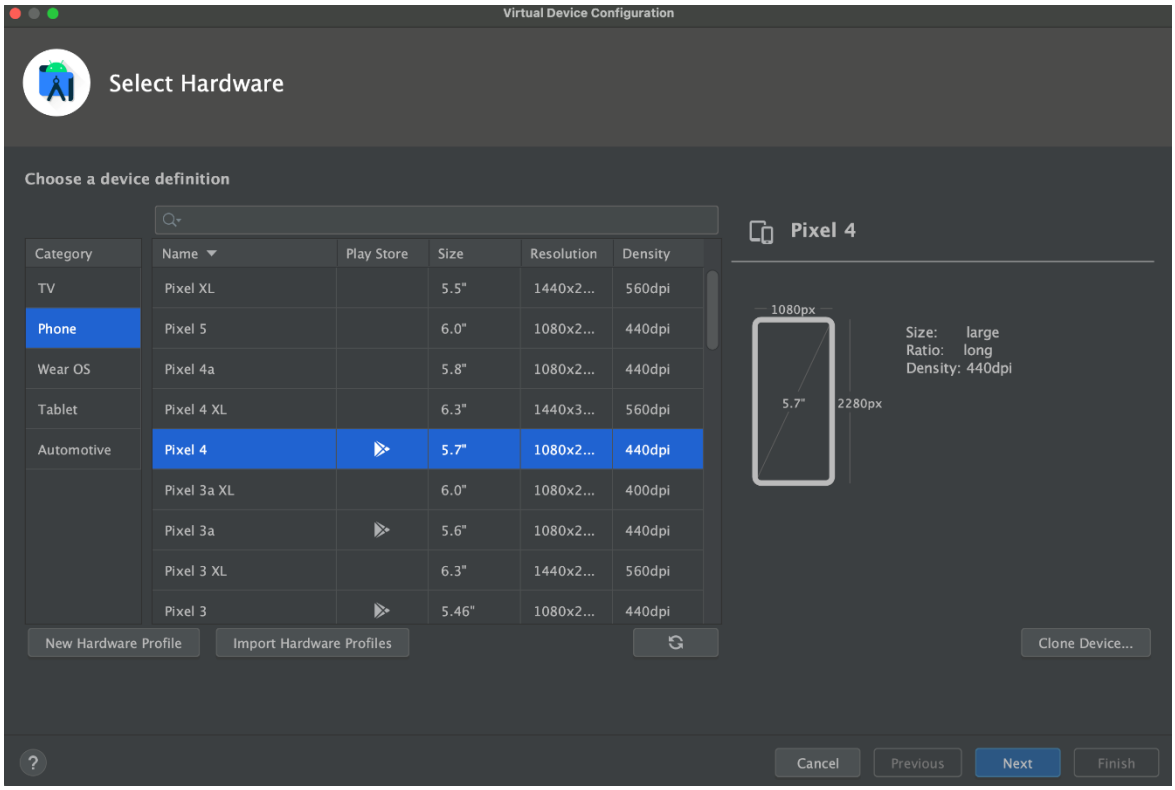
# Guía Cómo crear un AVD

**Sugerencia:** Si quieres iniciar tu app en un emulador, en lugar de ejecutarla desde Android Studio y luego en el diálogo **Select Deployment Target** que aparece, haz clic en **Create New Virtual Device**.

Para crear un AVD nuevo:

1. Abre el Administrador de dispositivos.
2. Haz clic en **Create device**.

   Aparecerá la ventana **Select Hardware**.

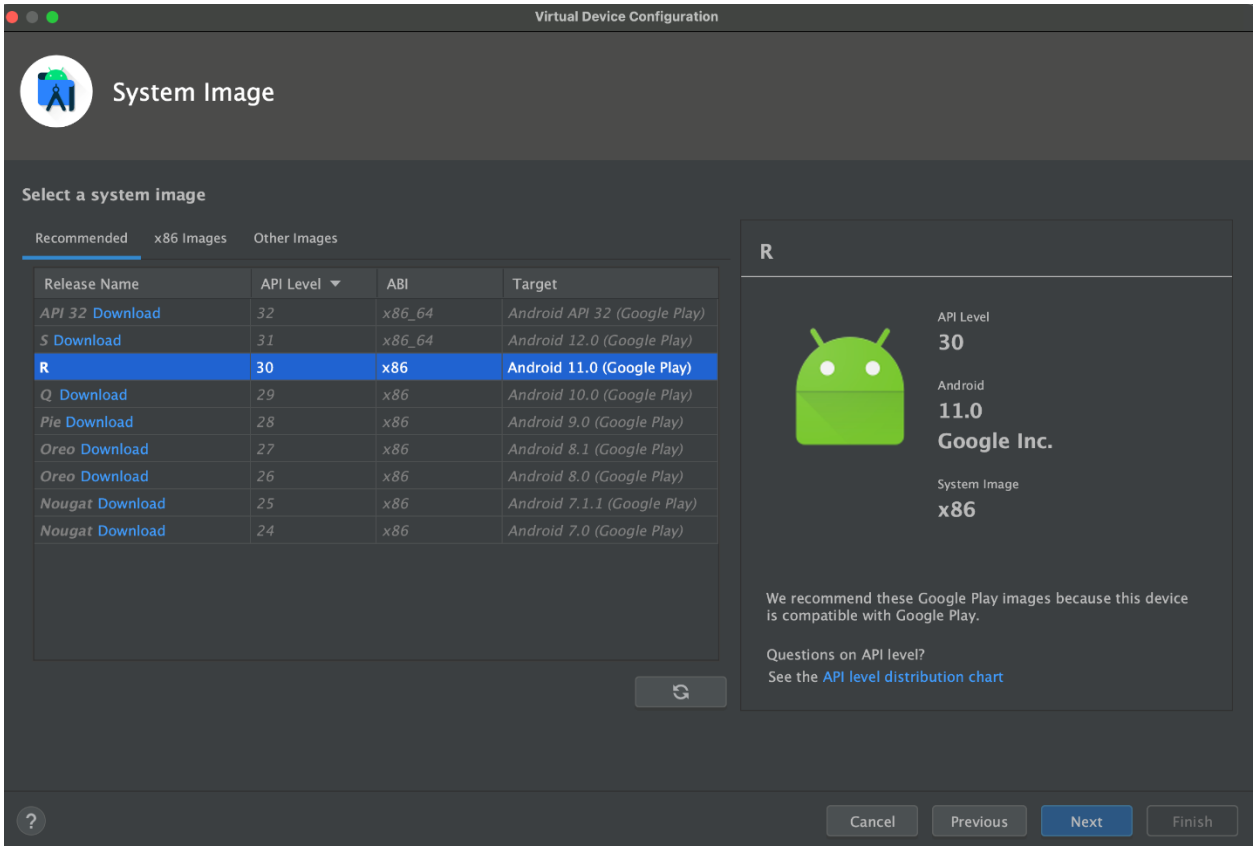



   Ten en cuenta que solo algunos perfiles de hardware incluyen **Play Store** por indicación. Esto indica que esos perfiles son totalmente compatibles con CTS y pueden usar imágenes del sistema que incluyan la app de Play Store.

3. Selecciona un perfil de hardware y haz clic en **Next**.

   Si no ves el perfil de hardware que deseas, puedes crear o importar uno.

   Aparecerá la página **System Image**.




4. Selecciona la imagen del sistema para un nivel de API determinado y haz clic en **Next**.

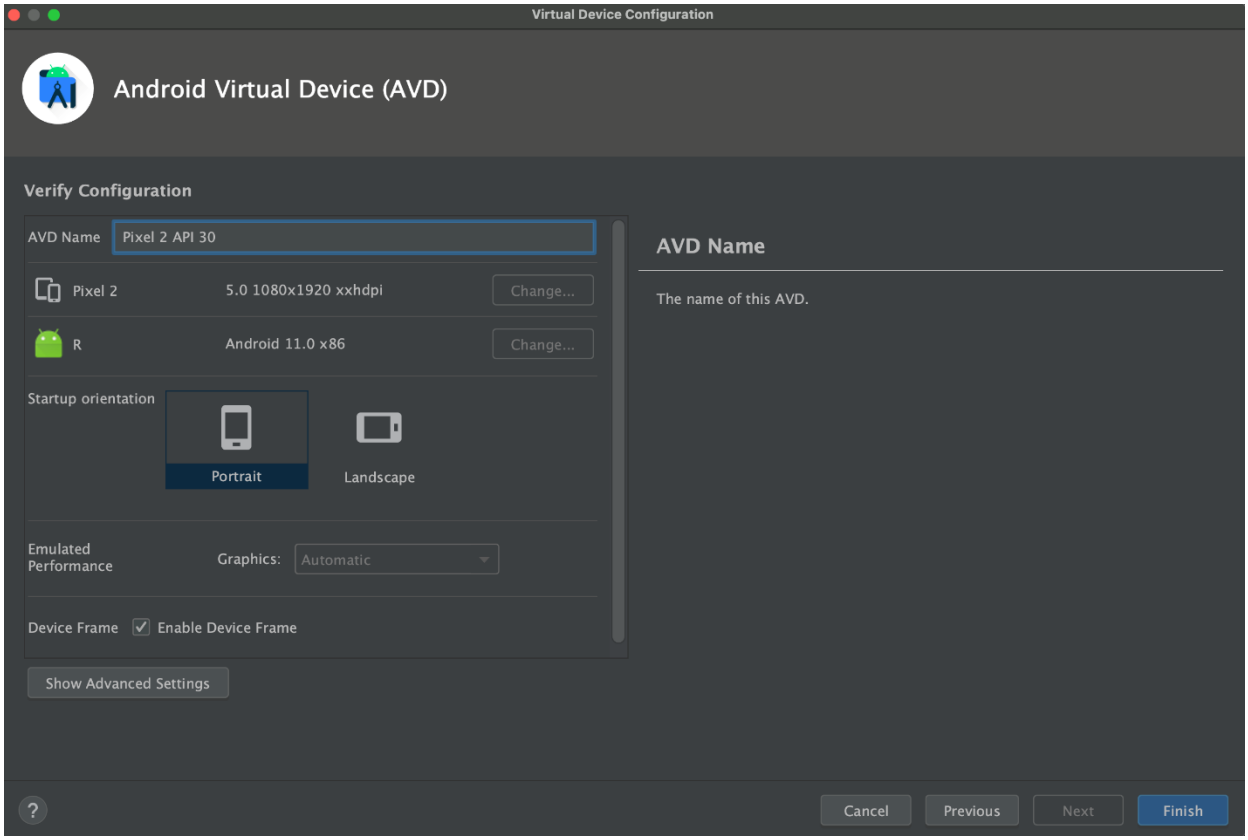
   En la pestaña **Recommended**, aparecen listas de imágenes de sistema recomendadas. En las otras pestañas, se incluye una lista más completa. En el panel derecho, se describe la imagen de sistema seleccionada. Las imágenes x86 son las que se ejecutan más rápido en el emulador.

   Si ves **Download** junto a la imagen de sistema, debes hacer clic en la opción para descargarla. Para concretar la descarga, debes estar conectado a Internet.

   El nivel de API del dispositivo de destino es importante, ya que no podrá ejecutarse tu app en una imagen del sistema con un nivel de API inferior al que se necesita, según se especifica en el atributo `minSdkVersion` del archivo de manifiesto de la app. Para obtener más información sobre la relación entre el nivel de API del sistema y `minSdkVersion`, consulta el Control de versiones de tus apps.

   Si tu app declara un elemento `<uses-library>` en el archivo de manifiesto, requerirá una imagen de sistema en la que se encuentre esa biblioteca externa. Si deseas ejecutar tu app en un emulador, crea un AVD que incluya la biblioteca obligatoria. Si lo haces, es posible que debas usar un componente complementario para la plataforma de AVD; por ejemplo, el complemento de las API de Google contiene la biblioteca de Google Maps.

   Aparecerá la página **Verify Configuration**.




5. Modifica las propiedades de AVD según sea necesario y, luego, haz clic en **Finish**.

   Haz clic en **Show Advanced Settings** para que se muestren más configuraciones, como la máscara.

   El nuevo AVD aparecerá en la pestaña **Virtual** del Administrador de dispositivos y en el menú desplegable de destino.

## Propiedades de AVD

Puedes especificar las siguientes propiedades para las configuraciones de AVD en la página **Verify Configuration**. La configuración de AVD especifica la interacción entre la máquina de desarrollo y el emulador, así como las propiedades que deseas anular en el perfil de hardware.

Las propiedades de configuración de AVD anulan las de perfil de hardware. Ambas son anuladas por las propiedades del emulador que estableces mientras este se encuentra en funcionamiento.

| Propiedad del AVD | Descripción |
|---|---|
| Nombre de AVD | Indica el nombre del AVD. Puede contener mayúsculas o minúsculas, números del 0 al 9, puntos (.), guiones bajos (_), paréntesis ( () ), guiones (-) y espacios. El nombre del archivo que contiene la configuración de AVD deriva del nombre de este. |
| AVD ID (Advanced) | El nombre de archivo de AVD deriva del ID y puedes usarlo para hacer referencia al AVD desde la línea de comandos. |
| Hardware Profile | Haz clic en **Change** para seleccionar un perfil de hardware diferente en la página **Select Hardware**. |
| System Image | Haz clic en **Change** para seleccionar una imagen del sistema diferente en la página **System Image**. Se requiere una conexión activa de Internet para descargar una imagen nueva. |

| | |
|---|---|
| Startup Orientation | Selecciona una opción para la orientación inicial del emulador:<br><br>• Vertical: Orientación de mayor altura que ancho<br>• Horizontal: Orientación de mayor ancho que altura<br><br>Se habilita una opción solo si se encuentra seleccionada en el perfil de hardware. Al ejecutar el AVD en el emulador, puedes modificar la orientación si el perfil de hardware admite tanto el formato vertical como el horizontal. |
| Camera (Advanced) | Para habilitar la cámara, selecciona una o ambas opciones:<br><br>• Front: La lente se orienta hacia el lado opuesto al del usuario.<br>• Back: la lente se orienta hacia el usuario.<br><br>Con el ajuste **Emulated**, se obtiene una imagen generada mediante software, mientras que **Webcam** usa la cámara web de tu computadora de desarrollo para tomar una foto.<br><br>Esta opción solo está disponible si se selecciona en el perfil de hardware. No está disponible para Wear OS ni Android TV. |
| Network: Speed (Advanced) | Selecciona un protocolo de red para determinar la velocidad de la transferencia de datos:<br><br>• GSM: Sistema de telefonía global para comunicaciones móviles<br>• HSCSD: Datos de alta velocidad con conmutación de circuitos<br>• GPRS: Servicio genérico de radiocomunicaciones por paquetes<br>• EDGE: Velocidades de datos mejoradas para la evolución de GSM<br>• UMTS: Sistema universal de telecomunicaciones móviles<br>• HSDPA: Acceso a paquetes de alta velocidad por enlace descendente<br>• LTE: Evolución a largo plazo<br>• Full (predeterminada): Transfiere datos tan rápido como lo permite la computadora |
| Network: Latency (Advanced) | Selecciona un protocolo de red a fin de establecer el tiempo (demora) necesario para que el protocolo transfiera un paquete de datos de un punto a otro. |
| Emulated Performance: Graphics | Selecciona el modo en que se representan los gráficos en el emulador:<br><br>• Hardware: usa la tarjeta de gráficos de tu computadora para una presentación más rápida.<br>• Software: emula los gráficos en el software; esto resulta útil si tienes un problema con la representación en tu tarjeta de gráficos.<br>• Auto: permite que el emulador determine la mejor opción según tu tarjeta de gráficos. |
| Emulated Performance: Boot option (Advanced) | • Cold boot: Enciende el dispositivo activándolo a partir del estado de apagado.<br>• Quick Boot: Enciende el dispositivo cargando el estado de este a partir de una instantánea guardada. Para obtener detalles, consulta Cómo ejecutar el emulador con Quick Boot. |
| Emulated Performance: Multi-Core CPU (Advanced) | Selecciona la cantidad de núcleos del procesador de tu computadora que quisieras usar para el emulador. El uso de más núcleos acelera el emulador. |
| Memory and Storage: RAM | El volumen de memoria RAM en el dispositivo. El fabricante del hardware fija este valor, pero puedes anularlo si es necesario, por ejemplo, para lograr que el emulador funcione más |

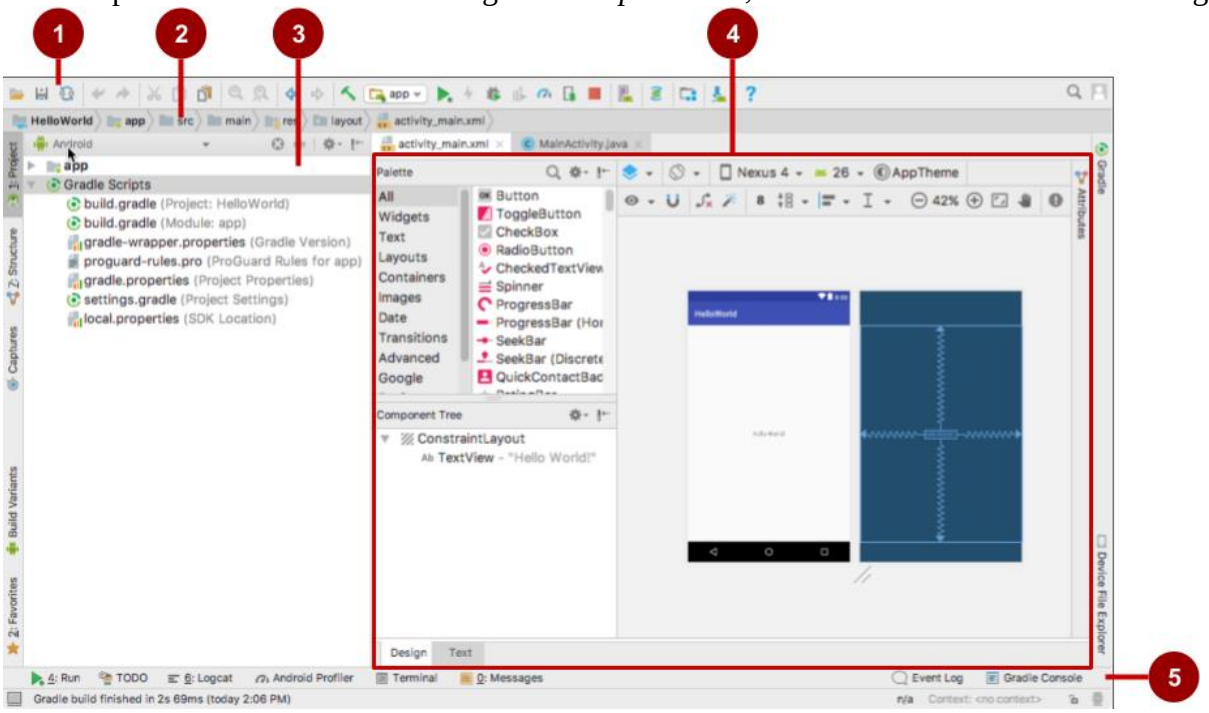
| | |
|---|---|
| | rápido. El aumento de tamaño ocupa más recursos de tu computadora. Escribe un tamaño de memoria RAM y selecciona las unidades: B (bytes), KB (kilobytes), MB (megabytes), GB (gigabytes) o TB (terabytes). |
| Memory and Storage: VM Heap | Tamaño del montón de VM. El fabricante del hardware fija este valor, pero puedes anularlo si es necesario. Ingresa un tamaño de montón y selecciona las unidades: B (bytes), KB (kilobytes), MB (megabytes), GB (gigabytes) o TB (terabytes). Para obtener más información sobre las VM de Android, consulta Administración de memoria para diferentes equipos virtuales. |
| Memory and Storage: Internal Storage | Espacio disponible de memoria no extraíble en el dispositivo. El fabricante del hardware fija este valor, pero puedes anularlo si es necesario. Ingresa un tamaño y selecciona las unidades; B (bytes), KB (kilobytes), MB (megabytes), GB (gigabytes) o TB (terabytes). |
| Memory and Storage: SD Card | Espacio disponible de memoria extraíble para almacenar datos en el dispositivo. Para usar una tarjeta SD administrada por Android Studio, selecciona **Studio-managed**, ingresa un tamaño y selecciona las unidades: B (bytes), KB (kilobytes), MB (megabytes), GB (gigabytes) o TB (terabytes). Se recomiendan como mínimo 100 MB para usar la cámara. A fin de administrar el espacio en un archivo, selecciona **External file** y haz clic en **…** a fin de especificar el archivo y la ubicación. Para obtener más información, consulta mksdcard y el directorio de datos de AVD. |
| Device Frame: Enable Device Frame | Habilita un marco alrededor de la ventana del emulador que reproduzca la apariencia de un dispositivo real. |
| Custom Skin Definition (Advanced) | Selecciona una máscara que controle la apariencia del dispositivo cuando se muestre en el emulador. Recuerda que especificar un tamaño de pantalla demasiado grande para la resolución puede hacer que la pantalla quede cortada y no puedas verla completa. Para obtener más información, consulta Cómo crear una máscara de emulador. |
| Keyboard: Enable Keyboard Input (Advanced) | Selecciona esta opción si deseas usar tu teclado de hardware para interactuar con el emulador. Está inhabilitada para Wear OS y Android TV. |

## Webgrafia


- https://developer.android.com/studio/run/managing-avds?hl=es-419

# Explorando un proyecto

Un proyecto de Android Studio contiene todo el código fuente y todos los recursos de una aplicación. Los recursos incluyen diseños, cadenas, colores, dimensiones e imágenes. La ventana principal de Android Studio se compone de varias áreas lógicas o *paneles* , como se muestra en la siguiente figura.




En la figura de arriba:

1. Barra de herramientas: proporciona una amplia gama de acciones, incluida la ejecución de la aplicación de Android y el inicio de herramientas de Android.
2. Barra de navegación: navegue por el proyecto y abra archivos para editarlos.
3. **Panel Proyecto** : muestra los archivos del proyecto en una jerarquía. La jerarquía seleccionada en la figura anterior es **Android** .
4. Editor: El contenido de un archivo seleccionado en el proyecto. Por ejemplo, después de seleccionar un diseño (como se muestra en la figura anterior), el panel del editor muestra el editor de diseño con herramientas para editar el diseño. Después de seleccionar un archivo de código Java, el panel del editor muestra el código Java con herramientas para editar el código.
5. Pestañas a la izquierda, a la derecha y en la parte inferior de la ventana: puede hacer clic en las pestañas para abrir otros paneles, como **Logcat** para abrir el panel **Logcat** con mensajes de registro o **TODO** para administrar tareas.

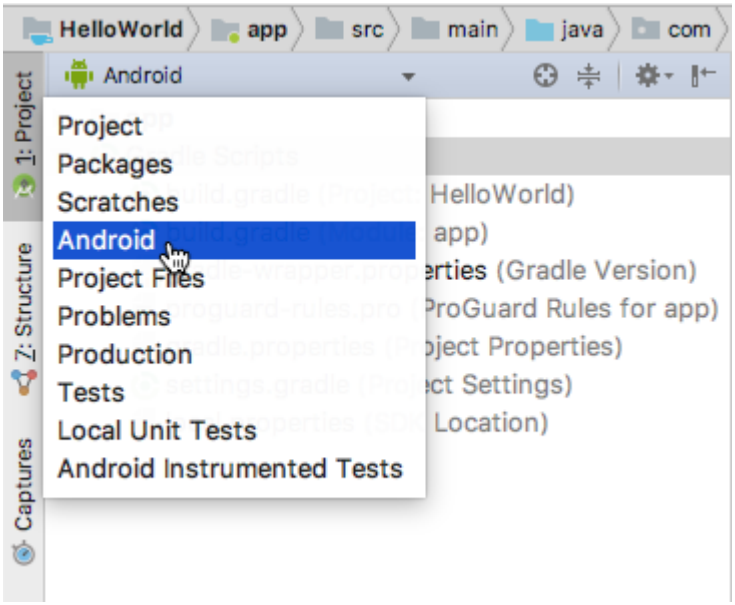
La barra de estado en la parte inferior de la ventana de Android Studio muestra el estado del proyecto y el propio Android Studio, así como cualquier advertencia o mensaje. Puede ver el progreso de la compilación en la barra de estado.

**Sugerencia** : puede organizar la ventana principal para disponer de más espacio en la pantalla ocultando o moviendo paneles. También puede usar métodos abreviados de teclado para acceder a la mayoría de las funciones. Consulte Métodos abreviados de teclado para obtener una lista completa.

## Uso del panel Proyecto

Puede ver la organización del proyecto de varias formas en el panel Proyecto. Si aún no está seleccionado, haga clic en la pestaña **Proyecto.** (La pestaña **Proyecto** está en la columna de pestaña vertical en el lado izquierdo de la ventana de Android Studio).

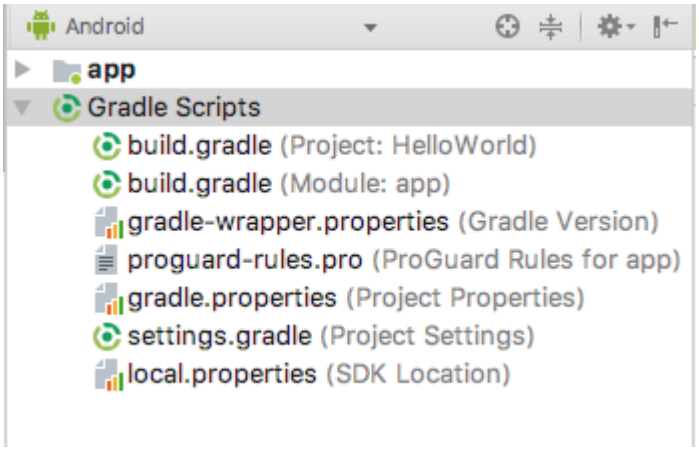
Aparece el panel Proyecto. Para ver el proyecto en la jerarquía de proyectos estándar de Android, seleccione **Android** en la flecha hacia abajo en la parte superior del panel **Proyecto.**




**Nota:** este capítulo y otros capítulos hacen referencia al panel Proyecto, cuando se establece en **Android** , como el panel **Proyecto** > **Android** .

**archivos Gradle**

Cuando crea un proyecto de aplicación por primera vez, aparece el panel **Proyecto** > **Android**`Gradle Scripts` con la carpeta expandida como se muestra a continuación. Si la `Gradle Scripts`carpeta no está expandida, haga clic en el triángulo para expandirla. Esta carpeta contiene todos los archivos que necesita el sistema de compilación.

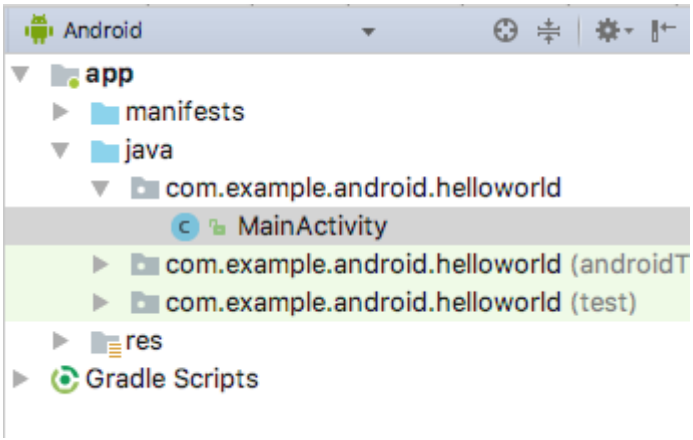



El `build.gradle(Module:app)` archivo especifica bibliotecas adicionales y la configuración de compilación del módulo. La `Activity`plantilla que seleccione crea este archivo. El archivo incluye el `minSdkVersion`atributo que declara la versión mínima de la aplicación y el `targetSdkVersion`atributo que declara la versión más alta (más nueva) para la que se optimizó la aplicación.

Este archivo también incluye una lista de *dependencias* , que son bibliotecas requeridas por el código, como la `AppCompat`biblioteca para admitir una amplia gama de versiones de Android.

**código de la aplicación**

Para ver y editar el código Java, expanda la `app`carpeta, la `java`carpeta y la `com.example.android.helloworld`carpeta. Haga doble clic en el `MainActivity` `java`archivo para abrirlo en el editor de código.




La `java`carpeta incluye archivos de clase Java. Cada `Activity`, `Service`u otro componente (como un `Fragment`) se define como una clase Java, normalmente en su propio archivo. Las pruebas y otros archivos de clase de Java también se encuentran aquí.

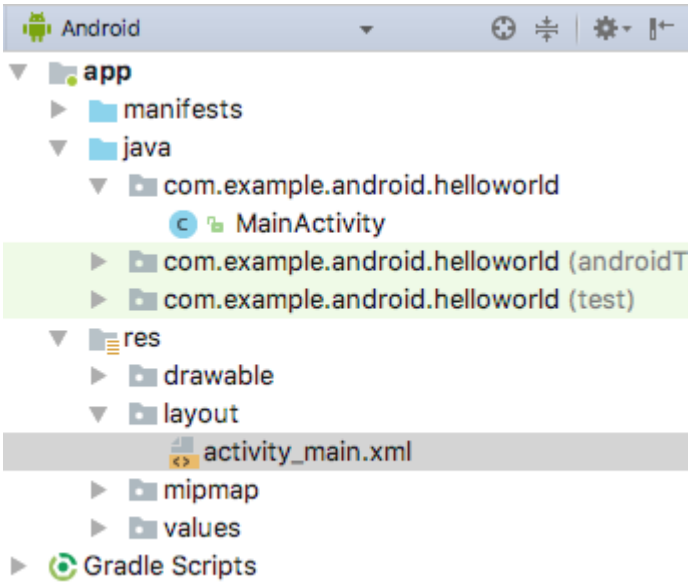
La `java`carpeta contiene tres subcarpetas:

- `com.example.hello.helloworld`(o el nombre de dominio que haya especificado) **:** todos los archivos de un paquete están en una carpeta con el nombre del paquete. Para su aplicación Hello World, hay un paquete y solo contiene `MainActivity.java`. La primera `Activity`(pantalla) que ve el usuario, que también inicializa los recursos de toda la aplicación, suele llamarse `MainActivity`. (La extensión del archivo se omite en el panel **Proyecto > Android ).**
- `com.example.hello.helloworld(androidTest)`: Esta carpeta es para sus pruebas instrumentadas y comienza con un archivo de prueba de esqueleto.
- `com.example.hello.helloworld(test)`: Esta carpeta es para sus pruebas de unidad y comienza con un archivo de prueba de unidad de esqueleto creado automáticamente.

**Archivos de diseño**

Para ver y editar un archivo de diseño, expanda la `res`carpeta y la `layout`carpeta para ver el archivo de diseño. En la siguiente figura, el archivo de diseño se llama `activity_main.xml`.

Haga doble clic en el archivo para abrirlo en el editor de diseño. Los archivos de diseño están escritos en XML.




**Archivos de recursos**

La `res`carpeta contiene recursos, como diseños, cadenas e imágenes. Por lo general, se asocia con un `Activity`diseño de vistas de interfaz de usuario que se definen como un archivo XML. Este archivo XML suele llevar el nombre de su `Activity`. La `res`carpeta incluye estas subcarpetas:

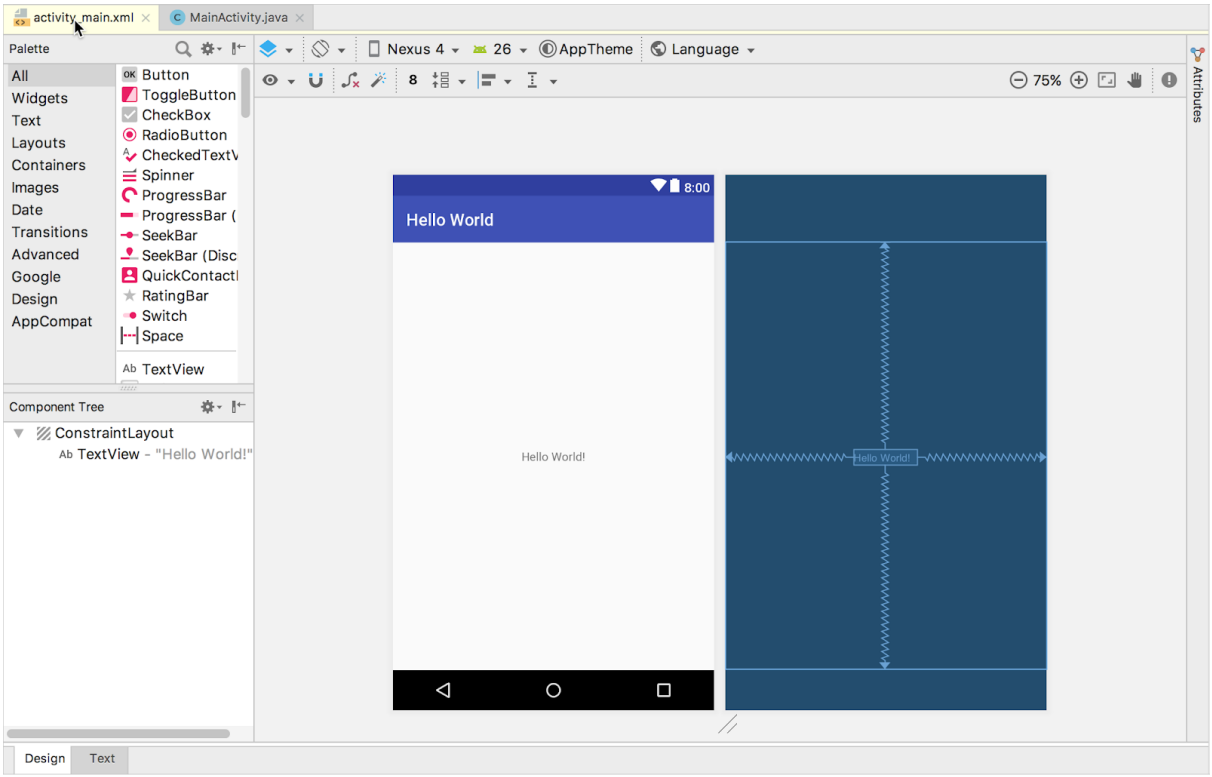
- `drawable`: almacene todas las imágenes de su aplicación en esta carpeta.
- `layout`: Cada `Activity`tiene al menos un archivo de diseño XML que describe la interfaz de usuario. Para Hello World, esta carpeta contiene `activity_main.xml`.
- `mipmap`: Los iconos del iniciador se almacenan en esta carpeta. Hay una subcarpeta para cada densidad de pantalla admitida. Android usa la densidad de la pantalla (la cantidad de píxeles por pulgada) para determinar la resolución de imagen requerida. Android agrupa todas las densidades de pantalla reales en densidades generalizadas, como media (mdpi), alta (hdpi) o extra-extra-extra-high (xxxhdpi). La `ic_launcher.png`carpeta contiene los íconos de inicio predeterminados para todas las densidades admitidas por su aplicación.
- `values`: En lugar de codificar valores como cadenas, dimensiones y colores en sus archivos XML y Java, se recomienda definirlos en sus respectivos `values`archivos. Esta práctica facilita cambiar los valores y mantener la coherencia de los valores en toda la aplicación.

La `values`subcarpeta incluye estas subcarpetas:

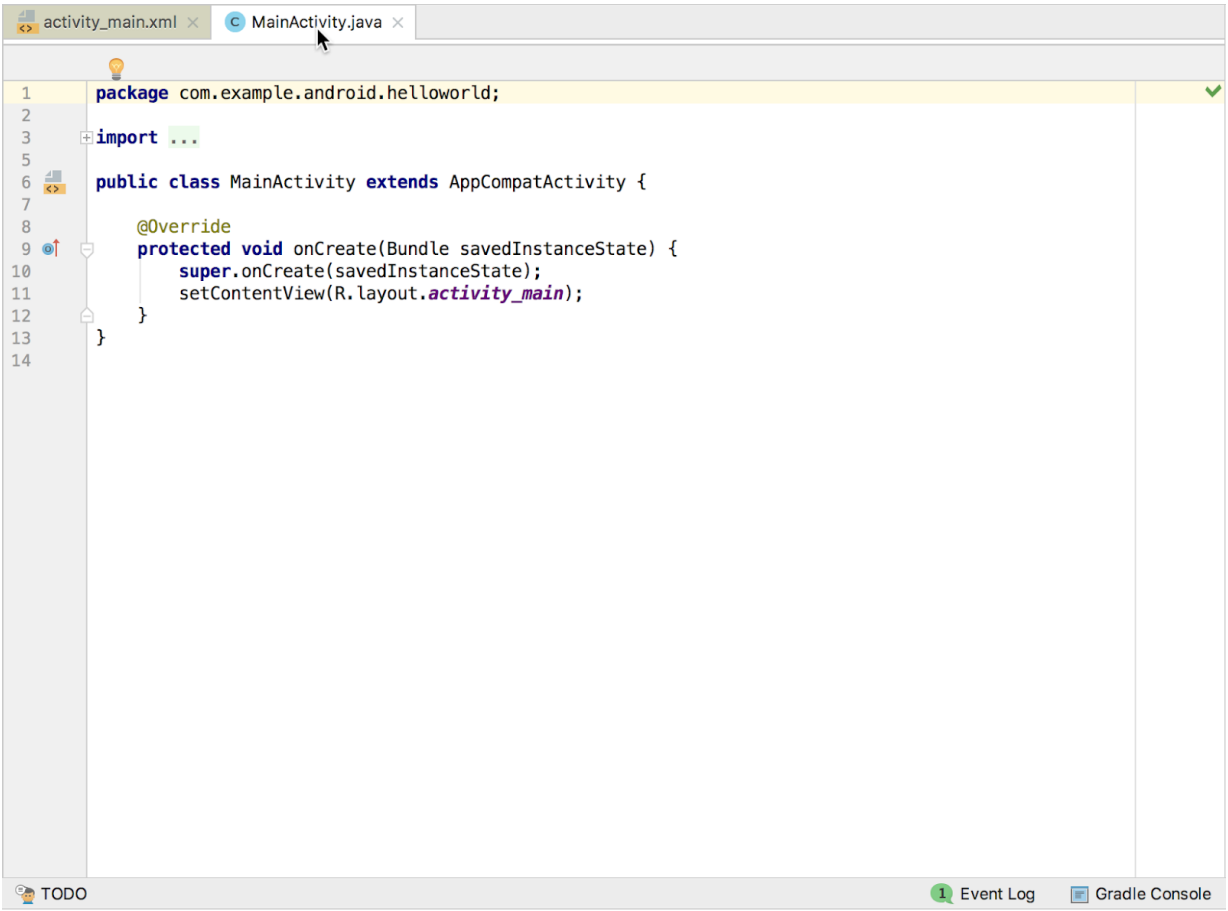
- `colors.xml`: muestra los colores predeterminados para el tema elegido. Puede agregar sus propios colores o cambiar los colores según los requisitos de su aplicación.
- `dimens.xml`: almacena los tamaños de vistas y objetos para diferentes resoluciones.
- `strings.xml`: Crea recursos para todas tus cadenas. Hacer esto facilita la traducción de las cadenas a otros idiomas.
- `styles.xml`**:** Todos los estilos para su aplicación y tema van aquí. Los estilos ayudan a darle a su aplicación una apariencia uniforme para todos los elementos de la interfaz de usuario.

## Uso del panel del editor

Si selecciona un archivo, aparece el panel del editor. Aparece una pestaña para el archivo para que pueda abrir varios archivos y alternar entre ellos. Por ejemplo, si hace doble clic en el archivo de diseño **activity_main.xml en el panel Proyecto > Android** , el editor de diseño aparece como se muestra a continuación.




Si hace doble clic en el archivo **MainActivity en el panel Proyecto > Android** , el editor cambia al editor de código como se muestra a continuación, con una pestaña para **MainActivity.java** :




En la parte superior del `MainActivity.java`archivo hay una `package`declaración que define el paquete de la aplicación. Esta instrucción de paquete va seguida de un `import`bloque condensado con . . ., como se muestra en la figura anterior. Haga clic en los puntos para expandir el bloque para verlo. Las `import`declaraciones importan las bibliotecas necesarias para la aplicación. Por ejemplo, la siguiente instrucción importa la `AppCompatActivity`biblioteca:

```
import android.support.v7.app.AppCompatActivity;
```

Cada uno `Activity`en una aplicación se implementa como una clase de Java. La siguiente declaración de clase amplía la `AppCompatActivity`clase para implementar funciones de manera que sea compatible con versiones anteriores de Android:

```
public class MainActivity extends AppCompatActivity {
    // ... Rest of the code for the class.
}
```

# Comprender el manifiesto de Android

Antes de que el sistema Android pueda iniciar un componente de aplicación como un `Activity`, el sistema debe saber que `Activity`existe. Lo hace leyendo el `AndroidManifest.xml`archivo de la aplicación, que describe todos los componentes de su aplicación de Android. Cada uno `Activity`debe estar incluido en este archivo XML, junto con todos los componentes de la aplicación.

Para ver y editar el `AndroidManifest.xml`archivo, expanda la `manifests`carpeta en el panel **Proyecto > Android** y haga doble clic en `AndroidManifest.xml`. Su contenido aparece en el panel de edición:

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

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        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>
```

## Etiqueta de aplicación y espacio de nombres de Android

El manifiesto de Android está codificado en XML y siempre usa el espacio de nombres de Android:

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

La `package`expresión muestra el nombre de paquete único de la nueva aplicación. No cambie la expresión del paquete después de publicar la aplicación.

La `<application>`etiqueta, con su etiqueta de cierre `</application>`, define la configuración del manifiesto para toda la aplicación.

## Copia de seguridad automática

El `android:allowBackup`atributo habilita la copia de seguridad automática de datos de la aplicación:

```
android:allowBackup="true"
```

Establecer el `android:allowBackup`atributo en `true`permite que la aplicación se respalde automáticamente y se restaure según sea necesario. Los usuarios invierten tiempo y esfuerzo para configurar aplicaciones. Cambiar a un nuevo dispositivo puede cancelar toda esa cuidadosa configuración. El sistema realiza esta copia de seguridad automática para casi todos los datos de la aplicación de forma predeterminada y lo hace sin que el desarrollador tenga que escribir ningún código de aplicación adicional.

Para las aplicaciones cuya versión objetivo del SDK es Android 6.0 (nivel de API 23) y superior, los dispositivos que ejecutan Android 6.0 y superior crean automáticamente copias de seguridad de los datos de la aplicación en la nube porque el `android:allowBackup`atributo se establece de forma predeterminada `true`si se omite. Para las aplicaciones <nivel de API 22, debe agregar explícitamente el `android:allowBackup`atributo y establecerlo en `true`.

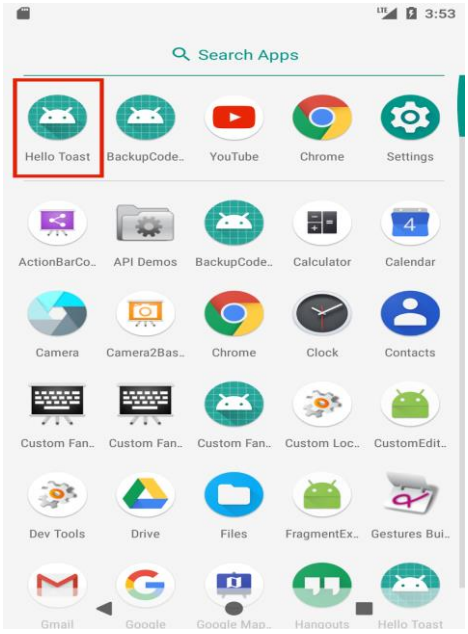
**Sugerencia** : para obtener más información sobre la copia de seguridad automática para aplicaciones, consulte Configuración de la copia de seguridad automática para aplicaciones .

## El ícono de la aplicación

El `android:icon`atributo establece el icono de la aplicación:

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

El `android:icon`atributo asigna a la aplicación un icono en la `mipmap`carpeta (dentro de la `res`carpeta en el panel **Proyecto > Android ).** El icono aparece en la pantalla de inicio o en la pantalla Buscar aplicaciones para iniciar la aplicación. El ícono también se usa como ícono predeterminado para los componentes de la aplicación.




## Etiqueta de aplicación y recursos de cadena

El `android:label`atributo muestra la cadena `"Hello World"`resaltada. Si hace clic en la cadena, cambia para mostrar el recurso de cadena `@string/app_name`:

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

**Sugerencia** : para ver el menú contextual, haga Ctrl-clic o haga clic con el botón derecho `app_name`en el panel del editor. Seleccione **Ir a > Declaración** para ver dónde se declara el recurso de cadena: en el `strings.xml`archivo. Cuando selecciona **Ir a > Declaración** o abre el archivo haciendo doble clic `strings.xml`dentro de la `values`carpeta en el panel **Proyecto > Android** , el contenido del archivo aparece en el panel del editor.

Después de abrir el `strings.xml`archivo, puede ver que el nombre de la cadena `app_name`está establecido en `Hello World`. Puede cambiar el nombre de la aplicación cambiando la `Hello World`cadena a otra cosa. Los recursos de cadena se describen en una lección separada.

### tema de la aplicación

El `android:theme`atributo establece el tema de la aplicación, que define la apariencia de los elementos de la interfaz de usuario, como el texto:

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

El `theme`atributo se establece en el tema estándar `AppTheme`. Los temas se describen en una lección separada.

### Declaración de la versión de Android

Diferentes dispositivos pueden ejecutar diferentes versiones del sistema Android, como Android 4.0 o Android 4.4. Cada versión sucesiva puede agregar nuevas API no disponibles en la versión anterior. Para indicar qué conjunto de API están disponibles, cada versión especifica un nivel de API. Por ejemplo, Android 1.0 es el nivel de API 1 y Android 4.4 es el nivel de API 19.

El nivel de API permite que un desarrollador declare la versión mínima con la que la aplicación es compatible, utilizando la `<uses-sdk>`etiqueta de manifiesto y su `minSdkVersion`atributo. Por ejemplo, las API del proveedor de calendario se agregaron en Android 4.0 (nivel de API 14). Si su aplicación no puede funcionar sin estas API, declare el nivel de API 14 como la versión mínima admitida de la aplicación de esta manera:

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.android.helloworld">
    <uses-sdk android:minSdkVersion="14" android:targetSdkVersion="19" />
    // ... Rest of manifest information
</manifest>
```

El `minSdkVersion`atributo declara la versión mínima para la aplicación y el `targetSdkVersion`atributo declara la versión más alta (más nueva) que se ha optimizado dentro de la aplicación. Cada versión sucesiva de Android brinda compatibilidad con las aplicaciones que se compilaron con las API de versiones anteriores, por lo que la aplicación *siempre* debe ser compatible con versiones futuras de Android mientras usa las API de Android documentadas.

El `targetSdkVersion`atributo *no* impide que una aplicación se instale en versiones de Android superiores (más nuevas) que el valor especificado. Aun así, el atributo de destino es importante porque le indica al sistema si la aplicación debe heredar los cambios de comportamiento en las versiones más recientes.

Si no actualiza `targetSdkVersion`a la última versión, el sistema asume que su aplicación requiere comportamientos compatibles con versiones anteriores cuando se ejecuta en la última versión. Por ejemplo, entre los cambios de comportamiento en Android 4.4, las alarmas creadas con las `AlarmManager`API ahora son inexactas de manera predeterminada para que el sistema pueda procesar alarmas de aplicaciones por lotes y preservar la energía del sistema. Si su nivel de API de destino es inferior a `"19"`, el sistema conserva el comportamiento de la API anterior para su aplicación.

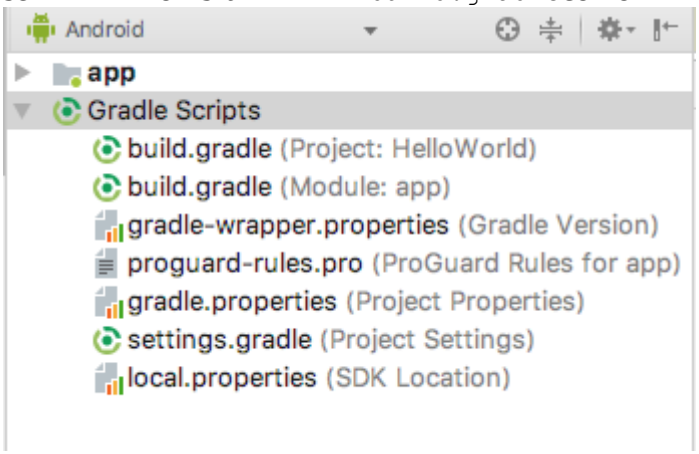
# Entendiendo el proceso de construcción

El paquete de aplicación de Android (APK) es el formato de archivo de paquete para distribuir e instalar aplicaciones móviles de Android. El proceso de compilación involucra herramientas y procesos que convierten automáticamente cada proyecto en un APK.

Android Studio usa Gradle como la base del sistema de compilación, con más capacidades específicas de Android proporcionadas por el complemento de Android para Gradle. Este sistema de compilación se ejecuta como una herramienta integrada desde el menú de Android Studio.

## Entendiendo los archivos build.gradle

Cuando crea un proyecto, Android Studio genera automáticamente los archivos de compilación necesarios en la `Gradle Scripts`carpeta del panel **Proyecto > Android .** Los archivos de compilación de Android Studio se nombran `build.gradle`como se muestra a continuación:




## build.gradle (Proyecto: *apptitle* )

Este archivo es el archivo de compilación de nivel superior para todo el proyecto, ubicado en la carpeta raíz del proyecto, que define las configuraciones de compilación que se aplican a todos los módulos de su proyecto. Este archivo, generado por Android Studio, no debe editarse para incluir dependencias de aplicaciones.

Si una dependencia no es una biblioteca local ni un árbol de archivos, Gradle busca los archivos en los repositorios en línea especificados en el bloque de repositorios de este archivo. De forma predeterminada, los nuevos proyectos de Android Studio declaran JCenter y Google (que incluye el repositorio de Google Maven ) como las ubicaciones del repositorio:

```
allprojects {
    repositories {
        google()
        jcenter()
    }
}
```

## build.gradle (Módulo: aplicación)

Android Studio crea `build.gradle (Module: app)` archivos separados para cada módulo. Puede editar la configuración de compilación para proporcionar opciones de empaquetado personalizadas para cada módulo, como tipos de compilación adicionales y tipos de productos, y anular la configuración en el manifiesto o en el archivo build.gradle de nivel superior. Este archivo suele ser el archivo que se debe editar al cambiar las configuraciones a nivel de la aplicación, como declarar dependencias en la `dependencies`sección. A continuación se muestra el contenido de un `build.gradle (Module: app)` archivo de proyecto:

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

android {
```

```
    compileSdkVersion 26
    defaultConfig {
        applicationId "com.example.android.helloworld"
        minSdkVersion 15
        targetSdkVersion 26
        versionCode 1
        versionName "1.0"
        testInstrumentationRunner
                        "android.support.test.runner.AndroidJUnitRunner"
    }
    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android.txt'),
                                                    'proguard-rules.pro'
        }
    }
}

dependencies {
    implementation fileTree(dir: 'libs', include: ['*.jar'])
    implementation 'com.android.support:appcompat-v7:26.1.0'
    implementation 'com.android.support.constraint:constraint-layout:1.0.2'
    testImplementation 'junit:junit:4.12'
    androidTestImplementation 'com.android.support.test:runner:1.0.1'
    androidTestImplementation
                    'com.android.support.test.espresso:espresso-core:3.0.1'
}
```

Los `build.gradle`archivos utilizan la sintaxis de Gradle. Gradle es un lenguaje específico de dominio (DSL) para describir y manipular la lógica de compilación mediante Groovy , que es un lenguaje dinámico para la máquina virtual Java (JVM). No necesita aprender Groovy para realizar cambios, porque el complemento de Android para Gradle presenta la mayoría de los elementos de DSL que necesita.

**Sugerencia** : para obtener más información sobre el complemento de Android DSL, lea la documentación de referencia de DSL .

### Complementos y bloques de Android

En el `build.gradle (Module: app)` archivo anterior, la primera instrucción aplica las tareas de compilación del complemento de Gradle específicas de Android:

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

android {
   compileSdkVersion 26
   // ... Rest of android block.
}
```

El `android { }`bloque especifica la versión del SDK de destino para compilar el código de la aplicación ( `compileSdkVersion 26`) y varios bloques de información.

### El bloque defaultConfig

La configuración principal y las entradas para la aplicación se especifican en el `defaultConfig { }`bloque dentro del`android { } block`:

```
defaultConfig {
    applicationId "com.example.android.helloworld"
    minSdkVersion 15
    targetSdkVersion 26
    versionCode 1
```

```
    versionName "1.0"
    testInstrumentationRunner
                "android.support.test.runner.AndroidJUnitRunner"
}
```

La configuración `minSdkVersion`y `targetSdkVersion`anula cualquier `AndroidManifest.xml`configuración para la versión mínima del SDK y la versión del SDK de destino. Consulte "Declaración de la versión de Android" anteriormente en este capítulo para obtener información básica sobre esta configuración.

La `testInstrumentationRunner`declaración agrega el soporte de instrumentación para probar la interfaz de usuario usando Espresso y UIAutomator. Estas herramientas se describen en una lección aparte.

**Tipos de compilación**

Los tipos de compilación para la aplicación se especifican en un `buildTypes { }`bloque, que controla cómo se compila y empaqueta la aplicación.

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

El tipo de compilación especificado es `release`para el lanzamiento de la aplicación. Otro tipo de compilación común es `debug`. La configuración de tipos de compilación se describe en una lección separada.

**dependencias**

Las dependencias de la aplicación se definen en el `dependencies { }`bloque, que es la parte del `build.gradle`archivo que es más probable que cambie a medida que comienza a desarrollar código que depende de otras bibliotecas. El bloque es parte de la API de Gradle estándar y pertenece *fuera* del `android { }`bloque.

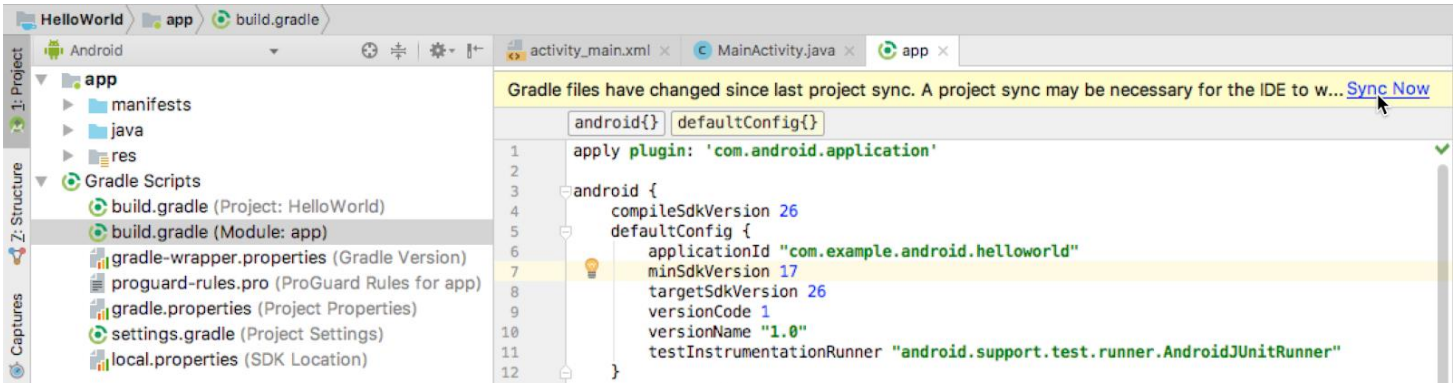
```
dependencies {
    implementation fileTree(dir: 'libs', include: ['*.jar'])
    implementation 'com.android.support:appcompat-v7:26.1.0'
    implementation 'com.android.support.constraint:constraint-layout:1.0.2'
    testImplementation 'junit:junit:4.12'
    androidTestImplementation 'com.android.support.test:runner:1.0.1'
    androidTestImplementation
                    'com.android.support.test.espresso:espresso-core:3.0.1'
}
```

En el fragmento anterior, la declaración `implementation fileTree(dir: 'libs', include: ['*.jar'])`agrega una dependencia de todos los archivos ".jar" dentro de la `libs`carpeta.

## Sincronizando tu proyecto

Cuando realiza cambios en los archivos de configuración de compilación en un proyecto, Android Studio requiere que *sincronice* los archivos del proyecto. Durante la sincronización, Android Studio importa los cambios de configuración de compilación y ejecuta comprobaciones para asegurarse de que la configuración no genere errores de compilación.

Para sincronizar los archivos del proyecto, haga clic en **Sincronizar ahora** en la barra de notificación que aparece al realizar un cambio (como se muestra en la figura a continuación), o haga clic en el botón Sincronizar **proyecto con archivos Gradle** en la barra de menú.




Si Android Studio detecta algún error con la configuración, por ejemplo, si el código fuente usa funciones de API que solo están disponibles en un nivel de API superior al , aparece `compileSdkVersion`la ventana **Mensajes** para describir el problema.

## Ejecutar la aplicación en el dispositivo virtual

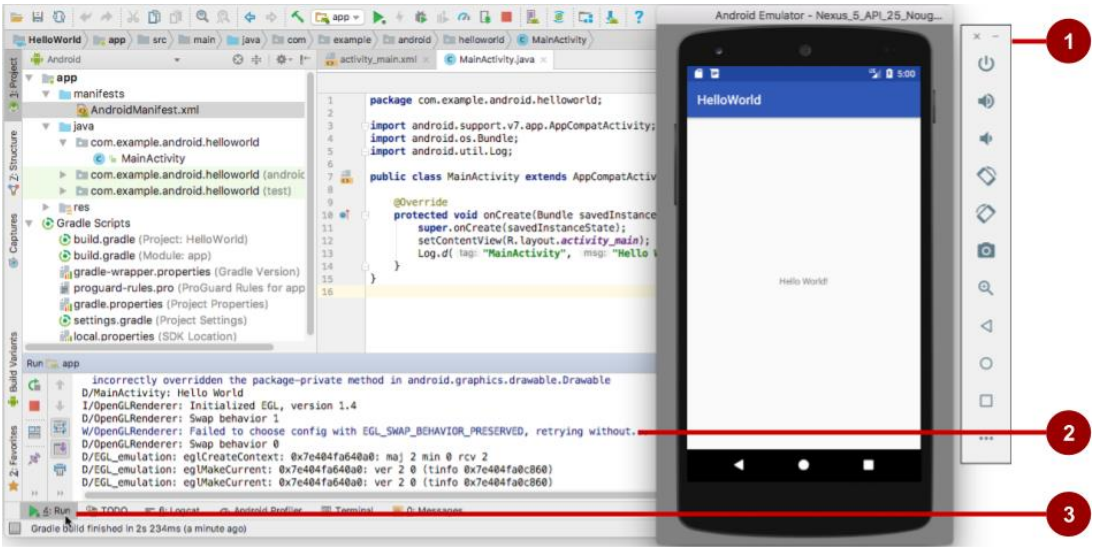
Para ejecutar la aplicación en el dispositivo virtual que creó en la sección anterior, siga estos pasos:

1. En Android Studio, seleccione **Ejecutar > Ejecutar aplicación** o haga clic en el **ícono Ejecutar** en la barra de herramientas.
2. En la ventana Seleccionar destino de implementación, en Emuladores disponibles, seleccione el dispositivo virtual que creó y haga clic en **Aceptar** .

El emulador se inicia y arranca como un dispositivo físico. Dependiendo de la velocidad de su computadora, el proceso de inicio puede demorar un tiempo. La aplicación se compila y, una vez que el emulador está listo, Android Studio carga la aplicación en el emulador y la ejecuta.

Debería ver la aplicación creada a partir de la plantilla Actividad vacía ("Hello World") como se muestra en la siguiente figura, que también muestra el panel **Ejecutar de Android Studio que muestra las acciones realizadas para ejecutar la aplicación en el emulador.**

**Sugerencia** : al realizar pruebas en un dispositivo virtual, es una buena práctica iniciarlo una vez, al comienzo de la sesión. No lo cierre hasta que haya terminado de probar su aplicación, para que su aplicación no tenga que volver a pasar por el proceso de inicio del dispositivo. Para cerrar el dispositivo virtual, seleccione **Salir** en el menú o presione **Control-Q** en Windows o **Comando-Q** en macOS.




La figura anterior muestra el emulador y el registro de ejecución:

1. El emulador que ejecuta la aplicación.
2. El panel **Ejecutar** , que muestra las acciones realizadas para instalar y ejecutar la aplicación.
3. La pestaña **Ejecutar** , en la que hace clic para abrir o cerrar el panel **Ejecutar .**

### Ejecutar la aplicación en un dispositivo físico

Siempre pruebe sus aplicaciones en un dispositivo físico. Si bien los emuladores son útiles, no pueden mostrar todos los estados posibles del dispositivo, como lo que sucede si se produce una llamada entrante mientras se ejecuta la aplicación. Para ejecutar la aplicación en un dispositivo físico, necesita lo siguiente:

- Un dispositivo con Android, como un teléfono o una tableta.
- Un cable de datos para conectar su dispositivo con Android a su computadora a través del puerto USB.
- Si está utilizando un sistema Linux o Windows, es posible que deba realizar pasos adicionales para ejecutar en un dispositivo de hardware. Consulte la documentación Uso de dispositivos de hardware . Es posible que también deba instalar el controlador USB adecuado para su dispositivo. Consulte Controladores USB OEM .

Para permitir que Android Studio se comunique con su dispositivo con Android, debe activar la depuración de USB en el dispositivo. Habilite la depuración de USB en la configuración de **opciones de desarrollador** del dispositivo . (Tenga en cuenta que habilitar la depuración USB no es lo mismo que rootear su dispositivo).

En Android 4.2 y versiones posteriores, la pantalla de **opciones del desarrollador** está oculta de forma predeterminada. Para mostrar las opciones de desarrollador y habilitar la depuración USB:

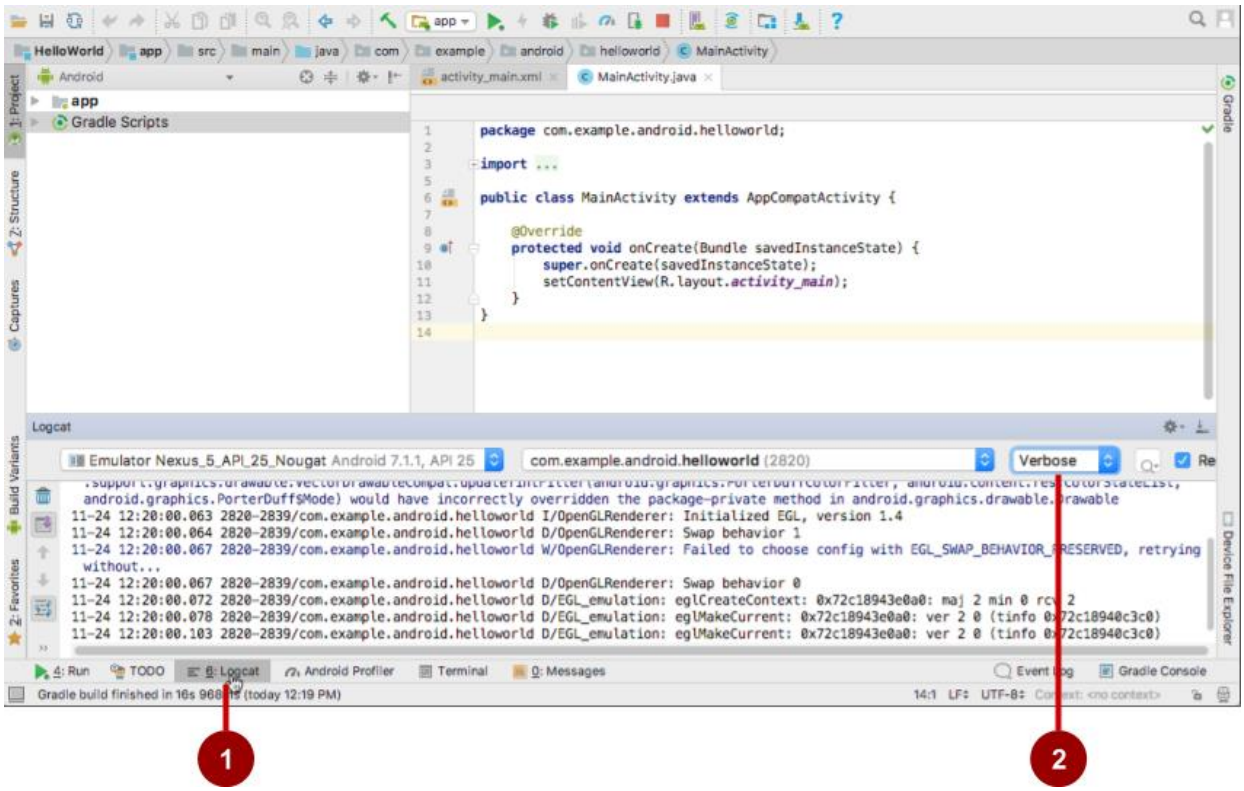
1. En su dispositivo, abra **Configuración > Acerca del teléfono** y toque **Número de compilación** siete veces.
2. Vuelve a la pantalla anterior ( **Configuración** ). **Las opciones de desarrollador** aparecen en la parte inferior de la lista. Presiona **Opciones de desarrollador** .
3. Seleccione **Depuración USB** .
4. Conecte el dispositivo y ejecute la aplicación desde Android Studio.

# Usando el registro

El registro es una poderosa herramienta de depuración que puede usar para buscar valores, rutas de ejecución y excepciones. Después de agregar declaraciones de registro a una aplicación, sus mensajes de registro aparecen junto con mensajes de registro generales en el panel de **Logcat** .

## Visualización de mensajes de registro

Para ver el panel **Logcat , haga clic en la pestaña Logcat** en la parte inferior de la ventana de Android Studio como se muestra en la figura a continuación.




En la figura de arriba:

1. La pestaña **Logcat** para abrir y cerrar el panel **Logcat** , que muestra información sobre su aplicación mientras se ejecuta. Si agrega `Log`declaraciones a su aplicación, los `Log`mensajes aparecen aquí.
2. El `Log`menú de nivel establecido en **Detallado** (predeterminado), que muestra todos los `Log`mensajes. Otras configuraciones incluyen **Depuración** , **Error** , **Información** y **Advertencia** .

## Agregar declaraciones de registro a su aplicación

Las declaraciones de registro agregan cualquier mensaje que especifique al registro. Agregar declaraciones de registro en ciertos puntos del código permite al desarrollador observar valores, rutas de ejecución y excepciones. Por ejemplo, la siguiente declaración de registro agrega `"MainActivity"`y `"Hello World"`al registro:

```
Log.d("MainActivity", "Hello World");
```

Los siguientes son los elementos de esta declaración:

- `Log`: la `Log`clase para enviar mensajes de registro al panel de **Logcat** .
- `d`: la configuración del nivel de **depuración**`Log` para filtrar la visualización de mensajes de registro en el panel de **Logcat** . Otros niveles de registro son `e`para **Error** , `w`para **Advertencia** y `i`para **Información** . Asigne un nivel de registro para que pueda filtrar los mensajes de registro usando el menú desplegable en el centro del panel de **Logcat** .
- `"MainActivity"`: El primer argumento es una etiqueta que se puede usar para filtrar mensajes en el panel de **Logcat** . Esta etiqueta es comúnmente el nombre de `Activity`donde se origina el mensaje. Sin embargo, puede nombrar la etiqueta como cualquier cosa que le resulte útil para la depuración.
- `"Hello world"`: El segundo argumento es el mensaje real.

Por convención, las etiquetas de registro se definen como constantes para `Activity`:

```
private static final String LOG_TAG = MainActivity.class.getSimpleName();
```

Use la constante en las declaraciones de registro:

```
Log.d(LOG_TAG, "Hello World");
```

Después de agregar la `Log.d`declaración que se muestra arriba, siga estos pasos para ver el mensaje de registro:

1. Si el panel **Logcat aún** no está abierto, haga clic en la pestaña **Logcat** en la parte inferior de Android Studio para abrirlo.
2. Cambie el `Log`nivel en el panel de **Logcat** a **Depurar** . (También puede dejar el `Log`nivel como **Verbose** porque hay muy pocos mensajes de registro).
3. Ejecute su aplicación en un dispositivo virtual.

El siguiente mensaje debería aparecer en el panel de **Logcat** :

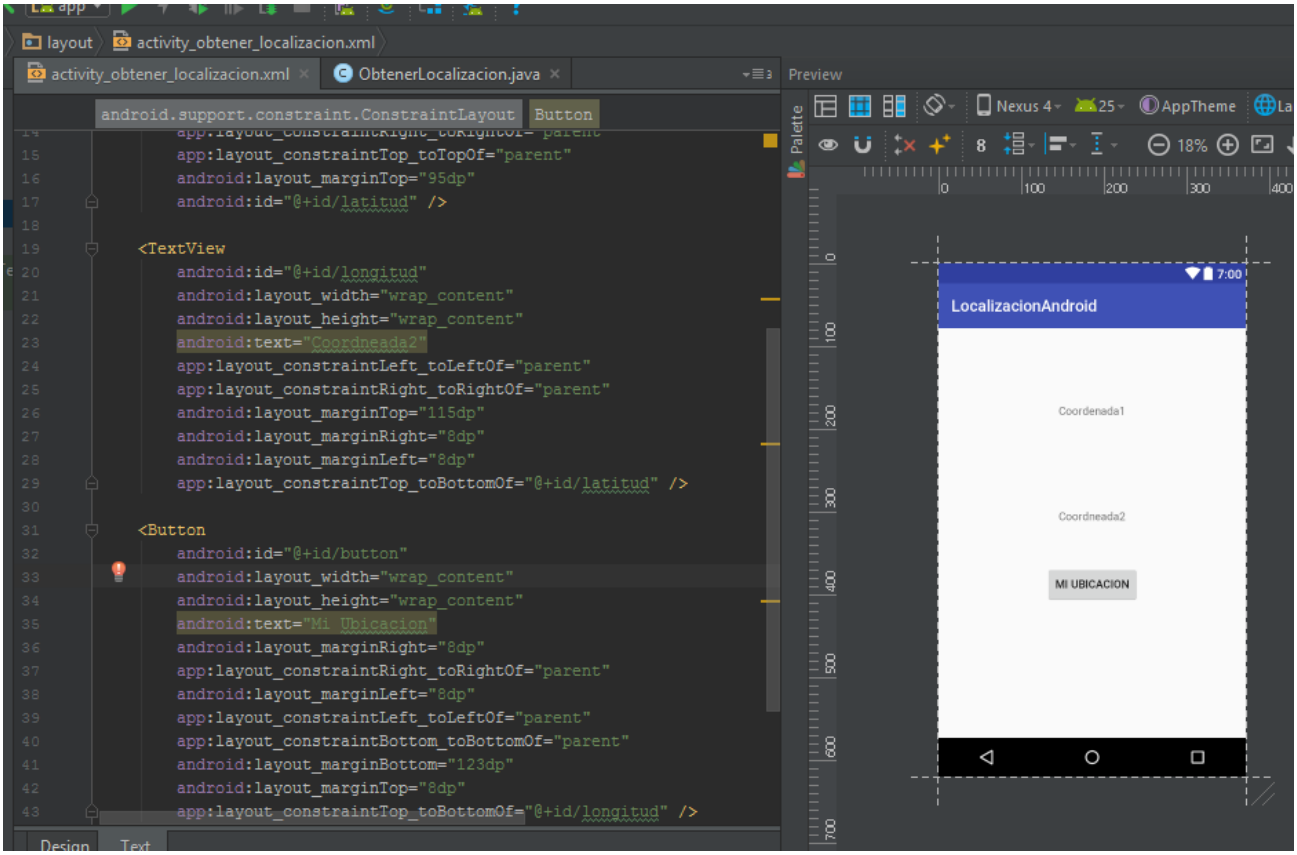
```
11-24 14:06:59.001 4696-4696/? D/MainActivity: Hello World
```

Acceso a los datos de localización o ubicación actual del dispositivo en el que se está ejecutando una aplicación Android.

Obtenemos como siempre las referencias a estos controles en el método `onCreate()` de la actividad principal, y crearemos un método auxiliar `updateUI()` que actualice los campos de latitud y longitud a partir de un objeto `Location`.

1. Creamos un Proyecto llamado **LocationProject**

2. Creamos en el Xml dos TextView y un botón donde mostraremos nuestras ubicación.




3. Referenciamos

```
ObtenerLocalizacion  onCreate()
age com.pomulitos.localizacionandroid;

rt ...

ic class ObtenerLocalizacion extends AppCompatActivity {
Button button;
TextView latitud, longitud;

LocationManager locationManager;
Location location;

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_obtener_localizacion);
    button= (Button) findViewById(R.id.button);
    latitud= (TextView) findViewById(R.id.latitud);
    longitud = (TextView) findViewById(R.id.longitud);
```

Declaramos dos variables **LocationMananger** que me permitirá obtener el Servicio.

Y **Location** que me permitirá obtener las coordenadas.

Ahora obtengo el servicio de localización:

```
    locationManager = (LocationManager) getSystemService(Context.LOCATION_SERVICE);// Obtengo el servicio de localizacion

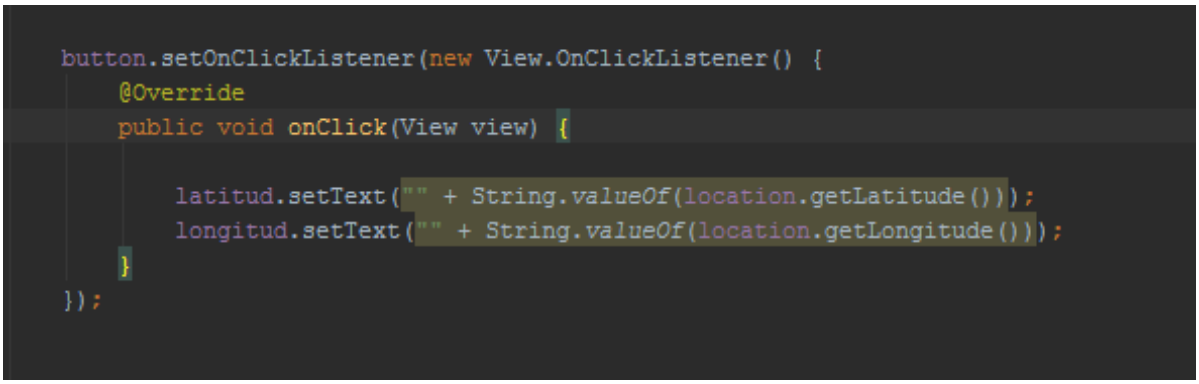
}
```

Ahora almacenamos en la variable location la última localización.

```
location =
locationManager.getLastKnownLocation(LocationManager.NETWORK_PROVIDER);
```

Como nos sale un error porque nos esta solicitando permisos damos clic en el bombillo o ALT + Enter.

Teniendo ya nuestras posiciones las mostraremos en nuestros texview utilizando los métodos del location getLatitude , getLongitude.



```
button.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {

        latitud.setText("" + String.valueOf(location.getLatitude()));
        longitud.setText("" + String.valueOf(location.getLongitude()));
    }
});
```

```
public class MapsActivity extends FragmentActivity implements
OnMapReadyCallback {
    LocationManager locationManager;
    Location location;
    private GoogleMap mMap;
    private Marker marker;
    double latitud = 0.0;
    double longitud = 0.0;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_maps);
        // Obtain the SupportMapFragment and get notified when the map is
ready to be used.
        SupportMapFragment mapFragment = (SupportMapFragment)
getSupportFragmentManager()
                .findFragmentById(R.id.map);
        mapFragment.getMapAsync(this);
    }


    @Override
    public void onMapReady(GoogleMap googleMap) {
        mMap = googleMap;
        miUbicacion();

           }

    public void agregarMarcador(double latitud, double longitud) {

        LatLng coordenada = new LatLng(latitud, longitud);
        CameraUpdate miubicacion =
CameraUpdateFactory.newLatLngZoom(coordenada, 16);
        if (marker != null) marker.remove();
        marker = mMap.addMarker(new MarkerOptions()
                .position(coordenada)
                .title("Mi posicion Actual")

.icon(BitmapDescriptorFactory.fromResource(R.mipmap.ic_launcher))
        );
        mMap.animateCamera(miubicacion);
    }

    public void actualizarUbicacion(Location location) {
        if (location != null) {
            latitud = location.getLatitude();
            longitud = location.getLongitude();
            agregarMarcador(latitud, longitud);
        }
```

```
    }

    LocationListener locationListener = new LocationListener() {
        @Override
        public void onLocationChanged(Location location) {
            actualizarUbicacion(location);

        }

        @Override
        public void onStatusChanged(String s, int i, Bundle bundle) {

        }

        @Override
        public void onProviderEnabled(String s) {

        }

        @Override
        public void onProviderDisabled(String s) {

        }
    };


    private void miUbicacion() {


        if (ActivityCompat.checkSelfPermission(this,
android.Manifest.permission.ACCESS_FINE_LOCATION) !=
PackageManager.PERMISSION_GRANTED &&
ActivityCompat.checkSelfPermission(this,
android.Manifest.permission.ACCESS_COARSE_LOCATION) !=
PackageManager.PERMISSION_GRANTED) {

            return;
        }
        locationManager = (LocationManager)
getSystemService(Context.LOCATION_SERVICE);
        location =
locationManager.getLastKnownLocation(LocationManager.NETWORK_PROVIDER);
        actualizarUbicacion(location);

locationManager.requestLocationUpdates(LocationManager.GPS_PROVIDER,15000
,0,locationListener);

     }
}
```

# MAPS ANDROID

Instructor Gustavo Salazar Escobar



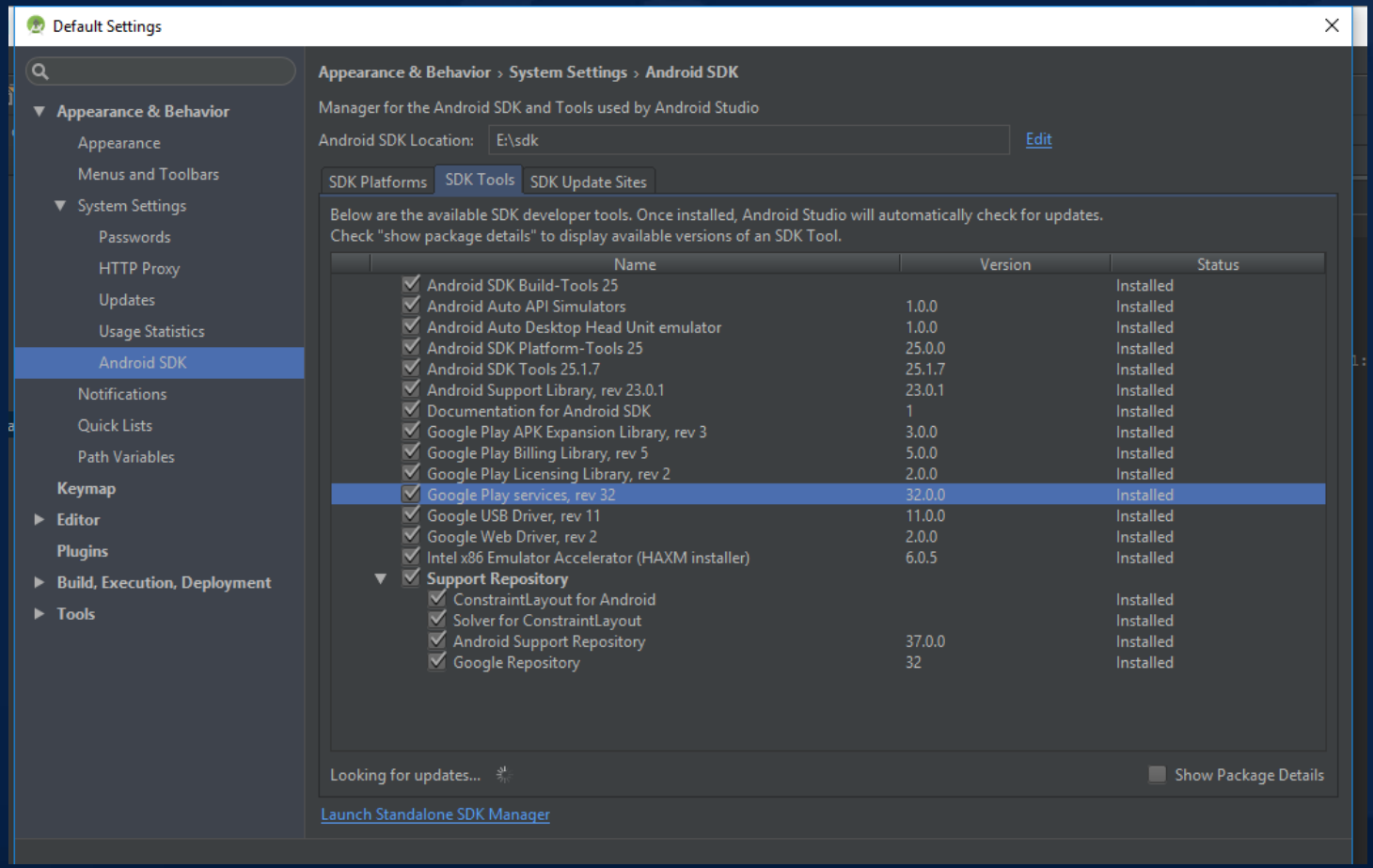
ANDROID

# Que necesitamos?

- 1. Verificar instalacion Google play services:
- 2.MapsActivity
- 2.Key(Consola Developer)
- 3.Permisos
- 4.Coordenadas



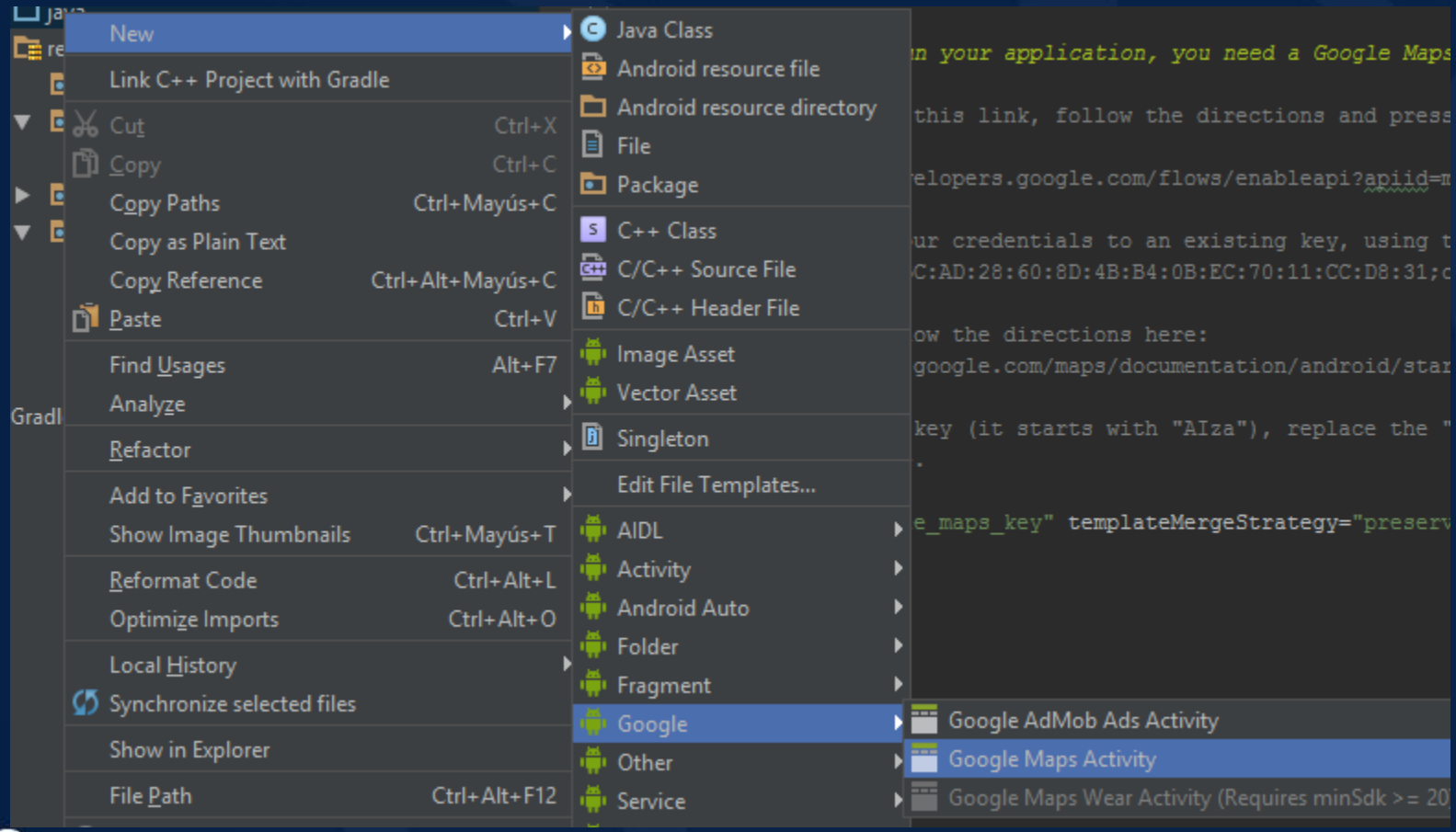
# Ejecutamos nuestro SDK MANANGER





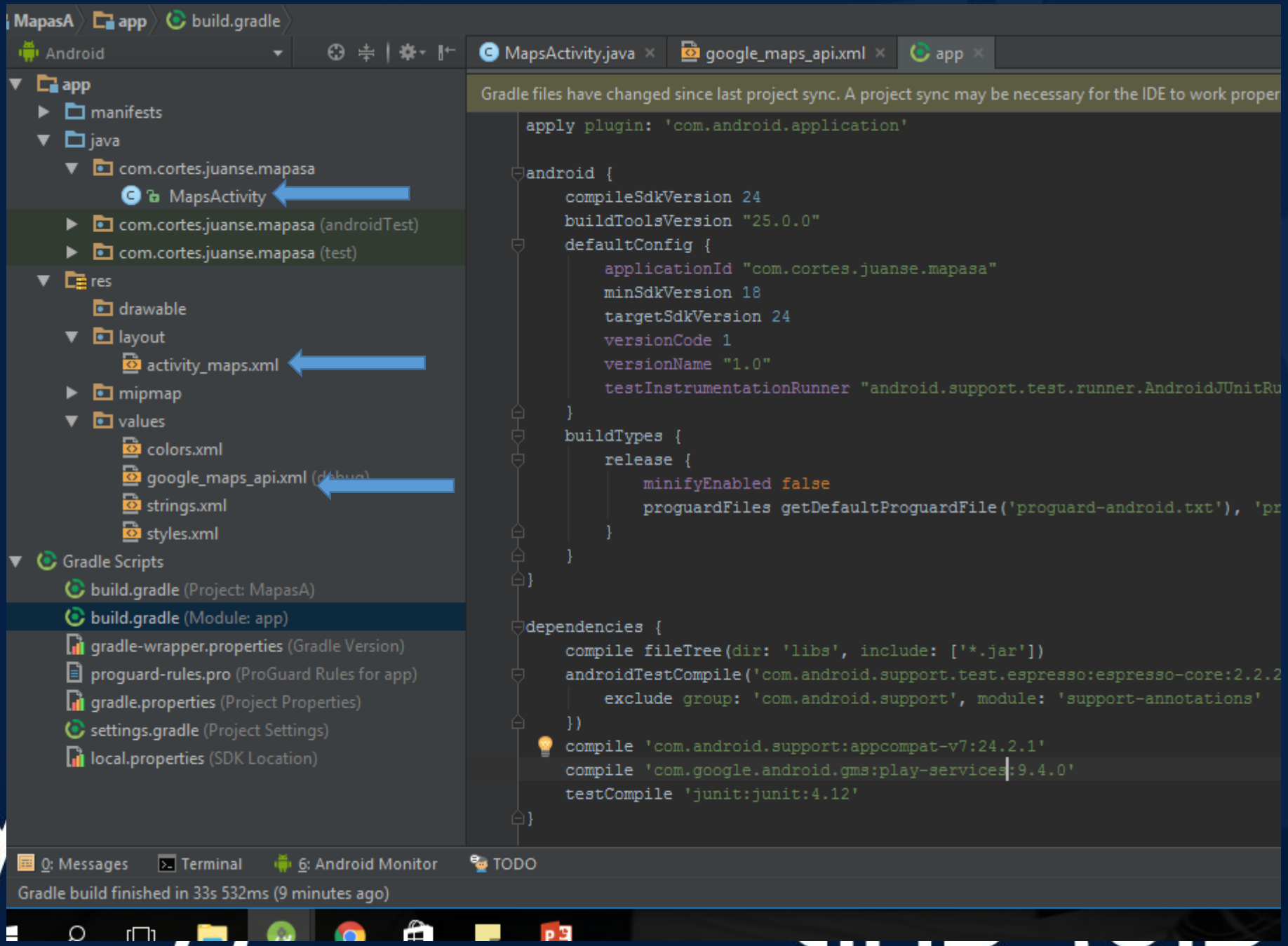

android

# Crear Google MapsActivity

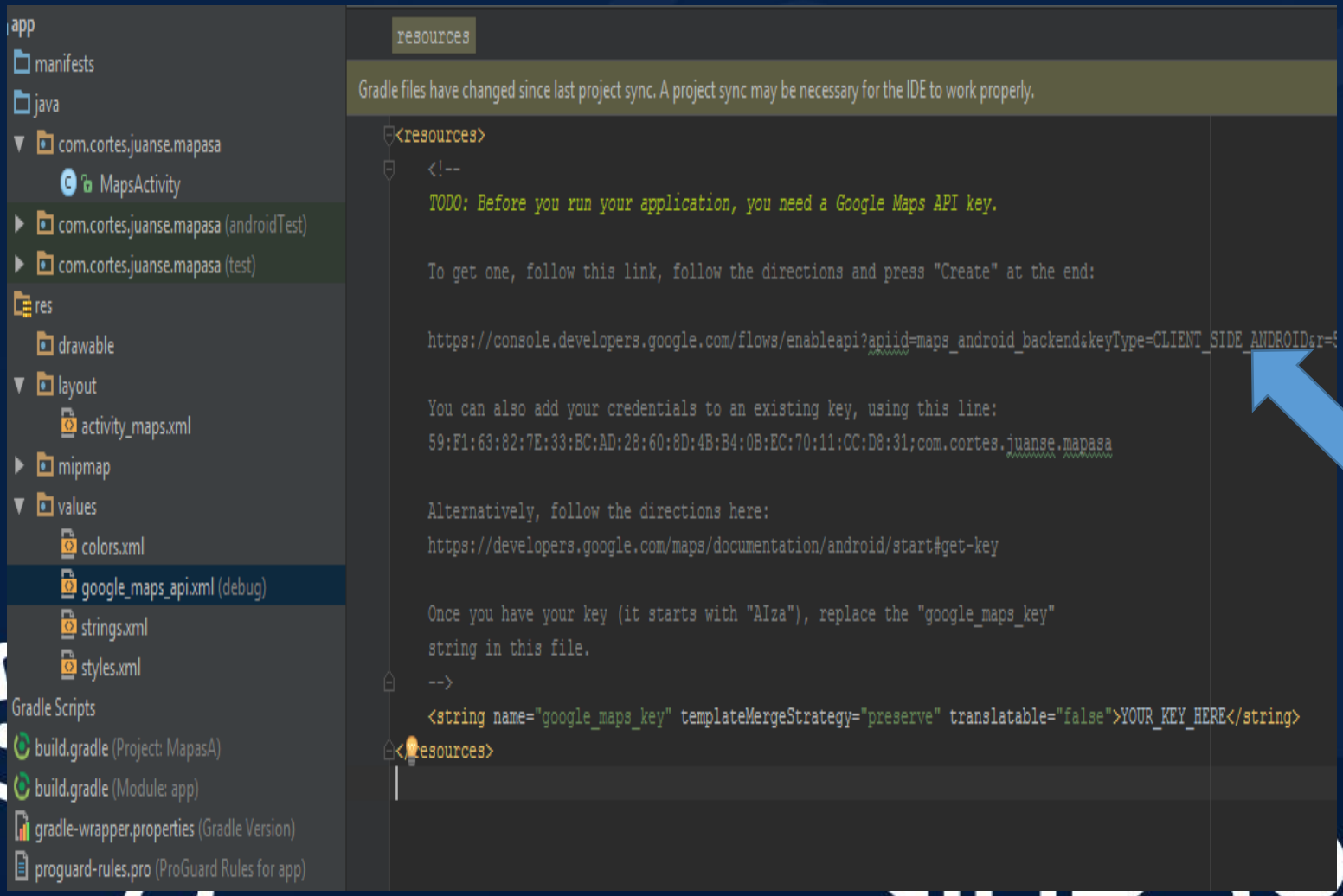




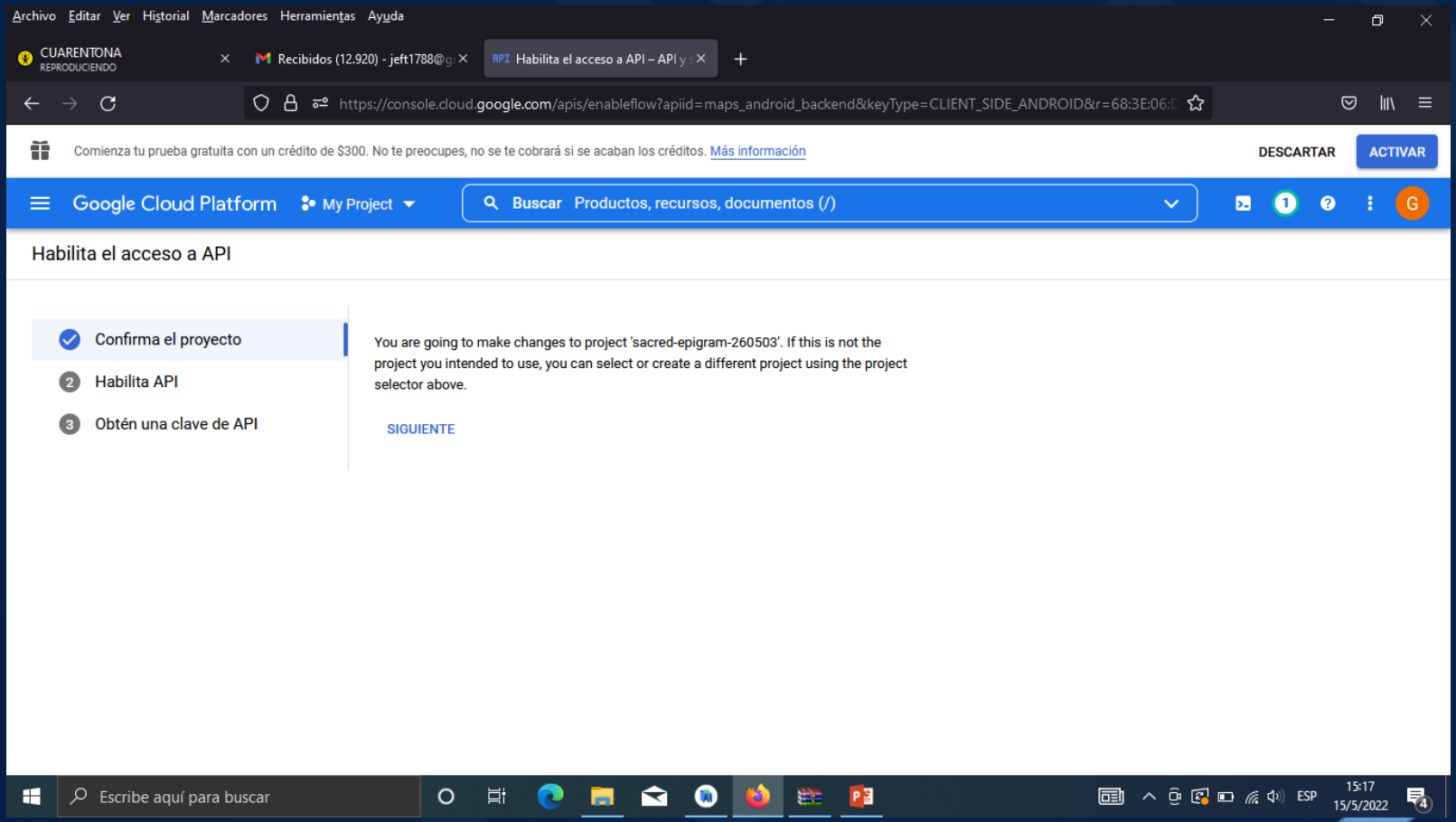

ANDROID


MapasA
app
build.gradle
Android
MapsActivity.java
google_maps_api.xml
app
Gradle files have changed since last project sync. A project sync may be necessary for the IDE to work proper
app
manifests
java
com.cortes.juanse.mapasa
MapsActivity
com.cortes.juanse.mapasa (androidTest)
com.cortes.juanse.mapasa (test)
res
drawable
layout
activity_maps.xml
mipmap
values
colors.xml
google_maps_api.xml
strings.xml
styles.xml
Gradle Scripts
build.gradle (Project: MapasA)
build.gradle (Module: app)
gradle-wrapper.properties (Gradle Version)
proguard-rules.pro (ProGuard Rules for app)
gradle.properties (Project Properties)
settings.gradle (Project Settings)
local.properties (SDK Location)
apply plugin: 'com.android.application'
android {
compileSdkVersion 24
buildToolsVersion "25.0.0"
defaultConfig {
applicationId "com.cortes.juanse.mapasa"
minSdkVersion 18
targetSdkVersion 24
versionCode 1
versionName "1.0"
testInstrumentationRunner "android.support.test.runner.AndroidJUnitRu
}
buildTypes {
release {
minifyEnabled false
proguardFiles getDefaultProguardFile('proguard-android.txt'), 'pr
}
}
}
dependencies {
compile fileTree(dir: 'libs', include: ['*.jar'])
androidTestCompile('com.android.support.test.espresso:espresso-core:2.2.2
exclude group: 'com.android.support', module: 'support-annotations'
})
compile 'com.android.support:appcompat-v7:24.2.1'
compile 'com.google.android.gms:play-services:9.4.0'
testCompile 'junit:junit:4.12'
}
0: Messages
Terminal
6: Android Monitor
TODO
Gradle build finished in 33s 532ms (9 minutes ago)

# Abrimos google_maps_api y copiamos el enlace para crear key

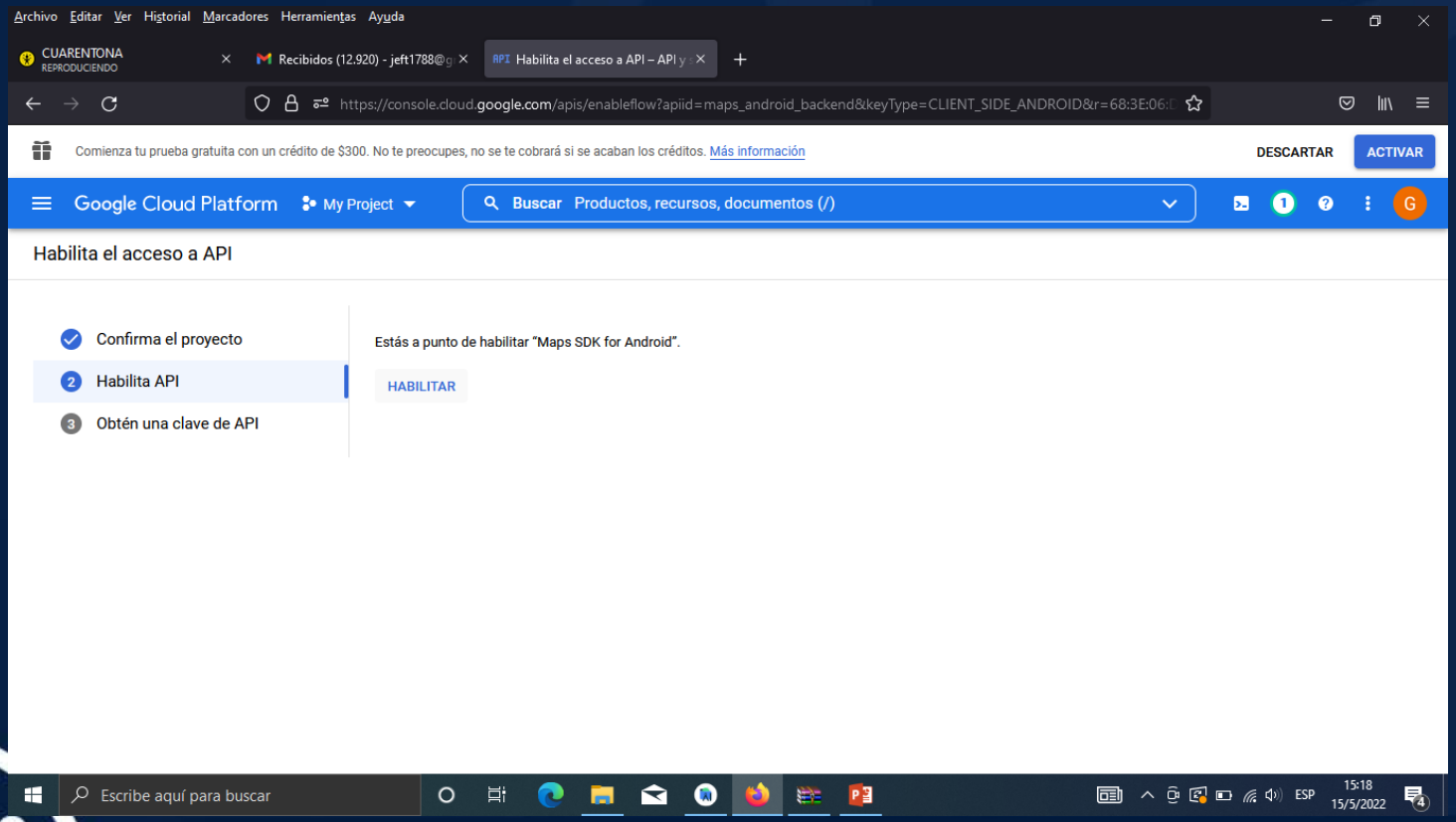


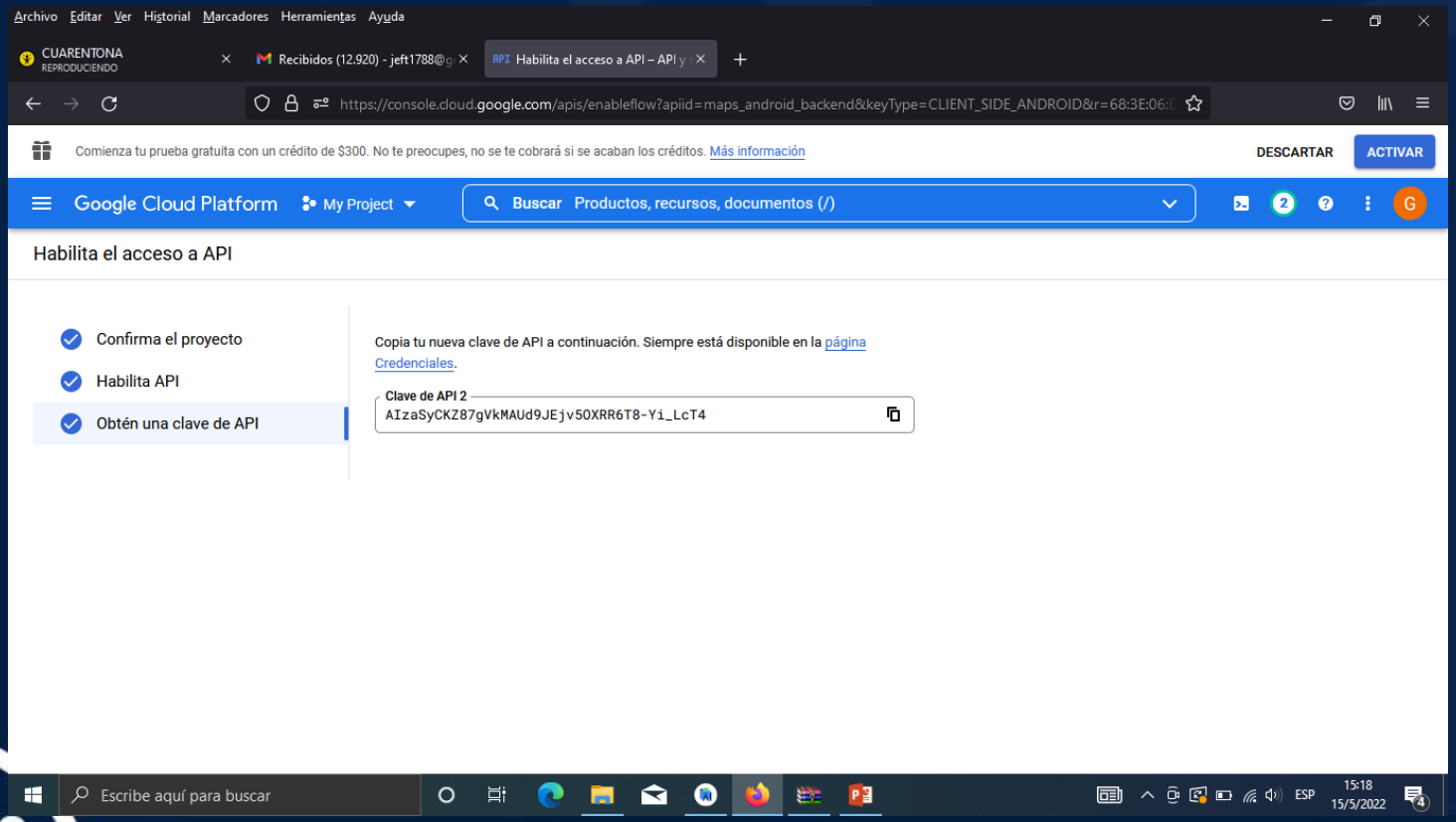
# Crear Proyecto



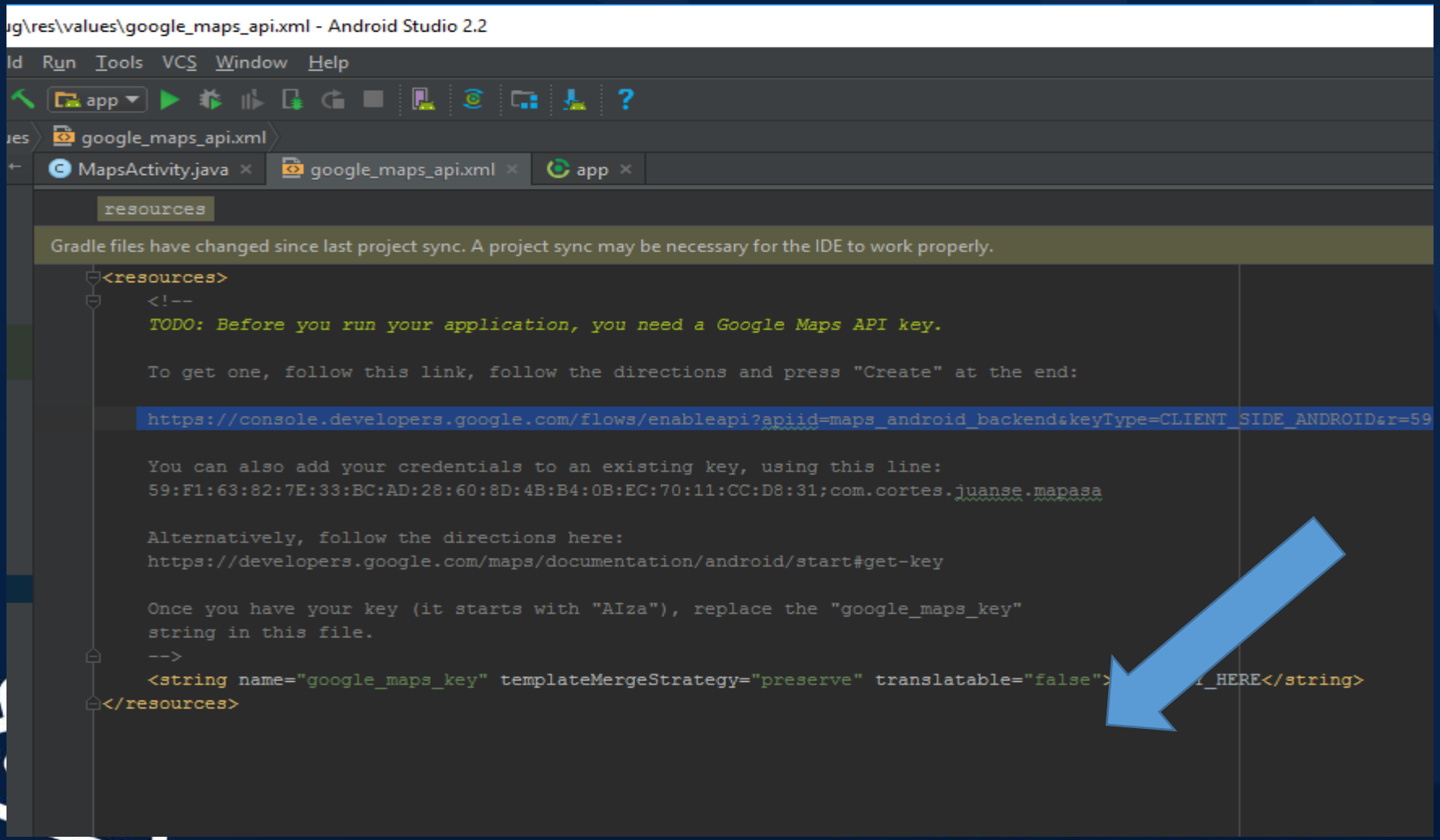






Archivo Editar Ver Historial Marcadores Herramientas Ayuda
CUARENTONA
REPRODUCIENDO
Recibidos (12.920) - jeft1788@g
Habilita el acceso a API – API y
https://console.cloud.google.com/apis/enableflow?apiid=maps_android_backend&keyType=CLIENT_SIDE_ANDROID&r=68:3E:06:
Comienza tu prueba gratuita con un crédito de $300. No te preocupes, no se te cobrará si se acaban los créditos. Más información
DESCARTAR
ACTIVAR
Google Cloud Platform
My Project
Buscar Productos, recursos, documentos (/)
Habilita el acceso a API
Confirma el proyecto
Habilita API
Obtén una clave de API
Estás a punto de habilitar "Maps SDK for Android".
HABILITAR
Escribe aquí para buscar
ESP
15:18
15/5/2022


Archivo Editar Ver Historial Marcadores Herramientas Ayuda
CUARENTONA
REPRODUCIENDO
Recibidos (12.920) - jeft1788@g
Habilita el acceso a API – API y
https://console.cloud.google.com/apis/enableflow?apiid=maps_android_backend&keyType=CLIENT_SIDE_ANDROID&r=68:3E:06:
Comienza tu prueba gratuita con un crédito de $300. No te preocupes, no se te cobrará si se acaban los créditos. Más información
DESCARTAR
ACTIVAR
Google Cloud Platform
My Project
Buscar Productos, recursos, documentos (/)
Habilita el acceso a API
Confirma el proyecto
Habilita API
Obtén una clave de API
Copia tu nueva clave de API a continuación. Siempre está disponible en la página Credenciales.
Clave de API 2
AIzaSyCKZ87gVkMAUd9JEjv5OXRR6T8-Yi_LcT4
Escribe aquí para buscar
ESP
15:18
15/5/2022

# PEGAR API-KEY



# IR ACTIVITY MAPS

```
     */
    @Override
    public void onMapReady(GoogleMap googleMap) {
        mMap = googleMap;

        // Add a marker in Sydney and move the camera
        LatLng sydney = new LatLng(-34, 151);
        mMap.addMarker(new MarkerOptions().position(sydney).title("Marker in Sydney"));
        mMap.moveCamera(CameraUpdateFactory.newLatLng(sydney));
    }
}
```





# TIPOS DE MAPAS

- **Normal**
- **Híbrido**
- **Tierra**
- **Ninguno**



ANDROID

# PROPIEDADES

```
GoogleMap map;
...
// Sets the map type to be "hybrid"
map.setMapType(GoogleMap.MAP_TYPE_HYBRID);
```



ANDROID

# Icono marcador

```
private static final LatLng MELBOURNE = new LatLng(-37.813, 144.962);
  private Marker melbourne = mMap.addMarker(new MarkerOptions()
                        .position(MELBOURNE)
                        .title("Melbourne")
                        .snippet("Population: 4,137,400")
                        .icon(BitmapDescriptorFactory.fromResource(R.drawable.arrow)));
```



