

Tema 2: Introducción a ▶ Android

Laura Sacristán Matesanz

Objetivos del tema

Conocer los fundamentos de las aplicaciones en Android

Distinguir los componentes de una aplicación

Utilidad del fichero Android Manifest

Identificar y usar con fluidez los recursos de una aplicación

Instalar y configurar Android Studio como entorno de desarrollo

Comprender el ciclo de vida de una aplicación

Identificar y manejar los métodos asociados al ciclo de vida de una actividad

Realizar la primera aplicación Android

Fundamentos de una aplicación Android

- ▶ Sistema operativo multiusuario basado en Linux
- ▶ Las aplicaciones pueden solicitar permiso para acceder a datos, recursos o funcionalidades del dispositivo. Se solicita en el momento de la instalación y los otorga el usuario.
- ▶ Hasta ahora el lenguaje de programación era Java apoyado con XML pero a día de hoy se utiliza más Kotlin.

Componentes de una aplicación I

- ▶ Activity (actividades): componente principal. Se encarga de gestionar las interacciones con el usuario. Contiene toda la lógica de aplicación.
- ▶ View: componentes básicos con los que se construye la interfaz gráfica de la aplicación.
 - ▶ Android pone a nuestra disposición una gran cantidad de controles básicos, como cuadros de texto, botones, listas desplegables o imágenes, permitiéndonos también crear nuestros propios controles.

Componentes de una aplicación II

- ▶ Service (servicios): son componentes sin interfaz gráfica que se ejecutan en segundo plano.
 - ▶ pueden realizar cualquier tipo de acciones, por ejemplo actualizar datos, lanzar notificaciones, o incluso mostrar elementos visuales, si la aplicación necesita en algún momento interactuar con el usuario
- ▶ Intent: son los encargados de las comunicaciones internas y externas de la aplicación. Pueden lanzar activitys, o pedir a otras aplicaciones externas que ejecuten una acción (ej. que se abra una galería de fotos).
 - ▶ Mediante un intent se puede ejecutar una activity desde cualquier otra, iniciar un servicio, enviar un mensaje broadcast, iniciar otra aplicación, etc.
 - ▶ Cuando se crea un nuevo proyecto, el sistema incluye en el fichero AndroidManifest.xml, una referencia a la clase MainActivity que contiene un intent (<intent-filter>) indicando que esa es la clase principal (clase Main) y que debe ejecutarse cuando se inicie el proyecto.

Componentes de una aplicación III

- ▶ Content provider (proveedor de contenido): comparte datos entre aplicaciones. Mediante estos componentes es posible compartir determinados datos de nuestra aplicación sin mostrar detalles sobre su almacenamiento interno, su estructura, o su implementación.
- ▶ Broadcast receiver (emisores de notificaciones): reaccionan a intents específicos y pueden ejecutar acciones. Por ejemplo, lanzar una determinada activity o devolver otro intent al sistema para que siga el proceso.
 - ▶ También pueden detectar y reaccionar ante determinados mensajes o eventos globales generados por el sistema (por ejemplo: “Batería baja”, “SMS recibido”, “Tarjeta SD insertada”) o por otras aplicaciones

Las aplicaciones Android están compuestas por una o mas de estos componentes

Instalación Android Studio

Instalación Android Studio I

- ▶ Fue presentado en 2013 y reemplazó a Eclipse como IDE oficial.
- ▶ Se basa en IntelliJ IDEA de JetBrains.
- ▶ Es licencia libre y se puede obtener para cualquier sistema operativo.
- ▶ Su instalación es muy sencilla y se puede hacer directamente desde la página oficial de Android Studio.
- ▶ Es conveniente instalar el emulador AVD (Android Virtual Device). No es obligatorio pero si recomendable.
- ▶ También preguntará la ubicación del entorno. Es conveniente ponerlo en un lugar de fácil acceso en unidades rápidas.
- ▶ Al finalizar la instalación hará una revisión y actualización de algunos componentes necesarios para el desarrollo (SDK, HAXM o AVD entre otros)

Instalación Android Studio II

BUSCAR INFORMACIÓN SOBRE LAS NECESIDADES
HARDWARE DE ANDROID STUDIO.

CUANTA MEMORIA RAM NECESITAMOS PARA INSTALAR LA
ÚLTIMA VERSIÓN.

¿HAY ALGUNA NECESIDAD MÁS SI INSTALAMOS EL
EMULADOR?

¿Y DE ESPACIO EN DISCO? ¿CUÁNTO SE NECESITA?

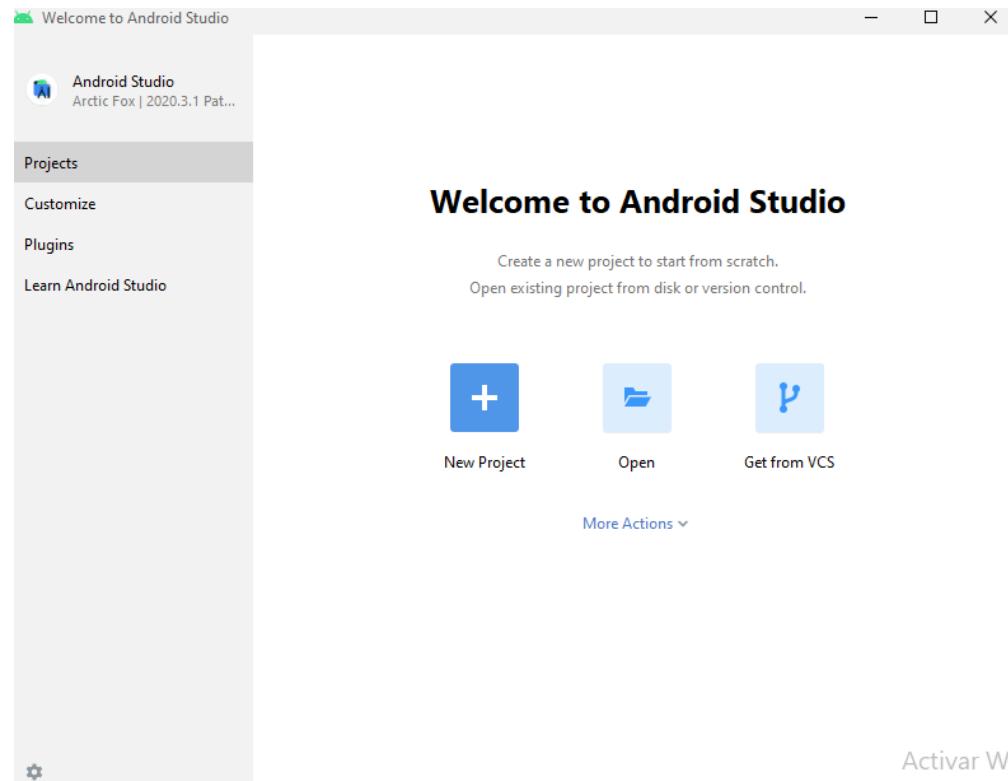
¿HAY ALGÚN SOFTWARE MÁS QUE NECESITEMOS
INSTALAR?

ESPERO VUESTRAS RESPUESTAS

Primer proyecto Android

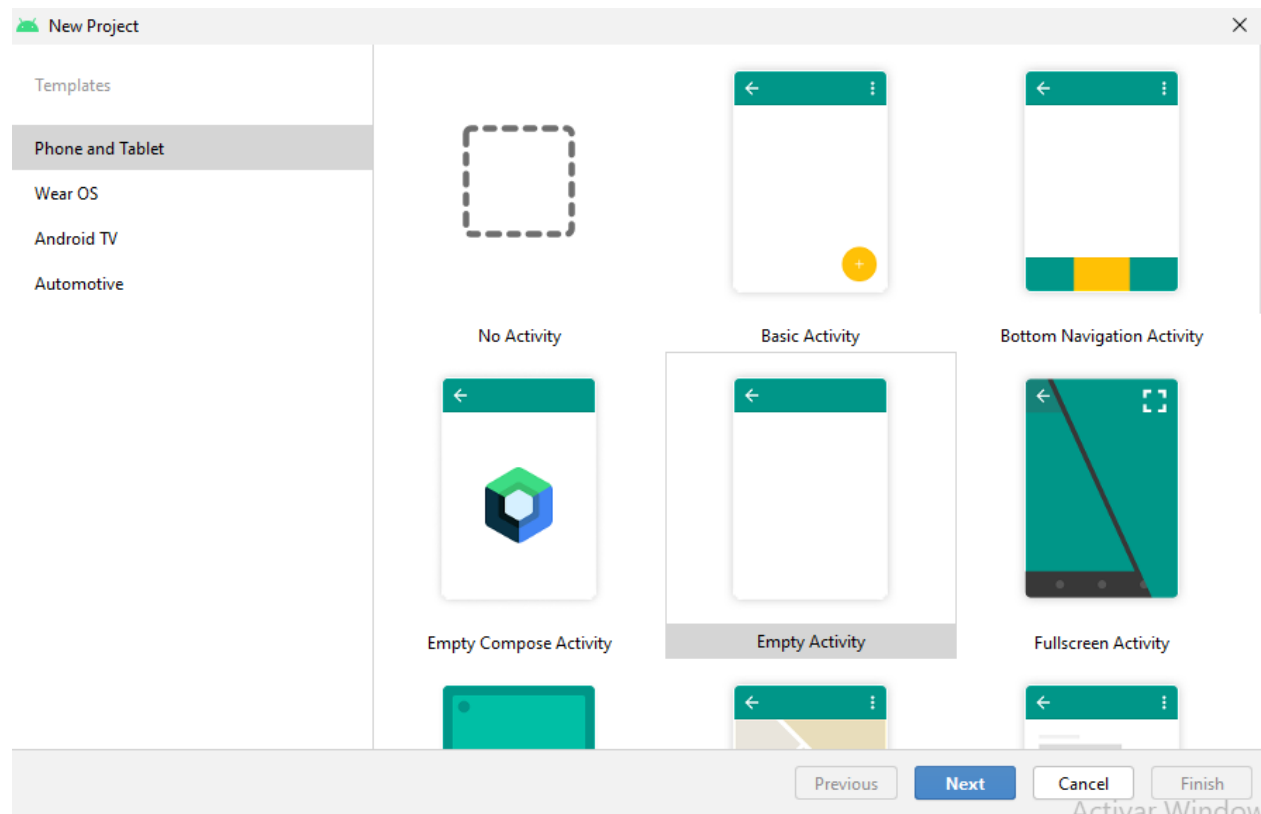
Iniciar un nuevo proyecto I

- ▶ Cuando iniciamos el por primera vez Android Studio o cuando cerremos el proyecto activo nos aparecerá la siguiente pantalla



Iniciar un nuevo proyecto III

- Elegimos la plantilla que vamos a utilizar para nuestra aplicación.



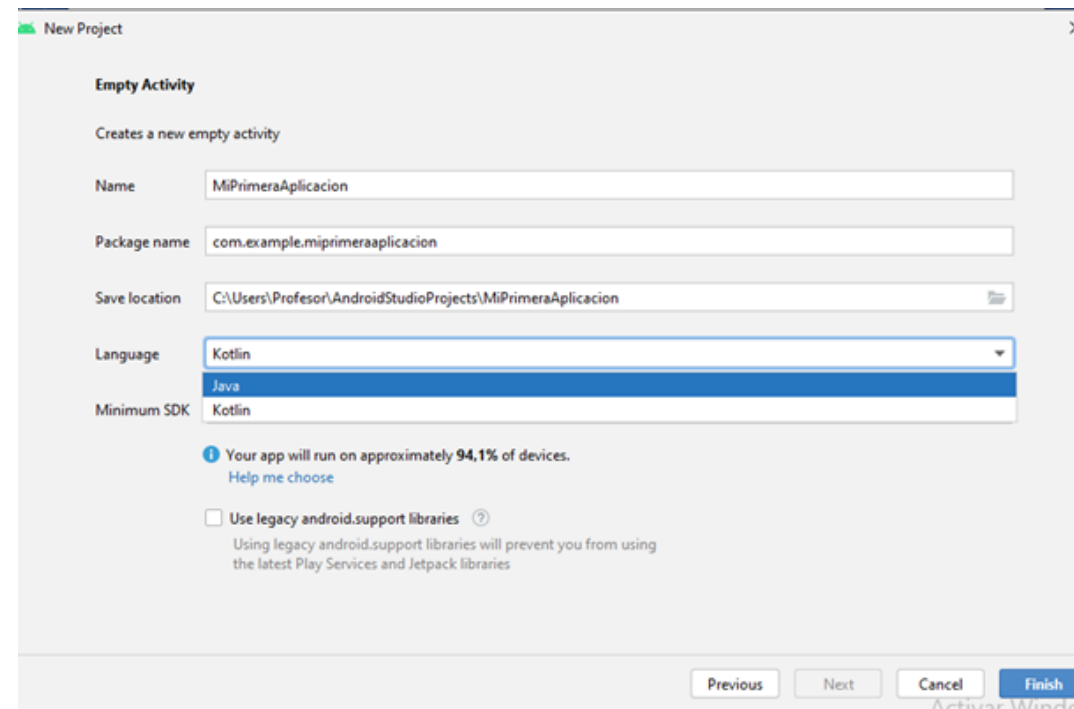
Iniciar un nuevo proyecto III

Si queremos abrir un nuevo proyecto teniendo uno activo ejecutaremos:

- ▶ File → New → New Project...

Escribimos el nombre de nuestra aplicación y la ruta en la que va a almacenarse y pulsamos Next.

Nos aparece la pantalla:

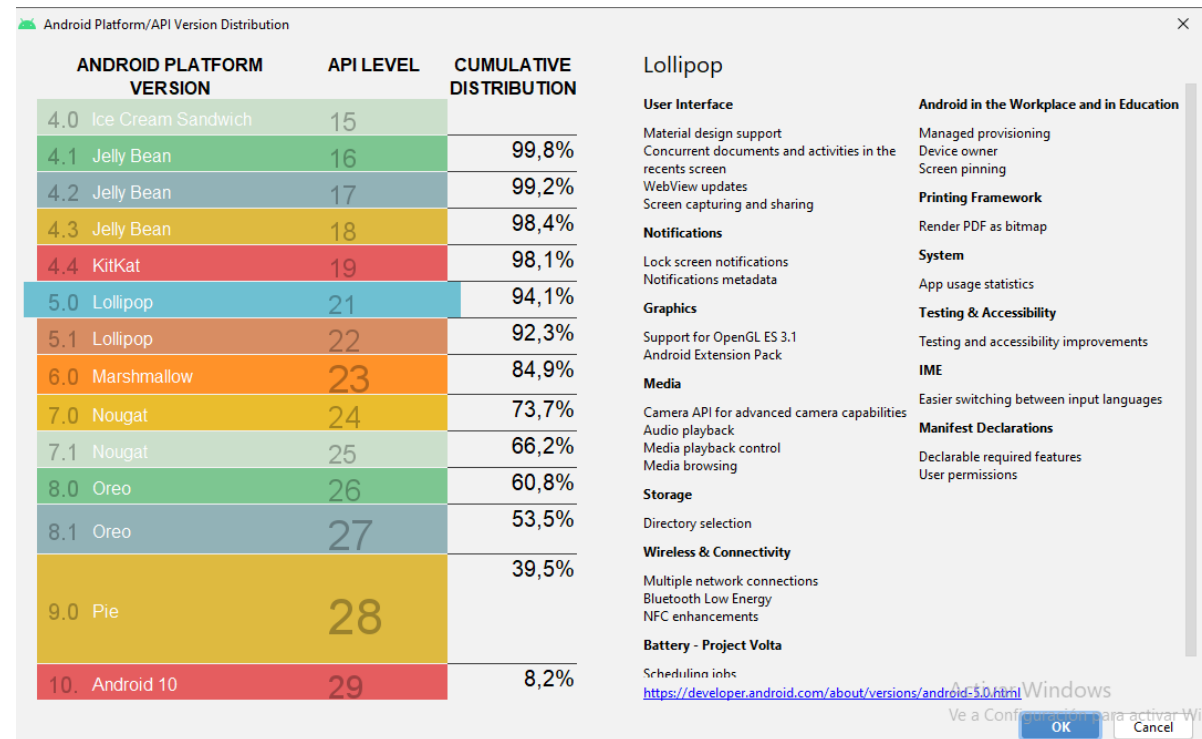


The screenshot shows the 'New Project' dialog box in Android Studio. The dialog is titled 'New Project' and has a close button (X) in the top right corner. It contains the following fields and options:

- Empty Activity**: A section header.
- Creates a new empty activity**: A descriptive text.
- Name**: A text field containing 'MiPrimeraAplicacion'.
- Package name**: A text field containing 'com.example.miprimeraaplicacion'.
- Save location**: A text field containing 'C:\Users\Profesor\AndroidStudioProjects\MiPrimeraAplicacion'.
- Language**: A dropdown menu with 'Kotlin' selected. A blue highlight is visible over the 'Java' option.
- Minimum SDK**: A dropdown menu with 'Kotlin' selected.
- Information**: A blue information icon followed by the text 'Your app will run on approximately 94,1% of devices.' and a link 'Help me choose'.
- Use legacy android.support libraries**: A checkbox that is currently unchecked, followed by a question mark icon and a note: 'Using legacy android.support libraries will prevent you from using the latest Play Services and Jetpack libraries'.
- Navigation buttons**: At the bottom, there are four buttons: 'Previous' (disabled), 'Next' (disabled), 'Cancel' (disabled), and 'Finish' (active).

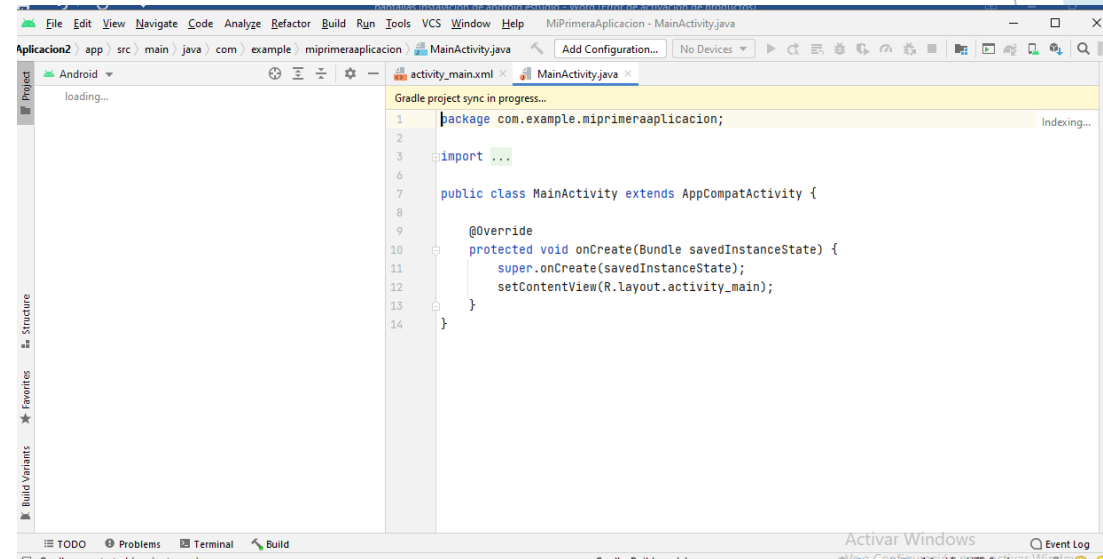
Iniciar un nuevo proyecto IV

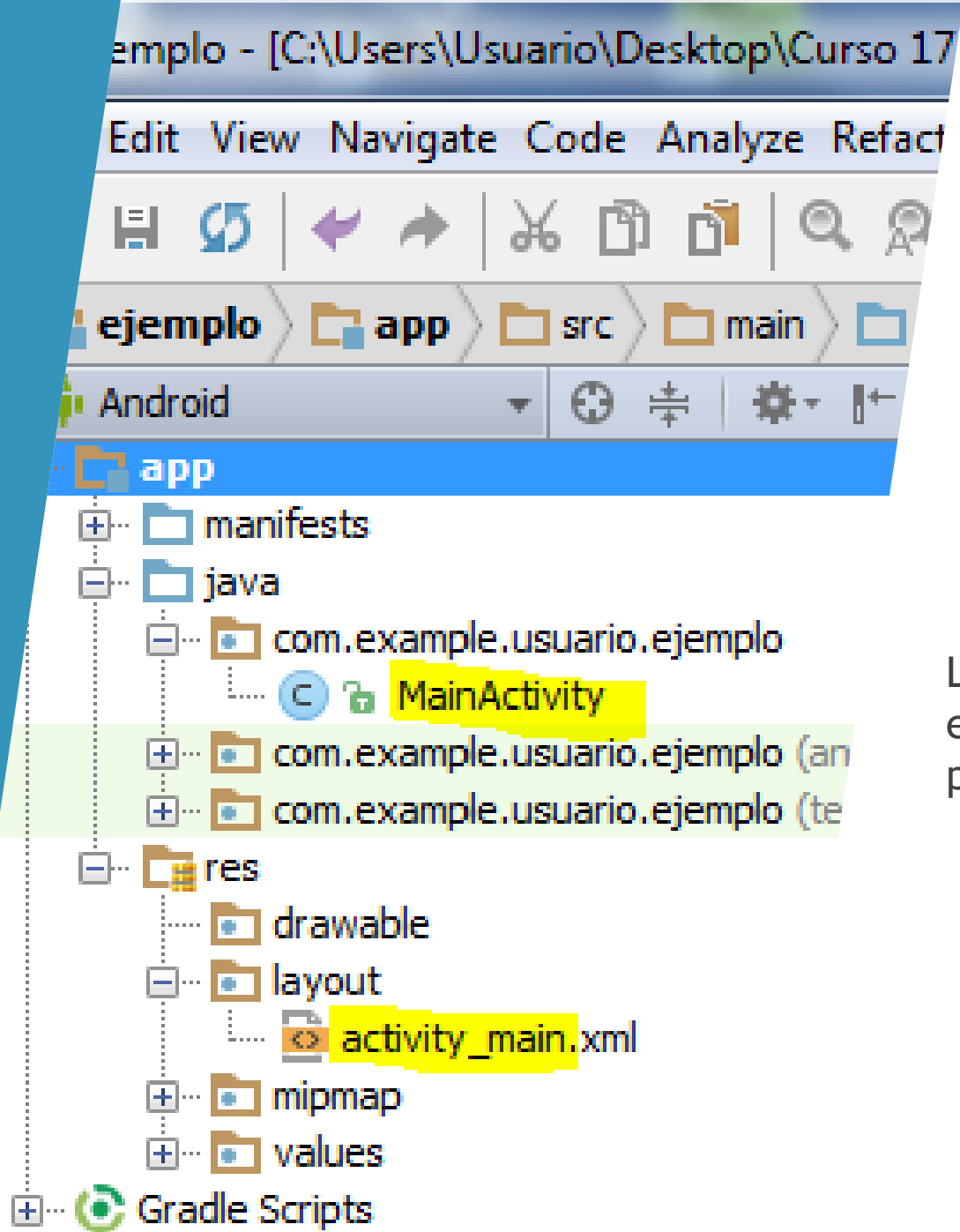
- ▶ En la pantalla anterior elegimos el/los dispositivos en los que se va descargar nuestra aplicación y la versión mínima de Android que puede soportarla.
- ▶ La opción Help me choose nos muestra el porcentaje de dispositivos que incluyen la versión mínima elegida.



Iniciar un nuevo proyecto V

- ▶ Posteriormente nos muestra el nombre de la actividad principal y el de su layout asociado, que podemos modificar.
- ▶ Pulsamos Finish.
- ▶ Para ver el contenido del proyecto:
 - ▶ View --> Tool Windows --> Project





Iniciar un nuevo proyecto VI

Lo que veremos una vez hechos todos estos pasos será la siguiente pantalla.

Iniciar un nuevo proyecto V

Observamos que, en principio, nuestro proyecto consta de dos documentos importantes y que abre directamente:

- ▶ MainActivity.java: que es un programa java donde se creará nuestro programa principal
- ▶ activity_main.xml: Contiene un documento XML, denominado layout, con información sobre lo que se va a mostrar visualmente en la pantalla del dispositivo.

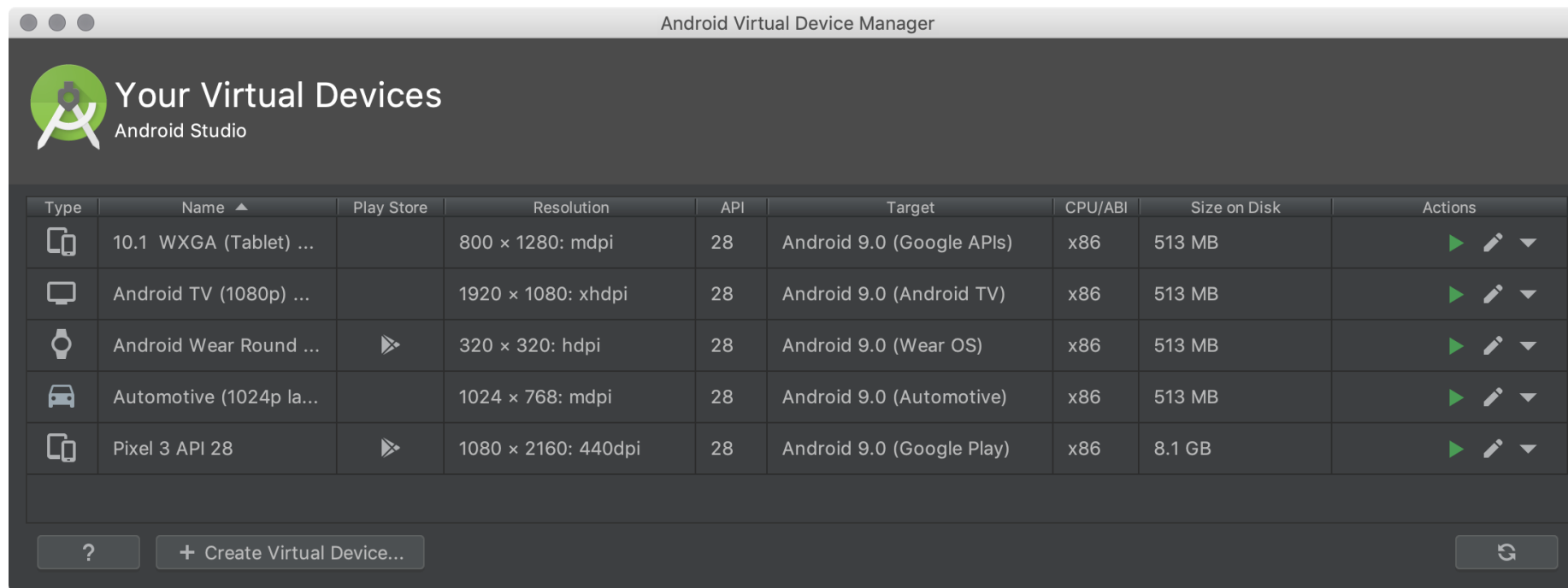
Uso del AVD Manager

Uso del AVD Manager I

- ▶ Un dispositivo virtual de Android (AVD) es una configuración que define las características de un teléfono o una tablet Android, o de un dispositivo Wear OS, Android TV o Automotive OS, que desees simular en Android Emulator.
- ▶ El Administrador de AVD es una interfaz que puedes iniciar desde Android Studio y te permite crear y administrar los AVD.
- ▶ Para abrir el Administrador de AVD, realiza una de las siguientes acciones:
 - ▶ Selecciona Tools > AVD Manager.
 - ▶ En la barra de herramientas, haz clic en AVD Manager Ícono del Administrador de AVD.

Uso del AVD Manager II

- ▶ Cuando accedemos al AVD Manager nos aparece la siguiente ventana:

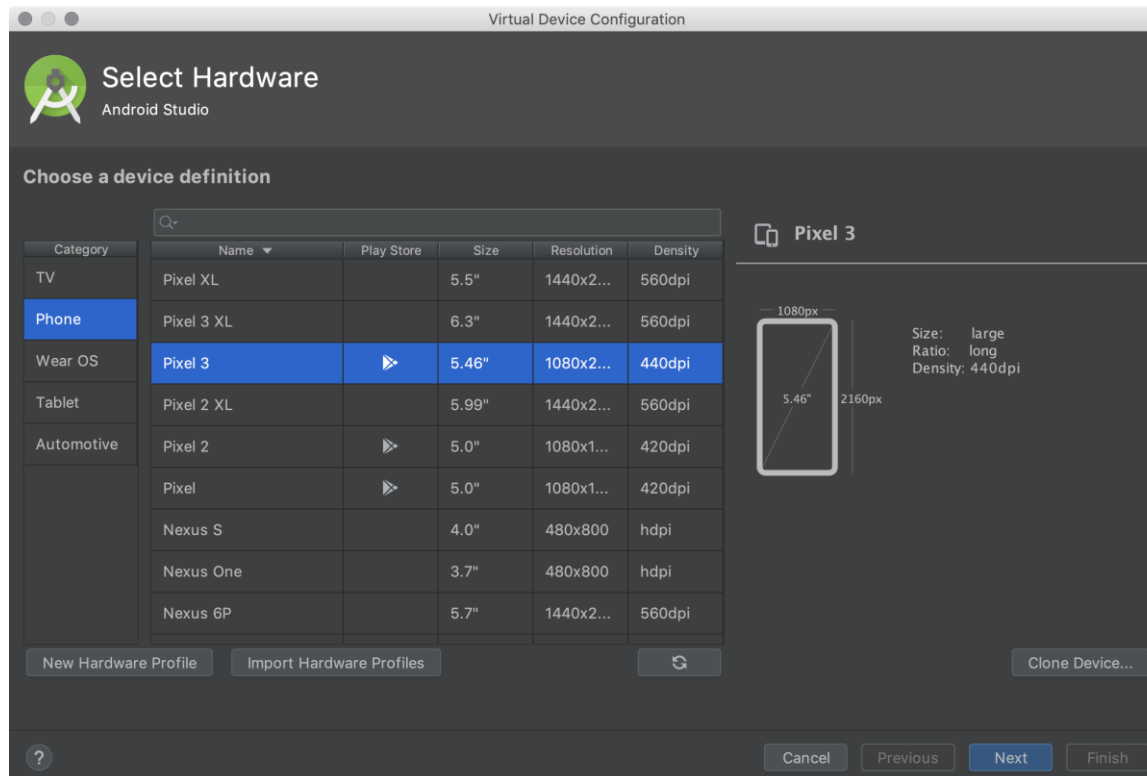


Uso del AVD Manager III

- ▶ En esa venta aparece la siguiente información:
 - ▶ Tipo de dispositivo del emulador
 - ▶ El nombre del emulador
 - ▶ Play Store: si tiene la etiqueta con el logo de Google Play tiene instaladas las APIs de Google (Gmail, maps, librerías...) Si queremos probar algunas de estas aplicaciones en el dispositivo tendrán que tenerlo instalado.
 - ▶ API: tiene que ser igual o superior a la API que nosotros hayamos elegido para nuestra aplicación. Si no es así no se podrá ejecutar.
 - ▶ Target: versión del sistema operativo instalado
 - ▶ CPU
 - ▶ El tamaño que ocupan en el disco
 - ▶ Acciones que se pueden realizar.

Uso del AVD Manager IV

- ▶ Vamos a pulsar en crear un nuevo dispositivo virtual (Create Virtual Device).
- ▶ Cuando lo pulsamos aparece la siguiente pantalla



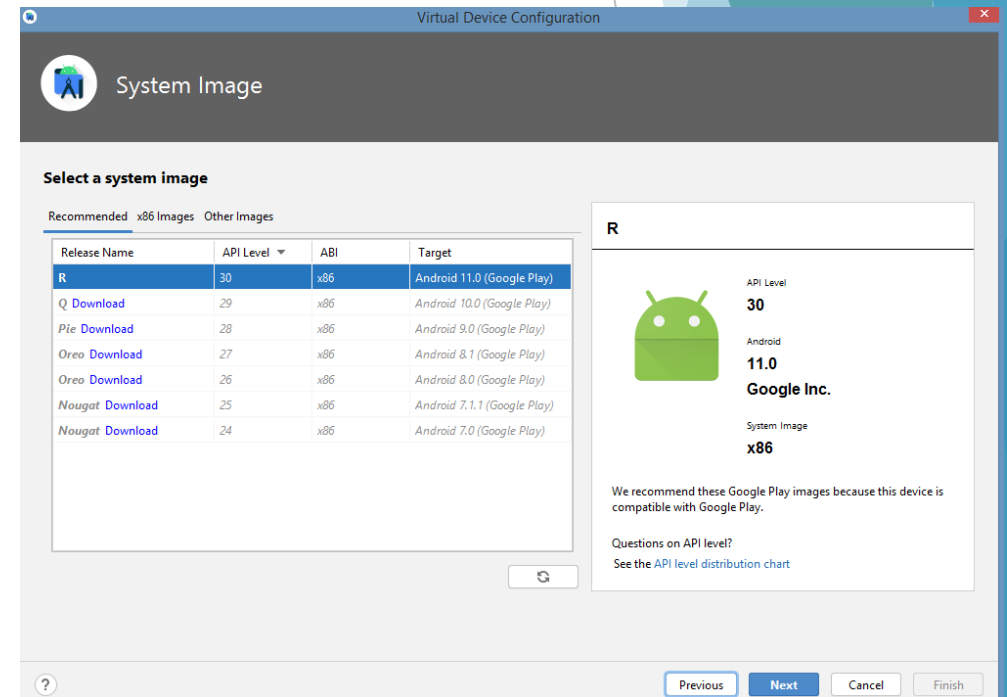
Uso del AVD Manager V

- Podemos elegir una definición existente o podemos hacer una a nuestra medida para poder utilizarla después.

**Práctica: Crear un nuevo emulador con el tamaño de
vuestro teléfono móvil habitual.**

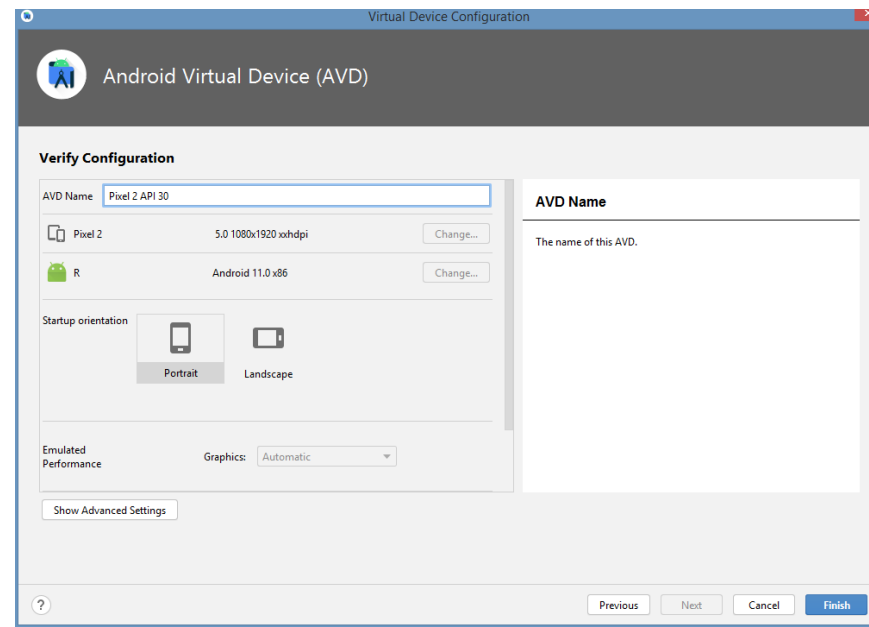
Uso del AVD Manager VI

- ▶ Una vez elegido el hardware con el que queremos trabajar, deberemos elegir el S.O que vamos a instalar en el dispositivo.
- ▶ En la pestaña Recommended aparecen los S.O. recomendados para el dispositivo.
- ▶ Si el S.O. que quieres instalar no están descargado aparecerá la opción de descarga. Se necesita estar conectado a Internet para proceder a la descarga.



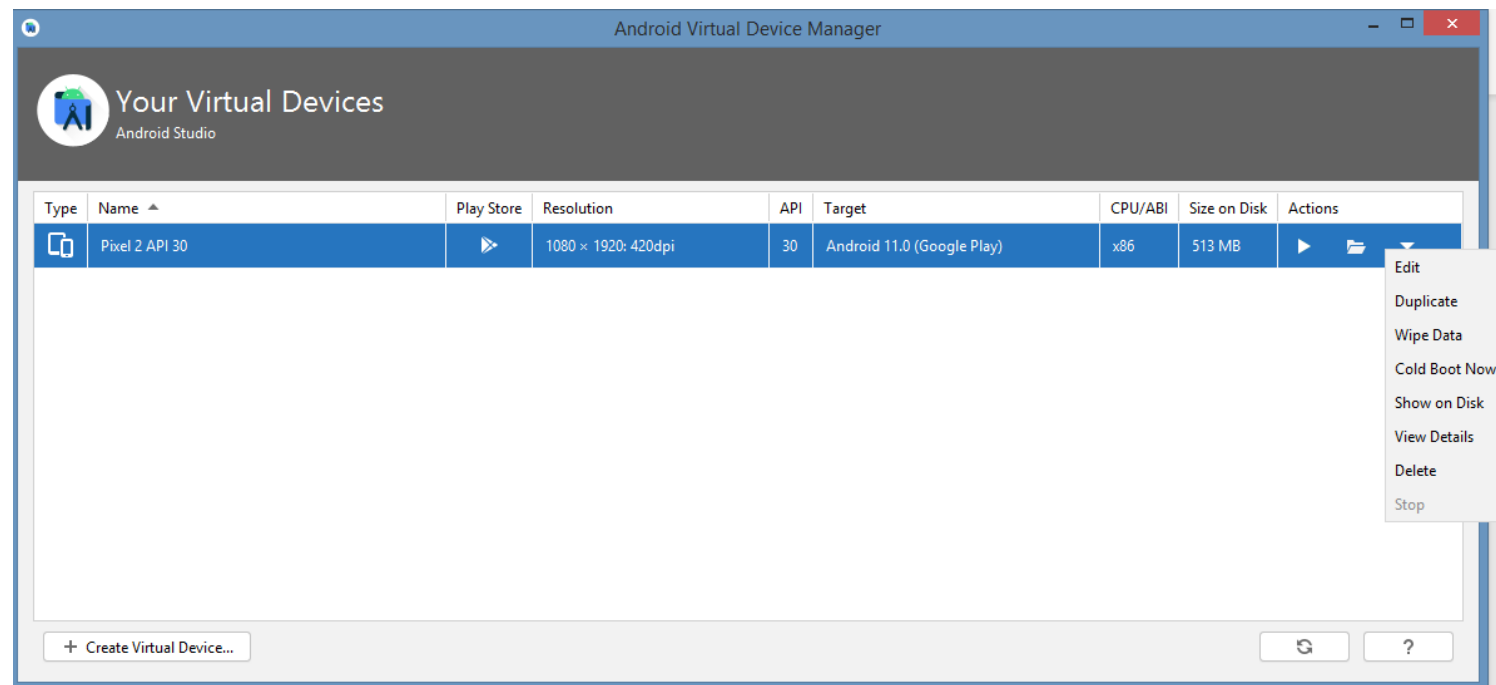
Uso del AVD Manager VII

- ▶ Para finalizar nos aparecerá una pantalla de configuración final del dispositivo. Se puede configurar desde la cámara, la velocidad de la red, como gestionar los gráficos o la cantidad de RAM que se le puede dar al dispositivo.
- ▶ Una vez realizado la selección se pulsa en el botón finish y ya tenemos generado nuestro dispositivo.



Uso del AVD Manager VIII

- Una vez creado el simulador volveremos a la pantalla inicial y aparecerá en el listado.

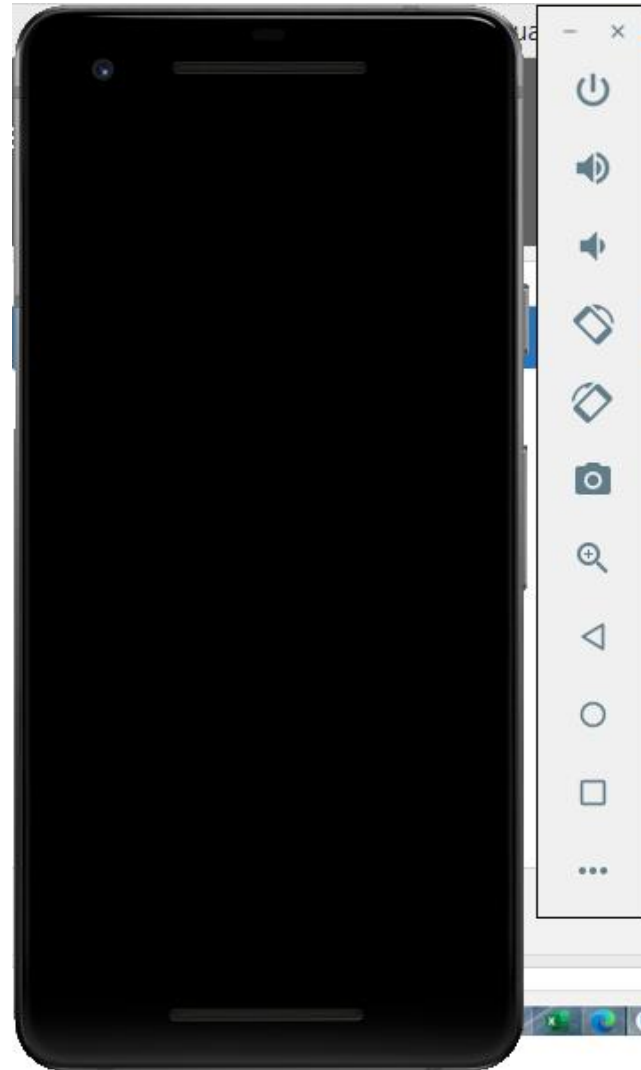


Uso del AVD Manager IX

- ▶ Dentro de las acciones podemos hacer varias cosas:
 - ▶ El primer icono es el de arrancar el simulador
 - ▶ El segundo icono es el la ubicación
 - ▶ El tercer icono es el de más opciones: editar, duplicar y borrar como las más usadas.
- ▶ Para comenzar a usar el simulador pulsaremos en el icono de arrancar.
- ▶ Aparecerá un dispositivo similar a este

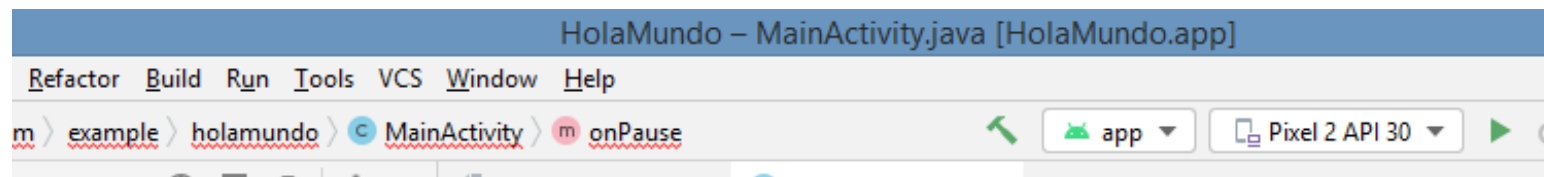
Uso del AVD Manager X

- ▶ El dispositivo tardará unos minutos en arrancar.
- ▶ El aspecto es muy similar al de un dispositivo móvil al uso.
- ▶ La barra de herramientas de la derecha simula todos los botones que tiene el móvil en la pantalla.
- ▶ Si se pulsan los tres puntos finales, aparecen muchas más opciones de configuración del dispositivo.



Uso del AVD Manager IX

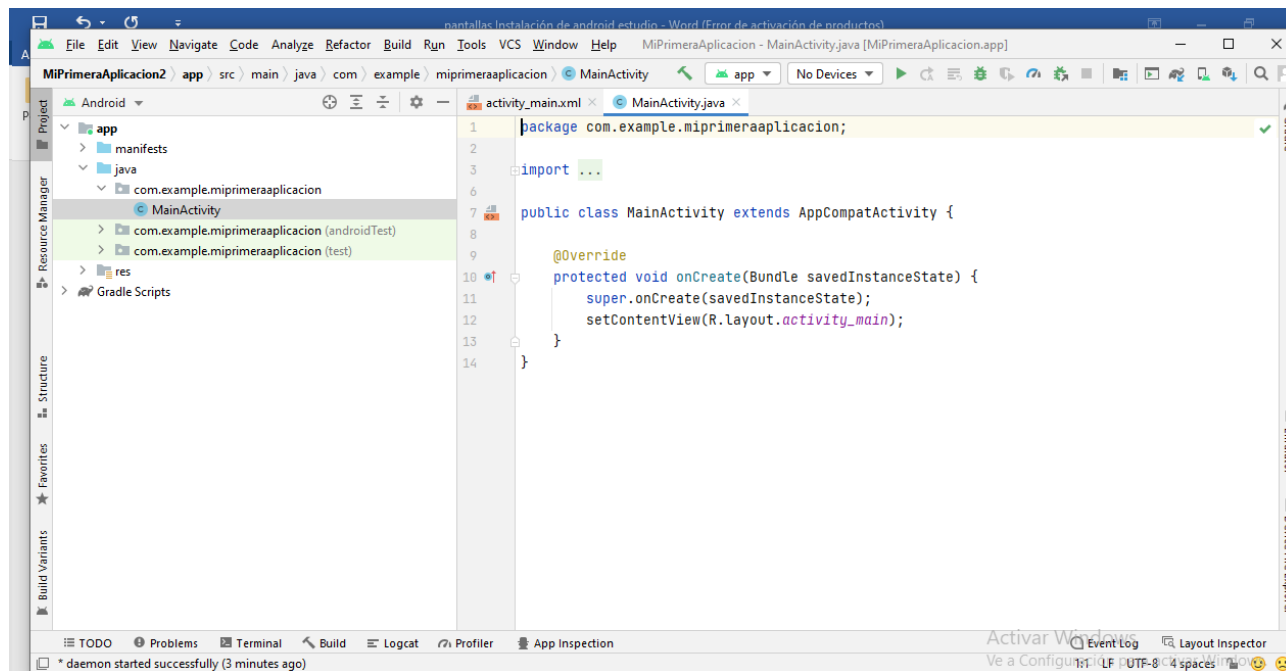
- ▶ Para ejecutar el dispositivo desde Android Studio, es suficiente con pulsar el icono de Run y directamente se abrirá el simulador.
- ▶ Si decidiéramos que el simulador creado no es el adecuado y quisiéramos cambiarlo, justo al lado del botón de Run hay un desplegable en el que te da la opción de volver al AVD Manager y poder cambiarlo o crear otro



Estructura de un proyecto

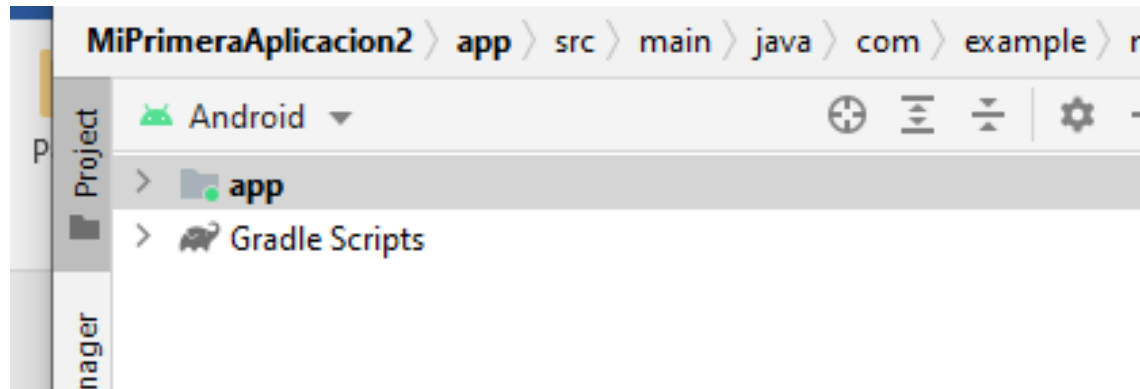
Estructura de un proyecto I

- Cuando se abre el nuevo proyecto nos aparece la siguiente estructura en el lado izquierdo de la pantalla.



Estructura de un proyecto II

- ▶ Aparecen dos carpetas principales:
 - ▶ App: es el módulo que Android Studio genera para aplicaciones móviles y Tablet. Si generamos un proyecto para relojes, televisión o coches aparecerá otro nombre diferente.
 - ▶ Gradle Scripts: es la configuración del proyecto a nivel de versiones. También se utiliza para gestionar la instalación de librerías externas. La gran mayoría de librerías que podemos utilizar lo haremos mediante la instalación de paquetes en este fichero.



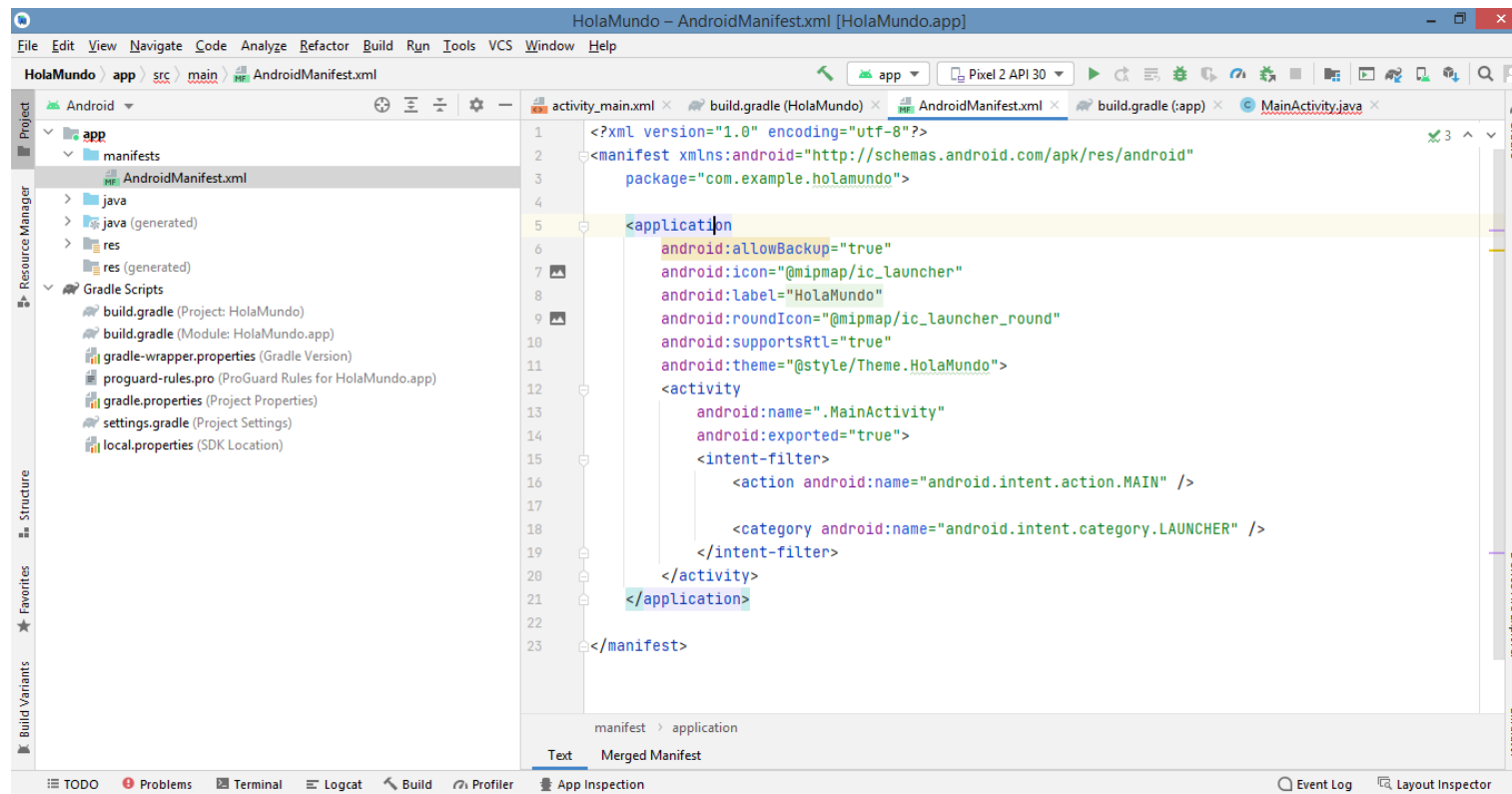
Estructura de un proyecto III (Manifest)

- ▶ Dentro de la carpeta APP está la carpeta manifest y dentro de esta el archivo AndroidManifest.
 - ▶ Es un código XML que incluye configuraciones a nivel del módulo de aplicación móvil.
 - ▶ Incluye como se llama el nombre del paquete principal donde tendremos todos los ficheros java del proyecto.
 - ▶ En la etiqueta <Application> se definen:
 - ▶ El icono de la aplicación
 - ▶ El nombre de la aplicación
 - ▶ El componente Activity: este componente hace referencia al único fichero que tenemos .java que a su vez define un componente Activity.

Estructura de un proyecto IV (Activity)

- ▶ El componente Activity define una pantalla de la aplicación.
- ▶ Cada pantalla de la aplicación estará gestionada por un componente Activity.
- ▶ Tenemos tantos componentes Activity como pantallas tenga la aplicación.
- ▶ El componente `<intent-filter>` de Activity solo puede aparecer una vez dentro de la aplicación y define cual es el Activity principal.
- ▶ El Activity principal se lanzará cuando nuestra aplicación se inicie desde el menú de aplicaciones del dispositivo.
- ▶ Si `<intent-filter>` apareciera dos veces, aparecerían dos iconos en el menú de aplicaciones, cada uno con el nombre del Activity que iniciaría en su caso.
- ▶ Solo debe aparecer una pantalla principal.

Estructura del proyecto. AndroidManifest



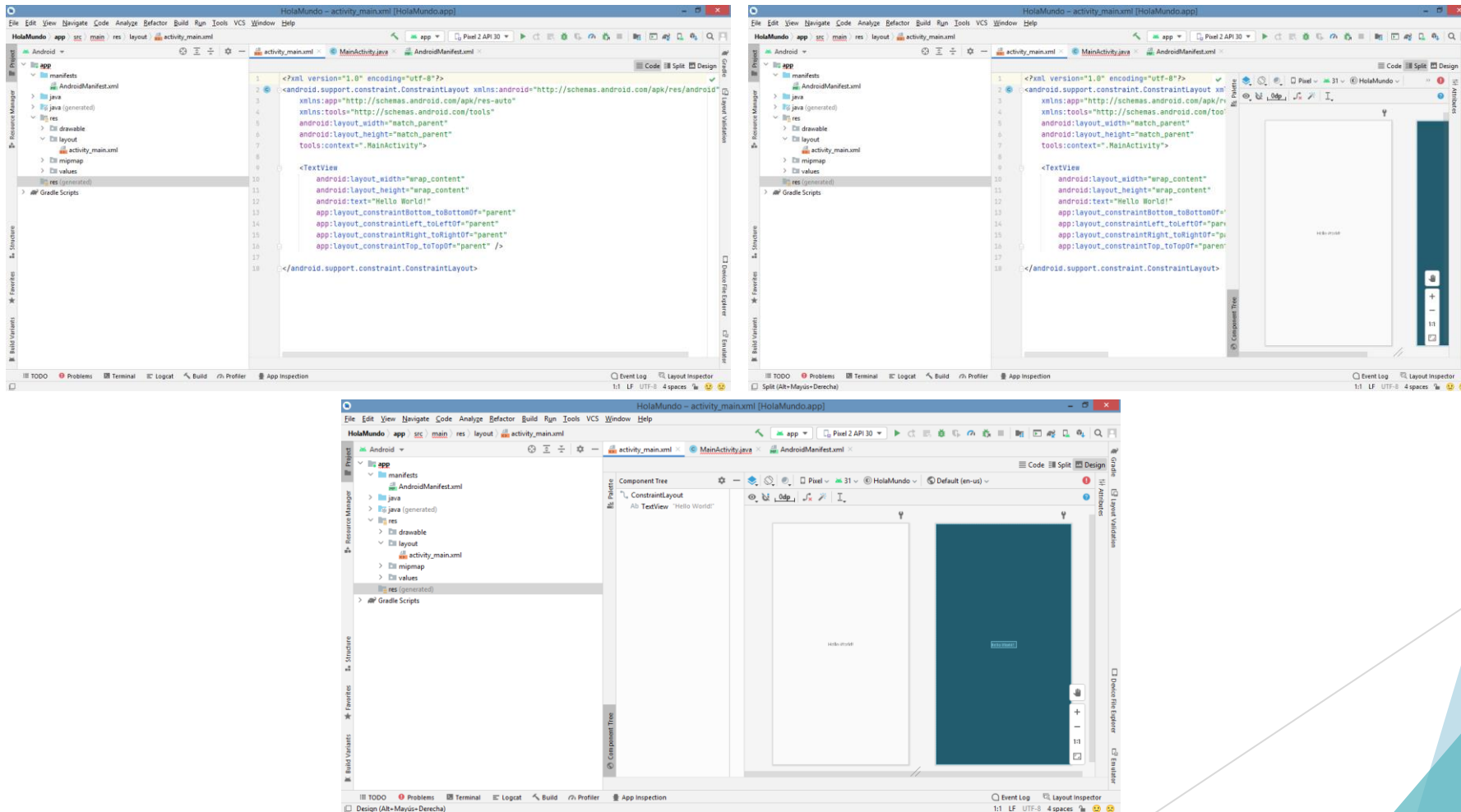
Estructura de un proyecto V (java)

- ▶ Dentro de la carpeta java tenemos tres subcarpetas:
 - ▶ `com.example.miprimeraaplicacion`
 - ▶ `com.example.miprimeraaplicacion (android test)`
 - ▶ `com.example.miprimeraaplicacion (test)`
- ▶ Las dos carpetas de test son para realizar pruebas en nuestra aplicación.
- ▶ En la carpeta `com.example.miprimeraaplicacion` es el paquete principal y su nombre se debe al id de la aplicación. Dentro de esta carpeta está el archivo `MainActivity.java`.
 - ▶ `MainActivity.java` gestiona una pantalla de la aplicación. En este caso la principal, pero se pueden generar todos los archivos `.java` que se necesiten.

Estructura de un proyecto VI (res)

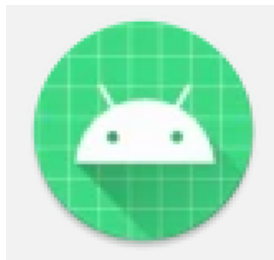
- ▶ La carpeta res es una abreviatura de resource que significa recurso.
- ▶ Almacena todos los recursos adicionales al código de nuestro fichero .java.
- ▶ Dentro de la carpeta res tenemos:
 - ▶ Carpeta drawable: contiene ficheros de imágenes decorativas de la aplicación, iconos, elementos para botones, imágenes descriptivas, .png, vectoriales,...)
 - ▶ Carpeta Layout: contiene el archivo activity.mail.xml que es el interfaz de usuario de la aplicación.
 - ▶ Cada vez que queremos crear una interfaz de usuario , crearemos un nuevo fichero que almacenaremos dentro de esta carpeta.
 - ▶ Se puede previsualizar con una vista de diseño para ver el resultado visual o lo podemos ver en formato XML o con las dos pantallas a la vez.

Estructura de un proyecto VII (res)(layout) activity_main.xml



Estructura de un proyecto VIII (res)

- ▶ En la carpeta mipmap se definen los iconos de la aplicación. Hay dos tipos de iconos, almacenados en dos carpetas diferentes:
 - ▶ ic_launcher: que almacena iconos en formato cuadrado
 - ▶ ic_launcher_round: que almacena iconos en formato redondo
- ▶ En cada carpeta hay iconos generados a diferente densidad de pixeles para que se vean con la resolución correcta en los distintos dispositivos.
- ▶ Android decidirá que imagen es correcta para cada dispositivo.
- ▶ La elección se hace en tiempo de ejecución



Estructura de un proyecto IX (res)

- ▶ La última carpeta es la carpeta values.
- ▶ En esta carpeta se definen diferentes variables a nivel de:
 - ▶ Colores (colors): variables con diferentes códigos de color que podemos usar en nuestro programa.
 - ▶ Cadenas de texto (strings): variables de cadenas de texto que podemos utilizar en la aplicación. Muy importante porque nos permite crear variaciones para hacer traducciones a otros idiomas.
 - ▶ Temas (themes): ficheros de temas para definir, sobrescribir o modificar temas en los que está basada la aplicación. Se puede modificar lo que hay en el fichero.

Estructura de un proyecto X (Gradle Scripts)

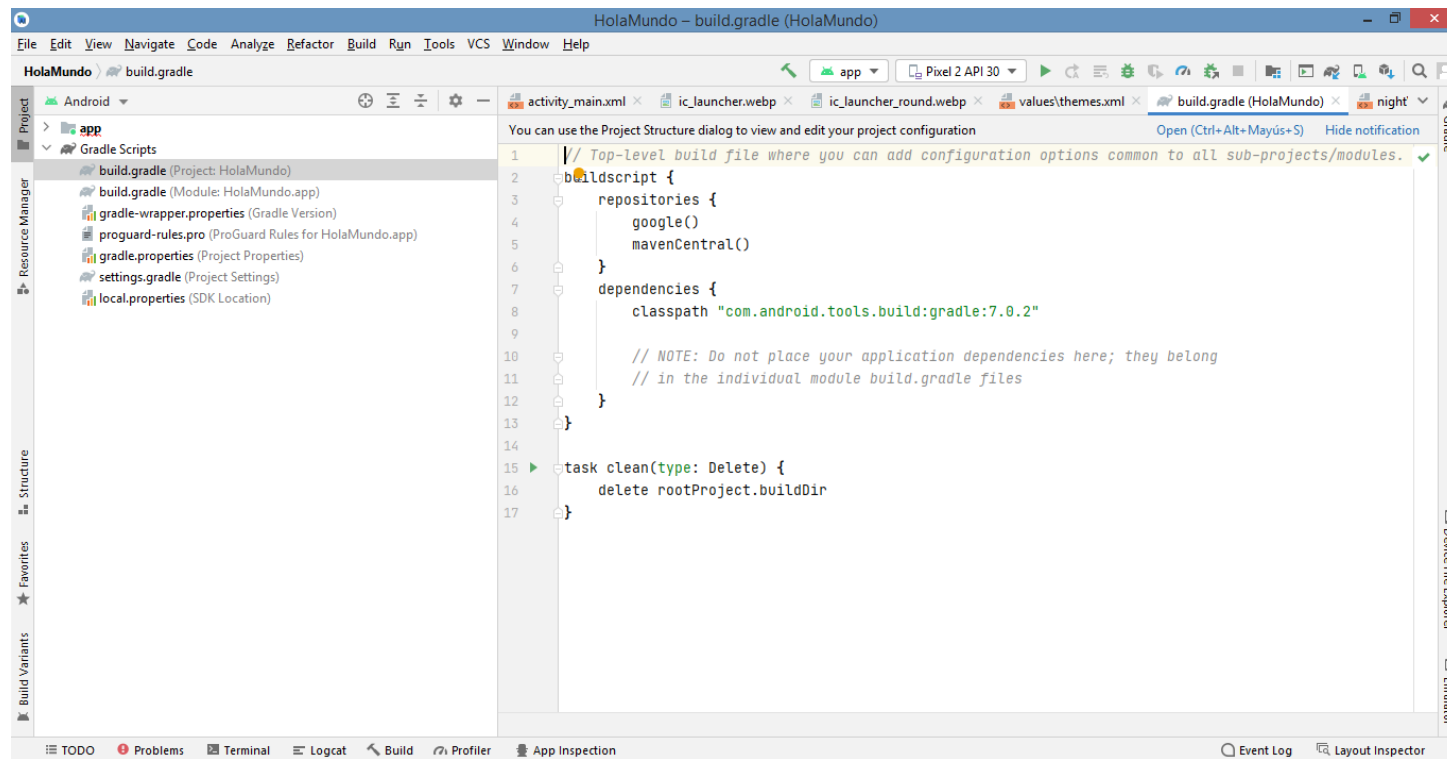
- ▶ La segunda carpeta principal de nuestro proyecto es Gradle Scripts.
- ▶ Es la configuración del proyecto a nivel de versiones.
- ▶ También se utiliza para gestionar la instalación de librerías externas.
- ▶ La gran mayoría de librerías que podemos utilizar lo haremos mediante la instalación de paquetes de dependencia en este fichero.
- ▶ Dentro de esta carpeta hay varios archivos de los cuales vamos a ver los más importantes que son:
 - ▶ Built.gradle (Project: MiPrimeraAplicacion)
 - ▶ Built.gradle(Module: MiPrimeraAplicacion.app)

Estructura de un proyecto XI

(build.gradle project)

- ▶ Incluyo este fichero para ver la diferencia con el build.gradle module.
- ▶ En este fichero configuramos a nivel del proyecto completo.
- ▶ Es un fichero que no se suele modificar y ya que las modificaciones se realizarán a nivel de módulo.
- ▶ El aspecto del fichero es:

Archivo (build.gradle project)

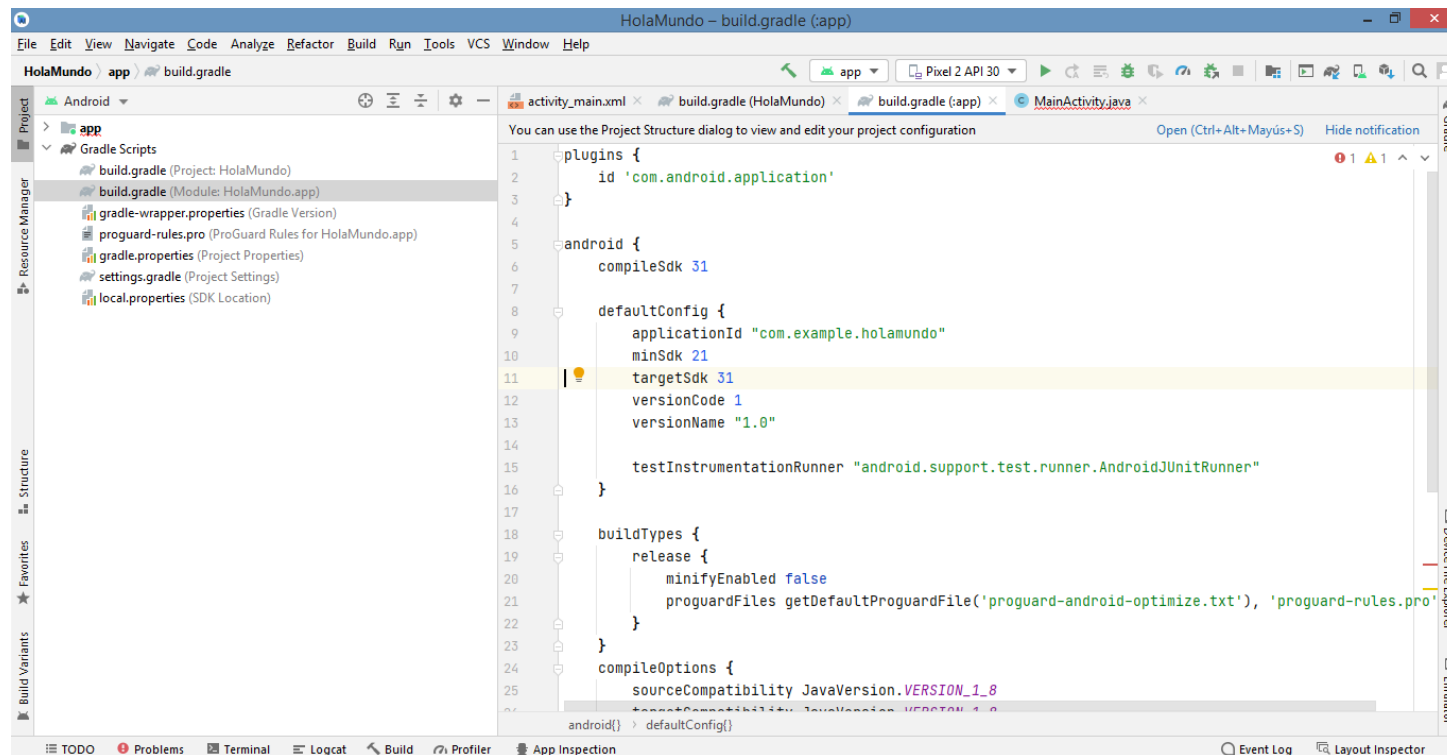


Estructura de un proyecto XII

(build.gradle module)

- ▶ En el fichero build.gradle(Module MiPrimeraAplicacion.app) se realizan las configuraciones a nivel del módulo de aplicación.
- ▶ En la configuración por defecto aparece:
 - ▶ la versión del SDK
 - ▶ la versión exacta de la librería de herramientas
 - ▶ El nombre principal ID de la aplicación
 - ▶ La versión mínima de android para poder instalar la aplicación
 - ▶ La versión code: siempre se modificará cada vez que el programa se modifique. Se aumenta de uno en uno.
 - ▶ La versión Name: es el nombre que le vamos a dar a la versión. No tiene porque aumentar de uno en uno aunque si se debe modificar si hay alguna modificación en la aplicación.

Archivo (build.gradle module)

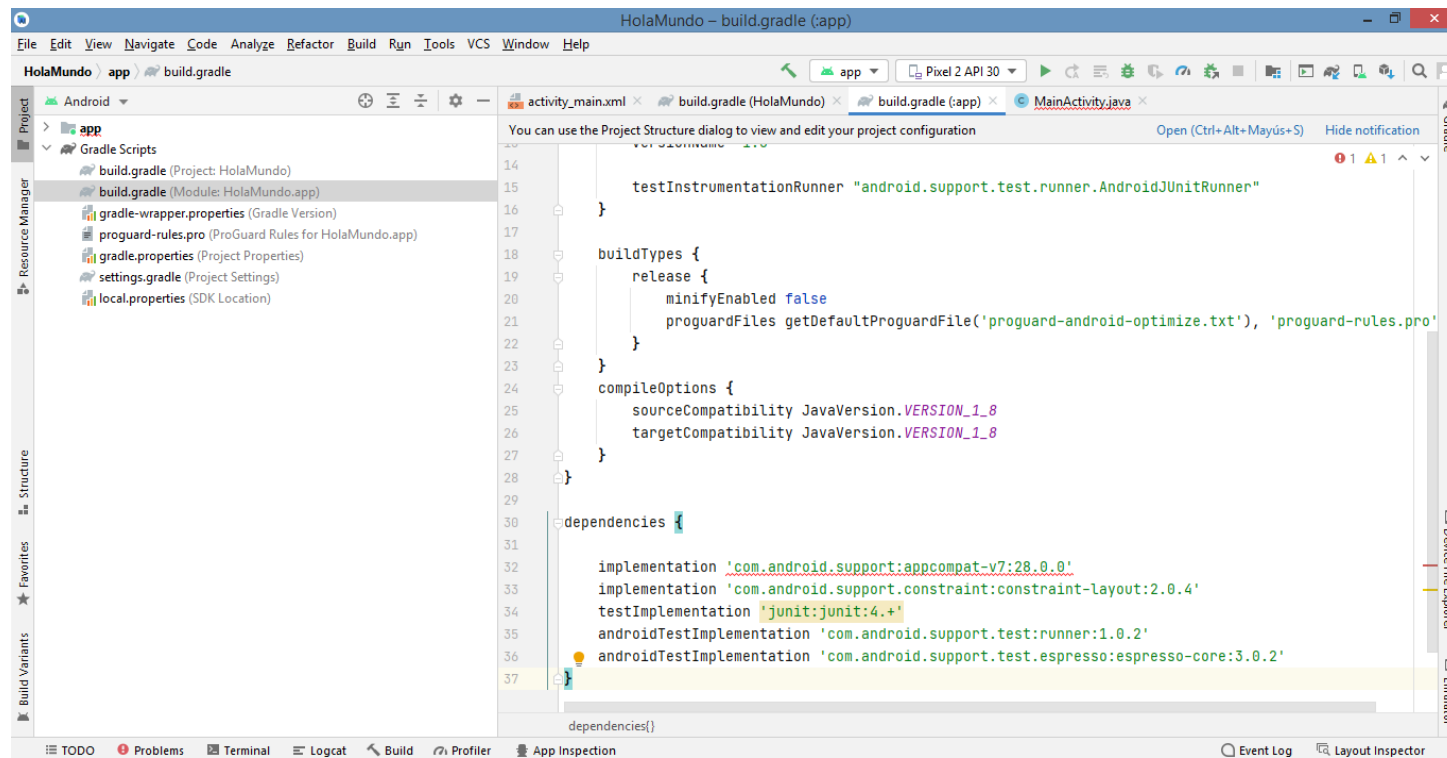


Estructura de un proyecto XIII

(build.gradle module)

- ▶ Dentro del archivo en el bloque de dependencias se incluyen las instalaciones de las librerías externas.
- ▶ Para añadir una librería pondríamos la palabra reservada `compile` indicando la ruta de la librería.
- ▶ Hay diferentes repositorios de donde obtener las librerías. Por ejemplo en el archivo `build.gradle (Project)` están los repositorios de Google y de `mavenCentral`.
- ▶ Si necesitamos librerías de otros repositorios deberemos añadirlos en este archivo `build.gradle (Project)`

Archivo (build.gradle module)

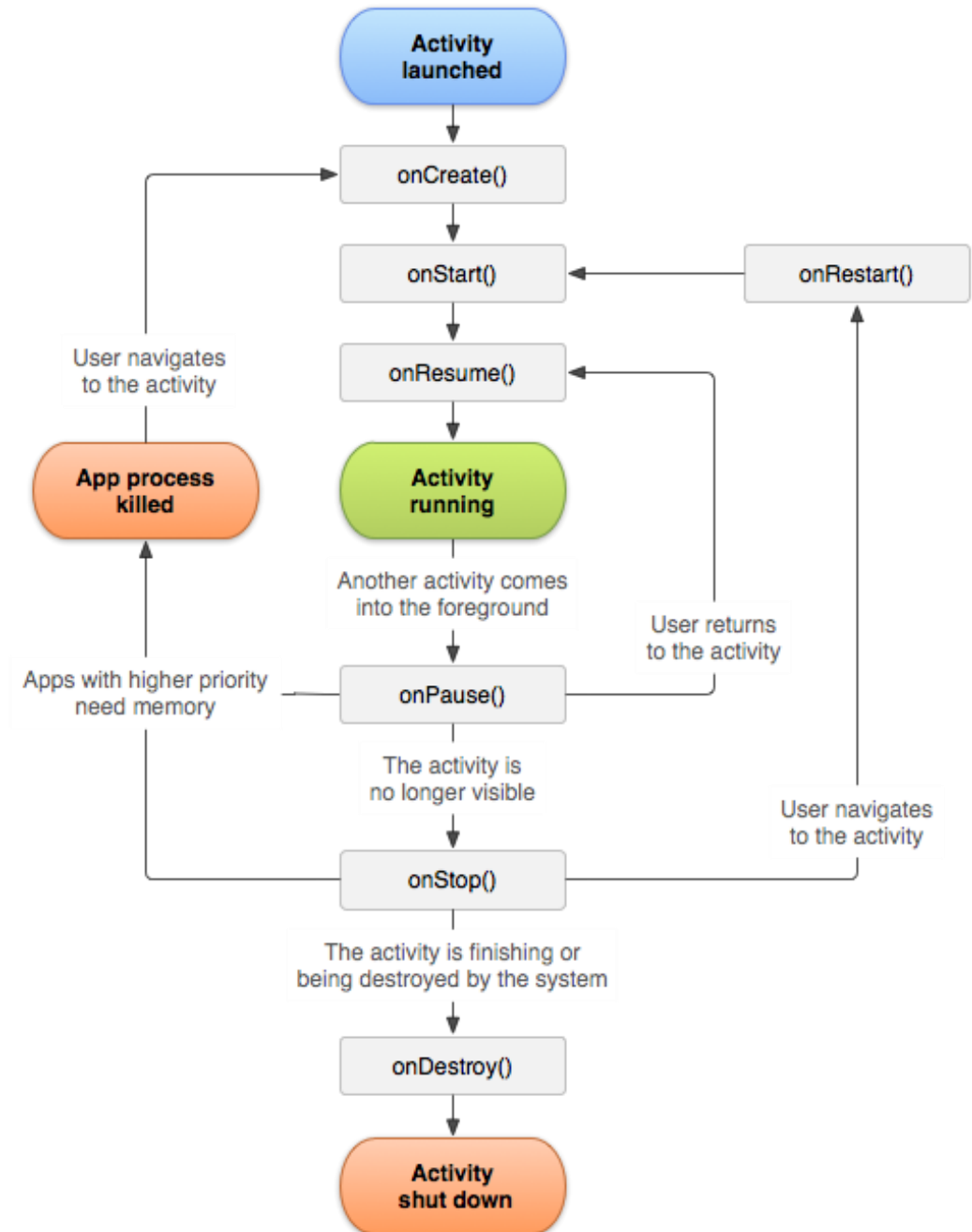


Ciclo de vida de un Activity

Ciclo de vida de un Activity I

En función de la navegación que se haga en una aplicación, va a pasar por diferentes fases.

En la imagen aparece el ciclo de vida de un Activity.



Ciclo de vida de un Activity II

Pasos del ciclo:

1. Cuando arrancamos el la aplicación desde nuestro dispositivo se lanza el primer Activity, el que hayamos definido en el fichero `androidManifest.xml`.
2. Una vez iniciado el primer método que se lanza es el `onCreate`
3. El siguiente método que se lanzará será el `onStart`
4. Seguidamente el método `onResume`

Cuando se inicia por primera vez una APP siempre se ejecutan los tres métodos.

La mayoría del código inicial estará en el método `onCreate`.

Una vez se haya pasado por los tres métodos la aplicación está en ejecución.

Ciclo de vida de un Activity III

Los métodos por los que puede pasar un Activity durante su ciclo de vida son:

1. `onCreate`: se activa cuando el sistema crea la actividad por primera vez.
2. `onStart`: el usuario ve la actividad mientras se prepara para ser interactiva. No se mantiene demasiado tiempo en este estado. Rápidamente pasa a `onResume`.
3. `onResume`: la actividad pasa a primer plano e interactúa con el usuario. La APP permanece en este estado hasta que otro evento le quita la prioridad.

Ciclo de vida de un Activity IV

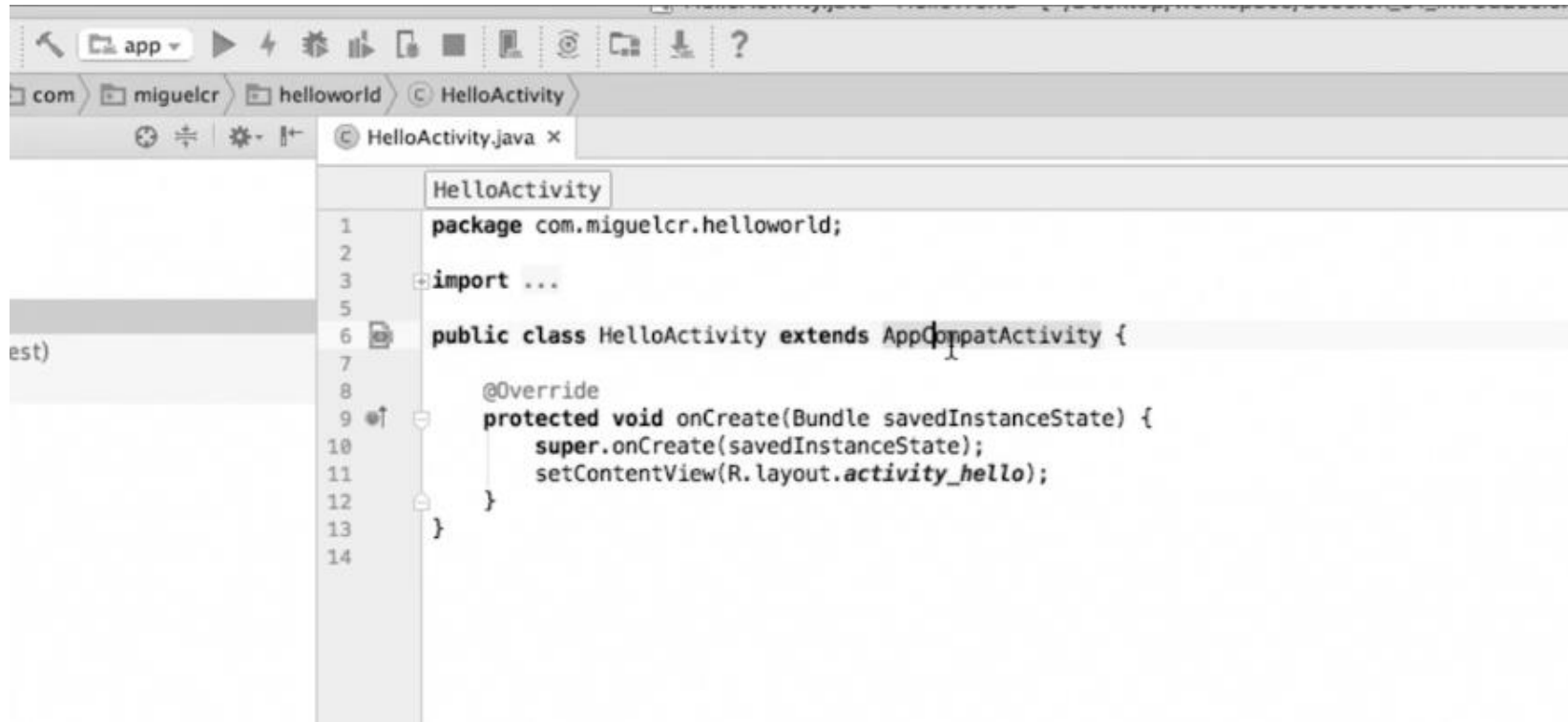
3. onPause: la actividad pasa a este estado cuando se produce un evento de interrupción. El usuario ha abandonado la actividad pero no la ha finalizado. Desde esta fase se pasa directamente al estado onResume.
4. onStop: cuando el usuario no puede ver la actividad ha entrado en fase de parada. Sucede cuando otra aplicación ocupa toda la pantalla o se está finalizando el uso de la aplicación. Desde este estado se puede pasar a onStart y de este a onResume.
5. onDestroy: la actividad ha finalizado. Si estamos en este estado, necesitaremos pasar de nuevo por todos los estados cuando utilicemos de nuevo la aplicación.

Ciclo de vida de un Activity V

- ▶ Ejemplo para ver como pasa por los distintos métodos de un activity.
- ▶ Vamos a utilizar el método estático `log.i()` para imprimir un mensaje cada vez que la aplicación pase por uno de los métodos vistos anteriormente.
- ▶ Este método escribe un mensaje informativo en el logcat.
- ▶ Podemos hacer una búsqueda para que nos muestre por donde ha ido pasando la aplicación.

Comprobación del ciclo de vida de una aplicación I

- ▶ Dentro del archivo MainActivity.java tenemos el método onCreate.
- ▶ AppCompatActivity hace que podamos tener compatibilidad con versiones anteriores.



```
1 package com.miguelcr.helloworld;
2
3 import ...
4
5
6 public class HelloActivity extends AppCompatActivity {
7
8     @Override
9     protected void onCreate(Bundle savedInstanceState) {
10         super.onCreate(savedInstanceState);
11         setContentView(R.layout.activity_hello);
12     }
13 }
14
```

Comprobación del ciclo de vida de una aplicación II

```
1 package com.miguelcr.helloworld;
2
3 import ...
4
5
6 public class HelloActivity extends AppCompatActivity {
7
8     @Override
9     protected void onCreate(Bundle savedInstanceState) {
10         super.onCreate(savedInstanceState);
11         setContentView(R.layout.activity_hello);
12     }
13
14     @Override
15     protected void onStart() {
16         super.onStart();
17     }
18
19     @Override
20     protected void onResume() {
21         super.onResume();
22     }
23
24 }
25
```

Comprobación del ciclo de vida de una aplicación III

```
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.util.Log;

public class HelloActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_hello);
        Log.i("TAG cicloVida ", "Ciclovida: onCreate");
    }
}
```


Comprobación del ciclo de vida de una aplicación IV

```
16
17
18 @Override
19 protected void onStart() {
20     super.onStart();
21     Log.i("TAG cicloVida ", "Ciclovida: onStart");
22 }
23
24 @Override
25 protected void onResume() {
26     super.onResume();
27     Log.i("TAG cicloVida ", "Ciclovida: onResume");
28 }
29
30 @Override
31 protected void onPause() {
32     super.onPause();
33     Log.i("TAG cicloVida ", "Ciclovida: onCreate");
34 }
```

Comprobación del ciclo de vida de una aplicación V

```
/com.miguelcr.helloworld I/TAG cicloVida: Ciclovida: onCreate  
/com.miguelcr.helloworld I/TAG cicloVida: Ciclovida: onStart  
/com.miguelcr.helloworld I/TAG cicloVida: Ciclovida: onResume
```

- Enviamos un mensaje para salir de la aplicación y que pase por el método onPause y onStop