Programación Multimedia y Dispositivos Móviles

UD 1 - Introducción e instalación

Antonio Calabuig Puigvert

Departamento de Informática

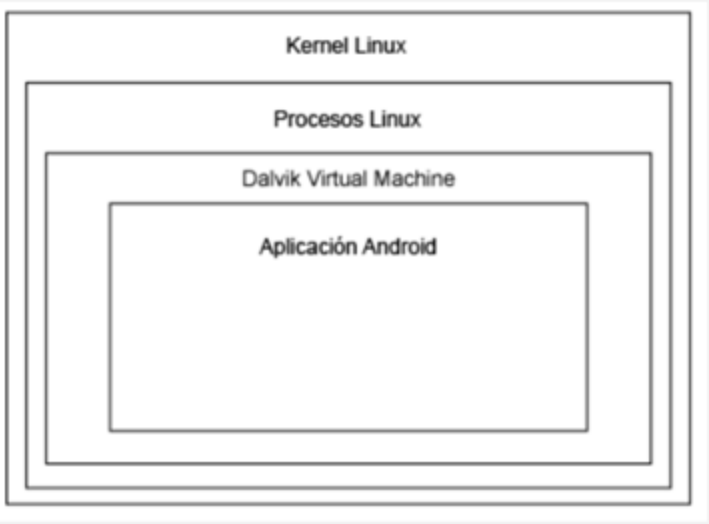
Android Studio

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

Platform	Android Studio package	Size	SHA-256 checksum
Windows (64-bit)	android-studio-2021.2.1.16-windows.exe Recommended	929 MiB	214fc7339060990d615bb02f2576474a3d6c152249fb67b03124162e111da4c7
	android-studio-2021.2.1.16-windows.zip No .exe installer	940 MiB	78575a5b779d66630b757a32df0a26b8c052995623d649ed31787fb64eef14b1
Mac (64-bit)	android-studio-2021.2.1.16-mac.dmg	1017 MiB	df46f2199fcf4c7e6b882ba16151ea1d2dd48a15f5c87d30224f1b5401d2b648
Mac (64-bit, ARM)	android-studio-2021.2.1.16-mac_arm.dmg	1014 MiB	d4e06bcc6f614cd4b261fc6034529edb205b31b0e56824490a91350c3640806a
Linux (64-bit)	android-studio-2021.2.1.16-linux.tar.gz	964 MiB	aa5773a9e1da25bdb2367a8bdd2b623dbe0345170ed231a15b3f40e8888447dc
Chrome OS	android-studio-2021.2.1.16-cros.deb	817 MiB	b020a9a664d8237711e74198d5d07087858d993c0b7ade1f35cbaff668e8acd5

Instalación

Ejecución en máquina virtual



- Las aplicaciones de Android que están escritas en Java se ejecutan dentro de una máquina virtual denominada Dalvik Virtual Machine, que es una tecnología de código abierto.
  - Cada aplicación de Android se ejecuta dentro de una instancia de la Dalvik VM, que a su vez permanece dentro de un proceso gestionado por el kernel Linux.

Componentes de una aplicación Android

Componente	Descripción
ACTIVITY	
VIEW	
SERVICE	
BROADCAST RECEIVER	
CONTENT PROVIDER	

Componentes de una aplicación Android

Componente	Descripción
ACTIVITY	Las activities son clases Java que heredan (extends) de la superclase Activity o de otras derivadas de ella. Contiene toda la lógica de la aplicación
VIEW	Los objetos view son los 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 Android

Componente	Descripción
SERVICE	Los servicios son componentes sin interfaz gráfica que se ejecutan en segundo plano. En concepto, son exactamente iguales a los servicios presentes en cualquier otro sistema operativo. Los servicios 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.

Componentes de una aplicación Android

Componente	Descripción
INTENT	Los intents 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). ● Se pueden entender como los mensajes o peticiones que son enviados entre los distintos componentes de una aplicación o entre distintas aplicaciones. ● Mediante un intent se puede ejecutar una activity desde cualquier otra, iniciar un servicio, enviar un mensaje broadcast, iniciar otra aplicación, etc. ● Debemos tener en cuenta que un proyecto Android no tiene un único punto de entrada, sino que sus aplicaciones (activities) reaccionan como resultado de intents específicos. ● 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 Android

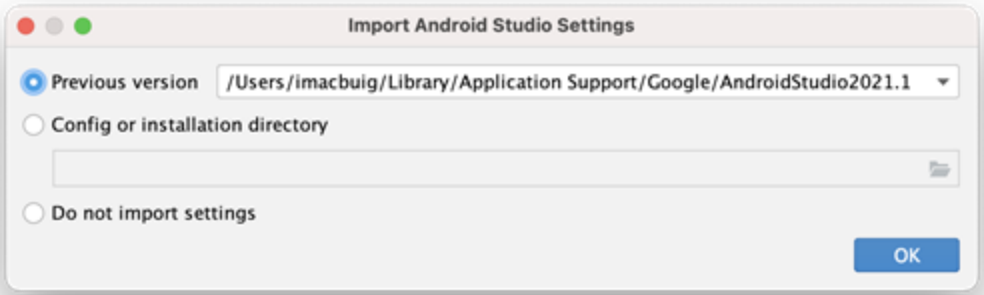
Componente	Descripción
BROADCAST RECEIVER	Los Broadcast Receiber son componentes que reaccionan a *intents* específicos y pueden ejecutar acciones. Por ejemplo, lanzar una determinada activity o devolver otro *intent* al sistema para que siga el proceso. ● También pueden detectar y reaccionar ante determinados mensajes o eventos globales generados por el sistema (por ejemplo: “Batería baja”, “SMS recibido”, “Tarjeta SD insertada”) o por otras aplicaciones.

Componentes de una aplicación Android

Componente	Descripción
CONTENT PROVIDER	Un "proveedor de contenidos" es el mecanismo que se ha definido en Android para compartir 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. De la misma forma, nuestra aplicación podrá acceder a los datos de otra a través de los content provider que se hayan definido.

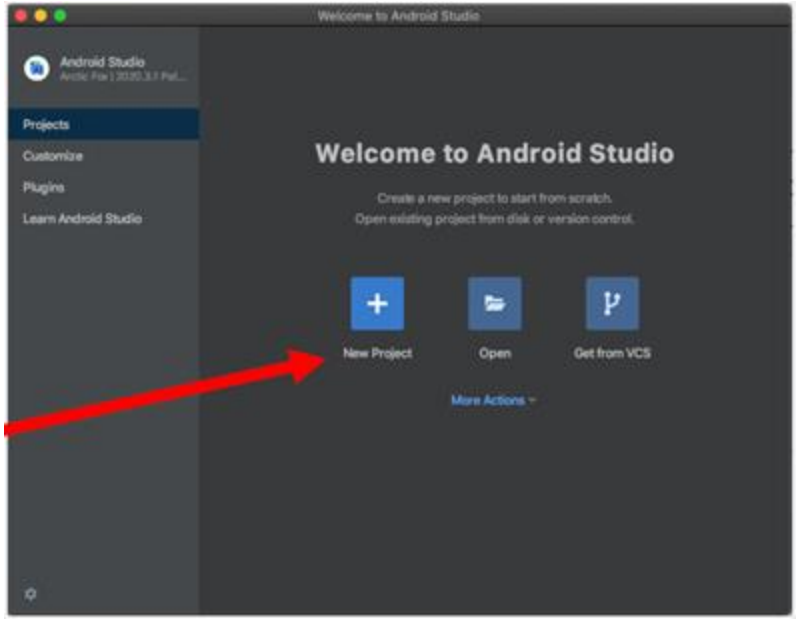
Primer proyecto Android

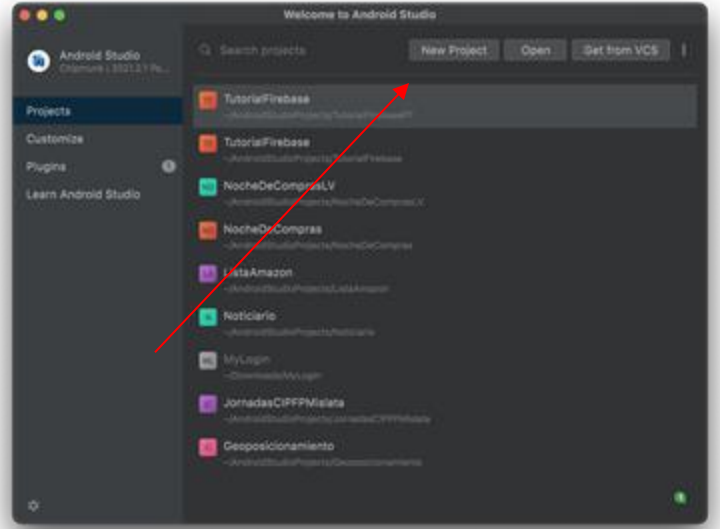
- Configuración del directorio de trabajo



Primer proyecto Android

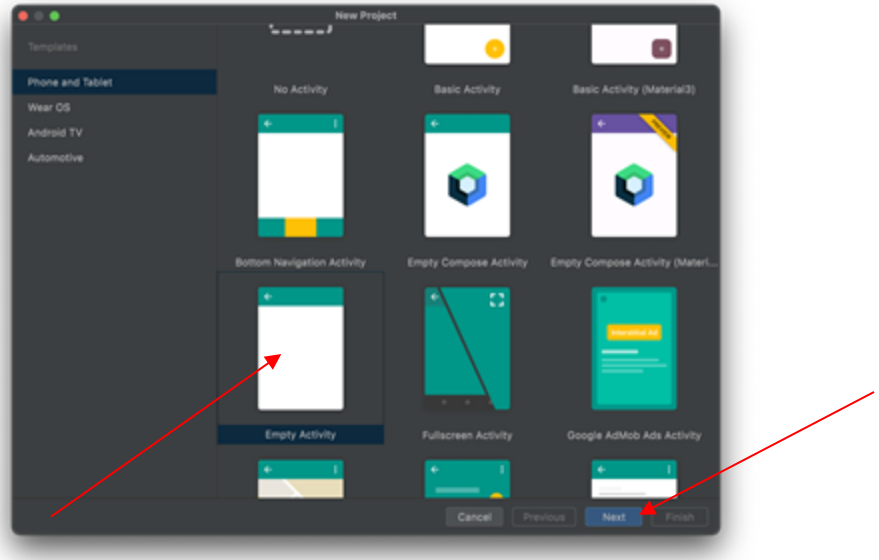
- Nuevo proyecto





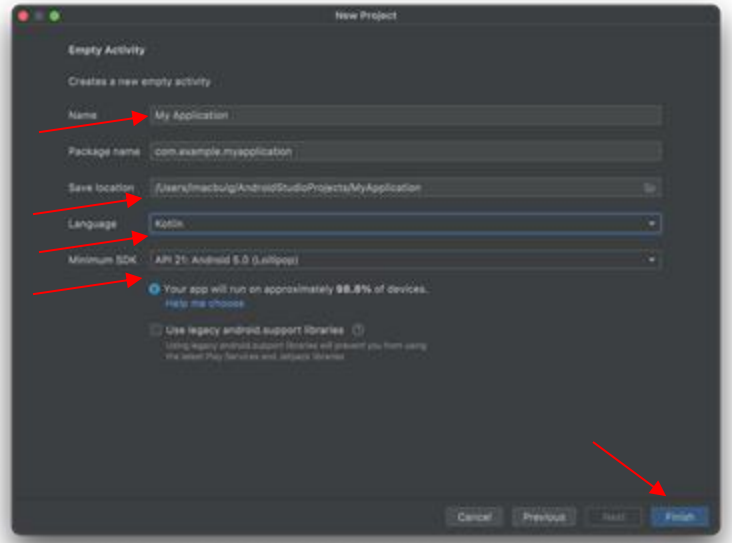
Primer proyecto Android

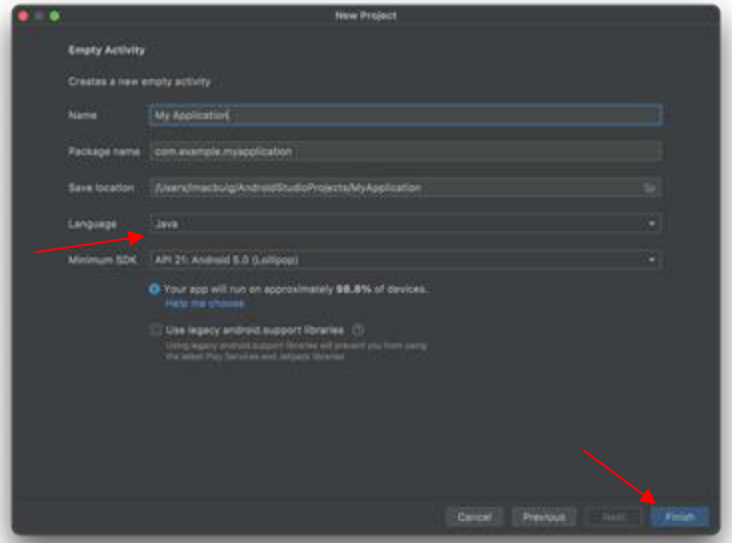
- Menú Activity



Primer proyecto Android

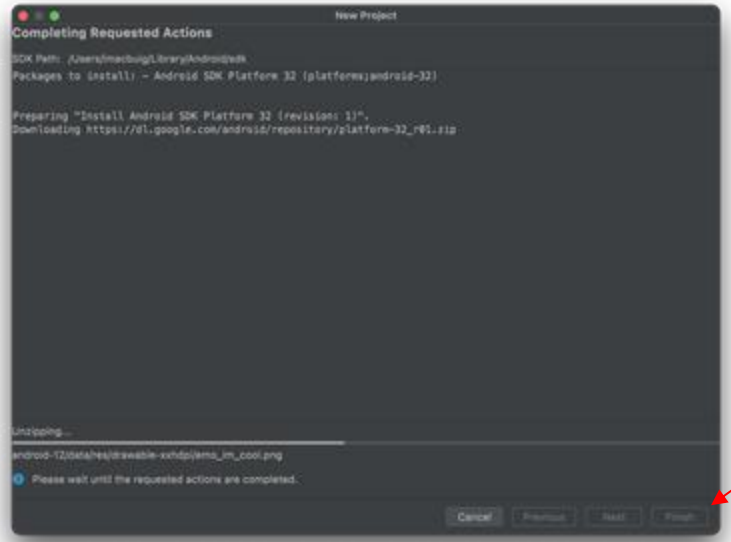
- Menú Project





Primer proyecto Android

- Esperamos mientras hace cosas

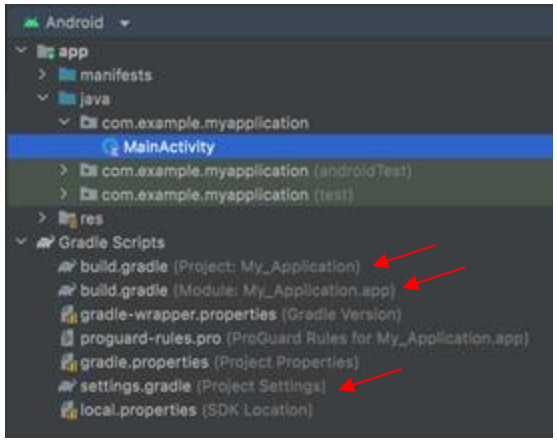


Al acabar pulsamos sobre finish y ya tenemos listo el proyecto.

Gradle

Android Studio usa Gradle, un paquete de herramientas de compilación avanzadas, para automatizar y administrar el proceso de compilación y, al mismo tiempo, definir configuraciones de compilación personalizadas y flexibles.

En un proyecto se visualiza como un archivo de configuración. En cada proyecto encontraremos tres ficheros Gradle diferentes.



El Gradel no es nuestro enemigo pero todos lo odiamos 😈

MainActivity.kt

MainActivity.kt: Contiene una clase Java que hereda de la superclase **AppCompatActivity**.

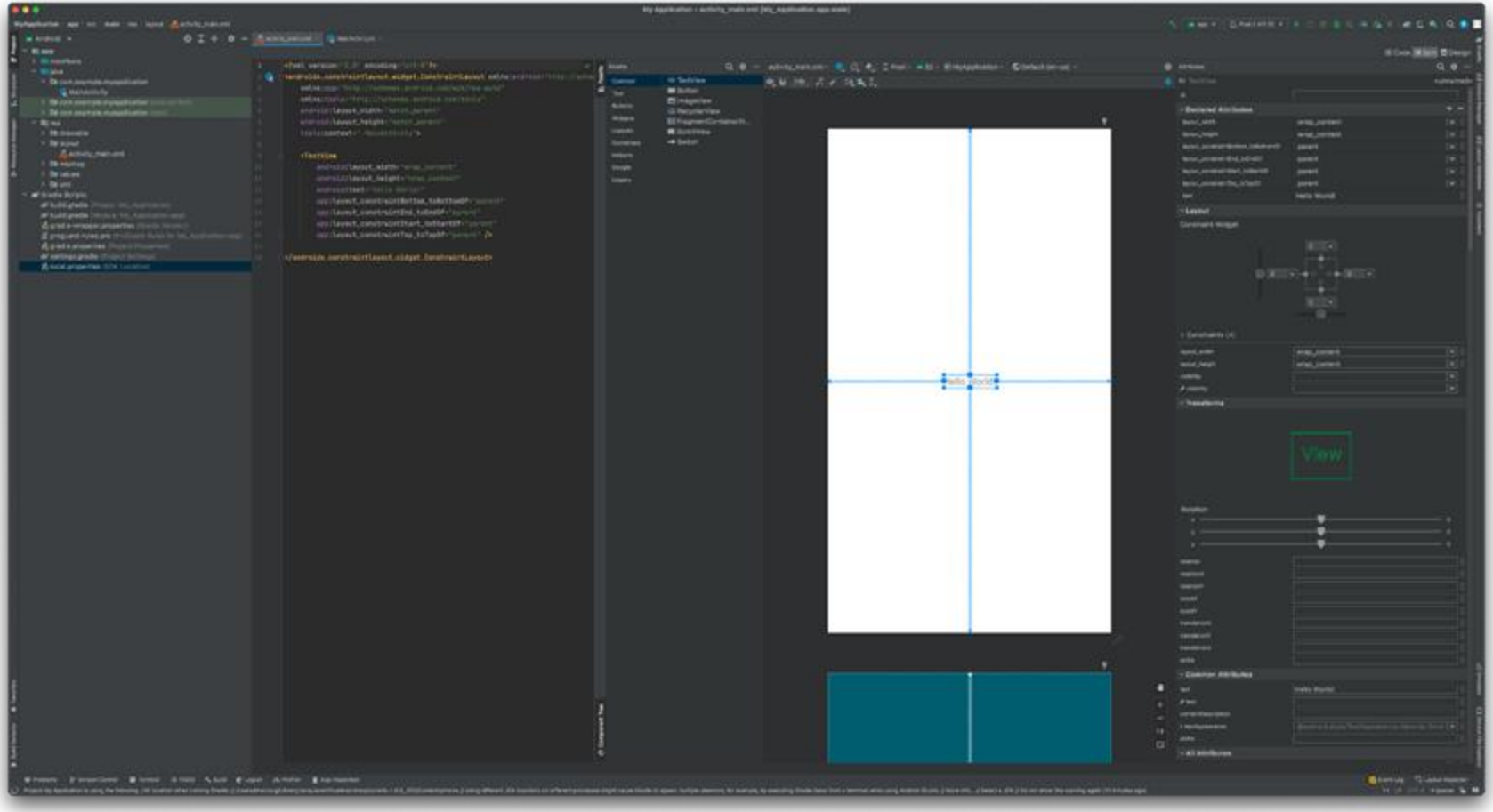
- Una aplicación Android contiene múltiples clases, que se comunican entre sí por medio de eventos llamados **intents**.
- El proceso interno de la aplicación es llevado a cabo por estas clases, denominadas **Activitys**.

```
class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
    }
}
```

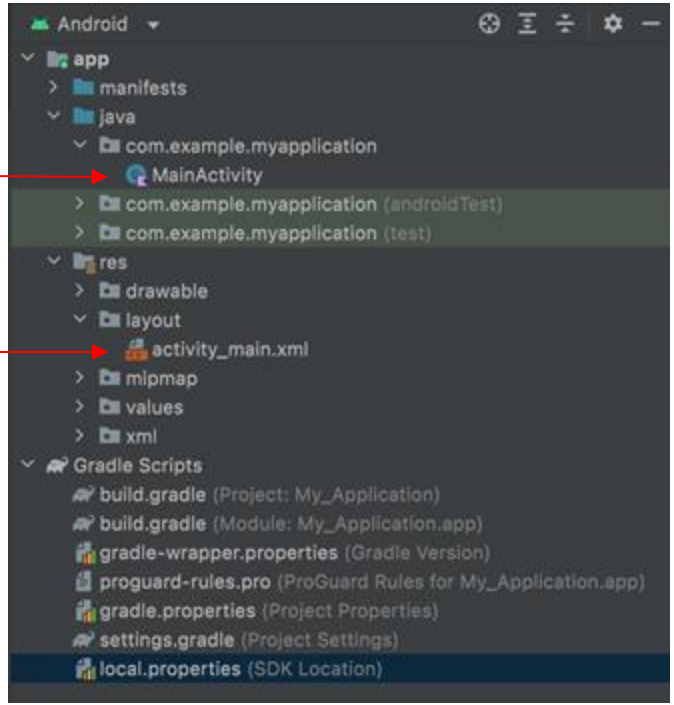
activity_main.xml

Contiene un documento XML, denominado **layout**, con información sobre lo que se va a mostrar en la pantalla del dispositivo.

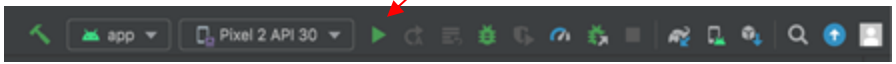
- Cada elemento del documento XML especifica un control, denominado **widget**, que aparecerá en la pantalla.
- Las características de estos controles se especifican mediante atributos.
- El elemento raíz se denomina **layout** y es un control contenedor, es decir que puede contener otros controles o incluso otros layouts.
- **activity_main** puede verse y modificarse también en modo diseño, según nos muestra la siguiente pantalla. En este caso debemos asegurarnos de que la API se corresponde con aquélla con la que hemos creado la aplicación.



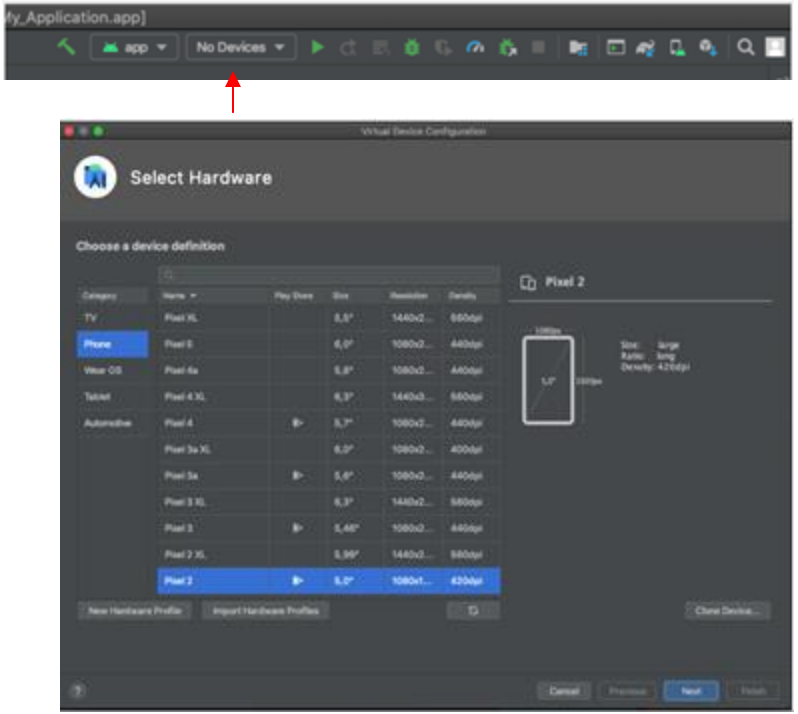
Ejecución de la aplicación



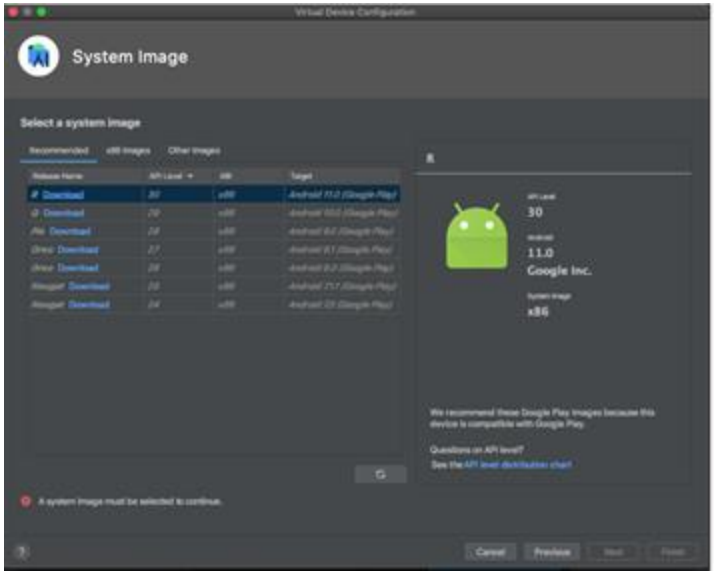
- En la clase **MainActivity** incluiremos las instrucciones necesarias para interactuar con los elementos contenidos en **activity_main**.
- Para activar un **layout** dentro de una **activity** se utiliza el método setContentView()
- Para ejecutar la aplicación: **Run → Run 'app'...**



Instalar un dispositivo



- La siguiente instalación depende un poco del SO Las pantallas pueden cambiar un poco, pero al final es lo mismo.

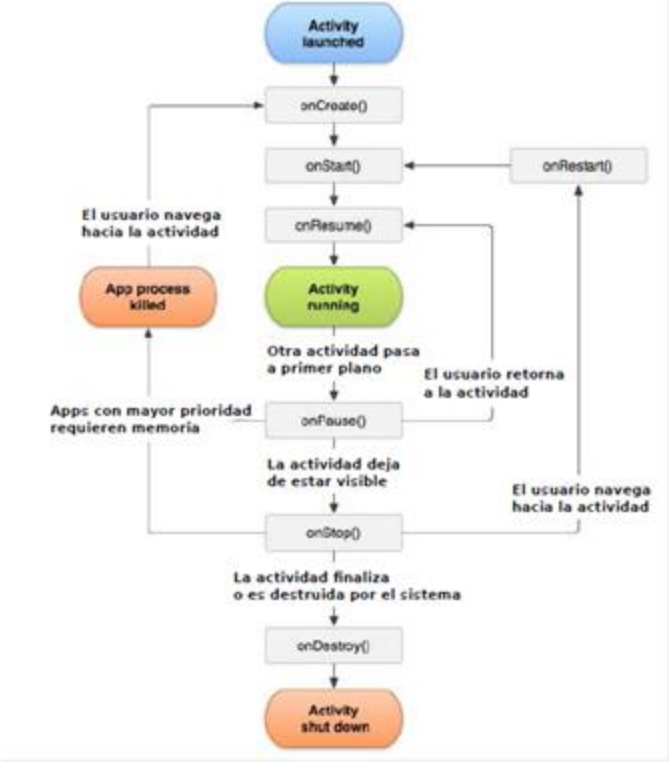


Ejercicio

https://developer.android.com/guide/topics/ui/declaring-layout?hl=es-419#kotlin

- Centrar el control del proyecto creado en la pantalla y modificar su contenido y apariencia.
- Añadir 2 controles "TextView" y practicar con sus atributos.
- Añadir un botón y una imagen

Ciclo de vida de una Activity



Ciclo de vida (Métodos) I

```
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_main)

    // La actividad está creada
}
```

```
override fun onStart() {
    super.onStart();
    Toast.makeText(this, "OnStart",
Toast.LENGTH_SHORT).show();

    // La actividad está a punto de hacerse visible
}
```

Ciclo de vida (Métodos) II

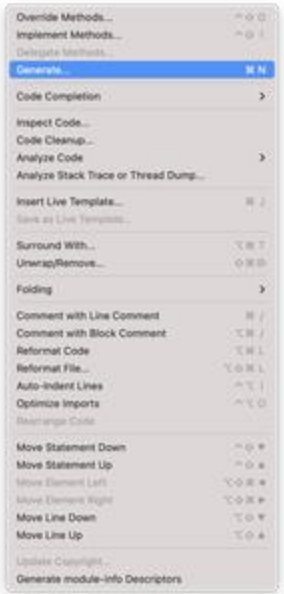
```
override fun onResume() {
    super.onResume();
    Toast.makeText(this, "OnResume", Toast.LENGTH_SHORT).show();

    // La actividad se ha vuelto visible (ahora se "reanuda")
}

override fun onPause() {
    super.onPause();
    Toast.makeText(this, "OnPause", Toast.LENGTH_SHORT).show();

    // Se enfoca en otra actividad (la actividad actual está a punto de ser
"detenida")
}
```

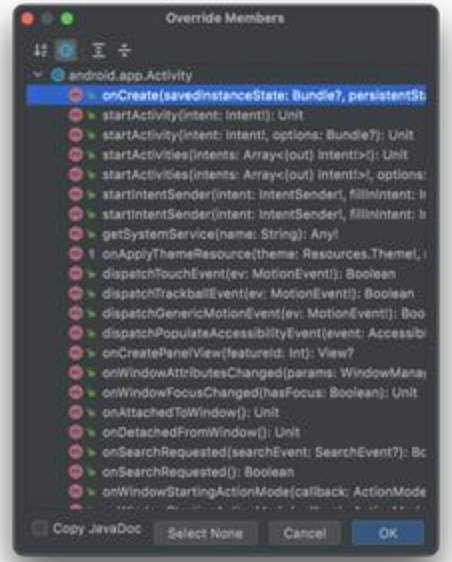
Ciclo de vida (Métodos) III

```
override fun onStop() {
    super.onStop();
    Toast.makeText(this, "OnStop", Toast.LENGTH_SHORT).show();

    // La actividad ya no es visible (ahora está detenida)
}

override fun onDestroy() {
    super.onDestroy();
    Toast.makeText(this, "OnDestroy", Toast.LENGTH_SHORT).show();

    // La actividad está a punto de ser detenida
}
```

Atajos de teclado para generar código

- https://developer.android.com/studio/intro/keyboard-shortcuts





Bibliografía

- https://developer.android.com/