**1.4.1 Recurso Web**

Explicad brevemente (máximo un párrafo) y con un ejemplo los siguientes aspectos de lenguaje Kotlin:

1. **Llamadas seguras (safe calls con ‘?.’)**

Este valor permite que una variable sea nula si se incorpora en la declaración de la misma, y permite comprobar si una variable es nula antes de acceder a las propiedades de esa variable. Por ejemplo, si se intenta acceder a la longitud de un String, se comprueba que dicho String no sea nulo antes de acceder a la propiedad length. En caso de que dicho String sea nulo, se devuelve este valor.

Ejemplo:

var pizza:String? = “pizza”

pizza = null

val porciones = pizza?.length

println(porciones) --> Mostrará null por consola

1. **El operador Elvis en Kotlin**

Si la expresión a la izquierda de ?: no es nula, el operador Elvis la devuelve. De ​​lo contrario, devuelve la expresión a la derecha. La expresión del lado derecho sólo se evalúa si la del lado izquierdo es nulo.

Ejemplos:

val a = b?.length ?: - 1

------------------------------------

fun foo (nodo : Nodo): ¿Cadena? {

val parent = nodo.getParent() ?: return null

nombre val = nodo.getName() ?: throw IllegalArgumentException ( "nombre esperado" )

// ...

}

1. **El operador “!!”**

El operador !! lo que nos permite es convertir cualquier valor a un tipo no nulo, y si éste fuese nulo, lanza una excepción (NullPointerException).

val l = b !!.length

1. **Las variables inmutables (read-only) y variables mutables (val/var)**

Es el equivalente en Java a marcar una variable con el modificador **final**.

Una variable inmutable (val) no alterará su valor en memoria a lo largo del programa (será una constante). Una variable mutable (var) podrá cambiar su valor a lo largo de la implementación.

Ejemplo:

val bbddConecction = “www.google.es:5217”

var flag = 1

1. **Las colecciones inmutables y las colecciones mutables.**

Las colecciones inmutables son colecciones que no se pueden modificar a lo largo de la implementación de un programa. La única forma de añadir un elemento a una lista inmutable, es crear una copia de lista y añadir el nuevo elemento.

val p1 = Person("p1", "s1")

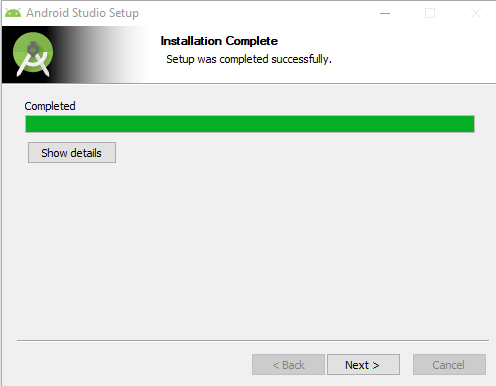
val p2 = p1.copy(surname = "s2")

**1.4.2 Android Studio**

Instala y configura Android Studio, el IDE oficial y recomendado para el desarrollo de aplicaciones Android.

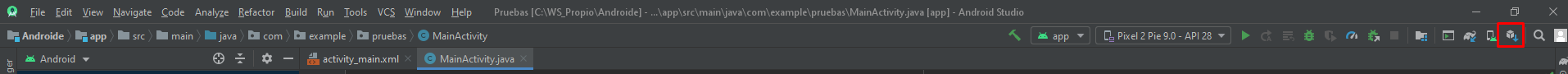
1. **¿En qué sistema operativo has instalado Android Studio? Explica si has tenido alguna incidencia durante el proceso.**

Windows 10. El único problema que he encontrado ha sido a la hora de vitualizar un dispositivo Android, ya que no tenía habilitada la opción Virtualization Technology.

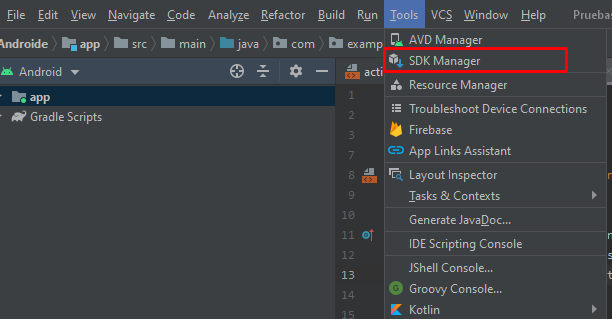


1. **Periódicamente aparecen nuevas versiones de Android y con ellas nuevas versiones del SDK oficial de Android. Si hubiera un nuevo cambio de versión, indica cómo acceder a la ventana que permite actualizar la versión más reciente del SDK.**

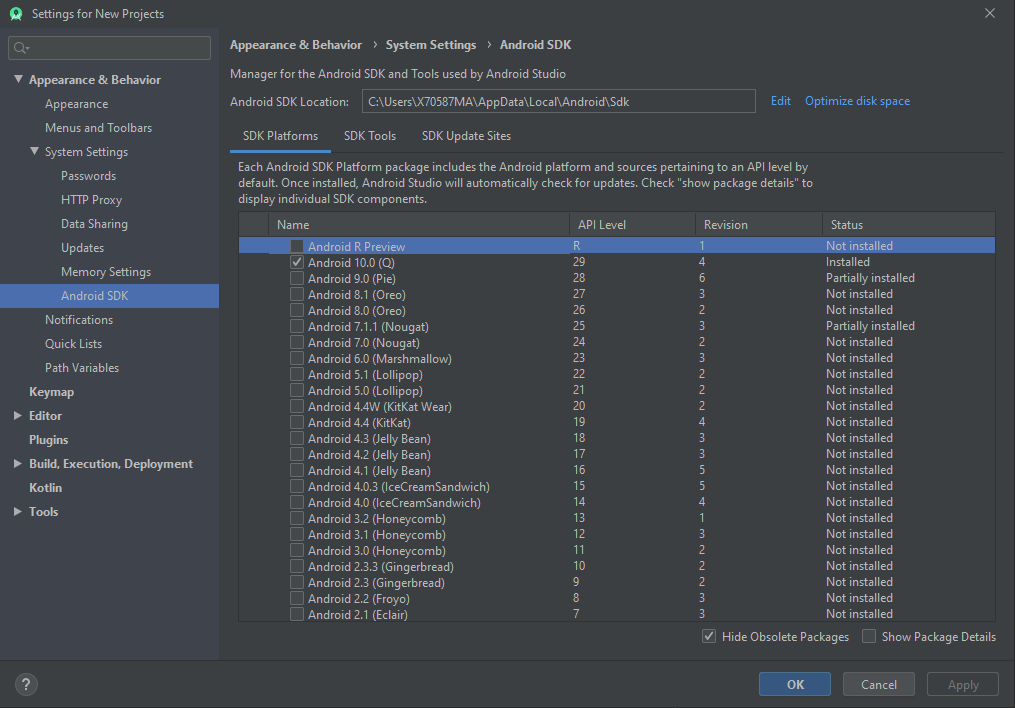
Se puede acceder a través del botón en la pantalla principal:



O a través del menú Tools > SDK Manager:

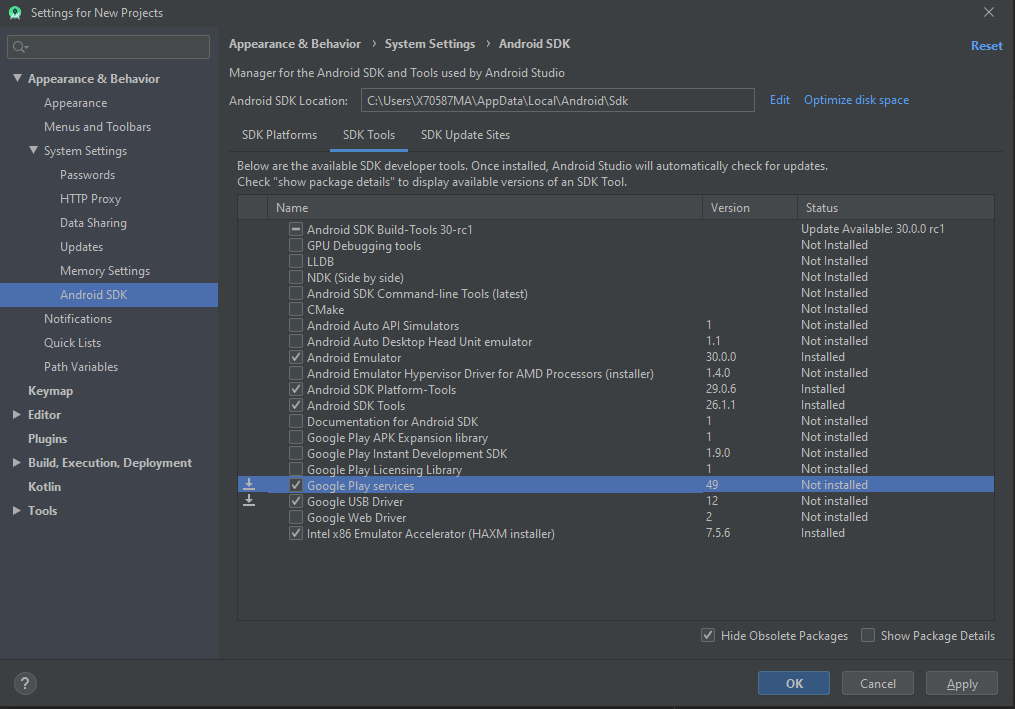


Desde este menú aparecen las versiones del SDK disponibles y podemos descargarlas si las seleccionamos:



1. **Una vez abierto el SDK Manager, instala el paquete Google USB driver (si el sistema operativo donde se ha instalado el Android Studio es Windows), Google Play Services y Intel x86 Emulator Accelerator (si el procesador del ordenador es Intel). Adjunta una captura de pantalla y explica para qué sirven.**

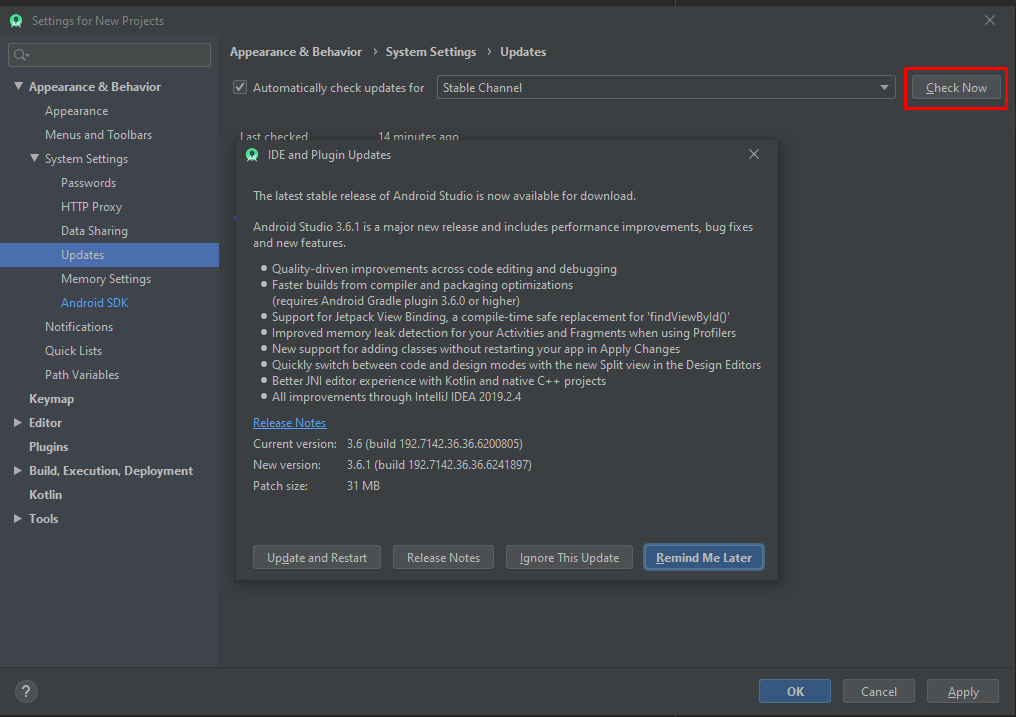
* **Google USB Driver**: Es necesario para realizar una depuración ADB (Android Debug Bridge). Esto permite conectar un dispositivo (como nuestro propio smartphone con Android) al ordenador, instalar la aplicación (apk) en él, y testearla en un dispositivo real.
* **Google Play Services:** Permite a nuestras aplicaciones utilizar los servicios de Google (como Maps, Gmail, Calendario, etc).
* **Intel x86 Emulator Accelerator**: Es un motor necesario para los equipos con procesador Intel que permite la virtualización de dispositivos Android por medio de una máquina virtual (integrada con Android Studio). También es necesario habilitar en la BIOS del sistema la opción “Virtualization Technology”



1. **Android Studio también se va actualizando cada cierto tiempo. Indica qué pasos se deben seguir para obtener la última actualización y dónde podríamos ver qué novedades incorpora.**

Desde el apartado “Updates” del SDK Manager se puede comprobar si existen actualizaciones pendientes para instalar en nuestro equipo.

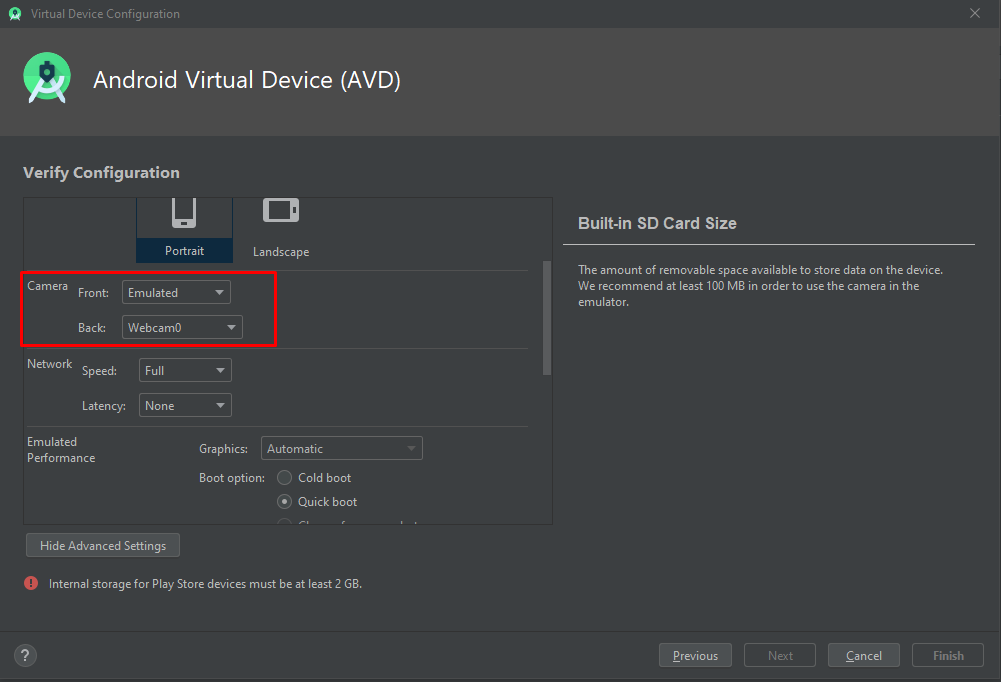
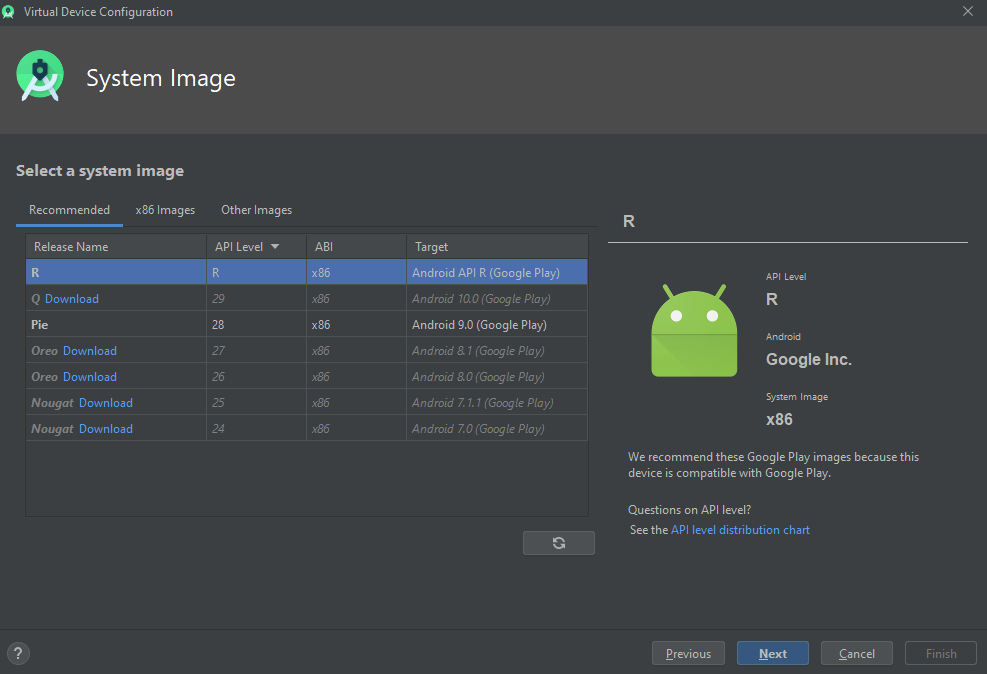
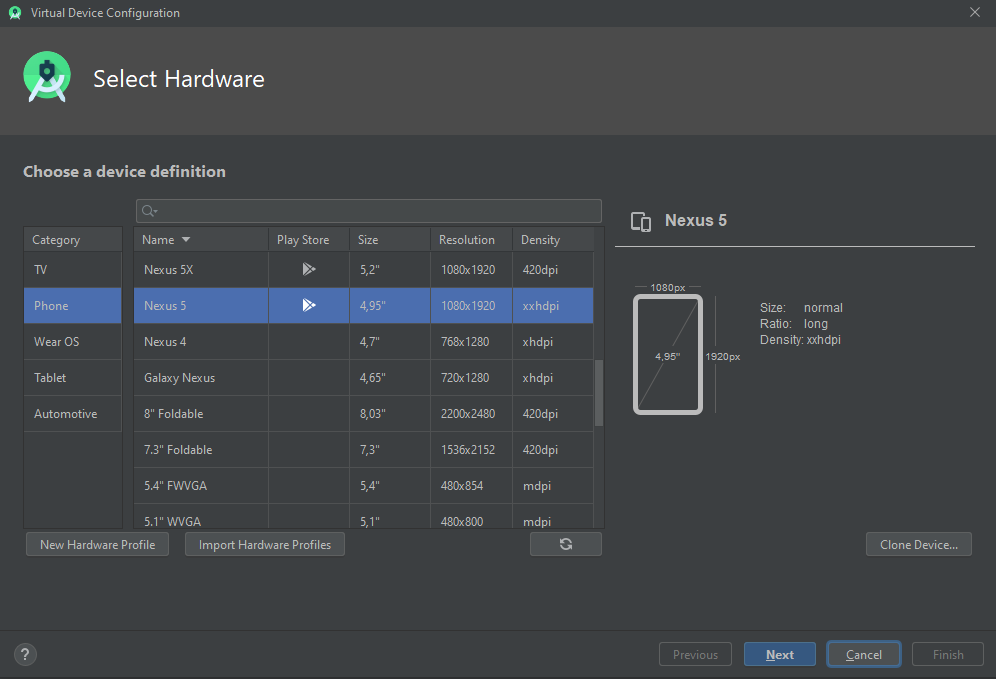
También se muestra una pequeña descripción de la actualización, incluyendo las nuevas características de la misma.

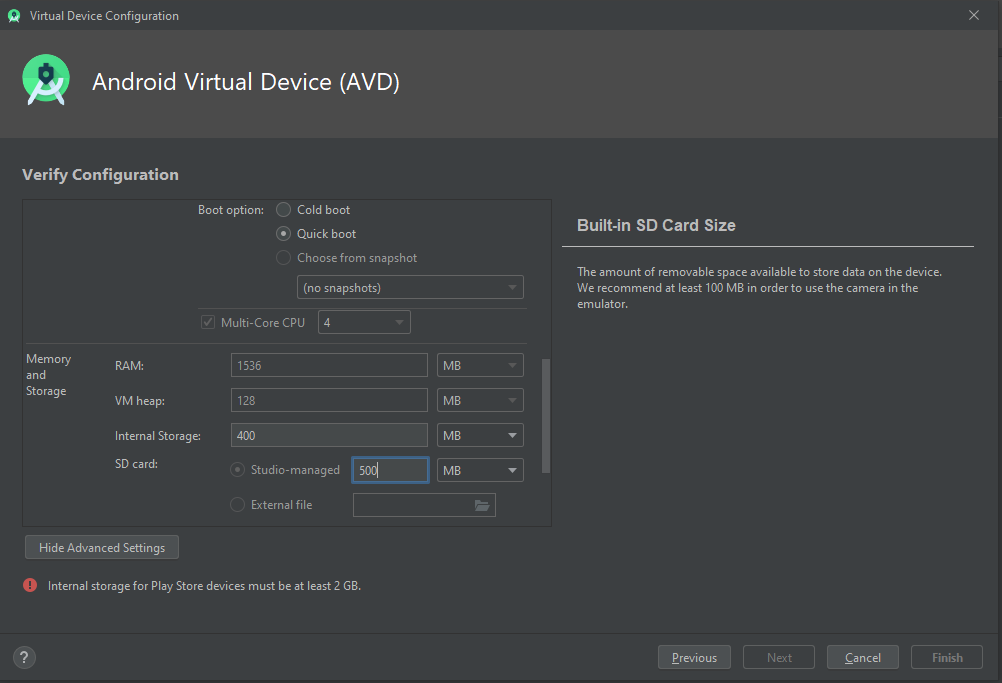


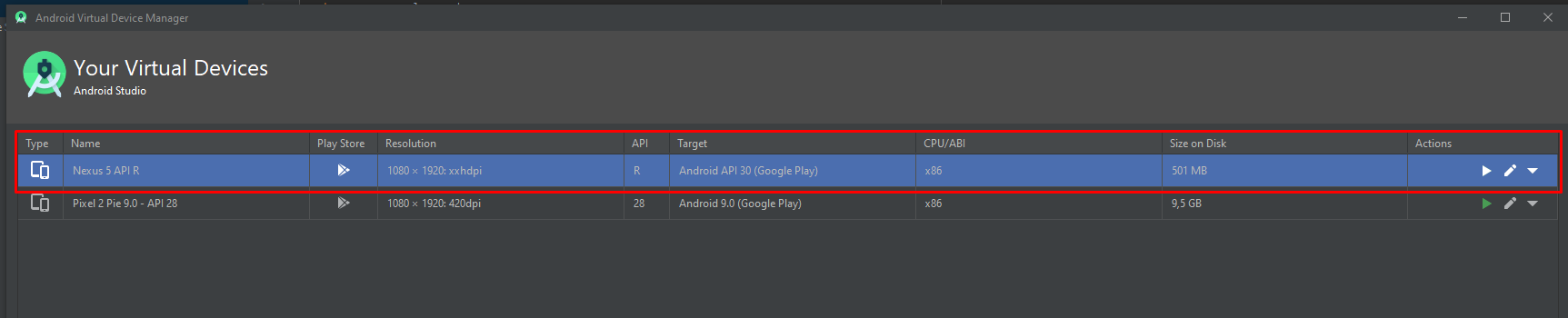
**1.4.3 Emulador**

a) Mediante Android Studio, crea un dispositivo virtual Android. Este dispositivo virtual debe tener las siguientes características:

* Nexus 5 con una pantalla de 4’95’’
* Ultima versión de Android disponible con la API de Google.
* La cámara de detrás debe ser la de la webcam
* Almacenamiento interno de 400 MB
* SD card de 500 MB







**b) (OPCIONAL) Si dispones de un dispositivo móvil con sistema operativo Android configura los drivers para que Android Studio reconozca el dispositivo cuando se conecte al ordenador. Indica qué dispositivo has conectado y explica los pasos que has seguido.**

Realme X2 Pro.

He seguido el siguiente Tutorial: <https://www.youtube.com/watch?v=eaOB-KS-Qjk>

**1.4.4 Componentes Android**

En este ejercicio nos centraremos en conocer los componentes de una aplicación Android, elementos esenciales para la creación de una aplicación Android. Relacionar los siguientes componentes con sus respectivas definiciones:

Componentes

1. Activity -> **e**
2. Service -> **g**
3. Content Provider -> **b**
4. Broadcast receiver-> **c**
5. Fragment -> **a**
6. View -> **f**
7. Layout -> **d**

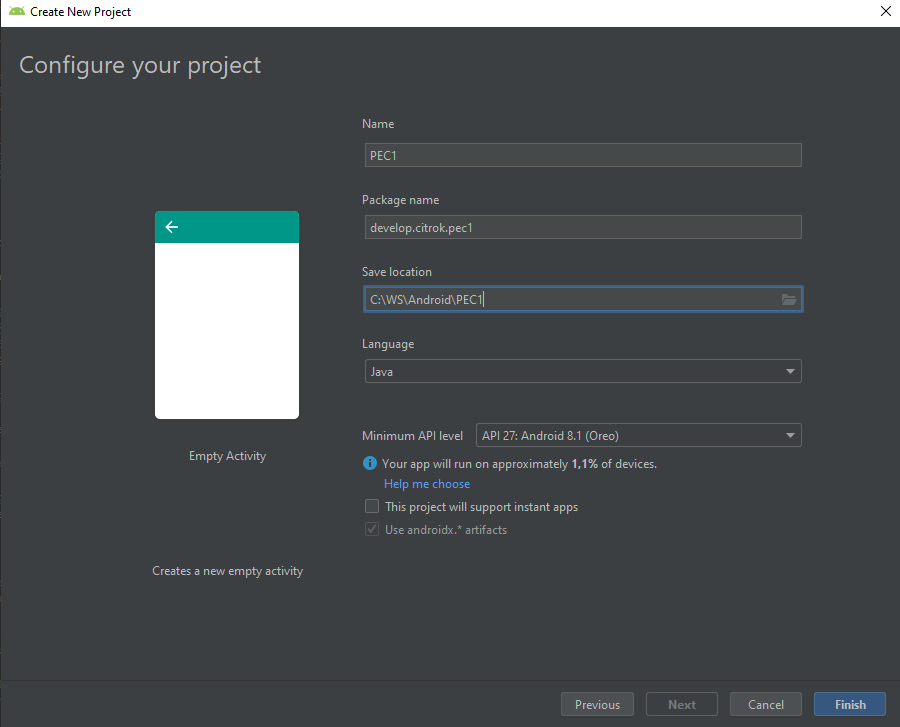
Definiciones

1. Porción de la interfaz de usuario.
2. Mecanismo para compartir datos entre aplicaciones
3. Componente dedicado a recibir y responder eventos
4. Contenedor que controla el formato visual de la pantalla.
5. Componente Android que proporciona una pantalla de la aplicación con su interfaz
6. Elementos de la interfaz de usuario que se dibujan en la pantalla
7. Tarea que se ejecutan en segundo plano

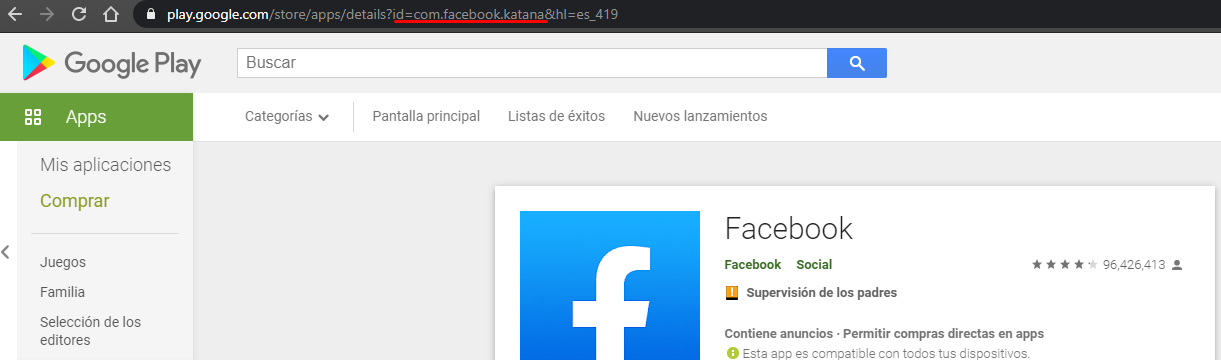
**1.4.5 Primera Aplicación (Valoración de un 30%).**

En este ejercicio se creará una aplicación y se ejecutará en el emulador o el dispositivo móvil. Tal y como se explica en el apartado 2.2. de la wiki, el primer paso para crear una aplicación es crear un proyecto en Android Studio.

1. Crea un proyecto Android (basado en el lenguaje Java) mediante Android Studio (escoge el tipo de proyecto “Empty Activity”). Explica todos los parámetros seleccionados para la creación del proyecto y justifica el motivo de su elección.



* **Name:** Nombre de la aplicación.
* **Package name:** Es el identificador único que utiliza Google Play Store para encontrar la aplicación. Este se puede consultar una vez subida la app a Google Play Store, mediante la URL:

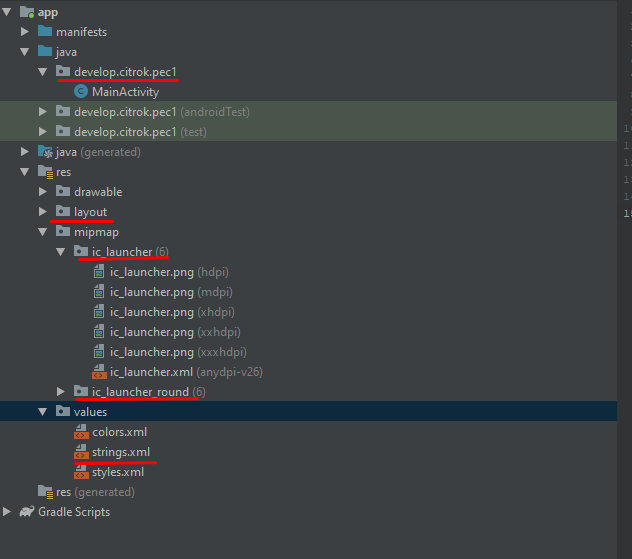


* **Save location:** Ubicación en la que se va a almacenar el proyecto dentro del equipo.
* **Language:** Lenguaje de programación en el que se va a desarrollar la aplicación (Java o Kotlin).
* **Minimum API level:** Versión mínima de Android necesaria en el teléfono para poder correr la aplicación.
* **This project will support instant apps:** Esta función significa que la aplicación pasa a formar parte de las Instant Apps de Google Play Store. Esta funcionalidad permite a los usuarios descargar parcialmente la aplicación y usar sus funcionalidades sin necesidad de descargarla completamente en su teléfono.
* **Use android.\* artifacts:** Indica si se quiere importar la librería de compatibilidad de androidx. Esta librería permite la compatibilidad de nuestra aplicación con versiones anteriores de Android.

1. Una vez creado el proyecto, estudia cómo está estructurado el proyecto. ¿Dónde encontramos…?
   * **El icono del proyecto:**

res/mipmap/ic\_launcher/ o res/mipmap/ic\_launcher\_round/

* + **El código fuente:** java/develop.XXX.nombreAplicacion/
  + **La interfaz gráfica:** res/layout/
  + **Los textos de la aplicación (strings):** res/values/strings.xml



1. Explica qué definen y para qué sirven los siguientes archivos:
   * **AndroidManifest.xml:**

Define parámetros propios de la aplicación

* + **Build.gradle(el que corresponde a la aplicación):**

1. Cambia el texto del TextView y escribe tu nombre. Adjunta una captura de pantalla del dispositivo con este cambio.
2. Después de ejecutar el proyecto, en qué ruta se encuentra el archivo .apk (archivo que contiene la aplicación generada).
3. Explica qué tipo de Logs hay en Android y añade uno al final del método onCreate() del archivo MainActivity. Como etiqueta (Tag), pon el nombre de la clase y como texto del mensaje tu nombre. Copia la línea de Log que has utilizado.
4. Añade una segunda actividad Java en el proyecto (New \ Activity \ Empty Activity) y, utilizando un Intent, iníciala desde la primera actividad. De la misma manera que en el punto d) modifica el texto del TextView indicando que se trata de la segunda actividad.
5. Utiliza la opción code de Android Studio para convertir la primera actividad (que está implementada en Java) en Kotlin (convert Java File to Kotlin File) y verifica la interoperabilidad del lenguaje Kotlin.