GENERALITAT VALENCIANA
Conselleria d'Educació, Cultura i Esport

ceedcv CENTRE ESPECÍFIC D'EDUCACIÓ A DISTÀNCIA DE LA COMUNITAT VALENCIANA

TEMA 2: DISEÑO DE LA INTERFAZ DE USUARIO

PROGRAMACIÓN MULTIMEDIA Y DISPOSITIVOS MÓVILES

Cicle Formativo de Grado Superior en Desarrollo de Aplicaciones Multiplataforma
Departamento de Informática // 2º curso 2024-2025

MARA VAÑÓ ALONSO

Cicles Formatius

Índice

1. OBJETIVOS DE LA UNIDAD

- Conocer la importancia de un buen diseño de interfaz de usuario
- Conocer la arquitectura Android con un poco más de profundidad
- Aprender a configurar el ED de Android Studio
- Iniciarnos en la programación en Android Studio
- Saber cómo publicar una aplicación en Google Play Store
- Conocer más sobre los permisos en Android

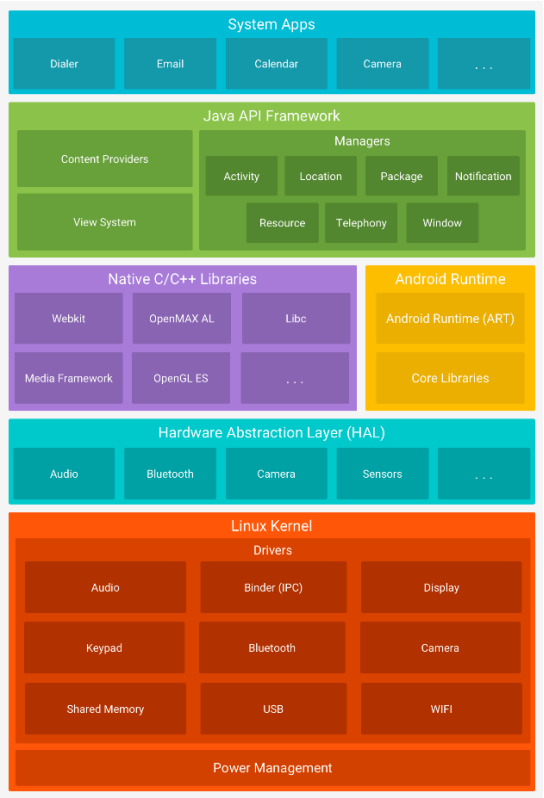
2. IMPORTANCIA DE UN BUEN DISEÑO DE INTERFAZ DE USUARIO

El Diseño de la interfaz de usuario cobra cada día más importancia en el desarrollo de una aplicación llegando a ser una parte importante en el éxito o fracaso del proyecto. Así, una buena interfaz será clave para llegar a nuestro usuario final. Es importante tener en cuenta que cuando desarrollamos en Android Studio la filosofía de diseño será muy diferente a la que hemos utilizado en otras plataformashasta el momento ya que tenemos que tener en cuenta las limitaciones como el uso de la RAM, la capacidad de ejecución del dispositivo o la limitación en cuanto a personalización o actualizaciones

2.1 ARQUITECTURA ANDROID

Tal vez quedaría mejor algo como:.Tal y como comentamos en la UD1, el hecho de que Android estábasado en Linux permite que su adaptación a multitud de dispositivos de diferentes tipos (móviles, tablets, IOT..) sea algo habi-

tual. Parte de esa facilidad de adaptación reside en la descomposición en capas de todo el sistema. Una forma gráfica de ver esos bloques puede verse en la siguiente figura:



Linux Kernel

Es la base de la plataforma Android y su uso permite que Android aproveche funciones de seguridad claves, además permite a los fabricantes desarrollar controladores de hardware para un Kernel conocido.

HAL (Capa de abstracción)

La capa de abstracción proporciona interfaces estándar y consiste de varios módulos de biblioteca queimplementan una interfaz para un hardware específico (cámara de fotos). Cuando una API hace una llamada para acceder a ese hardware, Android carga el módulo de biblioteca correspondiente.

Android Runtime

Actualmente cada app ejecuta sus procesos con sus instanciar del ART (Android Run Time), que estáescrito para ejecutar máquinas virtuales en dispositivos de memoria baja mediante archivos DEX, diseñados para optimizar el espacio ocupado.

Bibliotecas nativas

Se pueden usar en varios componentes de Android, están compiladas en código nativo del procesadory muchas son de código abierto.

Algunas son:

- System C library: adaptada para dispositivos embebidos basados en Linux.
- Media Framework: basada en OpenCORE de PacketVideo. Soporta codecs de reproduccióny grabación de audio y vídeo e imágenes.
- Surface Manager: maneja el acceso al subsistema de representación gráfica 2D/3D
- SGL: motor de gráficos 2D.
- FreeType: fuentes en bitmap y renderizado vectorial.
- SQLite: motor de bases de datos relacionales. SQLite está disponible para todas lasaplicaciones.
- SSL: proporciona servicios de encriptación Secure Socket Layer (capa de conexión segura).

JAVA API Framework

Es el marco de trabajo de la API de Java, una API es “un conjunto de definiciones y protocolos que seutilizan para desarrollar e integrar el software de las aplicaciones. API significa interfaz de programación de aplicaciones. Las API permiten que sus productos y servicios se comuniquen con otros, sin necesidad de saber cómo están implementados” utilizaremos APIs para crear

APPs mediantereutilización de componentes como, por ejemplo:

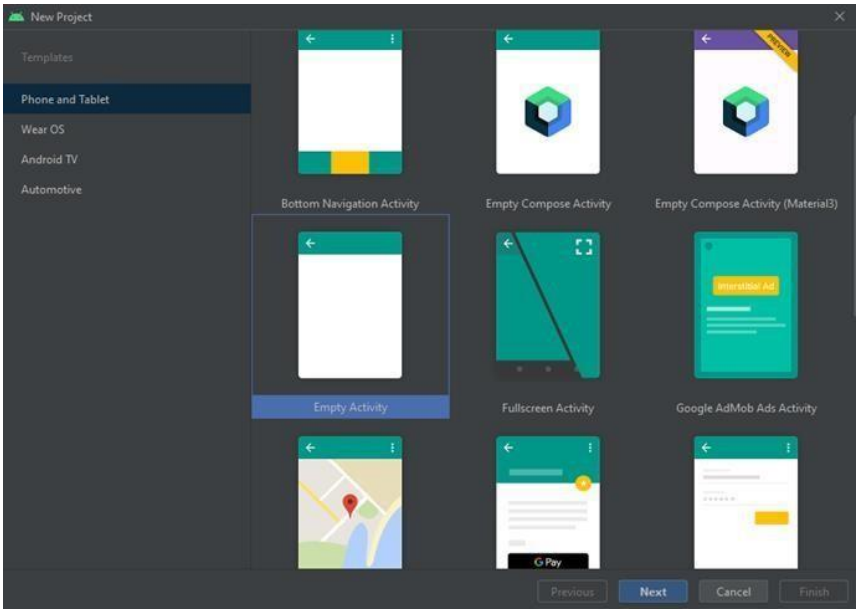
- Proveedores de contenido: acceso a datos desde otras apps.
- Administrador de actividad: que administra el ciclo de vida
- Administrador de notificaciones: permite mostrar alertas

APPs del sistema

Son aplicaciones básicas como las relacionadas con los SMS, el calendario, la cámara de fotos… En este caso el usuario no puede desinstalarlas ni hacer cambios sobre ellas. Pueden utilizarse para haceralgunas de las tareas de nuestra aplicación.

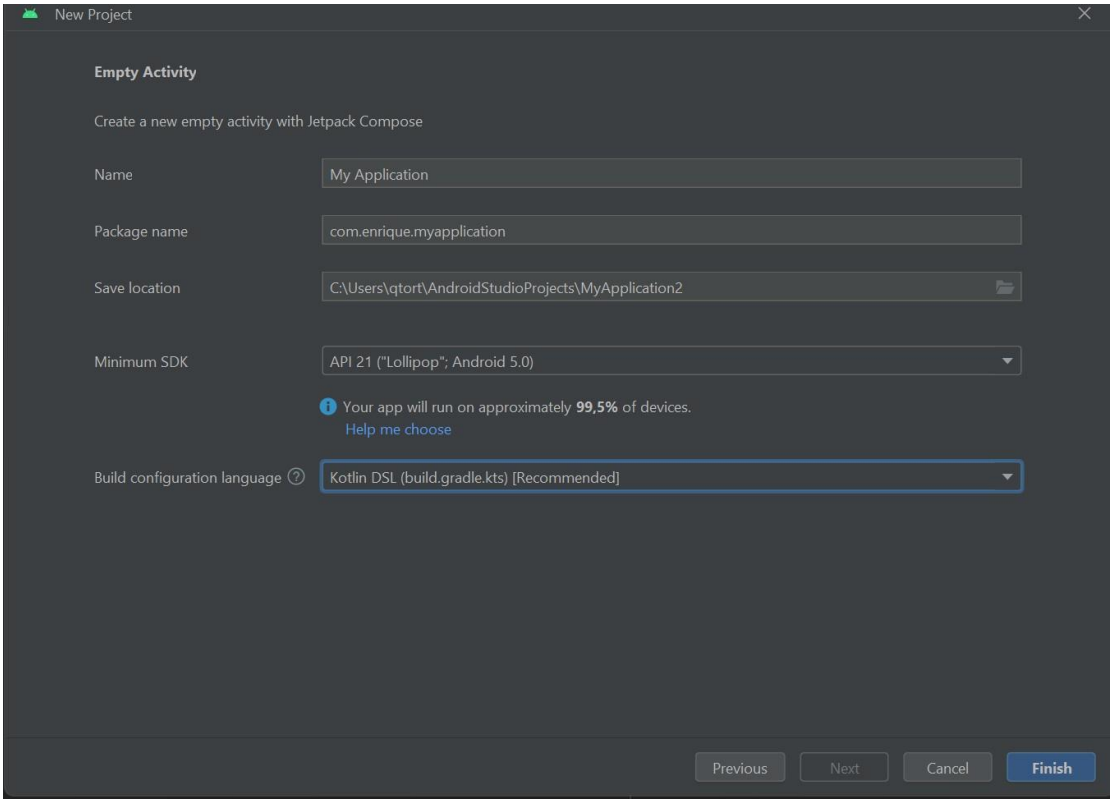
3. ENTORNO DE DESARROLLO EN ANDROID STUDIO

Una vez conocemos mejor la arquitectura de Android y hemos terminado la instalación, es el momentode iniciar la primera aplicación y para ello deberemos antes definir algunas características del ED. Lo primero que tenemos que elegir es el tipo de vista de la actividad principal, podemos elegir la queconsideremos mejor, aunque se recomienda elegir la “Empty Activity”.

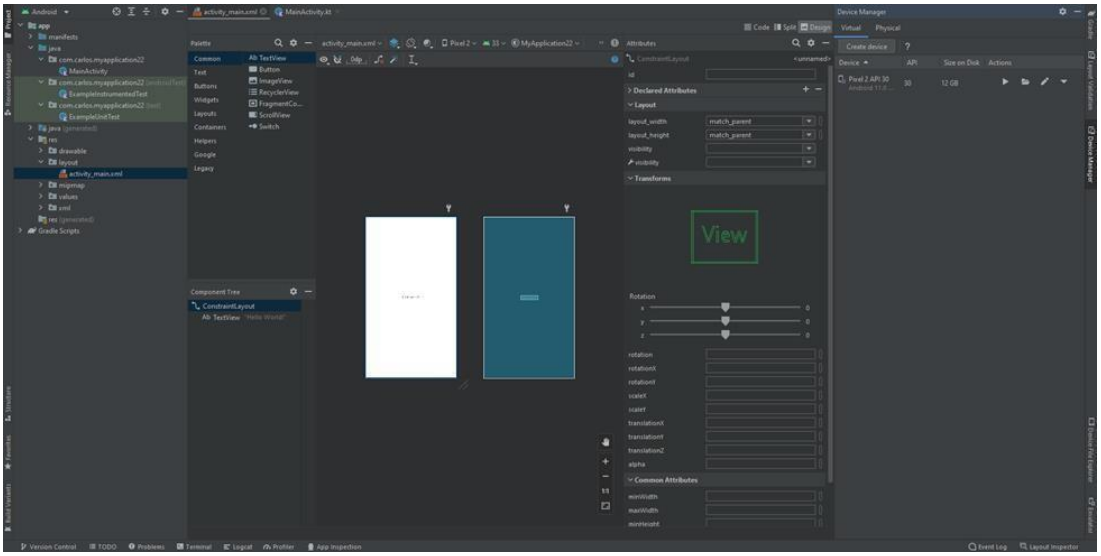


A continuación, nombraremos la aplicación, el dominio (PACKAGE) de la aplicación y su ubicación.

En cuanto a la versión mínima, en esta pantalla te muestra la compatibilidad que tendría la aplicación,si no estás seguro, puedes pulsar sobre "**Help me choose**" y ver la compatibilidad de cada versión. Encuanto al uso de "legacyandroid.support.libraries", hasta hace poco, casi todas usaban esto, pero Google anunció que ya no habría soporte para esto y que a partir de ahora se haría con una librería llamada AndroidX.



En la siguiente imagen vemos por un lado el explorador de proyectos donde veremos todas las carpetas y ficheros que componen nuestra aplicación. En la parte superior verás dos pestañas, una en la que aparece el ficheroXML y en otra el fichero JAVA. En XML declaramos los componentes y en JAVA los comportamientos.Además, puedes elegir en XML si prefieres ver el texto, una previsualización de la aplicación o ambas.



4. INTRODUCCIÓN DE PROGRAMACIÓN

Lo primero que tenemos que recordar es que no programaremos únicamente en Java, también veremos programación en XML (y mediante interfaz gráfica, como hemos visto). Recordemos que uno de los principios del diseño de soft- ware es separar DISEÑO, DATOS Y LÓGICA, siguiendo el patrón MVC que propone separar el código por sus responsabilidades, de tal forma que se mantienen capas (Modelos, Vistas, Controladores) encargadas de hacer ta- reas concretas, lo cual ofrece beneficios:

Separamos el Modelo de la Vista, es decir, los datos de su representación visual, siendo más sencillo agregar diferentes representaciones de los datos. Esto tiene las siguientes ventajas:

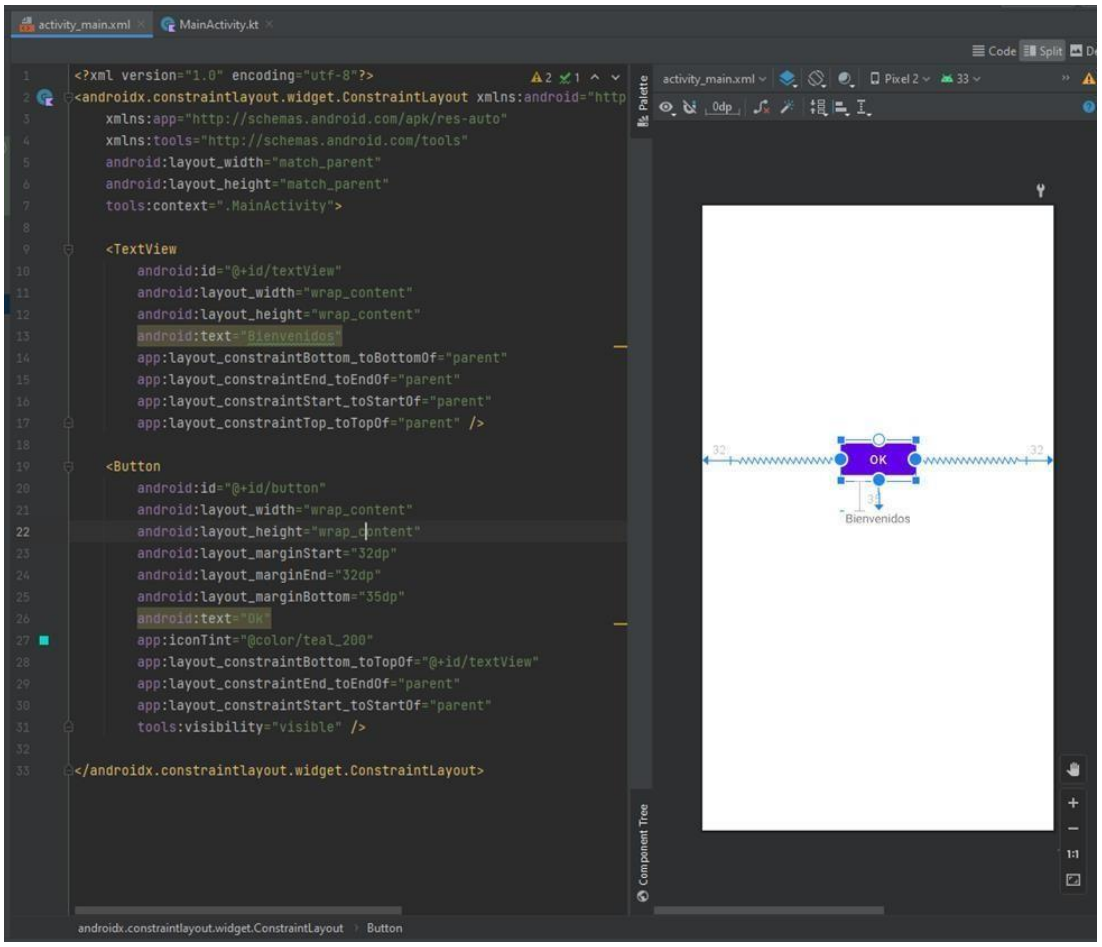
- Crea independencia de funcionamiento
- Se facilita el mantenimiento en caso de errores
- Permite escalar

Pero presenta algunos inconvenientes:

- Se agrega complejidad al sistema

- Se incrementa el número de archivos a mantener y desarrollar.

A continuación, vamos a ver uno de los ficheros. Accedemos a res/layout/activity_main.xml y vamos a la vista de código.



La interfaz está basada en una jerarquía de clases descendientes de View (vista). Una vista es un objeto que puede dibujarse y se utiliza como elemento de diseño

En la imagen de arriba vemos que se introduce un elemento: **androidx.constraintlayout.widget.ConstraintLayout** que contendrá al resto de elementos tipo view,con los diferentes atributos. Dentro de este, tenemos un elemento de TextView (vista de texto) con sus atributos. en este caso: android:text=""

Una vez tenemos claro cómo se estructura, utilizando los conocimientos

aprendidos en la asignatura de Programación podremos ir construyendo nuestro código con las diferentes subclases de View.

5. PUBLICAR UNA APP EN GOOGLE PLAY STORE

Una vez tenemos creada nuestra APP es necesario preparar la aplicación para ser publicada. Históricamente se ha hecho a través de APK, pero desde agosto de 2021 las apps deben publicarse con Android App Bundle. Puede verse la diferencia perfectamente explicada en:

https://www.xatakandroid.com/programacion-android/app-bundles-android-que-que-se-diferencian-apk

PK (Android Application Package) es el archivo donde los datos de la aplicación se encuentran comprimidos. Estos archivos podrán compartirse entre dispositivos y para abrirlos, simplemente pulsamos sobre ellos. ¿Es necesario y suficiente el archivo APK? El archivo APK será necesario siempre pero no siempre suficiente, esto implica que a veces será necesario descargar otros paquetes para que la aplicación funcione. Estos son los llamados XAPK, que contienen el APK y el OBB, que es un archivo con datos adicionales.

Para publicar la aplicación en la Play Store necesitaremos generar una APK firmada. **La firma identifica al autor de la aplicación a Google y a los usuarios que la deseen instalar**. Sin embargo, si mantenemos Android Studio en modo desarrollo no será necesario firmar la APK, pero recuerda será necesario para poder ponerla a disposición del público. Para generar una APKfirmada, seguiremos lo indicado en developer.android.com “Para firmar y publicar una app nueva en Google Play:

1. Genera una clave de carga y un almacén de claves.
2. Firma la app con la clave de carga.
3. Cómo configurar la firma de apps de Play

4. Sube la app a Google Play.
5. Prepara e implementa el lanzamiento de la app

En cambio, si tu app ya está firmada y publicada en Google Play Store con una clave de firma de la app existente, o si deseas usar una clave de firma de la app para una app nueva en lugar de la que creaGoogle, puedes generarla mediante los siguientes pasos:

1. Firma tu app con la clave de firma de la app y **selecciona** la opción para encriptar y exportar la clave de firma.
2. Sube la clave de firma de la app a la firma de apps de Play.
3. Recomendación: Genera y registra un certificado de carga para actualizaciones futuras de tu app. 4.Sube la app a Google Play.

5. Prepara e implementa el lanzamiento de la app

A la hora de hacer esto revisa siempre las instrucciones de Google, pues se actualizan constantemente y lo publicado aquí puede no estar actualizado en el momento en que vayáis a ejecutarlo.

6.1 ANDROID APP BUNDLE

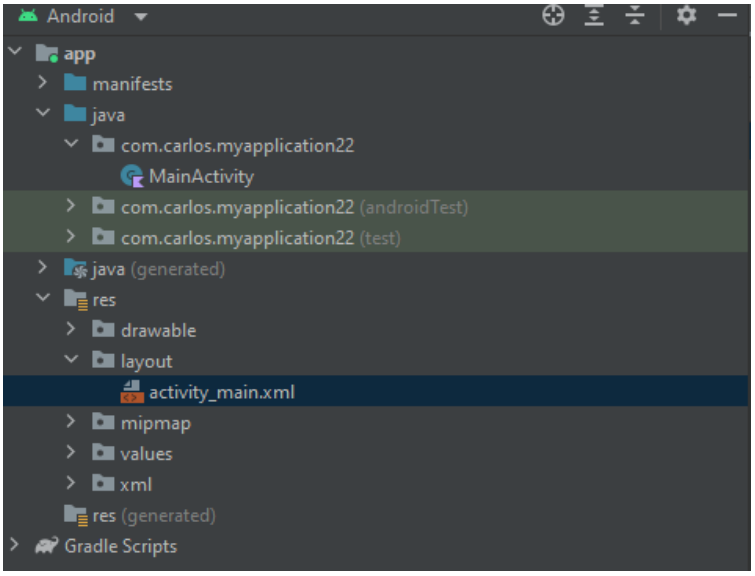
Desde el 21/08/2021 las aplicaciones deberán publicarse con App Bundle, un formato que incluye los recursos y el código, pero en el que la generación y la firma del APK la hará Google Play utilizando el App Bundle. Así, **se permite una mejor optimización ya que se utilizarán solo los recursos necesarios para el dispositivo en el que se vaya a realizar la instalación, logrando así una optimización de las descargas**. Otro de los cambios introducidos es que el límite del tamaño de descarga comprimido es de150 MB. Este límite afecta a módulos de funciones, pero NO afecta a los paquetes de elementos, que tienen sus propias restricciones de tamaño, que oscilan entre 512MB y los 2GB.

Importante

Si el peso del App Bundle es >150MB se producirá un error en la subida.

7. PERMISOS EN ANDROID

Una de las cosas más importantes a la hora de desarrollar para Android es tener en cuenta la seguridad. Los permisos los debemos declarar en el AndroidManifest.xml, y esta información será mostrada al usuario incluso antes de proceder a la instalación de la aplicación. Podremos verlo en el menú de la izquierda.



Los usuarios serán libres de conceder los permisos que consideren. Por ejemplo, para jugar aPOKEMON GO será necesario compartir nuestra ubicación, sin embargo, no podemos obligar al usuario y este tiene que dar su permiso de forma inequívoca. Otro ejemplo podría ser una aplicación de escaneado de documentos, en este caso deberíamos solicitar permisos para acceder a la cámara (para

tomar la captura) y para acceder al almacenamiento del teléfono para guardarla. Evidentemente, sin estos permisos, la aplicación no tendrá utilidad, sin embargo, no nos exime de preguntar.

Por ello, no solo debemos manifestar los permisos en el Manifest, también debemos asegurarnos de que se solicitan los permisos (en ocasiones incluso antes de cada uso).

Tan malo es no pedir un permiso que si vamos a necesitar para ejecutar la aplicación como pedir demasiados permisos que no son necesarios, ya que pueden hacer que el usuario cambie de opinión y desista en la instalación. Un consejo: puedes pedir los permisos necesarios para la instalación y la ejecución básica y luego, conforme el usuario necesite hacer uso de otras utilidades, solicitar el permisoen ese momento.

Android Manifest y la seguridad a nivel de una aplicación de Android

El archivo AndroidManifest.xml es un archivo XML que se encuentra en la raíz del proyecto de una aplicación de Android. Este archivo es esencial para la construcción de una aplicación de Android, ya que describe información importante sobre la aplicación a las herramientas de construcción de Android, el sistema operativo Android y Google Play1.

El archivo AndroidManifest.xml describe los siguientes elementos esenciales de una aplicación:

- **Los componentes de la aplicación**, incluidas todas las actividades, servicios, receptores de difusión y proveedores de contenido. Cada componente debe definir propiedades básicas, como el nombre de su clase

Kotlin o Java. También puede declarar capacidades, como qué configuraciones de dispositivo puede manejar y filtros de intención que describen cómo se puede iniciar el componente.

- **Los permisos que la aplicación** necesita para acceder a partes protegidas del sistema u otras aplicaciones. También declara cualquier permiso que otras aplicaciones deben tener si quieren acceder al contenido de esta aplicación.
- **Las características de hardware y software que requiere la aplicación**, lo que afecta a los dispositivos que pueden instalar la aplicación desde Google Play.

Las aplicaciones se ejecutan dentro de una sandbox gracias al funcionamiento de Android con la máquina virtual Dalvik. Así se evitan posibles fallos de seguridad entre aplicaciones y el propio sistema operativo. Como el sistema de procesos en sandbox no permite, por defecto, acceso a recursos del sistema que no sean los de la propia aplicación, es necesario que estos sean explícitamente declarados. Para declarar los permisos, cada aplicación proporciona dentro de su fichero .apk el archivo **AndroidManifest.xml**

Ejemplo de permisos de acceso a Internet y de escritura con las etiquetas usses-permission que declaran de forma explícita los permisos de la app:

```
<?xml version="1.0" encoding="utf-8"?>

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

    <application
         […]> […]
    </application>


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

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

</manifest>
```

Luego los permisos declarados en **AndroidManifest.xml** app deberá aceptar estos permisos durante la instalación en deberán ser aceptados durante la instalación por el usuario. También se pueden declarar permisos de acceso a otras aplicaciones. **No es hasta la versión 6.0 de Android que se permiten activar los permisos de manera independiente**.

Los permisos dela última versión de Android se pueden ver en :

https://developer.android.com/reference/android/Manifest.permission

Seguridad en la firma de aplicaciones

Por seguridad Android obliga el firmado de aplicaciones. Las aplicaciones que se intenten instalar sin haber sido firmadas son rechazadas tanto por Google Play como por el propio dispositivo de Android. Con la firma de la aplicación se identifica inequívocamente al creador de la aplicación. **Si dos aplicaciones tienen la misma firma, existe la posibilidad (configurándolo en el AndroidManifest.xml) de que cuando las dos aplicaciones se instalen en el mismo dispositivo compartan un mismo identificador de usuario UID**. De esta manera, las dos aplicaciones compartirán los recursos que les han sido asociados

Sin embargo, diferencia del App Store de Apple, Google no firma las aplicaciones Android ni proporciona certificados a los desarrolladores de confianza. En Google Play, **cada desarrollador puede generar su propio certificado autofirmado para firmar sus aplicaciones, lo que garantiza que la actualización de una aplicación o un grupo de aplicaciones sea del mismo desarrollador**. Aunque Google tiene un proceso para auditar las aplicaciones que se suben al Play Store y borra las aplicaciones que se reportan como malware, es bastante común encontrar muchas aplicaciones maliciosas en Google Play Store. **Estas aplicaciones maliciosas pueden ser copias de aplicaciones legítimas o aplicaciones que dicen ser una cosa pero tienen muchos permisos innecesarios, como la consulta de la lista de contactos**.

Es importante tener en cuenta que aunque Google Play Store tiene un proceso para auditar las aplicaciones que se suben al Play Store y borra las aplicaciones que se reportan como malware, **los usuarios deben tener precaución al descargar aplicaciones y verificar los permisos requeridos por la aplicación antes de instalarla.**

Recuerda, en la unidad 5 veremos el diseño de la interfaz con vistas de forma más exhaustiva.

8. BIBLIOGRAFÍA

i. Android. Programación Multimedia y de dispositivos móviles. Garceta.
ii. Programación Multimedia y Dispositivos Móviles. Editorial Síntesis.
iii. Android App Development. For Dummies.
iv. Beginning Android Programming with Android Studio. Wrox.
v. Java Programming for Android Developers. For Dummies.
vi. https://academiaandroid.com/
vii. https://developer.android.com/
viii. https://www.redhat.com
ix. https://desarrolloweb.com
x. https://www.xatakandroid.com/programacion-android/app-bundles-android-que-que-se-diferencian-apk
xi. https://developer.android.com/reference/android/Manifest.permission