Universidad de Valladolid

ESCUELA DE INGENIERÍA INFORMÁTICA

GRADO EN INGENIERÍA INFORMÁTICA
MENCIÓN EN TECNOLOGÍAS DE LA INFORMACIÓN

AppPerm: Apps para obtener conocimiento sobre la gestión de permisos Android

---

TRABAJO DE FIN DE GRADO

Alumno:
De la Cruz Garijo, Alejandro

Tutora:
Martínez González, Mercedes

Agradecimientos

Me gustaría agradecer a todas las personas que me han ayudado y apoyado durante estos años de carrera y en especial durante la realización de este trabajo. A mi tutora Mercedes por su apoyo y dedicación. A Amador Aparicio por sus consejos y su ayuda. A todos los profesores que durante estos años me han enseñado lo necesario para realizar este trabajo. A mis padres y familiares por apoyarme siempre en mis estudios y animarme con ellos. A mis amigos por animarme y estar siempre ahí para ayudarme. Finalmente, a mis compañeros de Invicsa Airtech que me admitieron antes de haber terminado la carrera y me han apoyado en todo lo necesario para poder completar este trabajo.

Resumen

Los permisos en Android son un mecanismo que permite a los desarrolladores y a la propia Google mantener la privacidad y seguridad de sus usuarios restringiendo el acceso a acciones y datos sensibles. En este trabajo se hará una exploración de cómo funcionan estos permisos y se responderá a algunas dudas que pueden surgir a los que estén empezando con ellos haciendo uso de una aplicación Android y otra para PC que desarrollaremos para este fin.

Abstract

Android permissions are one of the ways developers and Google can assure the privacy and security of their users by restricting the access to sensitive actions and data. By using an Android application and a PC application that we are going to develop in this project, we will explore how these permissions work and provide answers to some of the questions that beginners may encounter.

Índice general

Capítulo 1

Introducción

1.1. Contexto

Nos encontramos en una época en la que la gran mayoría de la población dispone o tiene acceso a dispositivos móviles con los que podemos hacer cada vez más cosas como llamadas telefónicas, compras online, tomar notas, actividades de entretenimiento, mensajería instantánea, etc. Todas estas funciones son posibles gracias a los avances tecnológicos que han sufridos estos dispositivos en la última década, sin embargo, este avance tan rápido ha hecho que surjan problemas con el uso que los usuarios hacen de estos dispositivos y el descuido de los desarrolladores con respecto a otras funciones que no todos los usuarios tienen en cuenta.

Una de esas funciones es la de la privacidad, hoy en día podemos tener una gran cantidad de datos personales en nuestros dispositivos, imágenes, audios, cuentas bancarias, información de localización, navegación web y muchos más. Para tratar de proteger a los usuarios contra el robo de esta información, los desarrolladores implementan en sus sistemas distintas medidas de seguridad. En este caso nos centraremos en el sistema de permisos de Android, un Sistema Operativo que utilizan la mayoría de los dispositivos móviles.

En los inicios de Android [7] estos permisos estaban muy descuidados y resultaba bastante sencillo acceder a los datos de los usuarios siempre que estos tuviesen una conexión a internet. En los últimos años, sin embargo, se han ido introduciendo nuevas medidas para poder controlar estos permisos en función de las necesidades de cada aplicación, el problema está en que hay una gran cantidad de permisos organizados en distintos grupos y no ofrece una cantidad suficiente de información en la web para los desarrolladores de Android. Estos permisos pueden ser de distintos tipos según su nivel de protección que afecta a la facilidad de conseguir acceso a ellos. Por esto, en este proyecto se realizará un estudio de estas funciones y estos permisos con el objetivo de realizar un plan de pruebas que nos permita ver como se gestionan los permisos en los sistemas Android.

1.2. Motivación

Como se ha dicho en el apartado anterior, la información que podemos encontrar en la web de Android para desarrolladores es muy limitada y no explica cómo es la organización de los permisos y sus grupos, cuáles son las opciones con respecto a estos y tampoco hay información que permita identificar fácilmente las diferencias entre los permisos de Android y los permisos creados por los desarrolladores.

El problema que genera esta falta de información es que es difícil para los desarrolladores entender cómo funcionan estos permisos y cuáles son estrictamente necesarios para que su aplicación funcione sin crear vulnerabilidades de seguridad para los usuarios. Por ejemplo, es posible que una aplicación pida permisos para acceder al almacenamiento para almacenar archivos creados por esta misma aplicación cuando realmente no es necesario ese permiso para eso creando así un acceso desde esa aplicación a archivos que no necesita acceder. Algunos ejemplos de los últimos años son:

- Caso CamScanner [8]: En este caso tenemos una aplicación con más de 100 millones de descargas que durante varios años fue segura pero que en 2019 recibió una actualización que incluía una biblioteca para mostrar publicidad que en realidad era un malware que, haciendo uso de los permisos de la aplicación, instalaba otros malware en el dispositivo que en general creaban anuncios emergentes pero que podían llegar a suscribir a los usuarios en servicios de pago.
- Caso Facebook [9]: Este es el caso de una aplicación de una empresa que en 2018 usó su aplicación y los permisos de esta para acceder a datos confidenciales de los usuarios, mediante el permiso de acceso multimedia podía acceder a imágenes y documentos sin que el usuario lo supiera y también podía acceder a ubicación, registros de llamadas, contactos o incluso métodos de pago.

Dada la importancia de la privacidad hoy, especialmente desde la salida de la ley de protección de datos [26], esta es una buena oportunidad de poner en práctica los conocimientos adquiridos a lo largo de la carrera al tiempo que investigo para mí mismo y para otros como funcionan los permisos en Android.

1.3. Objetivos

El objetivo del trabajo será crear una o varias aplicaciones que permitan comprender la relación entre permisos y grupos de permisos en Android, así como comprobar las posibilidades que ofrece Google a un usuario para modificar y alterar dichos permisos y grupos de permisos [12].

Otro objetivo será tratar de averiguar cuáles son los archivos de Android que almacenan toda la información de los permisos y ver si estos son accesibles por cualquier usuario y si pueden modificarse para obtener nuevos permisos o permisos restringidos, así como crear nuevos permisos.

1.4. Organización de la memoria

El presente documento se organiza en 8 capítulos de la siguiente manera:

- **Capitulo 1. Introducción:** Breve presentación de este trabajo.
- **Capítulo 2. Planificación:** Se describe la planificación inicial del proyecto y sus tareas, un análisis de riesgos y una estimación del presupuesto.
- **Capítulo 3. Fundamentos teóricos:** Aquí se explicarán los conocimientos teóricos necesarios para la comprensión del trabajo.
- **Capítulo 4. Análisis:** Se explicarán proceso, requisitos y casos de uso que se llevarán a cabo.
- **Capítulo 5. Diseño, implementación y pruebas:** Se detalla el diseño de la aplicación, los distintos scripts que se usarán y los resultados de las pruebas realizadas para comprobar su correcto funcionamiento.
- **Capítulo 6. Obtención de resultados:** En este apartado veremos cual a sido el proceso de obtener unos resultados usando las aplicaciones desarrolladas
- **Capítulo 7. Conclusiones y trabajo futuro:** Reflexión sobre el trabajo realizado y posibles vías de trabajo para continuar en un futuro.

Capítulo 2

Planificación del proyecto

En este capítulo se detallará la planificación del proyecto, así como la metodología que se seguirá. Primero se detallará el plan inicial, el cual contiene el tiempo dedicado, una lista de tareas con su duración y la descomposición de estas. Después, se detallará la metodología a usar y, finalmente, se hará un análisis de los posibles riesgos identificados.

2.1. Metodología

La metodología que se seguirá para este proyecto será un modelo en espiral. En este modelo, tenemos una serie de hitos u objetivos que hay que cumplir y tras cumplir cada uno de ellos o cada dos de ellos, se revisará el plan y se tomarán decisiones en función de los resultados obtenidos.

La ventaja que tenemos con estos modelos respecto a otros es que la planificación queda bastante abierta, es decir, que, aunque existe una planificación, en la fase de desarrollo, tenemos en cuenta que el plan se revisará y modificará varias veces. Esto nos beneficia en un proyecto como este en el que tenemos un campo de investigación bastante amplio y con bastantes dudas respecto a que se puede o no se puede hacer, teniendo una planificación más abierta las modificaciones que puedan ir surgiendo no tendrán un impacto tan grande como en otros planes más cerrados.

En cuanto a los hitos que habrán de alcanzarse, tenemos 5 hitos principales que serían, la planificación, la búsqueda de información, el desarrollo de la aplicación Android, el desarrollo de la aplicación Python y el desarrollo y redacción de la memoria

2.2. Plan inicial

Aunque no estamos realizando un proyecto de desarrollo de software convencional, seguimos teniendo unas fases de desarrollo claras que pueden resumirse en la siguiente figura:

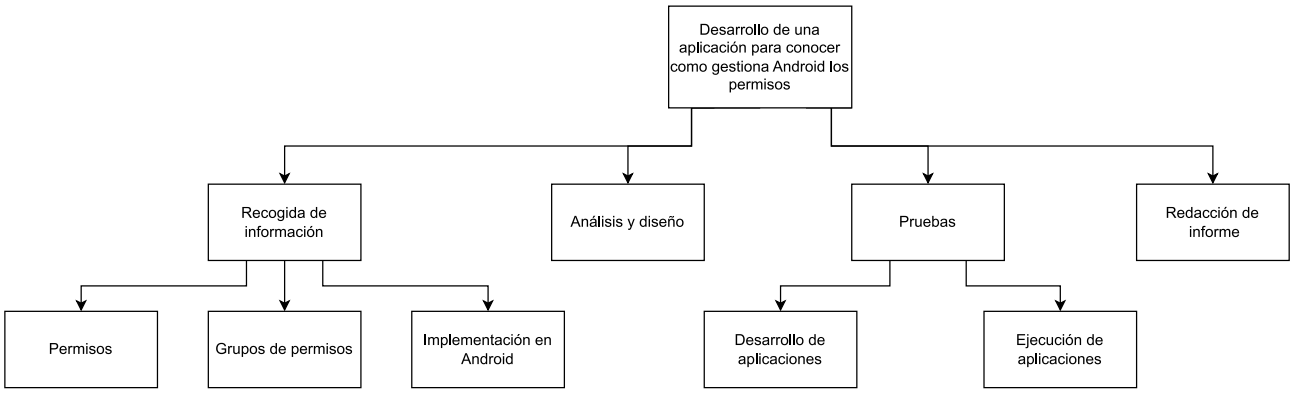

Figura 2.1: Estructura de descomposición de trabajo

En la figura podemos ver las siguientes fases:

- Recogida de información: La gestión de los permisos en Android es un campo muy extenso por lo que para poder ver todas las posibilidades que tenemos será necesario hacer un estudio de la información proporcionada por Google respecto a esta.
- Análisis y diseño: Una vez tenemos toda la información se hará un análisis de esta información para ver las opciones que tenemos un diseñar un plan de pruebas con el que cumplir con todos los objetivos del proyecto.
- Pruebas: En esta fase, se desarrollarán las aplicaciones planteadas en la fase anterior, se ejecutarán y se volverá a realizar un análisis con el que determinaremos si es necesario hacer más pruebas.
- Redacción del informe: Con toda la información recogida hasta esta fase, la última parte del proyecto será la de redactar el resto del informe en el que se expondrá toda esta información, el resultado de las pruebas y las conclusiones obtenidas.

A continuación, se detallarán las tareas identificadas, su duración y las fechas estimadas de inicio y final del proyecto.

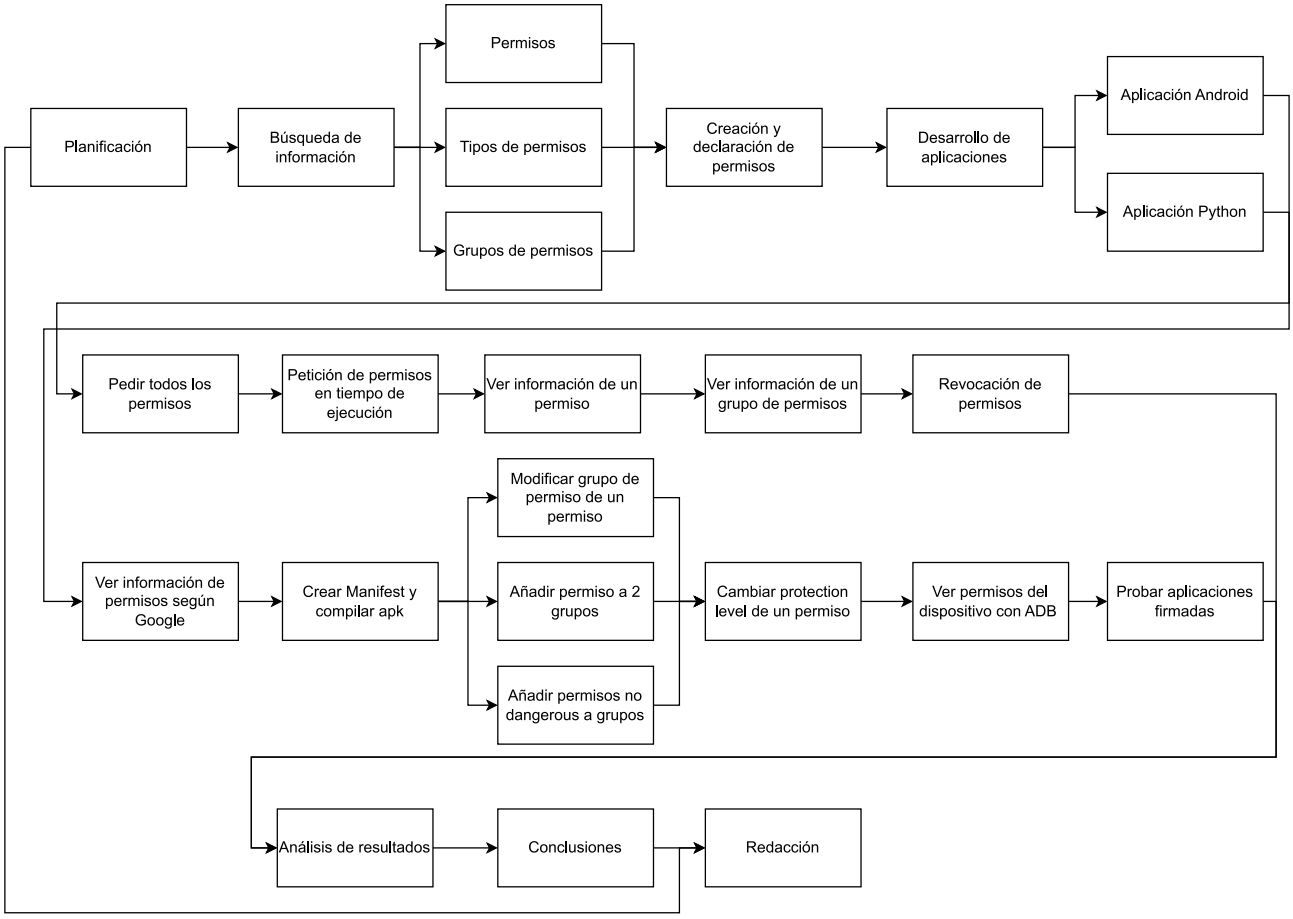

Figura 2.2: Diagrama de tareas

En el diagrama anterior podemos ver las tareas identificadas y que serán necesarias para completar el proyecto y el orden en el que han de ser completadas. Como vemos algunas pueden realizarse de forma simultánea o paralela si estas son similares. Por otro lado, tenemos la siguiente tabla donde vemos la duración estimada de cada una de estas tareas.

Id.	Actividad	Earliest Start	Duración	Earliest Finish
A	Planificación	0	3	3
B-E	**Búsqueda de información**	**3**	**4**	**7**
B	Qué permisos existen	3	1	4
C	Grupos de permisos	3	1	4
D	Tipos de permisos	3	1	4
E	Como crear y solicitar permisos	4	3	7
F-R	**Desarrollo de apps**	**7**	**30**	**37**
F-J	**Desarrollo de app Android**	**7**	**15**	**22**
F	Pedir todos los permisos	7	4	11
G	Petición de permisos en tiempo de ejecución	11	2	13
H	Ver información de un permiso	13	4	17
I	Ver información de un grupo de permisos	17	3	20
J	Revocación de permisos	20	2	22
K-R	**Desarrollo de app Python**	**7**	**30**	**37**
K	Ver información de permisos según Google	7	7	14
L	Crear Manifest y compilar apk	14	6	20
M	Modificar grupo de permisos de un permiso	20	3	20
N	Añadir permiso a 2 grupos	20	6	26
O	Añadir permisos no dangerous a grupos	20	3	23
P	Cambiar protection level de un permiso	26	3	29
Q	Ver permisos del dispositivo con ADB (Android Debug Bridge)	29	4	33
R	Probar aplicaciones firmadas	33	4	37
S-V	**Memoria del proyecto**	**3**	**44**	47
S	Análisis de resultados	37	1	38
T	Conclusiones	38	3	41
U	Redacción	3	42	45
V	Revisión y otros	45	2	47

Cuadro 2.1: Descomposición de tareas

En la tabla anterior, podemos ver que el proyecto tendrá una duración de 47 días de trabajo. También podemos ver que el camino crítico, es decir, las tareas que, de sufrir algún retraso, retrasarían todo el proyecto, es el siguiente:

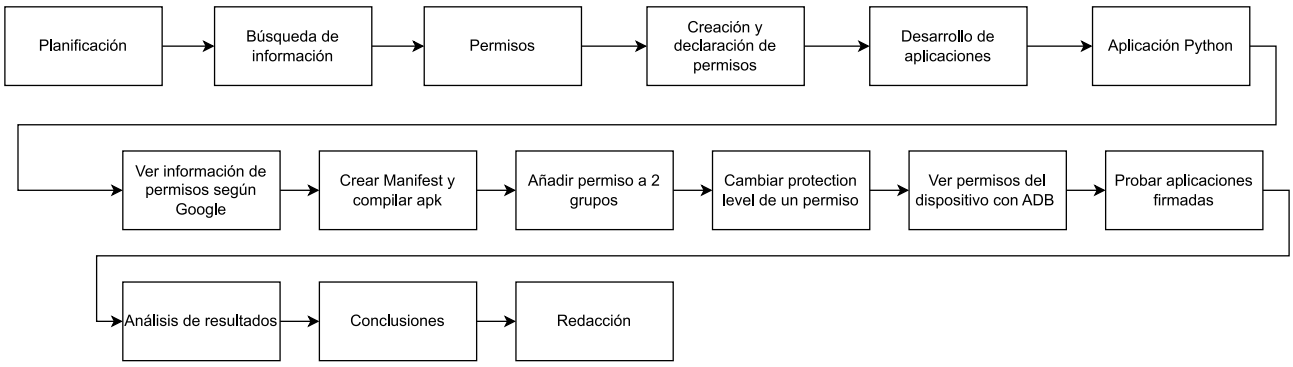

Figura 2.3: Diagrama de tareas

Finalmente, tenemos un diagrama de Gantt donde podemos ver las tareas y subtareas, la duración de cada una de ellas y el orden en el que deberán completarse:

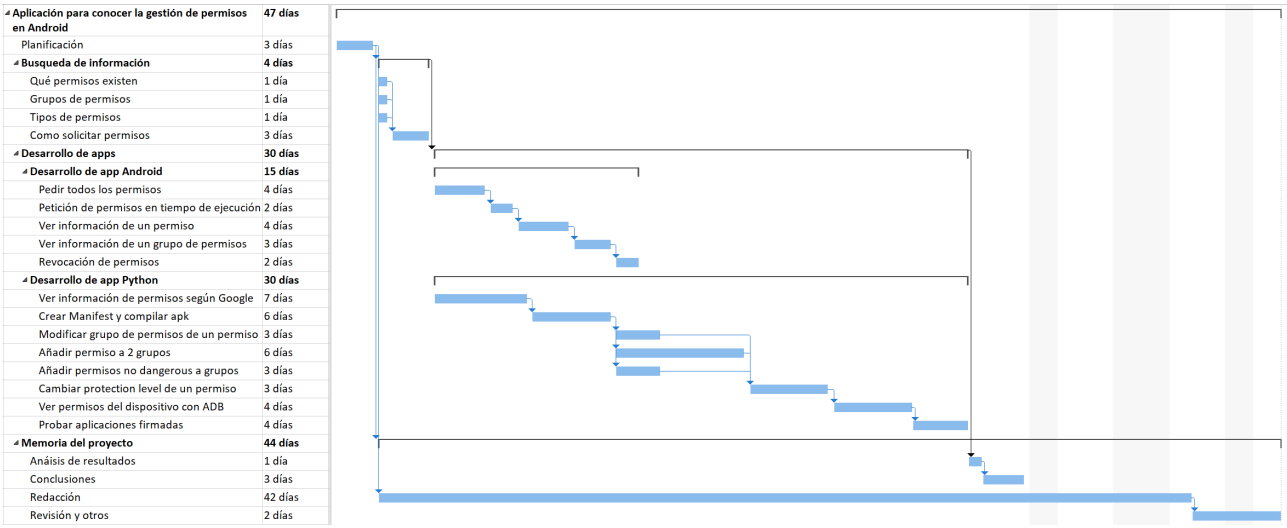

Figura 2.4: Diagrama de Gantt

2.3. Análisis de riesgos

En esta sección realizaremos un análisis de riesgos en donde veremos los riesgos que podrían surgir, con qué frecuencia, cuál sería su impacto y alguna forma de mitigarlos. Mitigar todos los riesgos consumiría demasiado tiempo, por lo que usaremos una matriz de riesgos como la de la Figura 2.3 para determinar el peligro que supone un riesgo en función de su probabilidad e impacto valorados en una escala del 1 al 5, siendo el 1 lo más bajo y 5 lo más alto, cuanto más se acerquen a la zona inferior derecha más peligrosos serán los riesgos.

		Impacto				
		1	2	3	4	5
Probabilidad	1	Baja	Baja	Baja	Media	Alta
	2	Baja	Baja	Media	Media	Alta
	3	Baja	Media	Media	Alta	Muy Alta
	4	Media	Media	Alta	Muy Alta	Muy Alta
	5	Alta	Alta	Muy Alta	Muy Alta	Muy Alta

Figura 2.5: Matriz de riesgos

A continuación, tenemos una lista de los riesgos identificados con su impacto y probabilidad, una descripción del riesgo y del impacto que pueden tener, así como una forma de mitigar cada uno de ellos y la probabilidad e impacto tras aplicar las mitigaciones si es que se aplican.

REGISTRO DE RIESGO					
ID Riesgo	1	**Título**	Estimación de trabajo incorrecta		
Descripción del riesgo Ya sea porque la tarea es más larga o complicada de lo previsto o por falta de tiempo es posible que alguna tarea se retrase.					
Descripción del impacto Dependiendo del retraso el proyecto entero podría llegar a retrasarse bastante					
Mitigación recomendada para el riesgo Evaluar durante la planificación la viabilidad y la simplicidad o complejidad de cada tarea.					
Valores de probabilidad e impacto					
		Probabilidad	**Impacto**	**Duración**	**Exposición**
Premitigación		4	5	-1 a -5 días	Muy Alta
Postmitigación		1	2	0 a -4 días	Baja

REGISTRO DE RIESGO					
ID Riesgo	2	**Título**	Equipamiento estropeado		
Descripción del riesgo Alguna pieza del equipamiento necesario se estropea, puede ser el ordenador o el dispositivo móvil que se usará para ejecutar las aplicaciones.					
Descripción del impacto El impacto sería bastante grande pues es difícil calcular de antemano el tipo de avería que pueden sufrir los dispositivos.					
Mitigación recomendada para el riesgo Tener dispositivos de repuesto en caso de que se estropee alguno.					
Valores de probabilidad e impacto					
		Probabilidad	**Impacto**	**Duración**	**Exposición**
Premitigación		1	5	1 a 10 días	Alta
Postmitigación		1	2	0 a 1 días	Baja

REGISTRO DE RIESGO					
ID Riesgo	3	**Título**	Enfermedad del alumno		
Descripción del riesgo Es posible que el alumno que se ponga enfermo.					
Descripción del impacto El impacto puede ser variado dependiendo de la gravedad de la enfermedad.					
Mitigación recomendada para el riesgo Realmente no hay muchas mitigaciones posibles en este caso, seguir las normas sanitarias comunes.					
Valores de probabilidad e impacto					
		Probabilidad	**Impacto**	**Duración**	**Exposición**
Premitigación		1	3	1 a 5 días	Baja
Postmitigación		1	3	1 a 5 días	Baja

REGISTRO DE RIESGO			
ID Riesgo	4	**Título**	Modificación de los objetivos
Descripción del riesgo Puede darse el caso de que según avanza el proyecto encontremos algún otro caso de estudio de interés dentro del tema por lo que se pueden añadir más objetivos.			
Descripción del impacto Añadir objetivos incrementaría la duración del proyecto, ya que se deben incluir en el plan de pruebas y desarrollar y ejecutar las aplicaciones necesarias.			
Mitigación recomendada para el riesgo Realizar un buen estudio y análisis previo al desarrollo del plan de pruebas y el establecimiento de los objetivos reduciría la probabilidad de que se dé el caso de tener que incluir nuevos objetivos.			

Valores de probabilidad e impacto				
	Probabilidad	**Impacto**	**Duración**	**Exposición**
Premitigación	4	3	1 a 3 días	Alta
Postmitigación	2	3	1 a 3 días	Media

REGISTRO DE RIESGO			
ID Riesgo	5	**Título**	Diseño del plan de pruebas inadecuado
Descripción del riesgo Dado que el campo de estudio es muy amplio y desconocido al inicio del proyecto, es posible que al diseñar el plan de pruebas no se planifiquen correctamente todos los casos.			
Descripción del impacto Si el plan de pruebas deja fuera algún caso tendríamos que añadirlo a mitad de proyecto lo que incrementaría la duración de este.			
Mitigación recomendada para el riesgo Realizar un buen estudio y análisis previo al desarrollo del plan de pruebas para no dejar casos sin probar.			

Valores de probabilidad e impacto				
	Probabilidad	**Impacto**	**Duración**	**Exposición**
Premitigación	3	4	1 a 3 días	Alta
Postmitigación	2	3	1 a 3 días	Media

REGISTRO DE RIESGO					
ID Riesgo	6	**Título**	Algún prototipo no puede realizarse		
Descripción del riesgo Los prototipos para desarrollar se han planteado antes de la recopilación de información por lo que es posible que alguno de ellos no sea posible realizarlo.					
Descripción del impacto En este caso el impacto sería menor pues el proyecto no se retrasaría, se adelantaría, en caso de que muchos de ellos no puedan realizarse podríamos no llegar al mínimo de 300h requeridas para el Trabajo de Fin de Grado.					
Mitigación recomendada para el riesgo Comprobar previamente que estas ideas pueden implementarse o buscar más ideas por si nos encontramos este imprevisto					
Valores de probabilidad e impacto					
	Probabilidad	**Impacto**	**Duración**	**Exposición**	
Premitigación	3	2	-2 a -4 días	Media	
Postmitigación	2	0	0 a -4 días	Baja	

Con todos los riesgos identificados y clasificados el siguiente paso será decidir qué hacer con ellos. Normalmente se pueden aplicar 3 opciones, transferir el riesgo, evitarlo o mitigarlo. En nuestro caso transferirlo es complicado por lo que optaremos por intentar mitigar aquellos riesgos una exposición alta o muy alta y aceptar el resto.

Capítulo 3

Fundamentos teóricos

El objetivo de esta sección será detallar los principales conceptos que hay que conocer para entender que son y cómo funcionan los permisos de Android, empezando por qué es Android [1] y para qué son los permisos y por qué son necesarios.

3.1. El sistema operativo Android

3.1.1. Visión general

Android es un sistema operativo creado en 2003 por Rich Miner, Nick Sears, Chris White y Andy Rubin con la intención de usarlo en dispositivos móviles. En el año 2005, Android fue comprado por Google y en 2008 se lanzó el primer teléfono móvil con Android, el HTC Dream.

Este sistema operativo está basado en el kernel de Linux con un enfoque en los dispositivos móviles, aunque en los últimos años han empezado a aparecer versiones adaptadas a ordenadores y televisores, así como a dispositivos más pequeños y sencillos como relojes o sistemas empotrados como los de los coches. Otra de las características principales de Android es que es de código abierto, es decir, cualquiera puede descargarlo, modificarlo y hasta venderlo.

En la actualidad, es el sistema operativo más usado [27] en dispositivos móviles y se encuentra en la versión 14. Una de sus mayores ventajas, aunque también ha generado algunos problemas es que, al ser de código abierto [28], cada empresa lanza dispositivos con su propia versión de Android, de forma que los componentes básicos son comunes pero algunos elementos de la interfaz, servicios y funcionalidades son distintos entre ellos por lo que cada fabricante debe asegurarse que su versión es compatible con el hardware de cada uno de sus dispositivos y con todas las aplicaciones desarrolladas para Android.
Debido a esto último, nos encontramos con que no todos los dispositivos cuentan con la última versión, a continuación, vemos el porcentaje de dispositivos que usaba cada versión de Android en enero de 2023 [5]:

Versión de Android	Distribución enero 2023
MARSHMALLOW (6.0)	2,8 %
NOUGAT (7.0 - 7.1)	3,7 %
OREO (8.0 - 8.1)	9,5 %
PIE (9.0)	13,2 %
Android 10 (10.0)	19,5 %
Android 11 (11.0)	24,4 %
Android 12 (12.0)	18,9 %
Android 13 (13.0)	5 %

Cuadro 3.1: Porcentaje de uso de Android

Cada una de estas versiones introduce cambios, algunos son solo mejoras de rendimiento u optimización de batería, sin embargo, desde Android 10, Google ha puesto especial atención al sistema de permisos de Android para garantizar la privacidad y seguridad de sus usuarios, generando una situación en la que los desarrolladores deben crear aplicaciones teniendo en cuenta que para versión de Android tienen que solicitar ciertos permisos y funcionar de forma distinta si quieren llegar a la mayoría de usuarios.

A todo el problema de las versiones, hay que añadir el que además de las versiones de Android que crean los distintos fabricantes, podemos encontrar cientos de ROMs personalizadas por los usuarios, algunas de ellas están hechas con el objetivo de dar soporte a dispositivos que los fabricantes han dejado de actualizar, pero otras son desarrolladas con intenciones maliciosas. Un ejemplo reciente es el de muchas TV boxes que pueden comprarse por internet y que en los últimos años se han convertido en una forma de distribuir sistemas con un malware similar a CopyCat, el cual infectó unos 14 millones de dispositivos en 2017 y que permite un acceso total al dispositivo de forma remota.

3.1.2. Estructura básica de Android

Como se ha dicho antes, Android debe adaptarse a una gran variedad de dispositivos con distintos componentes de hardware. Esto lo consigue gracias a los siguientes componentes que controlan el funcionamiento del sistema a distintos niveles [18]:

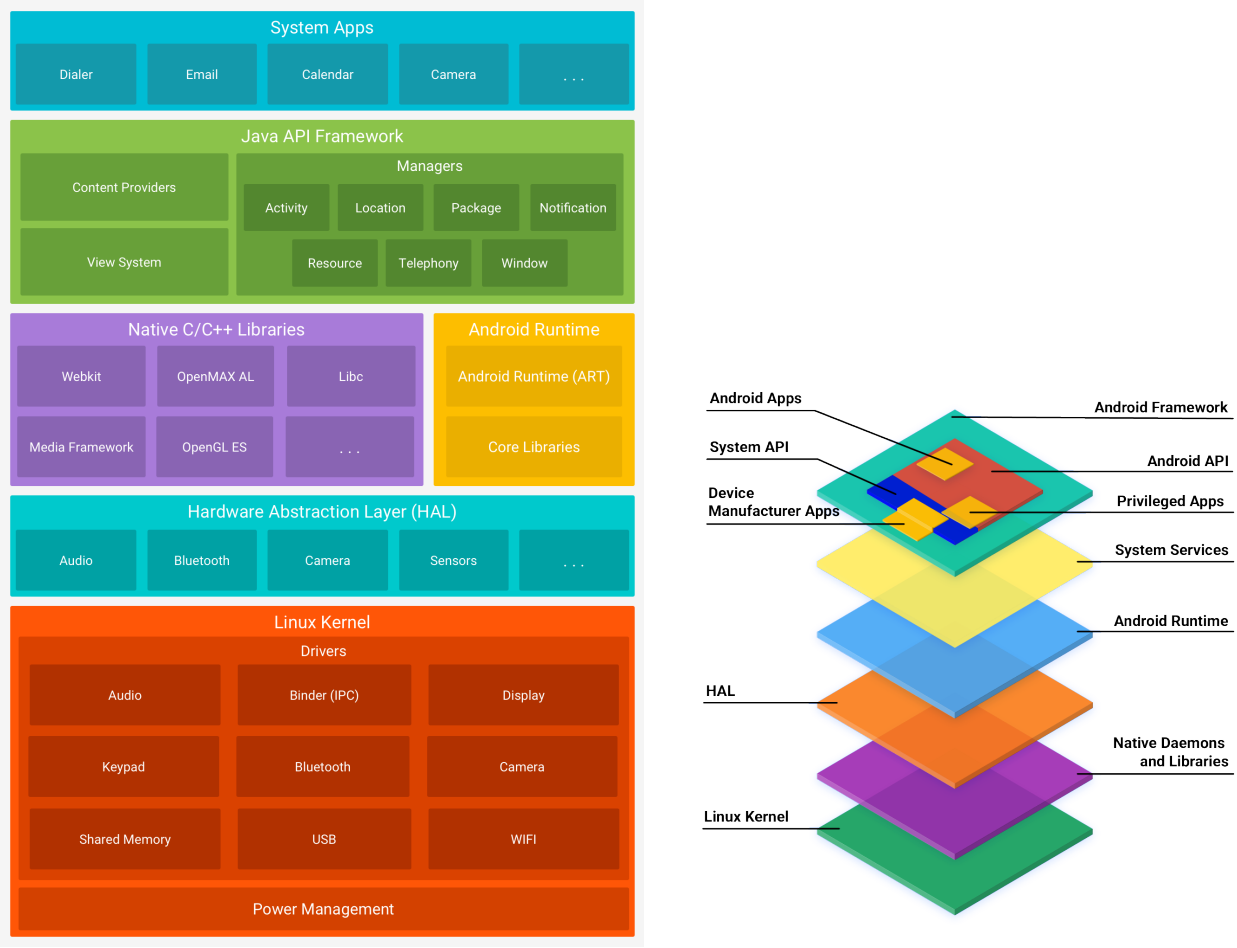

Figura 3.1: Pila de software de Android Figuras recogidas de "Descripción general de la arquitectura" [18] y "Introducción al desarrollo con Android" [30]

- **Kernel de Linux:** Android se basa en el núcleo de Linux, que gestiona los recursos del sistema como CPU, RAM y el resto de los controladores del sistema.
- **Hardware Abstraction Layer (HAL) [4]:** Esta es la capa de software que consigue que todas las aplicaciones y el propio Android puedan funcionar en cualquier dispositivo independientemente de su hardware. Cuando una aplicación requiere acceso a una función hardware, esta no accede directamente al componente físico si no que accede a esta capa HAL que gestiona el acceso al hardware haciendo que para la aplicación el acceso sea siempre igual.
- **Librerías nativas:** Librerías con funciones esenciales de Android que pueden usar tanto el SO en si como las aplicaciones, algunas de sus funciones son gráficos, seguridad o comunicación.
- **Android Runtime (ART) [3]:** Es el entorno de ejecución de Android, es decir, es la parte del sistema operativo que se encarga de ejecutar las aplicaciones de forma que el código escrito por el desarrollador sea entendido por el sistema transformándolo en instrucciones específicas del procesador. En versiones anteriores a Android 5.0 se usaba Dalvik como máquina virtual para ejecutar el bytecode dex, pero en la actualidad se usa ART que es compatible con Dalvik.

- **Marco de Aplicaciones:** Este componente proporciona a los desarrolladores herramientas y componentes para crear aplicaciones como actividades, servicios o receptores de difusión a los que se accede a través de la API de Android. Se encuentra en todos los dispositivos Android ya precompilado y se divide en dos partes, una disponible para todos los desarrolladores y otra solo accesible por los OEM a través de API del sistema.

- **Aplicación del fabricante del dispositivo:** Estas aplicaciones son creadas por el fabricante del dispositivo usando la API de Android y la API del sistema por lo que vienen preinstaladas y pueden tener acceso a funciones que las aplicaciones normales no tienen.

- **Aplicación privilegiada:** Al igual que las anteriores, usan ambas API y vienen preinstaladas en el dispositivo, la diferencia es que estas suelen ser de Google o de los partner del OEM y no suele ser posible desinstalarlas.

- **Aplicación Android:** Son las aplicaciones creadas usando únicamente el API de Android y son aquellas que podemos encontrar el Google Play Store y tiendas similares o instalar a partir de paquetes APK, la mayoría de las aplicaciones creadas para Android son de este tipo y son las aplicaciones con las que trabajaremos en este caso.

Otra de las características de Android es que ejecuta las aplicaciones en procesos separados siguiendo el método de "sandboxing", es decir, cada aplicación se ejecuta de forma completamente aislada gestionando la memoria de todas ellas de forma que se optimiza el rendimiento al tiempo que se optimiza la duración de la batería pues no podemos olvidar que su enfoque principal son dispositivos móviles que suelen tener poca potencia y dependen de baterías para poder funcionar.

A pesar de usar en método de "sandboxing", las aplicaciones Android son capaces de compartir datos, mensajes, periféricos, etc, esto es gracias a que el SO tiene un sistema de permisos que permite a las aplicaciones acceder a funciones concretas como el almacenamiento, el acceso a la cámara o suscribirse a los canales de difusión de otras aplicaciones para recibir o enviar mensajes.

3.1.3. Android Open Source Project (AOSP)

Una de las grandes ventajas de Android es que es de código abierto, es decir, el código fuente del sistema está disponible para ser descargado y modificado por cualquiera. El objetivo del proyecto es dar la oportunidad a todos los fabricantes y desarrolladores de competir en el campo de los sistemas móviles en igualdad de condiciones y, al mismo tiempo, crear entre todos los desarrolladores un sistema cada vez más potente y eficiente, capaz de funcionar en cualquier dispositivo de la forma más transparente posible.

Gracias a este proyecto tenemos gran cantidad de dispositivos que usan Android siendo cada uno muy distinto del resto, por ejemplo, tenemos los smartphones, aunque la gran mayoría usan Android, si usamos un dispositivo Xiaomi veremos una gran cantidad de diferencias respecto a un Samsung o un Google, esto es debido a que cada fabricante ha creado su propia versión a partir de AOSP.

También es gracias a este proyecto que tenemos Android en dispositivos tan diversos, desde smartphones, televisiones o relojes hasta proyectores o incluso la llave de un coche como en el caso del BMW Key Phone.

3.2. Seguridad en Android

Desde hace unos años la ciberseguridad ha ganado mucha importancia y Google también se propuso hacer de ella uno de los principales puntos de desarrollo para Android. Siempre ha habido un desarrollo continuo de seguridad en Android, pero desde la versión 10 del sistema operativo, esta se ha desarrollado con rapidez, especialmente todo aquello que puede suponer un riesgo para la privacidad de los usuarios. Algunas de las medidas que incorpora Android para garantizar la seguridad son [22]:

- **Sandboxing:** Desde el inicio Android a utilizado esta técnica como base de su seguridad. El sandboxing consiste en ejecutar cada aplicación como un proceso aislado, con sus propios recursos, de forma que una aplicación maliciosa no es capaz de acceder al sistema u otras aplicaciones. En este entorno se ejecuta todo lo necesario para que funcione la aplicación, desde la propia aplicación a las librerías y componentes de SDK de terceros.
- **Play Protect:** Play Protect es un componente integrado en la Play Store que cuenta con tres funciones principales:
  - Escaneo de aplicaciones: Play protect hace uso de técnicas de aprendizaje automático para escanear aquellas aplicaciones se instalan en el dispositivo, tanto desde Play Store como de otras fuentes, en busca de comportamientos anómalos. Cuando se detecta una aplicación con un comportamiento extraño, Play protect puede avisar al usuario o bloquear la aplicación completamente. Al inicio fue una aplicación muy criticada pues a pesar de su implementación gran cantidad de malware llega a los usuarios, sin embargo, gracias al aprendizaje automático, cada vez detecta más vulnerabilidades.
  - Medidas antirrobo: Play protect incorpora la capacidad de localizar, bloquear o incluso borrar el dispositivo a distancia desde nuestra cuenta de Google, dando al usuario opciones para evitar que sus datos se filtren en caso de robo.
  - Navegación segura: Su funcionamiento es similar al escaneo de aplicaciones, pero en el navegador Chrome, cuando detecta que el usuario accede a una web con comportamiento sospechoso, avisa al usuario.
- **Permisos:** Los permisos de aplicaciones en Android son otra medida de proteger la privacidad del usuario bloqueando el acceso a acciones y datos restringidos. Algunas de las acciones que requieren permisos son el acceso a internet, emitir notificaciones, acceso a la cámara, acceso a archivos multimedia, entre otros.
  Estos permisos deben ser solicitados por el desarrollador, y el usuario puede ver antes y después de instalar la aplicación qué permisos tendrá la aplicación, e incluso puede concederlos o bloquearlos por sí mismo. También, es posible ver que aplicaciones tienen un permiso y cuál es la función de dicho permiso si el desarrollador lo desea. El siguiente apartado se hará una exposición más profunda de cómo funcionan los permisos.
- **Almacenamiento seguro:** En Android contamos con varias medidas que garantizan la seguridad de los datos tanto de los usuarios como de las aplicaciones.
  En cuanto a las aplicaciones, estas cuentan con una función que hace que el almacenamiento de una aplicación este oculto para el resto, incluso los datos que se crean tras la instalación pueden quedar dentro de ese espacio reservado para la aplicación.
  Para los usuarios, Android cuenta con varias funciones, una de ellas son los permisos que permiten ver y decidir que aplicaciones pueden acceder al almacenamiento y ver la sección de almacenamiento que es compartido por las aplicaciones con este permiso.
  Otra de las medidas de seguridad es el cifrado del teléfono, desde Android 6.0 es posible cifrar la tarjeta SD del dispositivo, de forma que no sea posible acceder a sus datos desde otro dispositivo, además, contamos con la opción del encendido seguro que mantiene los datos del dispositivo desde que este se apaga hasta que se vuelve a encender y se introduce la contraseña.

- **Biometría:** Desde hace ya varios años, Android incluye distintas opciones para utilizar biometría para garantizar la seguridad, contamos con lectores de huellas, detección de rostro y detección de iris siempre que contemos con el hardware necesario. Todas las aplicaciones pueden hacer uso de este sistema para garantizar la seguridad de los datos del usuario.
- **Rooting:** Otra medida que usa Android es el sistema de rooting, es decir, los usuarios por defecto utilizan usuarios no root, de forma que no tienen acceso a la mayoría de los archivos de Android, en concreto solo tienen acceso a la carpeta emulated que contiene algunos archivos de aplicaciones, multimedia y poco más, esto garantiza que una aplicación no puede acceder a datos que estén fuera de esta carpeta.
- **Actualizaciones de seguridad:** Además de las actualizaciones de Android, contamos con este tipo de actualizaciones mensuales cuya función es introducir mejoras y parches de seguridad, desde Android 10 estas actualizaciones pueden distribuirse desde la Play Store haciendo más fácil su distribución pues, al igual que las actualizaciones de Android, estas han de ser distribuidas por el fabricante del dispositivo y no por Google por lo que, de nuevo, podemos ver que hay dispositivos que tienen distintos parches de seguridad, incluso puede darse el caso de contar con la última versión de Android pero estar varias por detrás en seguridad o al revés, tener una versión antigua de Android pero los últimos parches de seguridad.

3.3. Permisos y privacidad

Los permisos de Android es una las principales formas que tenemos para mantener la privacidad y seguridad al desarrollar una aplicación, su función es controlar el acceso a datos y acciones restringidos. Dado que este trabajo trata sobre estos permisos, a continuación, veremos más en detalle como son los permisos de Android.

3.3.1. Historia de los permisos

Antes de nada, debemos ver cómo ha evolucionado el sistema de permisos de Android y por qué es tan importante el tema mencionado anteriormente de que tengamos dispositivos con versiones antiguas de Android o sus parches de seguridad:

Android 1.0 - 5.1 (2008-2015)	La mayoría de las acciones no requerían permiso del usuario, se concedían al momento de la instalación .
Android 6.0 Marshmallow (2015)	Se introdujo el modelo de "Solicitud de permisos en tiempo de ejecución" por el cual algunos permisos deben solicitarse al usuario tras instalarse y pueden permitirse o bloquearse permisos individuales.
Android 7.0 Nougat (2016)	Se introdujeron los grupos de permisos.
Android 8.0 Oreo (2017)	Desde esta versión, podían revocarse permisos en tiempo real desde la configuración de la aplicación.
Android 9 Pie (2018)	Se restringió el acceso a la cámara y el micrófono en segundo plano y se unifico el uso de la seguridad biométrica.
Android 10 (2019)	El permiso de ubicación se modificó para que las aplicaciones solo pudiesen usarlo mientras estas se usaban y el servicio de actualizaciones de seguridad mediante Google Play
Android 11 (2020)	Esta versión incluyo permisos de un solo uso de forma que la próxima vez que sea necesario se vuelve a solicitar. También se incluyó el sistema de auto borrado de permisos después de un tiempo sin usar la app.
Android 12 (2021)	En este caso se introdujeron unas notificaciones que avisasen al usuario cuando se estuviese usando la cámara o el micrófono, se dividió el permiso de ubicación en aproximada o precisa y se añadió un apartado para ver cuánto tiempo cada app había usado la cámara, la ubicación, el micrófono, etc.
Android 13 (2022)	Incluye el permiso para mostrar notificaciones y restringe el acceso multimedia únicamente a los archivos que el usuario dese.

Cuadro 3.2: Evolución de los permisos en Android

3.3.2. Tipos de permisos

Android hace varias divisiones de permisos según el riesgo que suponen y el momento en el que se conceden o deniegan asignándolos después a un "protection level":

- Permisos en el momento de la instalación: Estos permisos se conceden directamente en el momento de instalar la aplicación, no es necesario abrirla. Se conceden directamente pues se considera que no tienen un gran impacto en otras aplicaciones o en el propio sistema. Dentro de estos tenemos dos tipos de permisos:

  1. Permisos normales: Estos son los que permiten acceso a datos y acceso fuera del sandbox de la aplicación pero que suponen un riesgo mínimo, por ejemplo, el acceso a la red o evitar el bloqueo del dispositivo. Se les asigna el nivel de protección normal.
  2. Permisos de firma: Son permisos que solo se conceden a aquellas aplicaciones que estén firmadas con el mismo certificado que la app o el SO que los declaró. Un ejemplo de permiso de firma es el acceso a estadísticas de la batería. Se les asigna el nivel de protección signature.

- Permisos de tiempo de ejecución: Son aquellos que proporcionan acceso a acciones o datos restringidos que si pueden suponer mayor riesgo para el usuario como conocer la ubicación o acceder al micrófono. Estos permisos deben solicitarse desde el código de la app y el usuario deberá conceder el permiso de forma explícita mediante una alerta que se muestra en pantalla. Un ejemplo de estos permisos es el acceso a la cámara. Se les asigna el nivel de protección dangerous.

- Permisos especiales: Estos permisos son aquellos que otorgan un acceso a funciones más avanzadas del sistema, el usuario debe acceder a la configuración de la app o del dispositivo y dar permiso a la aplicación de forma manual. Algunos ejemplos de permiso especial son INSTALL_PACKAGES, para instalar aplicaciones, o REBOOT, para reiniciar el dispositivo. Se les asigna un nivel de protección appop.

Dado que Android es un sistema de código abierto, es posible que el nivel de protección o incluso los permisos que encontramos en cada dispositivo varíe, normalmente los permisos de tipo dangerous

suelen ser comunes en todos los dispositivos y los más básicos de los normales como el acceso a internet también.
Donde suele haber muchos permisos que dependen del fabricante es en el grupo de signature, es conveniente tanto para el fabricante como para los usuarios pues estos permisos solo podrán ser usados por apps creadas por el fabricante y suelen permitir cosas como interacción entre esas apps, interacción con sistemas externos o acceso a servicios del fabricante.

3.3.3. Grupos de permisos

Otro elemento importante del sistema de permisos son los grupos de permisos [14]. Estos grupos se utilizan para minimizar la cantidad de solicitudes que se le hacen al usuario agrupando permisos similares en uno solo, por ejemplo, tenemos el grupo de SMS que contiene los permisos de enviar y recibir SMS junto a otros permisos relacionados con los SMS.
Estos grupos también pueden tener niveles de acceso, por ejemplo, en el caso del permiso de ubicación tenemos ubicación aproximada, ubicación precisa y ubicación en segundo plano (este último ha de darse manualmente desde los ajustes).

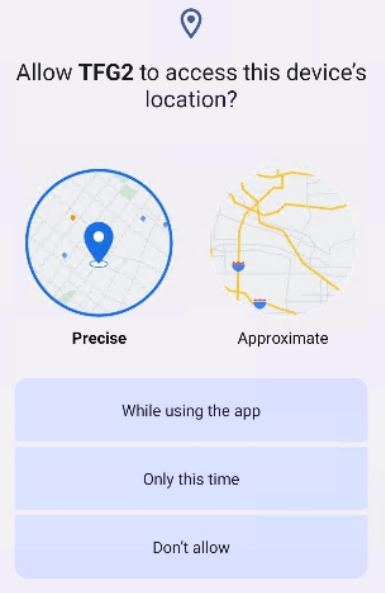

Figura 3.2: Solicitud de acceso a ubicación

Normalmente cuando un usuario decide revocar o conceder permisos lo que suele modificar es el grupo en sí y no el permiso individual, esto se hace para simplificar al usuario la gestión de los permisos.
Otra característica de estos grupos es que pueden cambiar, es decir, en una versión de Android un grupo puede tener 6 permisos y luego dividirse en 2 grupos con 3 permisos cada uno, esto supone que los desarrolladores debemos solicitar permisos y no grupos para mantener la mayor compatibilidad posible.

3.3.4. Solicitar permisos

Como se ha dicho en el apartado de tipos de permiso, tenemos varios tipos de permiso en función del momento en el que se conceden, sin embargo, una cosa común a todos ellos es que deben primero ser declarados en el archivo Manifest.xml de la aplicación. Esta obligación es una medida de transparencia que permite a los usuarios ver que permisos podría solicitar una aplicación y podemos verlo en el apartado de "Info de la app" en Google Play Store donde vemos los permisos que pueden solicitarse y para que puedan usarse.
En la página para desarrolladores de Android podemos encontrar bastantes guías sobre como deberían solicitarse los permisos y sobre todo se insiste en hacer un análisis de por qué se solicitan y hacer un resumen para que el usuario este informado en todo momento de que función cumple cada uno de los permisos, eso sí, es un paso opcional, no es necesario dar esta explicación:

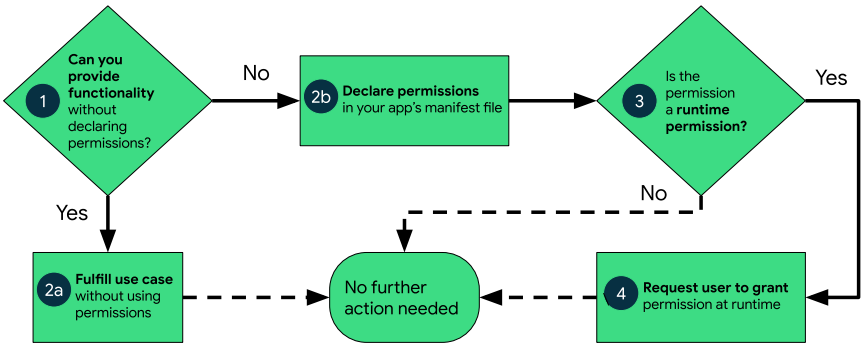

Figura 3.3: Flujo de trabajo general para usar permisos en Android
Figura obtenida de: "Permisos en Android" [12]

En cuanto al usuario, este puede decidir conceder o revocar acceso a permisos dangerous y especiales, la concesión se puede realizar desde la app cuando esta lo solicita en el caso de los dangerous o desde los ajustes del dispositivo en el caso de ambos tipos, la diferencia es que los de tipo dangerous se pueden ver en los ajustes de la app mientras que los especiales suelen estar en otros apartados, por ejemplo el acceso al almacenamiento está en el apartado "Permisos especiales" y el de mostrarse sobre otras apps estará en "Accesibilidad" o en "Seguridad" dependiendo del dispositivo.
La revocación siempre debe hacerse manualmente desde los ajustes, aunque la app puede pedir que se le revoquen los permisos desde Android 13. También existe, desde Android 11, un sistema de revocación automática de permisos para apps que no se hayan usado en unos meses.

3.4. Android Debug Bridge (ADB)

ADB (Android Debug Bridge) [11] es una herramienta de línea de comandos que se utiliza para interactuar con dispositivos Android y emuladores desde un ordenador. ADB es una parte esencial del kit de desarrollo de Android (SDK) y ofrece una amplia gama de capacidades para el desarrollo, depuración y administración de dispositivos Android. Algunas de las cosas que nos permite esta herramienta son:

1. Instalación de Aplicaciones: ADB permite instalar y desinstalar aplicaciones en dispositivos Android y emuladores.

2. Depuración: Facilita la depuración de aplicaciones Android al proporcionar acceso a registros, perfiles de rendimiento y más.
3. Transferencia de Archivos: Podemos copiar archivos entre nuestro ordenador y un dispositivo Android utilizando ADB.
4. Captura de Pantalla: ADB puede tomar capturas de pantalla de un dispositivo Android.
5. Grabación de Pantalla: En versiones más recientes, ADB permite grabar la pantalla de un dispositivo Android.
6. Shell Interactivo: Podemos abrir una shell interactiva en el dispositivo Android para ejecutar comandos en el entorno del sistema operativo Android directamente desde tu computadora, sería similar a usar una consola de Linux en el dispositivo.

Para poder usar adb, debemos activar en las opciones de desarrollador de nuestro dispositivo el modo "Depuración USB". Además, si queremos acceder a todas sus funciones, debemos tener un dispositivo con permisos root. Para activar la depuración USB debemos seguir los siguientes pasos:

1. Buscar en los ajustes del dispositivo la opción "Número de compilación", suele encontrarse en la sección de "Información de software" del dispositivo.
2. Pulsar repetidamente en esa opción hasta que nos salga un mensaje que diga que pulsemos X veces más para activar las opciones de desarrollador.
3. Seguir pulsando repetidamente hasta que nos diga que somos desarrolladores.
4. Buscar en ajustes la nueva sección "Opciones avanzadas", suele estar en el apartado "Sistema".
5. Dentro de "Opciones avanzadas" activar la depuración USB.

Capítulo 4

Análisis

En este capítulo veremos cuáles serán los objetivos del proyecto, cuál será el proceso que seguiremos para completarlos, cuáles serán los requisitos del sistema y los casos de uso con los que deben cumplir la aplicación Android y la aplicación de escritorio, al menos en un inicio, según avance el proyecto es posible que se encuentren nuevos casos de uso, en ese caso se mencionarán según vayan surgiendo.

4.1. Objetivos y proceso de investigación

4.1.1. Objetivos

En la sección de introducción ya se establecieron los objetivos del trabajo de forma general, en este apartado tenemos una serie de preguntas que se han identificado cuya respuesta podrá ayudarnos a cumplir con esos objetivos.

1. ¿A qué protection level asigna el sistema operativo los permisos, al declarado en el fichero Manifest.xml o al predefinido por Google?
2. ¿A qué grupo asigna el sistema operativo los permisos, al declarado en el fichero Manifest.xml o al predefinido por Google?
3. ¿Es posible asociar un permiso que no sea de tipo dangerous a un grupo de permisos?
4. ¿En qué fichero define Android los grupos de permisos soportados? ¿Puede conocerse vía adb?
5. ¿En qué fichero define Android los permisos soportados? ¿Puede conocerse vía adb?
6. ¿Puede un permiso estar en más de un grupo a la vez?
7. Dado un grupo de permisos, ¿es posible obtener qué permisos forman el grupo?
8. Si una app logra un permiso de tipo dangerous, ¿otra app firmada con el mismo certificado digital consigue automáticamente ese permiso (se lo concede el SO sin intervención del usuario)?
9. Si una app logra un permiso de tipo signature, ¿otra app firmada con el mismo certificado digital consigue automáticamente ese permiso?

4.1.2. Proceso de investigación

El proceso que seguiremos para dar respuesta a las preguntas expuestas anteriormente será el siguiente:

1. Obtener una lista de todos los permisos de Android, a que grupo pertenece cada uno y cuál es su protection level.
2. Crear una aplicación Android que solicite todos los permisos y algún permiso de usuario que se usará posteriormente.
3. Crear una aplicación que nos permita generar distintos apks para realizar pruebas.
4. Asociar un permiso dangerous a otro grupo.
5. Asociar un permiso normal a un grupo.
6. Asociar un permiso dangerous a varios grupos.
7. Crear aplicaciones Android firmadas con el mismo certificado y solicitar en ellas un permiso dangerous y un permiso signature.
8. Usar las aplicaciones para dar respuesta a las preguntas anteriores.
9. Investigar donde se encuentra el archivo con la declaración de permisos de Android.
10. Investigar el dispositivo de pruebas con adb (Android Debug Bridge).

4.2. Requisitos funcionales

Son aquellos requisitos que el sistema debe ser capaz de realizar, servicios que debe realizar y cómo reaccionará a distintas entradas o situaciones. Se dividirá en requisitos de la app Android y requisitos de la app de escritorio.

Para los requisitos de la app Android tenemos los siguientes:

- El sistema debe ser capaz de ejecutarse solicitando solo permisos normales.
- El sistema debe poder pedir todos los permisos dangerous.
- El sistema debe mostrar los permisos solicitados.
- El sistema debe mostrar que permisos se han concedido y cuáles no.
- El sistema debe mostrar el nivel de protección de cada permiso.
- El sistema debe mostrar el grupo al que pertenece cada permiso.
- El sistema debe mostrar todos los grupos.

Para los requisitos de la app de escritorio tendremos los siguientes requisitos funcionales.

- El sistema debe permitir ver los permisos por defecto de Android.

- El sistema debe permitir ver los grupos de permisos por defecto de Android.
- El sistema debe permitir ver los protection level por defecto de Android.
- El sistema debe permitir crear un AndroidManifest modificando el grupo de permisos de un permiso.
- El sistema debe permitir crear un AndroidManifest modificando el protection level de un permiso.
- El sistema debe permitir crear un AndroidManifest que asigne 2 grupos de permisos a un permiso.
- El sistema debe permitir ver información de permisos y grupos de permisos del dispositivo.
- El sistema debe permitir instalar apks.
- Es sistema debe permitir comparar dos apks para ver si están firmadas con el mismo certificado.
- El sistema debe poder comparar su lista de permisos con otra para ver si coinciden.

4.3. Requisitos no funcionales

Los requisitos no funcionales son aquellos que describen las restricciones que afectan a los servicios o funciones del sistema. Tendremos 3 grupos, requisitos comunes, requisitos para la app Android y requisitos para la app de escritorio.

Requisitos comunes:

- El sistema deberá estar disponible en español.
- El sistema deberá estar correctamente documentado.
- El sistema deberá incluir un manual de uso.
- La aplicación deberá responder de modo fiable y con un tiempo de respuesta tolerable por la media de usuarios.
- El sistema deberá permitir probar si dos apks firmadas con el mismo certificado comparten sus permisos, tanto los dangerous como los signature.

Requisitos de la app Android:

- El sistema debe usar la versión 13 de Android.

Requisitos de la app de escritorio:

- El sistema deberá funcionar sobre Windows 11.
- El sistema deberá funcionar sobre Linux.
- El sistema deberá incluir manual de instalación.

4.4. Requisitos de información

Estos requisitos describen como debe almacenarse la información.

- El sistema deberá obtener información de los permisos su grupo de permisos y su protection level del dispositivo Android.
- El sistema deberá obtener información de los permisos su grupo de permisos y su protection level del proyecto de AOSP.
- El sistema debe recoger todos los datos en un formato sencillo de ordenar según el permiso, grupo de permiso y protection level.
- Todos los datos recogidos por el sistema deben usar el mismo formato.

4.5. Casos de uso

En esta sección contaremos con un diagrama de casos de uso donde veremos las acciones que puede realizar un usuario y posteriormente se detallarán estos casos de uso describiendo la secuencia de actividades que el sistema realizará cuando intervenga el usuario. Tendremos 2 diagramas, con sus respectivos casos de uso, uno para la aplicación de escritorio y otro para la aplicación Android.

4.5.1. Casos de uso app Android

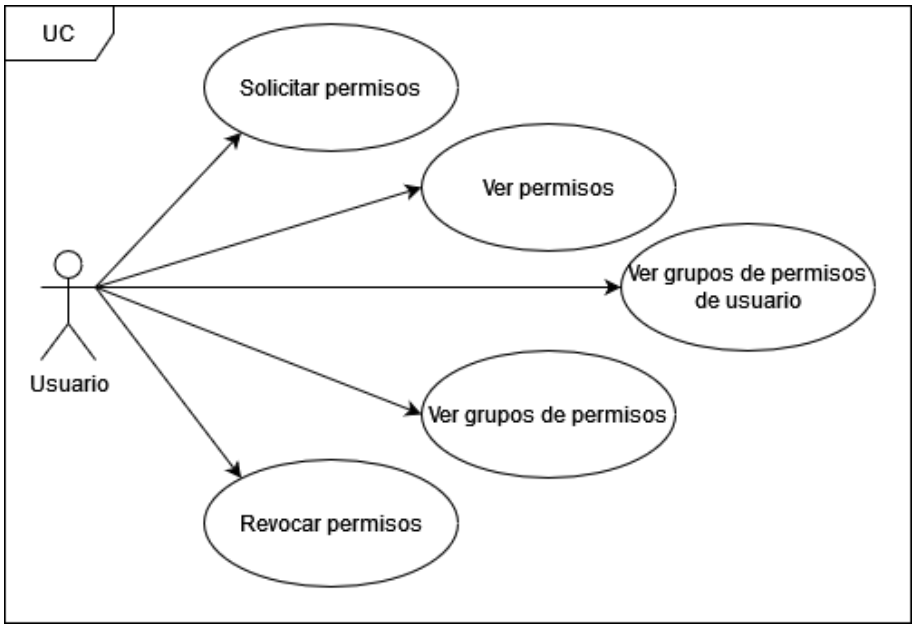

Figura 4.1: Flujo de trabajo general para usar permisos en Android.

UC-01	Solicitar permisos
Descripción	El usuario selecciona la opción de solicitar permisos
Secuencia normal	1. El usuario indica que quiere solicitar permisos. 2. El sistema solicita al usuario que permita o deniegue cada permiso. 3. El usuario decide si permite o deniega cada permiso.
Postcondición	Los permisos que el usuario haya permitido se han concedido

Cuadro 4.1: Caso de uso: Solicitar permisos

UC-02	Ver permisos
Descripción	El usuario selecciona la opción de ver los permisos
Secuencia normal	1. El usuario indica que quiere ver los permisos. 2. El sistema muestra una lista de permisos. 3. El sistema imprime una lista con toda la información de los permisos. 3. El usuario selecciona un permiso. 4. El sistema muestra la información del permiso.
Postcondición	Se ha creado una lista con la información de todos los permisos

Cuadro 4.2: Caso de uso: Ver permisos

UC-03	Ver grupos de permisos
Descripción	El usuario selecciona la opción de ver los grupos de permisos.
Secuencia normal	1. El usuario indica que quiere ver los grupos de permisos. 2. El sistema muestra una lista de grupos de permisos. 3. El sistema imprime una lista con toda la información de los grupos de permisos. 3. El usuario selecciona un grupo permiso. 4. El sistema muestra la información del grupo de permisos y sus permisos asociados.
Postcondición	Se ha creado una lista con la información de todos los grupos de permisos.

Cuadro 4.3: Caso de uso: Ver grupos de permisos

UC-03	Ver grupos de permisos de usuario
Descripción	El usuario selecciona la opción de ver los grupos de permisos usuario.
Secuencia normal	1. El usuario indica que quiere ver los grupos de permisos de usuario. 2. El sistema muestra una lista de grupos de permisos de usuario. 3. El sistema imprime una lista con toda la información de los grupos de permisos. 3. El usuario selecciona un grupo permiso de usuario. 4. El sistema muestra la información del grupo de permisos de usuario y sus permisos asociados.
Postcondición	Se ha creado una lista con la información de todos los grupos de permisos de usuario.

Cuadro 4.4: Caso de uso: Ver grupos de permisos de usuario

4.5.2. Casos de uso app de escritorio

A continuación, tenemos los casos de uso para la aplicación de escritorio:

UC-06	Revocar permisos.
Descripción	El usuario selecciona la opción de ir revocar permisos
Secuencia normal	1. El usuario indica que quiere revocar permisos. 2. El sistema explica al usuario como revocar permisos. 3. El usuario indica que quiere ir a los ajustes. 4. El sistema redirige al usuario a los ajustes de la aplicación.
Postcondición	Se han revocado los permisos que el usuario desea revocar.

Cuadro 4.5: Caso de uso: Revocar permisos

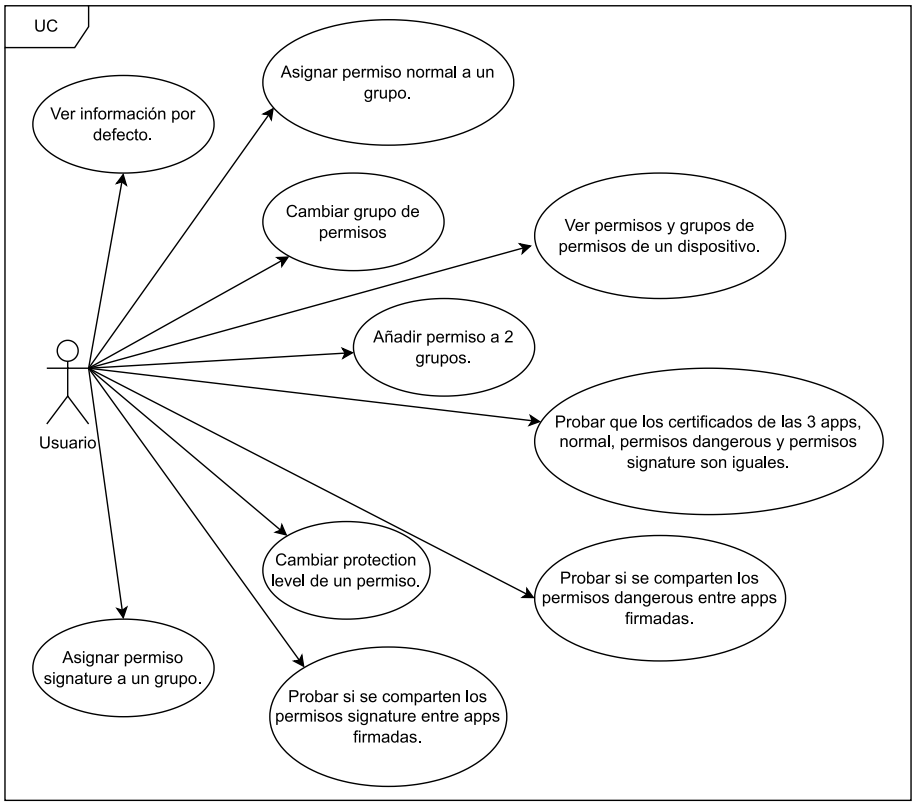

Figura 4.2: Flujo de trabajo general para la aplicación de escritorio.

UC-01	Ver información por defecto.
Descripción	El usuario quiere ver la información por defecto de algún permiso, grupo de permisos o protection level.
Secuencia normal	1. El usuario pulsa la opción “Grupos y permisos por defecto” 2. El sistema muestra 3 listas, una con permisos otra con grupos de permisos y otra con protection level 3. El usuario pulsa en un permiso 4. El sistema muestra el grupo de permisos y el protection level del permiso si existen, si no muestra un espacio blanco.
Flujos alternativos	3a. El usuario pulsa un grupo de permisos 4a. El sistema muestra los permisos que pertenecen a ese grupo
	3b. El usuario pulsa en un protection level 4b. El sistema muestra los permisos que tienen ese protection level
Postcondición	El usuario ha obtenido información sobre permisos, grupos de permisos y protection level.

Cuadro 4.6: Caso de uso: Ver información por defecto

UC-02	Cambiar grupo de permisos
Descripción	El usuario desea crear un apk con un permiso asignado a un grupo de permisos concreto
Secuencia normal	1. El usuario pulsa la opción "Cambiar grupo de permisos". 2. El sistema muestra una lista con permisos y otra con grupos de permisos. 3. El usuario selecciona un único permiso y grupo de permisos y pulsa siguiente. 4. El sistema compila el apk y muestra las opciones respecto al apk 5. El usuario pulsa en instalar el apk 6. El sistema instala el apk si hay un dispositivo compatible conectado.
Flujos alternativos	4a. El sistema muestra un error si no hay dispositivos compatibles conectados.
	5a. El usuario pulsa en abrir el explorador de archivos. 6a. El sistema abre el explorador de archivos.
Postcondición	Se ha obtenido un apk que solicita un permiso con un grupo elegido por el usuario.

Cuadro 4.7: Caso de uso: Cambiar grupo de permisos

UC-03	Añadir permiso a 2 grupos.
Descripción	El usuario quiere añadir un permiso a dos grupos de permisos.
Secuencia normal	1. El usuario selecciona la opción de "Añadir permiso a dos grupos". 2. El sistema muestra una lista con permisos y otra con grupos de permisos. 3. El usuario selecciona un único permiso y dos grupos de permisos y pulsa siguiente. 4. El sistema intenta compilar el apk y muestra un error de compilación y una opción. 5. El usuario pulsa en abrir el explorador de archivos. 6. El sistema abre el explorador de archivos.
Flujos alternativos	4a. El sistema muestra un error si no hay dispositivos compatibles conectados.
Postcondición	El usuario ve un mensaje de error al compilar el apk

Cuadro 4.8: Caso de uso: Añadir permiso a 2 grupos.

UC-04	Cambiar protection level de un permiso.
Descripción	El usuario desea crear un apk que intente cambiar el protection level de un permiso elegido por el usuario.
Secuencia normal	1. El usuario pulsa la opción de "Cambiar protection level" 2. El sistema muestra una lista de permisos y otra de protection level. 3. El usuario elige un permiso y un protection level. 4. El sistema compila el apk y muestra opciones. 5. El usuario pulsa en instalar el apk 6. El sistema instala el apk si hay un dispositivo compatible conectado.
Flujos alternativos	4a. El sistema muestra un error si no hay dispositivos compatibles conectados.
	5a. El usuario pulsa en abrir el explorador de archivos. 6a. El sistema abre el explorador de archivos.
Postcondición	Se ha generado un apk que intenta cambiar el protection level de un permiso.

Cuadro 4.9: Caso de uso: Cambiar protection level de un permiso.

UC-05	Asignar permiso normal a un grupo.
Descripción	El usuario quiere añadir un permiso normal a un grupo.
Secuencia normal	1. El usuario selecciona la opción “Asignar permiso normal a un grupo”. 2. El sistema muestra una lista de permisos normales y otra de grupos de e permisos. 3. El usuario selecciona un único permiso y grupo de permisos. 4. El sistema compila el apk y muestra opciones. 5. El usuario pulsa en instalar el apk 6. El sistema instala el apk si hay un dispositivo compatible conectado.
Flujos alternativos	4a. El sistema muestra un error si no hay dispositivos compatibles conectados.
	5a. El usuario pulsa en abrir el explorador de archivos. 6a. El sistema abre el explorador de archivos.
Postcondición	Se ha generado un apk que trata de añadir un permiso normal a un grupo.

Cuadro 4.10: Caso de uso: Asignar permiso normal a un grupo.

UC-06	Asignar permiso signature a un grupo.
Descripción	El usuario quiere añadir un permiso signature a un grupo.
Secuencia normal	1. El usuario selecciona la opción “Asignar permiso normal a un grupo”. 2. El sistema muestra una lista de permisos signature y otra de grupos de e permisos. 3. El usuario selecciona un único permiso y grupo de permisos. 4. El sistema compila el apk y muestra opciones. 5. El usuario pulsa en instalar el apk 6. El sistema instala el apk si hay un dispositivo compatible conectado.
Flujos alternativos	4a. El sistema muestra un error si no hay dispositivos compatibles conectados.
	5a. El usuario pulsa en abrir el explorador de archivos. 6a. El sistema abre el explorador de archivos.
Postcondición	Se ha generado un apk que trata de añadir un permiso signature a un grupo.

Cuadro 4.11: Caso de uso: Asignar permiso signature a un grupo.

UC-07	Ver permisos y grupos de permisos de un dispositivo.
Descripción	El usuario desea ver los permisos y grupos de permisos de un dispositivo.
Secuencia normal	1. El usuario selecciona la opción “Ver permisos y grupos de permiso del dispositivo”. 2. El sistema indica al usuario que debe hacer. 3. El usuario pulsa el botón para ejecutar adb. 4. El sistema muestra una lista de permisos y grupos de permisos del dispositivo.
Flujos alternativos	4a. El sistema muestra un error si no hay dispositivos compatibles conectados.
Postcondición	El usuario ve una lista de permisos y grupos de permisos del dispositivo.

Cuadro 4.12: Caso de uso: Ver permisos y grupos de permisos de un dispositivo.

UC-08	Probar si se comparten los permisos dangerous entre apps firmadas.
Descripción	El usuario quiere comprobar si 2 apps firmadas con el mismo certificado obtienen permisos dangerous si solo una de ellas lo solicita.
Secuencia normal	1. El usuario selecciona la opción “Pruebas con apps firmadas”. 2. El sistema muestra opciones con apps firmadas. 3. El usuario selecciona “Instalar apps firmadas dangerous”. 4. El sistema instala las 2 apps y se lo indica al usuario.
Flujos alternativos	4a. El sistema muestra un error si no hay dispositivos compatibles conectados.
Postcondición	Se han instalado 2 apps en el dispositivo

Cuadro 4.13: Caso de uso: Probar si se comparten los permisos dangerous entre apps firmadas.

UC-09	Probar si se comparten los permisos signature entre apps firmadas.
Descripción	El usuario quiere comprobar si 2 apps firmadas con el mismo certificado obtienen permisos signature si solo una de ellas lo solicita.
Secuencia normal	1. El usuario selecciona la opción "Pruebas con apps firmadas". 2. El sistema muestra opciones con apps firmadas. 3. El usuario selecciona "Instalar apps firmadas signature". 4. El sistema instala las 2 apps y se lo indica al usuario.
Flujos alternativos	4a. El sistema muestra un error si no hay dispositivos compatibles conectados.
Postcondición	Se han instalado 2 apps en el dispositivo

Cuadro 4.14: Caso de uso: Probar si se comparten los permisos signature entre apps firmadas.

UC-10	Probar que los certificados de las 3 apps, normal, permisos dangerous y permisos signature son iguales.
Descripción	El usuario quiere comprobar que, efectivamente, las apps para probar aplicaciones firmadas están firmadas con el mismo certificado.
Secuencia normal	1. El usuario selecciona la opción "Pruebas con apps firmadas". 2. El sistema muestra opciones con apps firmadas. 3. El usuario selecciona "Comprobar firmas". 4. El sistema muestra información sobre la firma de las 3 apps.
Flujos alternativos	4a. El sistema muestra un error si no hay dispositivos compatibles conectados.
Postcondición	El usuario ha podido comprobar que las 3 apps están firmadas con el mismo certificado.

Cuadro 4.15: Caso de uso: Probar que los certificados de las 3 apps, normal, permisos dangerous y permisos signature son iguales.

Capítulo 5

Diseño, implementación y pruebas

En esta sección se detallará como se organizan los proyectos de Android, el patrón de diseño que se empleará para desarrollar la aplicación Android, así como las tecnologías que usaremos para implementarla.

5.1. Organización de los proyectos Android

Para poder entender que patrón de diseño vamos a usar y algunos de los conceptos que se mencionarán, primero debemos saber cómo se estructura y organiza un proyecto Android.

Lo primero es que Android soporta de forma nativa dos lenguajes de programación, Java y Kotlin, siendo el segundo un lenguaje creado por Google para sustituir a Java ya que cuenta con una sintaxis más sencilla y su curva de aprendizaje es menor, en este proyecto usaremos kotlin. Por otro lado, los proyectos de Android que se realicen usando Android Studio contendrán los siguientes directorios y archivos:

- **Directorio app:**

  Este directorio contiene los archivos y recursos específicos de la aplicación. La mayoría del trabajo de desarrollo se realiza dentro de este directorio.

- **Directorio app/build:**

  Este directorio se genera automáticamente y contiene archivos generados por el proceso de compilación, como archivos APK y otros archivos intermedios. Por lo general, no necesitas modificar nada en este directorio.

- **Directorio app/src:**

  Este directorio es donde se almacenan todos los archivos fuente de la aplicación.

- **Directorio app/libs:**

  Este directorio puede contener bibliotecas de terceros en forma de archivos JAR (Java Archive) o AAR (Android Archive). Estas bibliotecas se utilizan en el proyecto.

- **Directorio .gradle:**

  Contiene archivos de configuración relacionados con Gradle, que es la herramienta de compilación utilizada por Android Studio.

- **Directorio gradle/wrapper:**

  Contiene archivos relacionados con el sistema de distribución de Gradle, como el archivo gradle-wrapper.properties, que especifica la versión de Gradle que se debe utilizar en el proyecto.

- **Archivo build.gradle:**

  Este archivo define la configuración de Gradle para el proyecto. Puede haber uno en el nivel del proyecto (archivo build.gradle) y otro en el nivel del módulo (app/build.gradle), que se utiliza para configurar la compilación y las dependencias específicas de la aplicación.

- **Archivo settings.gradle:**

  Este archivo contiene la configuración del proyecto de nivel superior, como la inclusión de módulos.

Dentro de esos directorios, nos interesa especialmente el directorio app/src/main donde encontramos lo siguiente:

- **Archivo AndroidManifest.xml [29]:** Es un archivo de configuración fundamental en el desarrollo de aplicaciones Android. Se encuentra en el directorio raíz del proyecto y contiene información crucial sobre la aplicación:

  1. Descripción de la Aplicación: El archivo AndroidManifest.xml contiene una descripción detallada de la aplicación Android. Esto incluye el nombre de la aplicación, el icono de la aplicación, el nombre del paquete de la aplicación y la versión del código y del nombre de la aplicación.
  2. Declaración de Actividades: Una de las funciones más importantes del archivo AndroidManifest.xml es declarar todas las actividades que componen la aplicación. Cada actividad debe estar registrada en este archivo junto con su nombre de clase y otra información relevante.
  3. Permisos Requeridos: El archivo también se utiliza para especificar qué permisos de sistema necesita la aplicación para funcionar correctamente. Si los permisos no están declarados aquí no será posible solicitarlos, ni siquiera los que requieren ser solicitados después en tiempo de ejecución.
  4. Configuración de Componentes: Además de las actividades, el archivo AndroidManifest.xml también se utiliza para declarar otros componentes de la aplicación, como servicios, receptores de difusión (broadcast receivers) y proveedores de contenido. Esto permite que el sistema Android conozca la estructura de la aplicación y cómo interactúan sus componentes.
  5. Filtros de Intenciones (Intents): Puedes definir filtros de intenciones en el archivo manifest para especificar qué componentes de la aplicación responden a qué tipos de intenciones. Esto es crucial para la navegación y la comunicación entre las diferentes partes de la aplicación.
  6. Información de Configuración: El archivo también incluye información de configuración, como el tema de la aplicación, la orientación de pantalla predeterminada, la compatibilidad con pantallas grandes o pequeñas y otros atributos que afectan la apariencia y el comportamiento de la aplicación.
  7. Declaración de Actividades de Inicio: Puedes especificar cuál será la actividad principal de inicio de la aplicación en el archivo AndroidManifest.xml. Esto es lo que Android lanzará cuando el usuario abra la aplicación.
  8. Definición de Iconos y Recursos: Puedes especificar los iconos de la aplicación y otros recursos gráficos en el archivo AndroidManifest.xml. Esto permite que el sistema Android acceda a estos recursos cuando sea necesario.

9. Etiquetas de Metadatos: Puedes incluir metadatos en el archivo manifest que son útiles para la aplicación. Por ejemplo, puedes incluir información sobre el autor, la licencia de la aplicación y otra información relevante.

- **Carpeta java:** esta carpeta contiene todos los archivos de código fuente java y como mínimo deberá contener una clase por actividad. Una actividad representa una sola pantalla con una interfaz de usuario y suele corresponder a una pantalla o ventana en la aplicación. Cada actividad debe estar definida en el archivo AndroidManifest.xml de la aplicación. Esto permite que el sistema Android conozca las actividades disponibles en la aplicación y cómo se pueden iniciar.

- **Carpeta res:** Es la carpeta de recursos de la aplicación. Estos recursos pueden incluir elementos de interfaz de usuario, archivos de diseño, imágenes, cadenas de texto y más. La carpeta "res" se divide en subdirectorios, cada uno de los cuales almacena un tipo específico de recurso.

  1. res/drawable: Esta carpeta almacena recursos de imágenes y gráficos que se utilizan en la aplicación. Aquí es donde almacenamos imágenes en formato PNG, JPEG o GIF que se utilizarán en la interfaz de usuario. Podemos crear subdirectorios dentro de "drawable" para organizar las imágenes por densidad de píxeles, como "drawable-hdpi", "drawable-xhdpi", "drawable-xxhdpi", etc.
  2. res/layout: Aquí es donde se almacenan los archivos XML que definen la estructura de diseño de las pantallas de la aplicación. Cada archivo XML representa la disposición de una actividad o fragmento en la interfaz de usuario. Por ejemplo, si tenemos una actividad de inicio, su diseño se definirá en un archivo XML en esta carpeta.
  3. res/values: En esta carpeta se encuentran archivos XML que almacenan recursos de valores, como cadenas de texto, colores, dimensiones, estilos y temas. Los archivos "strings.xml" contienen cadenas de texto utilizadas en la aplicación, y otros archivos XML pueden definir colores, estilos de texto, tamaños de fuente y otros valores.
  4. res/mipmap: Esta carpeta almacena iconos de la aplicación que se utilizan en el lanzador del dispositivo y en otras áreas donde se necesitan iconos de alta calidad. Similar a "drawable", podemos crear subdirectorios para organizar iconos según la densidad de píxeles.
  5. res/menu: Si la aplicación utiliza menús, los archivos XML que definen la estructura y el contenido de los menús se almacenan en esta carpeta. Por ejemplo, los menús contextuales y de opciones se crean aquí.
  6. res/animator y res/anim: Aquí se almacenan archivos XML que definen animaciones utilizadas en la aplicación. "res/animator" se utiliza para las animaciones de Property Animation, mientras que "res/anim" se usa para las animaciones View Animation.
  7. res/xml: Podemos almacenar archivos XML adicionales en esta carpeta para varios propósitos. Esto podría incluir archivos de configuración personalizados, definiciones de estilos o cualquier otro recurso XML que la aplicación requiera.
  8. res/raw: Esta carpeta es para recursos que no se procesan automáticamente por Android. Los archivos en "res/raw" se mantienen en su forma original y pueden incluir datos binarios, archivos de audio, video o cualquier otro recurso que la aplicación necesite.
  9. res/drawable-nodpi: Si necesitamos recursos que no dependan de la densidad de píxeles del dispositivo, podemos almacenarlos aquí. Esto es útil para recursos como archivos SVG que no necesitan escalarse según la densidad de píxeles.
  10. res/font: A partir de Android 8.0 (Oreo), podemos almacenar fuentes personalizadas en esta carpeta y utilizarlas en nuestra aplicación.
  11. res/color: Esta carpeta se utiliza para almacenar archivos XML que definen paletas de colores utilizadas en la aplicación. Esto puede ayudar a mantener una consistencia en la apariencia de la aplicación.

5.2. Patrones de diseño utilizados

Los patrones de diseño son una solución general y reutilizable para un problema común que se encuentra al diseñar software.
La utilidad de los patrones de diseño es que proporcionan un lenguaje común y una estructura para que los desarrolladores comuniquen y documenten sus diseños. También promueven la flexibilidad, la escalabilidad y la mantenibilidad del código, ya que ofrecen soluciones probadas para problemas comunes y evitan la necesidad de reinventar la rueda cada vez que se enfrenta a un problema similar.

5.2.1. Patrón MVP

En este caso usaremos el patrón MVP (Modelo-Vista-Presentador [32] para la aplicación Android que es un patrón de diseño arquitectónico que se utiliza comúnmente en el desarrollo de aplicaciones de software, especialmente en aplicaciones de interfaz de usuario. El patrón MVP se divide en tres componentes principales:

1. **Model (Modelo):**
   - El Modelo representa la capa de datos de la aplicación.
   - Es responsable de la gestión de datos, lógica de negocio y la interacción con fuentes de datos como bases de datos, servicios web, etc.
   - Normalmente no tiene conocimiento de la Vista ni del Presentador.
   - Puede emitir eventos o notificaciones cuando los datos cambian.
2. **View (Vista):**
   - La Vista representa la capa de presentación de la aplicación.
   - Es responsable de mostrar los datos al usuario y de recoger la entrada del usuario.
   - Idealmente, la Vista debe ser lo más tonta posible, es decir, no debe contener lógica de negocio.
   - Se comunica con el Presentador para solicitar datos y realizar acciones.
3. **Presenter (Presentador):**
   - El Presentador actúa como un intermediario entre el Modelo y la Vista.
   - Contiene la lógica de negocio y la lógica de presentación.
   - Recibe las solicitudes de la Vista y coordina las operaciones necesarias en el Modelo.
   - Actualiza la Vista con los datos adecuados.
   - No debe contener referencias directas a componentes de la Vista, lo que permite que la Vista sea fácilmente reemplazada o probada.

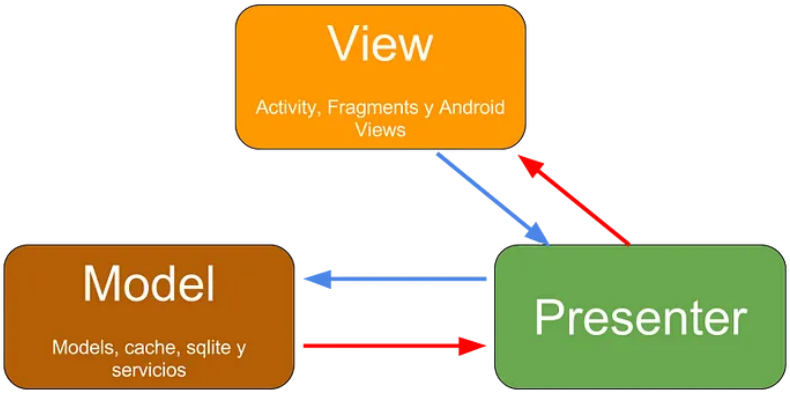

Figura 5.1: Patrón MVP
Figura obtenida de: "Model-View-Presenter ... otro post más" [32]

La implementación exacta en este caso consiste en 4 actividades o vistas con un presentador para cada una y un modelo de datos, detallaremos esto más adelante.

5.2.2. Patrón MVC

El patrón MVC(Modelo-Vista-Controlador) patrón es uno de los más utilizados y muy similar al anterior, consiste en modelos, vistas y controladores donde ningún modelo debe depender de los controladores o las vistas, las vistas no deben depender de controladores y modelos y los controladores no pueden depender de vistas y modelos. Lo usaremos en el desarrollo de la aplicación de escritorio.

- **Modelo:** El modelo representa los datos y la lógica de negocios de la aplicación. Se encarga de almacenar los datos, proporcionar acceso a ellos y realizar las operaciones necesarias para el negocio.
- **Vista:** La vista es responsable de la presentación de los datos al usuario. Se encarga de generar la interfaz de usuario y responder a las interacciones del usuario.
- **Controlador:** El controlador es el intermediario entre la vista y el modelo. Se encarga de recibir las entradas del usuario, actualizar el modelo y enviar las actualizaciones a la vista.

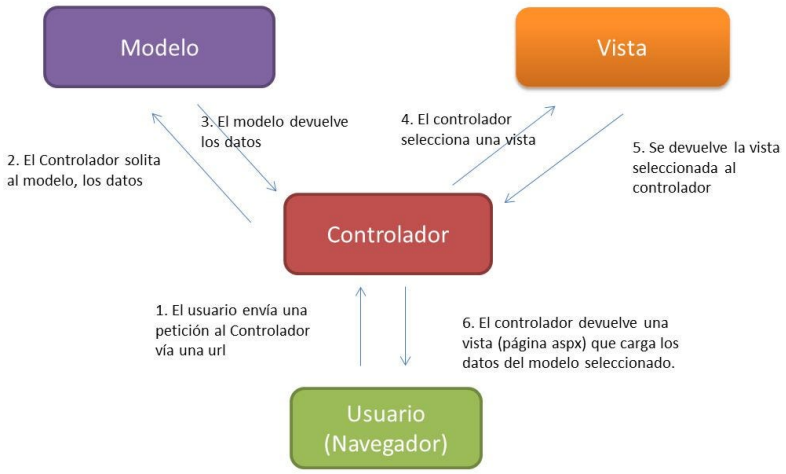

Figura 5.2: Patrón MVC
Figura obtenida de: CodeIgniter – Introducción [31]

El patrón de diseño MVC es similar al patrón MVP, pero tiene algunas diferencias clave. En MVP, el presentador se encarga de la lógica de presentación, mientras que en MVC, la vista se encarga de esta lógica. Dado que en Android las vistas estas formadas por archivos xml y en la aplicación de escritorio se genera por código python, el patrón MVP tiene más sentido en Android y el MVC tiene más sentido en escritorio.

5.3. Diagrama de despliegue

En esta sección veremos el diagrama de despliegue, es decir, como será la estructura física del sistema con sus nodos, componentes y relaciones.

- Nodo: Representan los elementos físicos de un sistema, como servidores, estaciones de trabajo, dispositivos móviles, etc.
- Componente: Representan los elementos de software de un sistema, como aplicaciones, bibliotecas, módulos, etc.
- Relación: Representan las conexiones entre los nodos y los componentes.

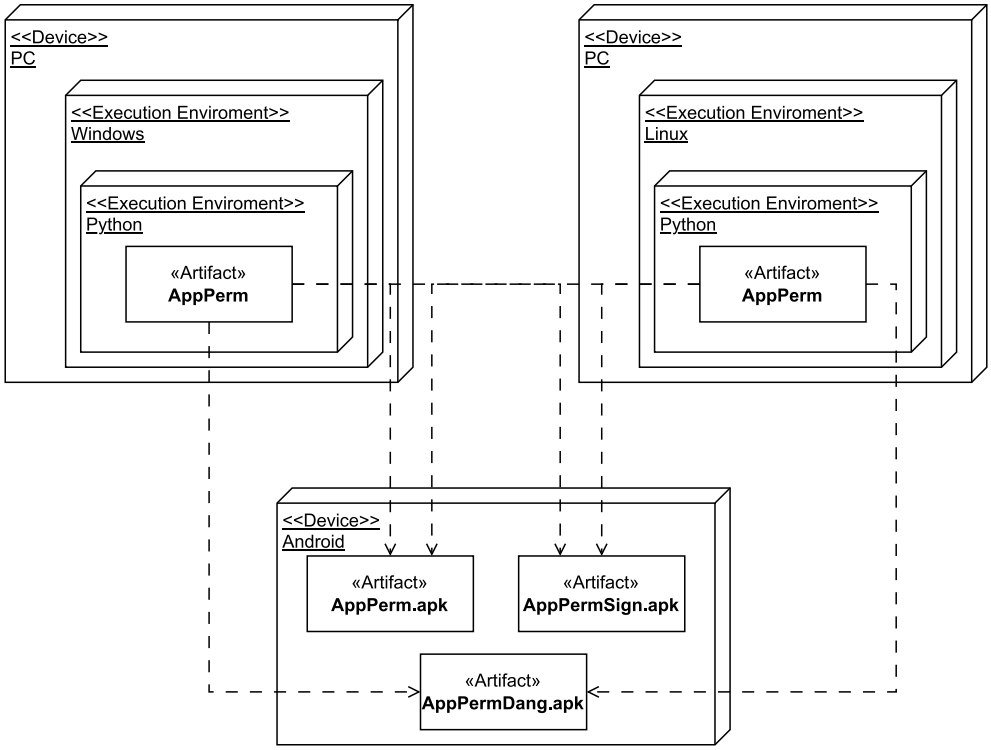

Figura 5.3: Diagrama de despliegue

En el diagrama anterior vemos 3 nodos, 2 PCs, uno con Linux y otro con Windows y un dispositivo Android. Cada uno de los PCs tiene a su vez 3 componentes, el sistema operativo, el entorno de ejecución Python y la aplicación AppPerm. El dispositivo Android contendrá 3 aplicaciones.
También, vemos que las 3 aplicaciones están relacionadas con las aplicaciones AppPerm indicando que las aplicaciones del dispositivo Android se instalan desde los PCs.

5.4. Diagramas de clases

un diagrama de despliegue representa la estructura estática de un sistema, mostrando las clases del sistema, sus atributos, operaciones (o métodos), y las relaciones entre los objetos.

- Clases: Representan los tipos de objetos que existen en un sistema.
- Atributos: Representan los datos que almacenan los objetos.
- Operaciones: Representan las acciones que pueden realizar los objetos.
- Relaciones: Representan las conexiones entre los objetos.

En este sistema encontramos 2 diagramas de clases, uno para cada aplicación:

5.4.1. Aplicación de escritorio

A continuación, tenemos el diagrama de clases para la aplicación de escritorio donde vemos que está formada por 7 vistas, cada una con su controlador y 4 modelos de datos. Las vistas tienen el formato de nombre VistaNombre, los controladores NombreContr y los modelos solo tienen el nombre de la clase.

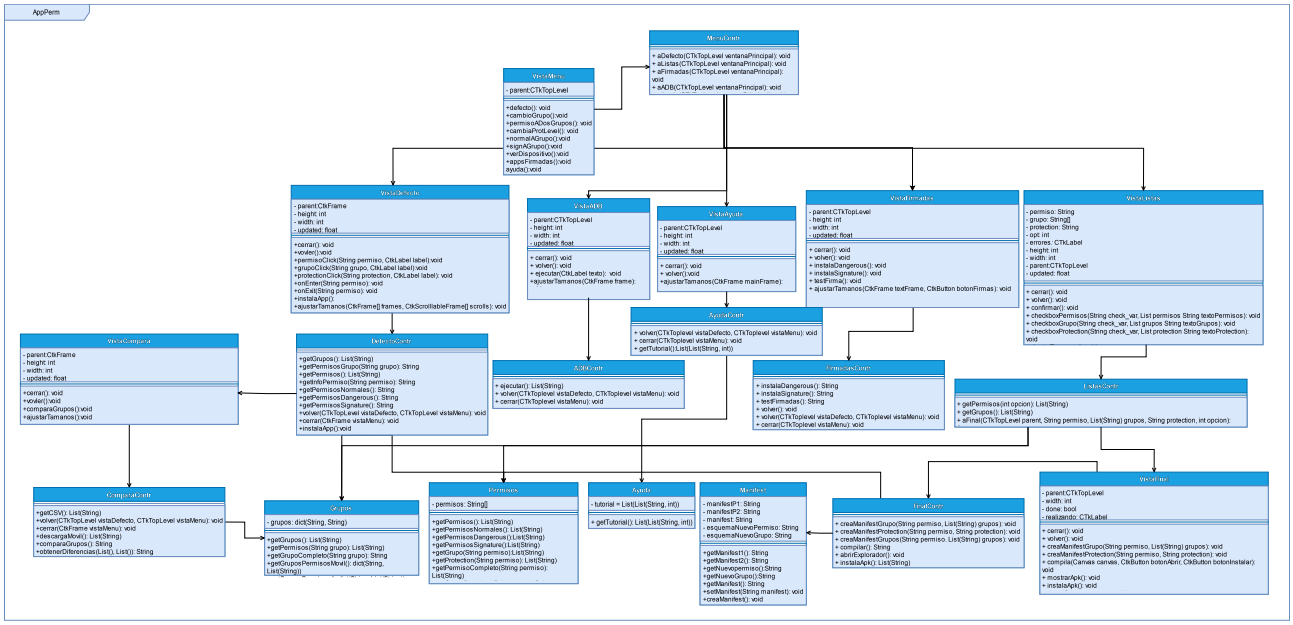

Figura 5.4: Diagrama de clases para la app Python

5.4.2. Aplicación Android

A continuación, tenemos el diagrama de clases para la aplicación Python donde vemos que está formada por 6 presentadores, cada una con su controlador y 1 modelo de datos. Los presentadores tienen el formato de nombre PresenterNombre, los controladores solo tienen el nombre de la clase y el modelo es PermisosMod.

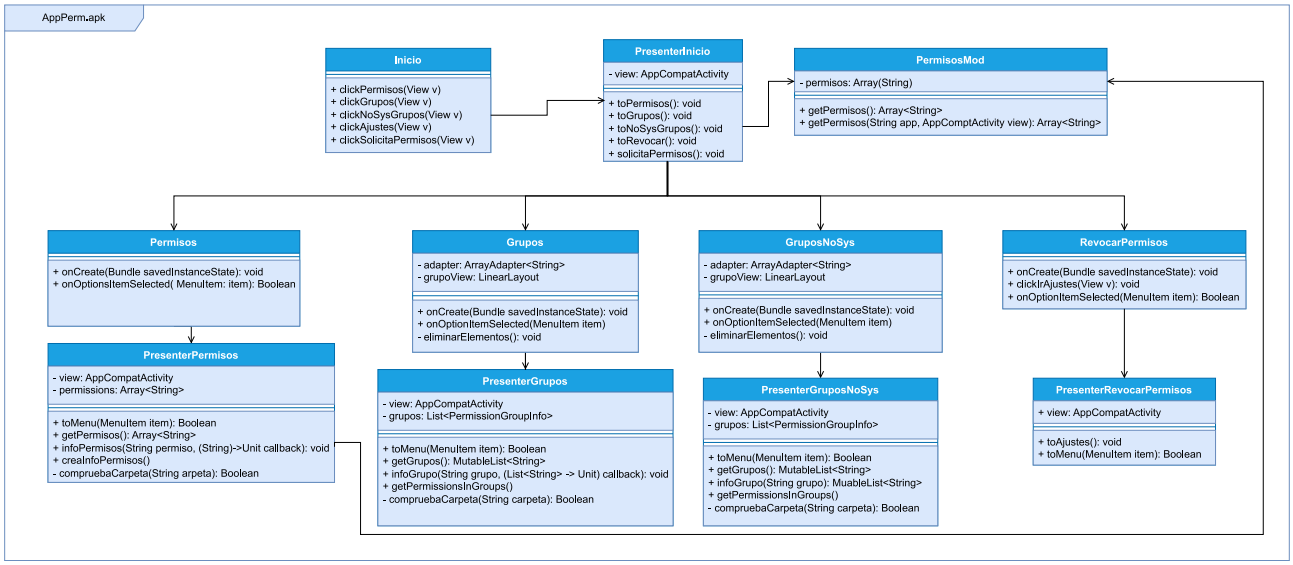

Figura 5.5: Diagrama de clases para la app Android

5.5. Implementación

A continuación, veremos como son las tecnologías usadas, como se organizan los datos y como se han conseguido estos.

5.5.1. Tecnologías usadas

Para la aplicación de escritorio encargada de crear distintas aplicaciones Android en función de lo que deseamos probar se ha usado Python, un lenguaje de programación muy popular con el que podemos conseguir, con unas pocas líneas, lo mismo que con otros requeriríamos varias, además es bastante sencillo y cuenta con una gran cantidad de librerías muy optimizadas para realiza gran cantidad de funciones.

Dado que la aplicación cuenta con interfaz de usuario y debe trabajar con datos, usamos el patrón MVC para su desarrollo. En la parte gráfica usamos la librería por defecto Tkinter y la librería CustomTkinter que, aunque tienen pocas opciones de personalización son sencillas de usar y darán una interfaz suficiente para nuestro objetivo.

En cuanto a la app Android, usaremos Kotlin, el lenguaje principal de programación junto a Java, aunque kotlin suele generar código más sencillo de leer y requiere escribir menos para obtener los mismos resultados, además, Android esta optimizado para ejecutar código Kotlin frente a Java.

De nuevo, tenemos una aplicación con interfaz de usuario y en este caso usaremos el patrón MVP que se ajusta mejor que MVC para apps Android como se explicó antes. En cuanto a la plataforma de desarrollo, usaremos Android Studio que nos ofrece las herramientas necesarias para desarrollar este tipo de apps sencillas.

5.5.2. Generar datos

Una de las funciones de la app será ver los permisos por defecto que encontramos en Android. Para ello, obtendremos una copia del Manifest de AOSP y, mediante un script Python independiente de la aplicación, generaremos un fichero csv con la información de los permisos.
El fichero puede obtenerse en este enlace https://github.com/aosp-mirror/platform_frameworks_base/blob/main/core/res/AndroidManifest.xml, una vez obtenido, buscaremos la sección RUNTIME PERMISSIONS donde se declaran todos los permisos, a continuación, recorremos todos los permisos y obtenemos aquellos de tipo normal, dangerous o signature, el resto no es posible obtenerlos desde una aplicación de terceros.

Dentro del proceso anterior encontramos 2 problemas, el primero es que algunos permisos signature, al intentar solicitarlos, nos darán un error en Android Studio indicando que solo puede concederse a aplicaciones del sistema. Para quedarnos solo con los permisos que nos interesan, añadimos una función al script que recibe un archivo con estos permisos y los elimina de la lista final.
Para obtener esos permisos, la única posibilidad es comprobarlos a mano, por tanto, lanzaremos el script, iremos a Android Studio y añadiremos los permisos que den problemas al fichero, sistema.txt, a continuación, volvemos a ejecutar el script y tendremos la lista final junto con unos permisos que no

se encuentran en el Manifest pero que usaremos para algunas pruebas.

Otro problema que encontramos es que, en este Manifest, los permisos se declaran como pertenecientes al grupo UNDEFINED, este grupo se refiere solo al grupo de usuario, luego se le asigna otro grupo de sistema, pero ese no podemos verlo en este archivo, por tanto, para hacer la lista final, lo que haremos será asignar a todos los permisos dangerous al último grupo de permisos creado antes de dicho permiso. Una vez tengamos la aplicación Android que solicita estos permisos, podremos comprobar desde esta que la asignación es correcta.

Otros datos que generamos es una lista de permisos y grupos de permisos como la anterior pero generada en el dispositivo Android, esta es la lista que usaremos para comparar con la anterior y comprobar que los grupos son correctos.

5.5.3. Generación de APKs

Otra de las funciones importantes de AppPerm es la creación de apks, por defecto tenemos una ya creada que es la que contiene y solicita todos los permisos obtenidos de la forma descrita anteriormente y también están precompiladas las apks para pruebas con apps firmadas.

Por tanto, quedan las apks que se crean para comprobar el cambio de grupos y protection level de los para ello, AppPerm viene con el SDK para Android 14 ya incluido y en el modelo de la aplicación tenemos toda la información necesaria para generar un Manifest que después se añade a la estructura de la app móvil y se compila usando el SDK teniendo en cuenta el sistema operativo que se está usando.

5.6. Pruebas

Una vez explicadas las herramientas que usaremos y como vamos a desarrollar las aplicaciones, tenemos que comprobar que estas funcionan. Estas herramientas tienen el objetivo de ayudarnos a dar respuesta a varias cuestiones sobre los permisos Android, por ello es importante asegurarnos de que funcionan correctamente, por tanto, en este capítulo vamos a probar las funcionalidades de ambas aplicaciones y a comprobar que el resultado es el esperado o, al menos, correcto.

5.6.1. Aplicación de escritorio

PFDesk-01	
Objetivo	Se quiere ver información de los permisos, los grupos de permisos y los protection level
Pasos	1. Se pulsa en “Grupos y permisos por defecto” 2. Se pulsa en un permiso, grupo de permisos o protection level.
Resultado esperado	Se obtiene información del elemento pulsado
Resultado de la prueba	Correcto

PFDesk-02	
Objetivo	Se quiere ver los permisos del dispositivo conectado
Pasos	1. Se pulsa la opción "Ver permisos y grupos del dispositivo" 2. Se pulsa en "Ejecutar adb"
Resultado esperado	Se muestra una lista donde salen los grupos de permisos y los permisos que estos contienen
Resultado de la prueba	Correcto

PFDesk-03	
Objetivo	Se quiere crear un apk con un AndroidManifest modificado
Pasos	1. Se pulsa una de las opciones para modificar permisos. 2. Se seleccionan los cambios que queremos para un permiso y se pulsa "Siguiente". 3. Esperamos y nos dará la opción de ver la ubicación del apk o instalarla.
Resultado esperado	Si el cambio es válido, se compilará un apk con éxito
Resultado de la prueba	Correcto

PFDesk-04	
Objetivo	Se quiere instalar un apk modificado
Pasos	1. Se pulsa una de las opciones para instalar un apk 2. Se selecciona el permiso y los cambios deseados 3. Se conecta un dispositivo con depuración USB 4. Se espera a que compile y se pulsa "Instalar apk"
Resultado esperado	Se instala una app en el dispositivo si las opciones seleccionadas son válidas
Resultado de la prueba	Correcto

PFDesk-05	
Objetivo	Se quiere comprobar que la firma de las apks para pruebas de apps firmadas usa el mismo certificado
Pasos	1. Se pulsa "Pruebas con apps firmadas" 2. Se pulsa en "Comprobar firmas"
Resultado esperado	Se muestra la información de la firma digital de las 3 apks.
Resultado de la prueba	Correcto

5.6.2. Aplicación Android

PFDesk-06	
Objetivo	Se quiere ver el apk creada
Pasos	1. Se pulsa una de las opciones que crean un apk. 2. Se seleccionan las opciones correctas y se pulsa "Siguiente". 3. Se espera que termine de compilar. 4. Se pulsa .ªbrir ruta apk".
Resultado esperado	Se muestra la carpeta que contiene el apk en el explorador de archivos.
Resultado de la prueba	Correcto

PFDesk-07	
Objetivo	Se quiere comparar la lista de permisos por defecto de la app con los del dispositivo.
Pasos	1. Se pulsa la opción "Grupos y permisos por defecto". 2. Se seleccionan Çomparar permisos".
Resultado esperado	Se muestran las diferencias entre ambos si las hay
Resultado de la prueba	Correcto

PFAnd-01	
Objetivo	Se quiere ver los permisos solicitados por la app
Pasos	1. Se pulsa "Permisos"
Resultado esperado	Se muestra una lista de permisos solicitados por la app.
Resultado de la prueba	Correcto

PFAnd-01	
Objetivo	Se quiere ver los grupos de permisos del dispositivo
Pasos	1. Se pulsa "Grupos o NoSysGrupos".
Resultado esperado	Se ve una lista de grupos de permisos.
Resultado de la prueba	Correcto

PFAnd-02	
Objetivo	Se quiere solicitar los permisos dangerous
Pasos	1. Se pulsa "Solicitar Permisos"
Resultado esperado	Se solicita al usuario que acepte los permisos
Resultado de la prueba	Correcto

PFAnd-03	
Objetivo	Se quiere revocar permisos
Pasos	1. El usuario pulsa Revocar Permisos".
Resultado esperado	El usuario ve los ajustes de la app donde puede ir a la sección de permisos
Resultado de la prueba	Correcto

PFAnd-04	
Objetivo	Se quiere ver información de un grupo de permisos
Pasos	1. El usuario pulsa "Grupos.º "NoSysGrupos". 2. El usuario selecciona un grupo.
Resultado esperado	El usuario ve los permisos asociados al grupo pulsado
Resultado de la prueba	Correcto

PFAnd-05	
Objetivo	Se quiere ver información de un permiso
Pasos	1. El usuario pulsa "Permisos". 2. El usuario selecciona un permiso.
Resultado esperado	El usuario ve el grupo de permisos y el protection level del permiso seleccionado
Resultado de la prueba	Correcto

Capítulo 6

Obtención de resultados

Una vez explicadas las herramientas que usaremos, en este capítulo vamos a ir dando respuesta a cada una de las preguntas que se plantearon en el capítulo 4.4.1. El procedimiento será el siguiente, primero se analizará cuáles son las opciones que tenemos para dar respuesta a cada una de las preguntas, luego usaremos las herramientas creadas para realizar las pruebas y finalmente daremos una respuesta.

6.1. Obtener permisos de un grupo

Es este caso el objetivo será conocer cómo podemos obtener los permisos que forman un grupo dado el nombre de dicho grupo. Para ello usaremos la aplicación de Android que se instala en el dispositivo al entrar en la primera opción de la app de escritorio y en la cual encontramos 2 opciones relacionadas con los grupos, la primera es Grupos y la segunda GruposNoSys.
Tenemos 2 opciones pues en Android existen una distinción entre los grupos de sistema y los grupos de usuario y la manera de obtener información de estos es diferente.
Además, es necesario haber solicitado en el AndroidManifest el permiso GET_PERMISSIONS para que la app tenga acceso a esta información.

Para los grupos de sistema, tenemos la siguiente función callback que recibe el nombre de un grupo y devuelve los permisos de este y su protection level.

Listing 6.1: Obtención de permisos de un grupo de sistema.

```
fun infoGrupo(grupo: String?, callback: (List<String>) -> Unit) {
    if (grupo != null) {
        view.packageManager.getPlatformPermissionsForGroup(grupo, view.mainExecutor) {
            group ->
            val infoGrupo: MutableList<String> = mutableListOf<String>()
            infoGrupo.add("Permisos en el grupo $grupo:\n")
            //Comprobamos si el grupo tiene o no permisos
               if (group.isEmpty()) {
                infoGrupo.add("Este grupo no contiene permisos")
            } else {
                for (permiso in group) {
                    //Miramos el protection level del permiso
                    val protection =
                            when (view.packageManager.getPermissionInfo(permiso, 0).
                                protection) {
                             0 -> "Normal"
                             1 -> "Dangerous"
                             2 -> "Signature"
                             4 -> "Internal"
                              else -> "Desconocido"
                            }
                    infoGrupo.add("$permiso\n$protection\n")
                }
            }
            callback(infoGrupo)
        }
    }
}
```

Por otro lado, tenemos esta otra función con la que obtenemos la misma información, pero para permisos de usuario:

Listing 6.2: Obtención de permisos de un grupo de usuario.

```
fun infoGrupo(grupo: String): MutableList<String> {
    val info: MutableList<String> = mutableListOf()
    info.add("Permisos en el grupo $grupo:\n")

    val permisos = view.packageManager.queryPermissionsByGroup(grupo, 0)

    if (permisos.isEmpty()) {
        info.add("Este grupo no contiene permisos")
    } else {
        for (permiso in permisos) {
            //Miramos el protection level del permiso
            val protection = when (permiso.protection) {
                PermissionInfo.PROTECTION_NORMAL -> "Normal"
                PermissionInfo.PROTECTION_DANGEROUS -> "Dangerous"
                PermissionInfo.PROTECTION_SIGNATURE -> "Signature"
                PermissionInfo.PROTECTION_INTERNAL -> "Internal"
                else -> "Desconocido"
            }
            info.add("${permiso.name}\n$protection\n")
        }
    }
    return info
}
```

Como vemos, estas funciones nos permiten obtener la información requerida y, junto con otras funciones del código, conseguiremos mostrar esta información en la app y crear un fichero CSV con esta información por si quisiéramos contrastarla con lo disponible en la aplicación de escritorio.

Finalmente, podemos confirmar que sí, es posible dado un nombre de un grupo obtener la información de dicho grupo.

6.2. Declaración de permisos y ADB

Dos preguntas muy similares de las planteadas son la 4 y la 5: "¿En qué fichero define Android los grupos de permisos soportados? ¿Puede conocerse vía adb?" y "¿En qué fichero define Android los permisos soportados? ¿Puede conocerse vía adb?". Para darles respuesta, se revisó el código fuente de AOSP hasta dar con el archivo en el cual se encuentra en Manifest que los declara y llegamos al fichero que se usa para obtener los permisos por defecto y que se encuentra en el siguiente enlace: .

Por otro lado, tenemos la pregunta de si pueden conocerse vía adb, para eso, tenemos en la aplicación de escritorio una opción que nos permite ver los permisos y grupos de permisos del dispositivo conectado. Para hacer eso sencillamente debemos conectar un dispositivo Android con depuración USB activada, pulsar el botón y se nos mostrará una lista tal cual la recibimos de adb con el comando *adb shell pm list permissions -g* que nos devuelve la lista de permisos en el dispositivo ordenados por grupo.

Por tanto, la respuesta es que el fichero con los permisos y grupos de permisos de Android se encuentra en el repositorio de GitHub de AOSP, en concreto en el repositorio "platform_frameworks_base " y en la ruta "core/res/AndroidManifest.xml" y sí, se puede acceder a esa información mediante adb.

6.3. Modificación de grupos de permisos

En este apartado trataremos las preguntas relacionadas con los grupos de permisos, ¿es posible modificar el grupo de permisos de un permiso? y ¿es posible que un permiso pertenezca a dos o más grupos?
Para ello, en la aplicación de escritorio contamos con 2 opciones, "Cambiar grupo de permiso" y "Añadir permiso a dos grupos" que nos permitirán probar si es posible modificar el grupo de permisos de un permiso o añadir un permiso a 2 grupos respectivamente.

En cuanto a la modificación del grupo de permisos, el proceso es bastante sencillo, para asignarle un grupo a un permiso, tenemos que volver a declarar el permiso pues es el momento en el que se le asigna el grupo.
A pesar de ser algo sencillo, debemos tener en cuenta que no es lo mismo un permiso declarado por la aplicación que un permiso de sistema, por eso en este caso tendríamos que probar ambas cosas y eso es algo que AppPerm permite pues tenemos algunos permisos de usuario además de los permisos del sistema.
Por tanto, para probarlo tenemos 4 opciones usando permisos de usuario, permisos de sistema, grupos de sistema y grupos de usuario. El proceso sería seleccionar un permiso y un grupo, compilar el apk e instalarla y, por último, mirar dentro de la app si los cambios han surgido efecto o no. A continuación, tenemos una tabla donde vemos que ocurre en cada caso.

Prueba	¿Se ha modificado el permiso?
Añadir un permiso de sistema a un grupo de sistema	No
Añadir un permiso de sistema a un grupo de usuario	No
Añadir un permiso de usuario a un grupo de sistema	Si
Añadir un permiso de usuario a un grupo de usuario	Si

Cuadro 6.1: Pruebas con grupos de permisos

La otra prueba que debemos hacer es ver si un permiso puede añadirse a dos grupos, para ello tenemos el segundo botón mencionado antes "Añadir permiso a dos grupos".
Esta vez nos encontramos con la duda de como añadir un permiso a dos grupos pues en la web para desarrolladores de Android no encontramos referencias a esta posibilidad. Por lo tanto, las opciones serían declarar 2 veces el permiso cada vez con un grupo o en una sola declaración añadir ambos grupos separados por algún tipo de separador.

Para hacer la prueba, AppPerm intenta lo primero, declarar el permiso 2 veces. Se elige un permiso y 2 grupos, se compila y se instala. En este punto veremos que solo se ha realizado con éxito el último de los 2 cambios.
Para la otra prueba, deberemos acceder al manifest de la aplicación que se encuentra en la ruta *Android\App-Perm\app\src\main\AndroidManifest.xml* desde la raíz del proyecto y que abriremos con Android Studio, para probar el segundo método, debemos coger el grupo de permisos de la segunda declaración y añadirlo en la primera para con distintos separadores, espacio, coma, punto y coma, incluso podemos intentar añadir la etiqueta android:permission_group 2 veces, en todas ellas el propio Android Studio nos dará un error en el primer caso porque la etiqueta no está cerrada y en el segundo porque la etiqueta está duplicada.

Por ejemplo, en esta situación la aplicación creada contendrá estas declaraciones en su Manifest:

Listing 6.3: Añadir permiso a 2 grupos.

```
1    <permission
2     android:name="appperm.permission.TEST1"
3     android:permissionGroup="appperm.permission-group.TEST_GROUP"
4     android:protectionLevel="dangerous" />
5    <permission
6     android:name="appperm.permission.TEST1"
7     android:permissionGroup="appperm.permission-group.TEST_GROUP1"
8     android:protectionLevel="dangerous" />
```

Y tendríamos que llegar a este caso donde sustituiremos la coma que hay en la linea 3 por el separador deseado:

```
1    <permission
2     android:name="appperm.permission.TEST1"
3     android:permissionGroup="appperm.permission-group.TEST_GROUP" , "appperm.permission-
          group.TEST_GROUP1"
4     android:protectionLevel="dangerous" />
```

Obtendremos en el propio Android Studio un error indicando que la etiqueta no está cerrada. Y el otro caso a probar sería este:

```
1    <permission
2     android:name="appperm.permission.TEST1"
3     android:permissionGroup="appperm.permission-group.TEST_GROUP"
4     android:permissionGroup="appperm.permission-group.TEST_GROUP1"
5     android:protectionLevel="dangerous" />
```

Donde nos dice que la etiqueta está duplicada.

Finalmente, con toda esta información, podemos contestar a la pregunta 2 "¿A qué grupo asigna el sistema operativo los permisos, al declarado en el fichero Manifest.xml o al predefinido por Google?" que se asigna al predefinido por Google pues no se puede cambiar el grupo de permisos de los permisos por defecto y a la pregunta "¿Puede un permiso estar en más de un grupo a la vez?" que no, no es posible que un permiso pertenezca a 2 grupos a la vez.

6.4. Añadir permisos normales y signature a un grupo de permisos

Siguiendo con los grupos de permisos, cuando miramos los permisos que hay en estos grupos, no encontramos permisos normales o signature, por tanto ¿Es posible si lo deseamos añadirlos a un grupo? En caso afirmativo, si se concede el permiso a ese grupo, ¿se concede permiso también al permiso signature?
En este caso la forma de hacer esto de nuevo es redeclarar el permiso con los cambios deseados, si no hay problemas de compilación, se instala la app y se prueba si se ha asignado correctamente el permiso al grupo. De nuevo podemos distinguir entre permisos de sistema y permisos de usuario.

Para probarlo, tenemos 2 opciones en AppPerm "Añadir permiso normal a un grupoz "Añadir permiso signature a un grupo. Cuando entramos, veremos una lista con permisos normales y signature y la lista de grupos, por tanto, el procedimiento es similar al anterior, se elige un permiso, se elige un grupo, se compila la aplicación y se instala.
Una vez hecho esto, entramos en la aplicación en el móvil y comprobamos que los permisos de sistema no se han modificado, sin embargo, los permisos de usuario sí que se han asignado a los grupos de usuario correctamente, sin embargo, si solicitamos los permisos veremos que no se concede el permiso signature. El permiso normal sí que se concede pues estos se conceden al instalar.

Por último, daremos respuesta a la pregunta 3 "¿Es posible asignar un permiso que no sea de tipo dangerous a un grupo de permisos?´´ y la respuesta es que sí, siempre que sea un permiso de usuario, pero solo para organizar los permisos, no influirá en si el permiso es concedido o no.

6.5. Modificación de protection level de permisos

A continuación, veremos qué ocurre al intentar cambiar el protection level de un permiso. De nuevo, podemos distinguir entre permisos de usuario y permisos de sistema. Además, podríamos pensar que, dado que un permiso dangerous es más peligroso, no se permitiría convertirlo en normal o signature, pero uno normal si podría pasarse a dangerous o signature si así se requiere.

En AppPerm, podemos probar todo esto con la opción "Cambiar protection level´´, en este caso se nos muestra una lista de permisos como antes, pero en este caso no podemos seleccionar grupos si no protection levels, así podemos hacer la siguiente tabla para comprobar los casos anteriores:

Como vemos en la tabla anterior volvemos a encontrarnos con que los permisos de sistema son inmutables, sin embargo, los permisos de usuario podemos modificarlos como nosotros queramos.

	Permisos de sistema	Permisos de usuario
Dangerous-Normal	NO	SI
Dangerous-Signature	NO	SI
Signature-Normal	NO	SI
Signature-Dangerous	NO	SI
Normal-Dangerous	NO	SI
Normal-Signature	NO	SI

Cuadro 6.2: Modificación de protection level

Con estas pruebas, podemos dar respuesta a la pregunta 1 y decir que el sistema asigna el permiso al protection level declarado por Google en el caso de los permisos de sistema y al declarado por el usuario en el caso de permisos de usuario pues Google nunca les asigno un protection level pues no son permisos que existan por defecto en Android.

6.6. Aplicaciones firmadas

En este último caso, veremos qué ocurre con las apps firmadas y como afecta esto a la concesión de permisos, según la web para desarrolladores de Android, una app firmada con un certificado digital puede obtener los permisos signature de otra firmada con ese mismo certificado la pregunta es si eso se cumple y si es aplicable a los permisos dangerous.
Para probarlo, la única opción será crear tres aplicaciones, la primera solicitará 2 permisos, uno dangerous y otro signature, la segunda solo solicitará el permiso dangerous y la tercera solo el permiso signature, instalamos las 3 y miramos que permisos se conceden en cada uno.

Para esto, en AppPerm tenemos esas 3 aplicaciones ya creadas, solo será necesario instalarlas y comprobar. Para ello, vamos a la opción de "Pruebas con apps firmadas´´ y elegimos una de las opciones, "Instalar apps firmadas dangerous´´ o "Instalar apps firmadas signature´´, una vez hecho eso, se instalan la app que solicita ambos permisos y la app que solo solicita uno de los dos según lo elegido.
Una vez instaladas, vamos al dispositivo, abrimos una de ellas y, si estamos probando permisos dangerous, damos a solicitar permisos, una vez concedidos o si estamos probando permisos signature, podemos comprobar que se hayan concedido, si así es, vamos a la otra app y sin solicitar permisos comprobamos directamente si se han concedido los mismos permisos, veremos que en el caso del permiso signature es así, en el caso del permiso dangerous no.
Una última comprobación que podemos hacer desde AppPerm, es ver que efectivamente las apps están firmadas con el mismo certificado, para ello solo tenemos que pulsar el botón "Comprobar firmas´´ en la sección de "Pruebas con apps firmadas´´ y se nos mostrará la salida del comando *apksigner verify –print-certs ¡apk¿*, el cual es parte del SDK de Android y permite comprobar si una app está firmada y si es así con que certificado se firmó.

Finalmente, podemos dar respuesta a las 2 últimas preguntas respecto a la concesión de permisos en apps firmadas y decir que, en el caso de los permisos signature, así es, una app firmada con el mismo certificado digital que otra, tiene acceso a los mismos permisos signature que otra app firmada con ese mismo certificado, pero esto no se aplica para los permisos dangerous, solo a los permisos signature.

Capítulo 7

Conclusiones y trabajo futuro

En este último capítulo, veremos un resumen de los resultados obtenidos en el capítulo anterior, que nos indican estos resultados y que podríamos hacer en un futuro para mejorar el trabajo hecho.

7.1. Conclusiones

En el capítulo anterior dimos respuesta a las preguntas planteadas desde el inicio explicando el proceso de prueba en cada caso. En este apartado veremos esas respuestas de forma resumida en la tabla 7.1 **??** y a continuación, veremos cómo afecta eso al objetivo que teníamos originalmente.

El objetivo era comprender la relación entre permisos y grupos de permisos en Android, así como comprobar las posibilidades que ofrece Google a un usuario para modificar y alterar dichos permisos y grupos de permisos.

En cuanto al primer punto, hemos visto que, por defecto, Android solo tiene permisos de tipo dangerous asignados a algún grupo, lo cual tiene sentido para organizar estos grupos según su funcionalidad, facilitando que el usuario no tenga que aceptar demasiados permisos a la vez que no es posible engañar a un usuario para que acepte un permiso que luego haga otra cosa ya que estos grupos aglutinan permisos muy similares y no pueden modificarse.
Por otro lado, hemos visto que, si es posible añadir otros permisos a grupos de permisos siempre y cuando sean permisos de usuario, esto es porque los permisos de usuario solo afectan a permisos dentro de la propia aplicación y por tanto no representan ningún peligro para el usuario, si el desarrollador crea grupos será por conseguir más organización pues no afectará a la seguridad.

Por otro lado, vemos que los certificados digitales pueden ser usos cuando una compañía crea una aplicación que luego requiere servicios proporcionados por otra pues gracias a las firmas digitales no es necesario conceder permiso para acceder a las funciones de otras apps del mismo desarrollador en todas las aplicaciones, con darlo en una será suficiente.

El otro punto que era ver las posibilidades que ofrece Google para alterar permisos y grupos de permiso, podemos ver que Google no ofrece ninguna posibilidad en cuanto a los permisos de sistema, estos están diseñados para que solo Google pueda modificarlos, garantizando así la seguridad del

usuario. Sin embargo, sí que da la opción de crear permisos con sus correspondientes protection level y grupos de permisos a modo de organización.

Pregunta	**Respuesta**
¿A qué protection level asigna el sistema operativo los permisos, al declarado en el fichero Manifest.xml o al predefinido por Google?	En el caso de permisos de sistema, al predefinido por Google, en el caso de permisos de usuario, al del usuario.
¿A qué grupo asigna el sistema operativo los permisos, al declarado en el fichero Manifest.xml o al predefinido por Google?	En el caso de permisos de sistema, al predefinido por Google, en el caso de permisos de usuario, al del usuario.
¿Es posible asociar un permiso que no sea de tipo dangerous a un grupo de permisos?	Si no es un permiso de sistema, si, los permisos de sistema no pueden modificarse.
¿En qué fichero define Android los grupos de permisos soportados? ¿Puede conocerse vía adb?	Se encuentra en el fichero "core/res/AndroidManifest.xml" del repositorio platform_frameworks_base de AOSP y si puede conocerse mediante adb
¿En qué fichero define Android los permisos soportados? ¿Puede conocerse vía adb?	Se encuentra en el fichero "core/res/AndroidManifest.xml" del repositorio platform_frameworks_base de AOSP y si puede conocerse mediante adb
¿Puede un permiso estar en más de un grupo a la vez?	No, no es posible asignar un permiso a varios grupos de permisos.
Dado un grupo de permisos, ¿es posible obtener qué permisos forman el grupo?	Si, en Android tenemos varias formas de obtener esa información.
Si una app logra un permiso de tipo dangerous, ¿otra app firmada con el mismo certificado digital consigue automáticamente ese permiso (se lo concede el SO sin intervención del usuario)?	No, los permisos dangerous deben ser concedidos manualmente siempre que sean requeridos.
Si una app logra un permiso de tipo signature, ¿otra app firmada con el mismo certificado digital consigue automáticamente ese permiso?	Si, los permisos signature, una vez concedidos en una aplicación se conceden a todas aquellas que estén firmadas con ese mismo certificado.

Cuadro 7.1: Resumen de respuestas dadas en el capítulo 6.

7.2. Trabajo futuro

Los siguientes pasos para este proyecto podrían incluir crear un método para que, cada vez que salga una nueva versión de Android, la aplicación pueda descargar el nuevo SDK y el nuevo Manifest para que se mantenga como una aplicación relevante, ya que dentro de 5 o 6 años puede haber nuevos permisos, grupos o incluso que algún permiso haya cambiado de grupo.

Otra posible mejora sería la inclusión de pasados SDK y Manifest, de forma que podamos hacer todas estas pruebas para versiones antiguas y comparar la robustez del sistema de permisos de Android antes contra su robustez actual.

Finalmente, hacer la aplicación más adaptable a distintos tipos de pantalla y hacerla más rápida y efectiva siempre es una ruta de mejora, en este caso por ejemplo, si usamos la aplicación en una pantalla con un formato que no sea 16:9 toda la interfaz acabará descolocada dejándola inservible, también en

cuanto a rapidez, cuando entramos a las opciones que muestran listas, estas tardan en cargar pues la app debe acceder a disco para leer esa información, por tanto cargar toda esa información en memoria al inicio podría ser buena idea.

Referencias

[1] *Qué es Android*,
https://www.android.com/intl/es_es/what-is-android/
Último acceso: 10/09/2023

[2] *Análisis de Android, el sistema operativo para móviles de Google*,
https://ibertronica.es/blog/tutoriales/android-sistema-operativo/
Último acceso: 10/09/2023

[3] *Tiempo de ejecución de Android (ART) y Dalvik*,
https://source.android.com/docs/core/runtime?hl=es-419
Último acceso: 10/09/2023

[4] *Descripción general de la capa de abstracción de hardware (HAL)*,
https://source.android.com/docs/core/architecture/hal?hl=es-419
Último acceso: 10/09/2023

[5] *Porcentaje uso Android*,
https://www.xataka.com/moviles/ha-pasado-casi-medio-ano-lanzamiento-android-13-todavia-no-
Último acceso: 10/09/2023

[6] *copyCat*,
https://cso.computerworld.es/social-security/copycat-el-malware-que-ha-infectado-a-14-mill
Último acceso: 10/09/2023

[7] *Historia y evolución de Android: cómo un sistema operativo para cámaras digitales acabó conquistando los móviles*,
https://www.xatakandroid.com/sistema-operativo/historia-evolucion-android-como-sistema-ope
Último acceso: 10/09/2023

[8] *Caso CamScanner*,
https://www.kaspersky.es/blog/camscanner-malicious-android-app/19148/
Último acceso: 10/09/2023

[9] *Caso FaceBook*,
https://www.nytimes.com/es/2018/03/20/espanol/cambridge-analytica-facebook.html

Último acceso: 10/09/2023

[10] *Todas las versiones de Android de la historia*,
https://www.android.com/intl/es_es/what-is-android/
Último acceso: 10/09/2023

[11] *Android Debug Bridge*,
https://developer.android.com/studio/command-line/adb?hl=es-419
Último acceso: 10/09/2023

[12] *Permisos en Android*,
https://developer.android.com/guide/topics/permissions/overview?hl=es-419#perm-groups
Último acceso: 10/09/2023

[13] *Lista de permisos*,
https://developer.android.com/reference/android/Manifest.permission
Último acceso: 10/09/2023

[14] *Lista de grupos*,
https://developer.android.com/reference/android/Manifest.permission_group
Último acceso: 10/09/2023

[15] *Cómo declarar permisos de la app*,
https://developer.android.com/training/permissions/declaring?hl=es-419
Último acceso: 10/09/2023

[16] *Solicitud permisos de tiempo de ejecución*,
https://developer.android.com/training/permissions/requesting?hl=es-419
Último acceso: 10/09/2023

[17] *Temas principales del sistema operativo Android*,
https://source.android.com/docs/core?hl=es-419
Último acceso: 10/09/2023

[18] *Descripción general de la arquitectura*,
https://source.android.com/docs/core/architecture?hl=es-419
Último acceso: 10/09/2023

[19] *Firma de Apps*,
https://developer.android.com/studio/publish/app-signing?hl=es-419
Último acceso: 10/09/2023

[20] *How Google Play Protect kept users safe in 2019*,
https://security.googleblog.com/search/label/google%20play%20protect
Último acceso: 10/09/2023

[21] *Qué es Google Play Protect, cómo protege tu móvil y cómo configurarlo* ,
https://www.xataka.com/basics/que-google-play-protect-como-protege-tu-movil
Último acceso: 10/09/2023

[22] *Funciones de seguridad de Android*,
https://source.android.com/security/features?hl=es-419
Último acceso: 10/09/2023

[23] *Model-View-Presenter ... otro post más (1)*,
https://www.android.com/intl/es_es/what-is-android/
Último acceso: 10/09/2023

[24] *Seguridad móvil*,
https://www.android.com/intl/es_es/safety/
Último acceso: 10/09/2023

[25] *Google Security Blog*,
https://security.googleblog.com/
Último acceso: 10/09/2023

[26] *Ley Orgánica 3/2018, de 5 de diciembre, de Protección de Datos Personales y garantía de los derechos digitales.*,
https://www.boe.es/buscar/act.php?id=BOE-A-2018-16673
Último acceso: 10/09/2023

[27] *Android o iOS: ¿qué sistema operativo consigue más conversiones?*,
https://marketing4ecommerce.net/android-o-ios-que-sistema-operativo-consigue-mas-conversio
Último acceso: 10/09/2023

[28] *AndroideProyecto de código abierto*,
https://source.android.com/?hl=es-419,
Último acceso: 10/09/2023

[29] *Descripción general del manifiesto de una app*,
https://developer.android.com/guide/topics/manifest/manifest-intro?hl=es-419
Último acceso: 10/09/2023

[30] *Introducción al desarrollo con Android*,
https://wiki.uqbar.org/wiki/articles/android-introduccion.html
Último acceso: 21/03/2024

[31] *CodeIgniter – Introducción*,
http://www.antoniorios.net/blog/?p=373
Último acceso: 21/03/2024

[32] *Model-View-Presenter ... otro post más*,
https://devpicon.medium.com/model-view-presenter-otro-post-ms-1-93319862e610
Último acceso: 21/03/2024

[33] *Python*
https://www.python.org/
Último acceso: 14/03/2024

[34] *Github*
https://github.com/
Último acceso: 14/03/2024

[35] *TKinter*
https://docs.python.org/es/3/library/tkinter.html
Último acceso: 14/03/2024

[36] *CTKinter*
https://customtkinter.tomschimansky.com/
Último acceso: 14/03/2024

[37] *Android*
https://www.android.com/
Último acceso: 14/03/2024

[38] *Overleaf*
https://es.overleaf.com/
Último acceso: 14/03/2024

[39] *PyQt vs Tkinter (Spanish)*
https://dev.to/amigosmaker/pyqt-vs-tkinter-spanish-2n4k
Último acceso: 14/03/2024

[40] *csv — Lectura y escritura de archivos CSV*
https://docs.python.org/es/3/library/csv.html
Último acceso: 14/03/2024

[41] *subprocess — Subprocess management*
https://docs.python.org/3/library/subprocess.html
Último acceso: 14/03/2024

[42] I. M. Almomani and A. A. Khayer, .A Comprehensive Analysis of the Android Permissions Systemïn IEEE Access, vol. 8, pp. 216671-216688, 2020, doi: 10.1109/ACCESS.2020.3041432. keywords: *Security;Analytical models;Mathematical model;Static analysis;Virtual machining;Tools;Access control;analysis;android;android application;android security;API level;APK;formal model;permission evolution;permission system;user privacy;security attack;survey*

[43] R. Johnson, Z. Wang, C. Gagnon and A. Stavrou, .Analysis of Android Applications' Permissions", 2012 IEEE Sixth International Conference on Software Security and Reliability Companion, Gaithersburg, MD, USA, 2012, pp. 45-46, doi: 10.1109/SERE-C.2012.44. keywords: *Smart phones;Androids;Humanoid robots;Security;Educational institutions;Software;Java*

[44] N. S. A. A. Bakar and I. Mahmud, .Empirical Analysis of Android Apps Permissions", 2013 International Conference on Advanced Computer Science Applications and Technologies, Kuching, Malaysia, 2013, pp. 406-411, doi: 10.1109/ACSAT.2013.86. keywords: *Smart phones;Androids;Humanoid robots;Internet;Sensors;Correlation;Security;Android applications;sensitive data;empirical analysis;statistics;permission types*

[45] Primal Wijesekera and Arjun Baokar and Ashkan Hosseini and Serge Egelman and David Wagner and Konstantin Beznosov, Android Permissions Remystified: A Field Study on Contextual Integrity, 24th USENIX Security Symposium (USENIX Security 15), 2015, 978-1-939133-11-3, Washington, D.C., 499–514, https://www.usenix.org/conference/usenixsecurity15/technical-sessions/presentation/wijesekera, USENIX Association,

Apéndice A

Creación del entorno de pruebas

El objetivo de esta sección será guiar a los interesados en instalar el software necesario para probar las herramientas desarrolladas para que sean capaces de probar por sí mismos todo lo que se va probando en este trabajo. En concreto, instalaremos miniconda, Android Studio, el entorno virtual de python y, opcionalmente, un emulador de Android. Esta guía será únicamente para Windows, el software necesario para Linux es el mismo, el proceso de instalación puede variar. A continuación, tenemos los pasos necesarios a seguir:

1. Instalar Miniconda, puede descargarse del siguiente enlace a su web https://docs.anaconda.com/free/miniconda/index.html y será suficiente con dejar todas las opciones por defecto.

2. Añadir conda al PATH de Windows o abrir la consola de conda con el archivo creado en el menú de inicio .ªnaconda Powershell Prompt".

3. Instalar Android Studio, en este caso se usó la versión 2023.2.1.23 Iguana, la última debería funcionar, aunque puede haber diferencias. En este caso puede descargarse de la web parea desarrolladores de Android: https://developer.android.com/studio?hl=es-419

4. Una vez está instalado Android Studio, clonamos el repositorio de App-Perm descargando el zip de https://github.com/Loloncio/App-Perm y descomprimiéndolo o, si tenemos instalado git, con el comando: *git clone https://github.com/Loloncio/App-Perm.git*, recordar la ruta de guardado para los próximos pasos

5. Configuración de Android Studio:

   a) Una vez tengamos AppPerm ejecutamos Android Studio.

   b) Seguir los pasos para la instalación inicial.

   c) Una vez descargado abrimos pulsamos en abrir proyecto, vamos a la ruta donde descargamos AppPerm y abrimos el proyecto que se encuentra en .ªpp-Perm/Android/App-Perm".

   d) Una vez abierto vamos a Tools SDK Manager, seleccionamos el SDK que se encuentra en "App-Perm/Android/sdk" y comprobamos que se detecta correctamente la versión con API 33 en la pestaña platforms, que estén marcados Android SDK Platform-Tools, Google USB Driver (Si vamos a usar un dispositivo) o Android Emulator (Si vamos a usar un dispositivo virtual) e Intel x86 Emulator Accelerator (Si vamos a usar un dispositivo virtual y nuestro PC tiene procesador Intel). Si falta alguno, lo marcamos y seguimos los pasos para realizar la instalación.

e) Al instalar el acelerador, nos preguntará la máxima RAM que queramos dedicarle, es recomendable darle mínimo 4GB, si no funciona no supondrá un problema, pero el dispositivo virtual no ira tan fluido.

6. (Opcional) Instalar dispositivo virtual:

   a) En Android Studio, abrimos la ventana Device Manager y pulsamos en el icono + (Çreate virtual device").

   b) Seleccionamos el dispositivo Pixel 8 (O uno con densidad de pixeles similar).

   c) En la siguiente pestaña descargamos la versión "Tiramisu", la seleccionamos e instalamos.

   d) En el apartado "Verify Configuration"pulsamos en "Show Advanced Settingsz debemos asegurarnos de que la orientación sea vertical, que tenga al menos 2GB de RAM y 10 GB de almacenamiento (5GB serían suficientes, pero es posible que se ralentice mucho el dispositivo).

   e) Ya tenemos creado el dispositivo virtual que iniciará al pulsar play en Device Manager.

   Cada vez que vayamos a probar algo que requiera instalar un apk deberemos tener el dispositivo virtual en marcha para que pueda ser detectado por adb.

7. Ahora creamos el entorno virtual para Python:

   a) En una terminal de conda escribimos *conda create –name AppPerm python=3.10* y le escribimos zçuando lo pida.

   b) Activamos el entorno con *conda activate AppPerm*

   c) Nos desplazamos en la consola al directorio raíz de AppPerm.

   d) Instalamos las dependencias con *pip install -r requirements.txt*

   e) Comprobamos que se ha hecho todo correctamente ejecutando: *python ./src/View/VistaMenu.py* Si se abre el menú entonces está listo.

Ya tenemos todo listo para empezar, cada vez que queramos probar cosas será necesario conectar el dispositivo Android para realizar algunas de las pruebas. Si el dispositivo es externo (un dispositivo físico) tendremos que activar la depuración USB en las opciones de desarrollador que encontramos en las opciones del dispositivo tras pulsar repetidamente en la versión de compilación. Este paso no se detalla pues la localización de estos apartados depende de la marca del dispositivo que vayamos a usar.

Finalmente, recordar que para la ejecución de la app a de ejecutarse el archivo VistaMenu.py que se encuentra en la carpeta src/View/ haciendo uso del comando python.

Apéndice B

Guía de usuario

B.1. Introducción

El presente manual explicará el funcionamiento de la aplicación App-Perm, explicando la funcionalidad de cada uno de sus botones y las distintas opciones que ofrecen tanto en su versión de escritorio como en la versión de Android.

En su versión para escritorio, App-Perm cuenta con 8 opciones disponibles dispuestas en 2 filas de 4 como se ve en la siguiente imagen:

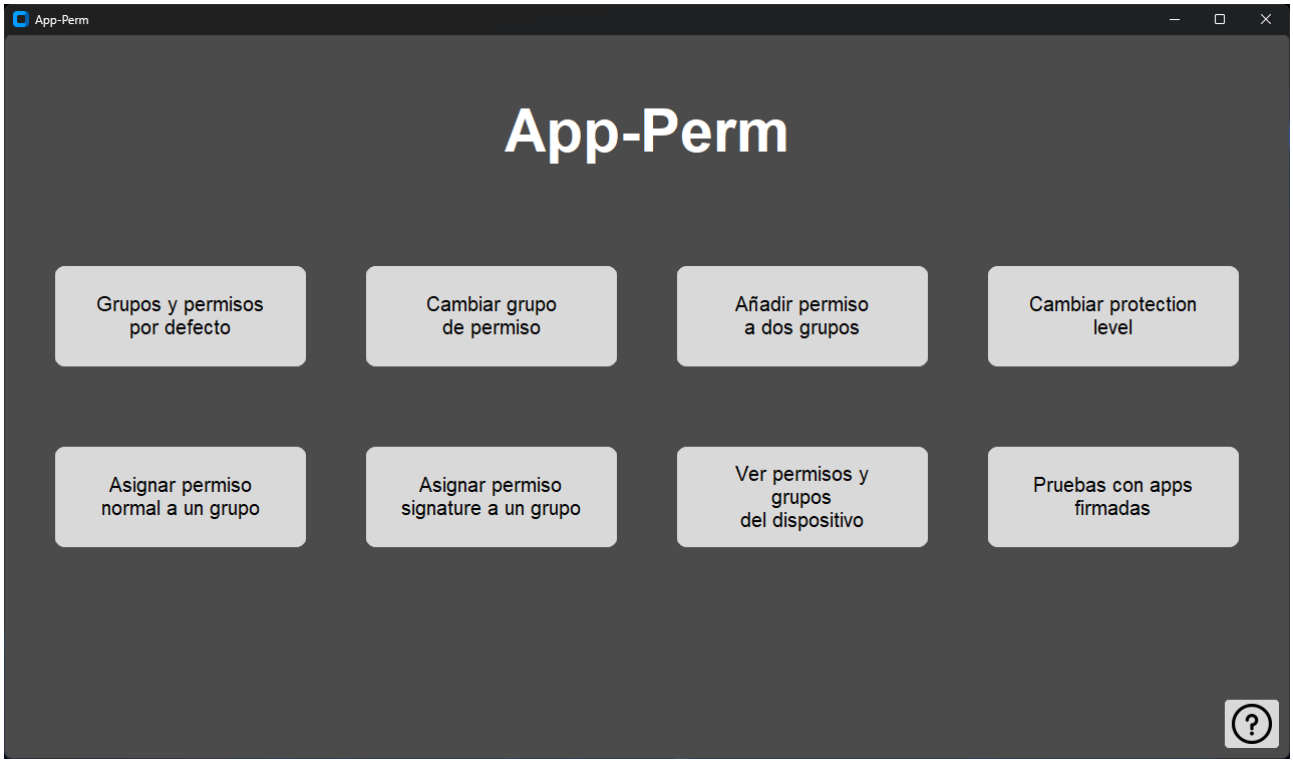

Figura B.1: Vista principal de la app de escritorio

En su versión para Android cuenta con 5 opciones dispuestas en 2 columnas:

A continuación, tenemos 2 secciones en las que explicaremos cada una de ellas:

Figura B.2: Vista principal de la app Android

B.2. App-Perm móvil

Esta aplicación permite ver los permisos solicitados en el manifiesto de la app, sus grupos de permisos y los protection level de cada permiso, también permite solicitar los permisos dangerous y nos dirige a los ajustes cuando queramos revocar alguno de ellos. Además, en segundo plano tenemos una función que crea un fichero con la información de los permisos, grupos de permisos y protection level para usar con la app de escritorio.

Tenemos 5 opciones, la opción "Permisos", nos permite ver una lista de los permisos declarados en el manifiesto de Android y obtener la información disponible sobre ellos: nombre, grupo y protection level.

La opción de "grupos", nos permite ver los grupos de permisos disponibles y que permisos forman su grupo. La siguiente opción, "nosysgrupos" nos permite ver los grupos que vería un usuario del sistema donde tenemos todos los permisos dangerous en un grupo llamado "Undefined" y el resto vacíos, también es en esta sección donde veremos los grupos que creamos nosotros.

La opción "Solicitar permisos" nos permite solicitar los permisos dangerous de forma que al volver a entrar en la opción "Permisos", veremos que la información cambia y los permisos dangerous previamente no concedidos ahora si lo están. La última opción, es la de "Revocar permisos" la cual nos lleva a los ajustes de la app por si es necesario en algún momento revocar los permisos ya concedidos.

La utilidad de la app es comprobar, por ejemplo, la información que tenemos en la ventana "Grupos y permisos por defecto" en la app de escritorio o ver cómo se comportan los permisos en el caso de que tengamos aplicaciones firmadas con el mismo certificado.

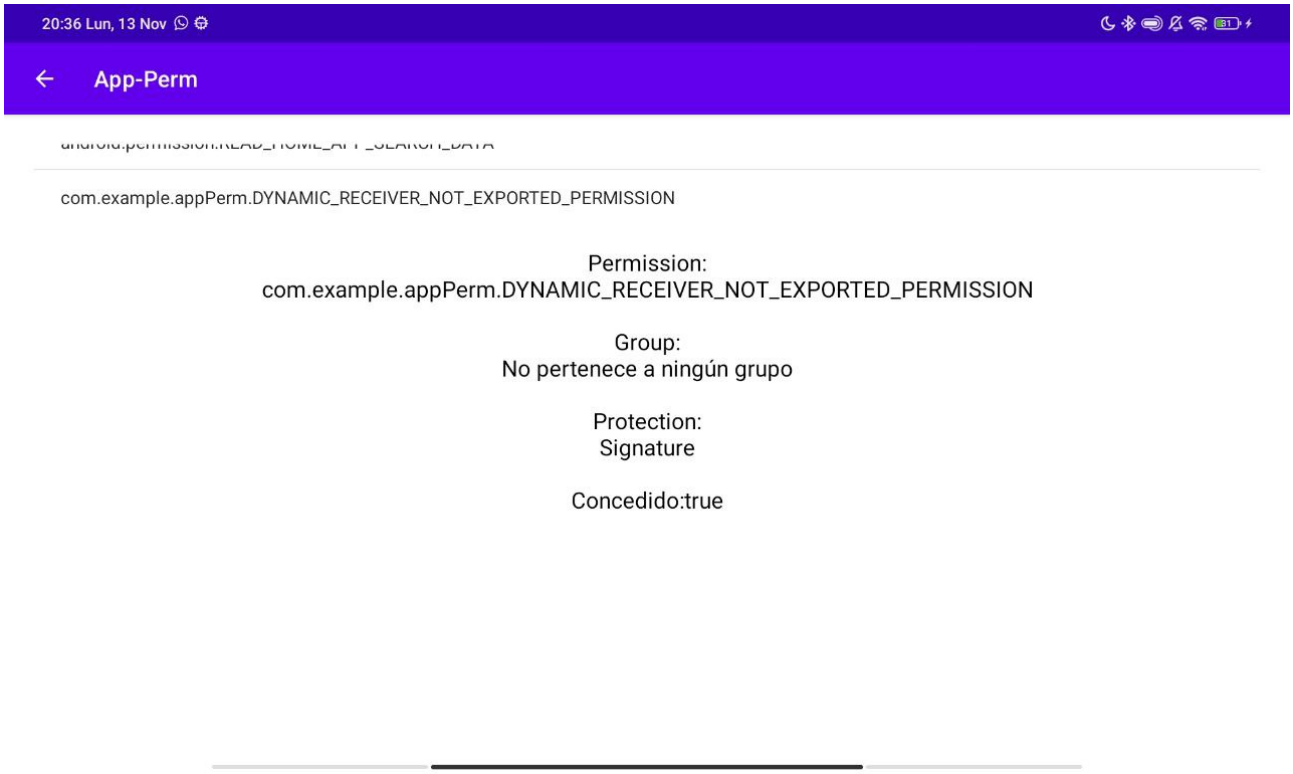

Figura B.3: Vista de la lista de permisos en Android

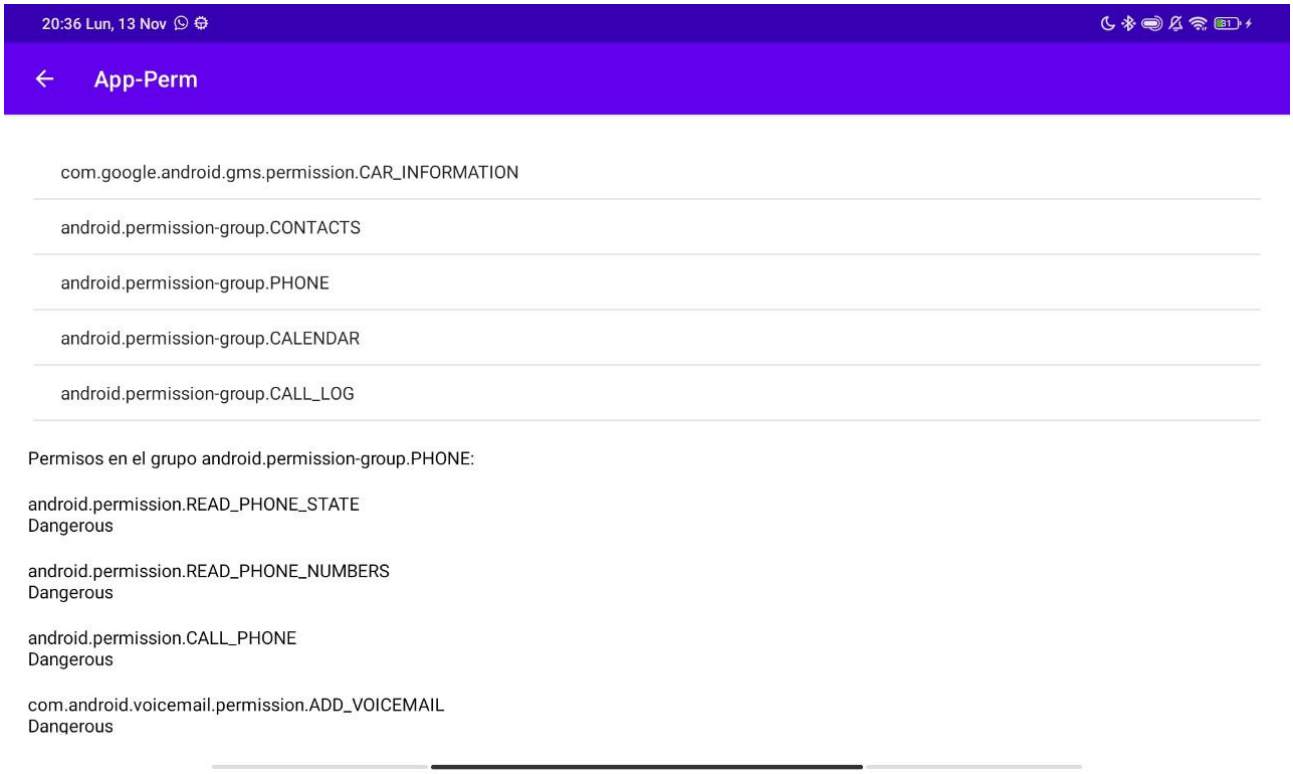

Figura B.4: Vista de la lista de grupos en Android

B.3. Aplicación de escritorio

Esta aplicación nos permitirá:

1. Ver información de los grupos de permisos, los permisos y sus protection level tal y como los define Google y comprobar con un dispositivo móvil que tenga App-Perm que esto es así.
2. Crear distintas apk con AndroidManifest modificados para intentar cambiar esas asignaciones por defecto e instalar esas apks en un dispositivo.
3. Instalar unas apks precompiladas que están firmadas con el mismo certificado y que solicitan los mismos permisos además de comprobar que efectivamente el certificado es el mismo.
4. Ver los permisos y grupos de permisos que hay en un dispositivo conectado.

Para explicar su funcionamiento, contaremos con 4 secciones pues agruparemos en una sola las 5 que se refieren a la modificación de los permisos por defecto.

B.3.1. Permisos según la web de desarrolladores de Android

Esta sección explica el funcionamiento del primer botón "Grupos y permisos por defecto", el cual nos permite ver la información sobre los grupos de permisos y sus permisos según viene definida en la web de desarrolladores de Android.

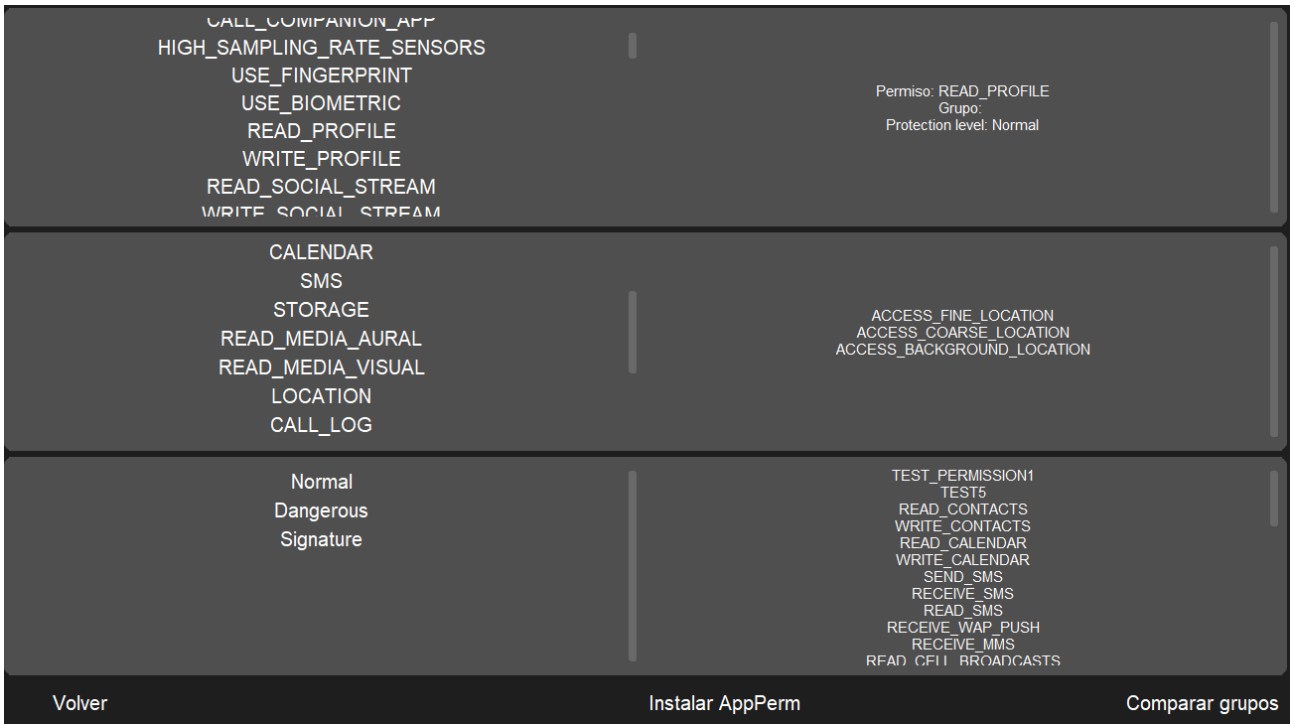

Figura B.5: Vista de grupos y permisos por defecto en la app de escritorio

El objetivo de este botón es permitir ver de forma sencilla la información que Google proporciona sobre los permisos y grupos de permisos. Como vemos en la siguiente imagen, tenemos 3 apartados, el primero contiene una lista de permisos en la cual podemos seleccionar uno y veremos su nombre,

a que grupo pertenece y su protection level. El segundo apartado contiene una lista de grupos de permisos donde, si pulsamos en uno de los grupos, veremos los permisos que componen dicho grupo. Finalmente tenemos el tercer apartado que nos presenta los protection level más importantes y donde podemos ver que permisos cuentan con dicho protection level si pulsamos sobre alguno de ellos. Con esta opción podemos ver todos los permisos dangerous y luego ir mirando uno a uno si pertenecen a un grupo y vemos que así es, todos los permisos dangerous pertenecen a un grupo de permisos.

Además de las funciones mencionadas, si se conecta un dispositivo con Android 13 o superior con depuración USB activada podremos pulsar en"Instalar AppPerm" que instalará en el dispositivo una app que solicita en el Manifest todos los permisos de la lista mostrada.

Una vez instalada, podemos pulsar en"Comparar grupos" para asegurarnos que la asignación de permisos a grupos de permisos hecha en la app de escritorio es la misma que la que contiene el dispositivo. En caso de que no sea así, se indicará que grupos tienen diferencias y dentro de ese grupo que permisos solo los encontramos en el dispositivo y cuales solo en la lista por defecto.

Para hacer uso de esta opción, primero hay que entrar en la app Android y pulsar en la sección permisos, eso generará una carpeta con un fichero csv en la carpeta de usuario del dispositivo y que contendrá los permisos y grupos de permisos del dispositivo.

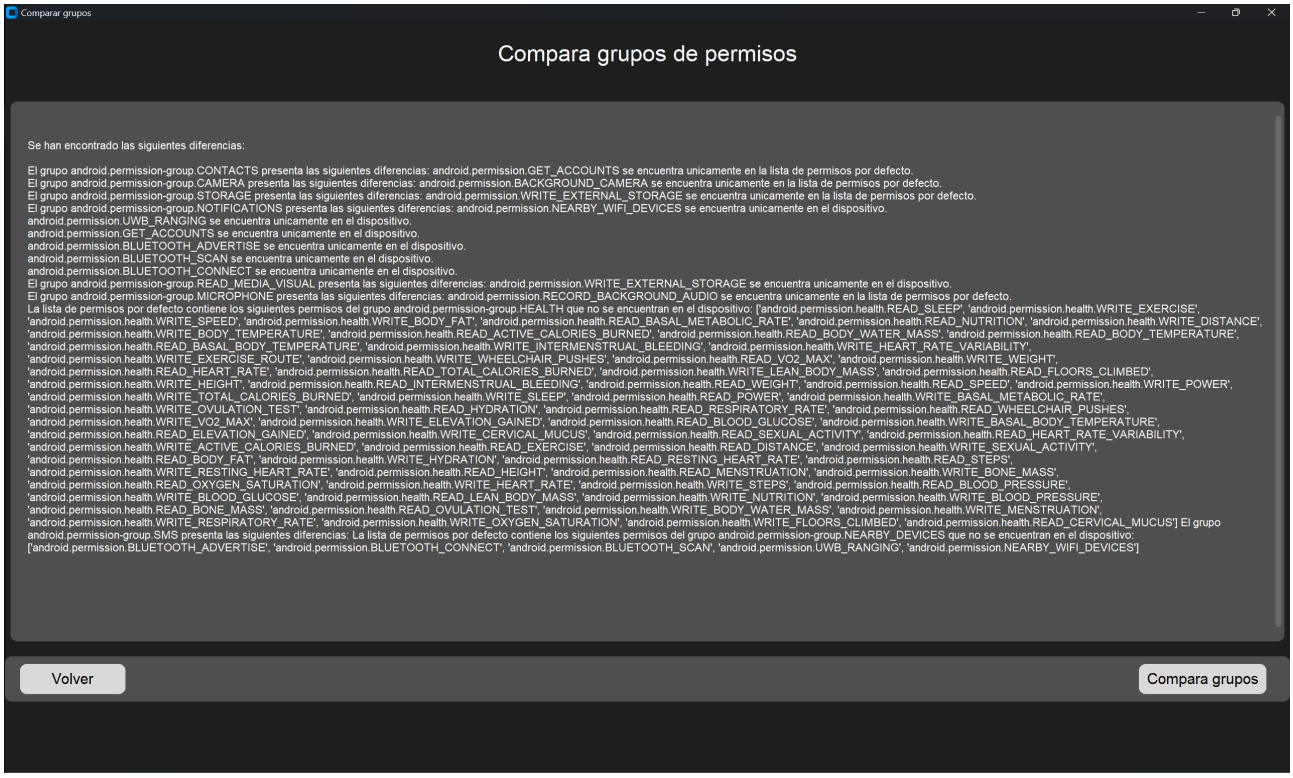

Figura B.6: Vista de comparación de permisos en la app de escritorio

B.3.2. Modificación de permisos

En esta sección veremos la funcionalidad de los siguientes 5 botones: "Cambiar grupo de permisos", "Añadir un permiso a dos grupos", "Cambiar protection level", "Asignar permiso normal a un grupo"

y “Asignar permiso signature a un grupo”.
Todos ellos tienen en común que crean un nuevo AndroidManifest con el cual luego creamos una apk que se puede instalar en el dispositivo para ver que ocurre.

También es importante saber que, en las listas de permisos que se muestran, tenemos unos permisos con nombre TESTX, estos son permisos que se crearán en el momento y nos servirán para ver en algunos casos como se comportan los permisos de usuario frente a los permisos de sistema.

Cambiar grupo de permisos

Figura B.7: Vista para seleccionar modificaciones para el Android Manifest

Como su nombre indica, esta opción nos permitirá cambiar el grupo de permisos de un permiso, para ello simplemente debemos seleccionar un permiso y el grupo al que queremos asignarlo. Una vez elegidos permiso y grupo, pulsamos sobre siguiente y se nos mostrará una ventana en la cual se crea un AndroidManifest.xml con la configuración indicada y a continuación se compilará un apk con ese AndroidManifest que posteriormente podremos instalar o abrirlo en el explorador de archivos.

Añadir permiso a dos grupos

En esta sección, podemos probar que ocurre si añadimos un permiso a 2 grupos, para ello elegimos el permiso, 2 grupos y pulsamos en siguiente. Se nos mostrará la misma ventana de antes, pero esta vez nos encontramos un error de compilación donde podemos ver una línea que dice que el elemento permission está duplicado, esto es un comportamiento esperado.

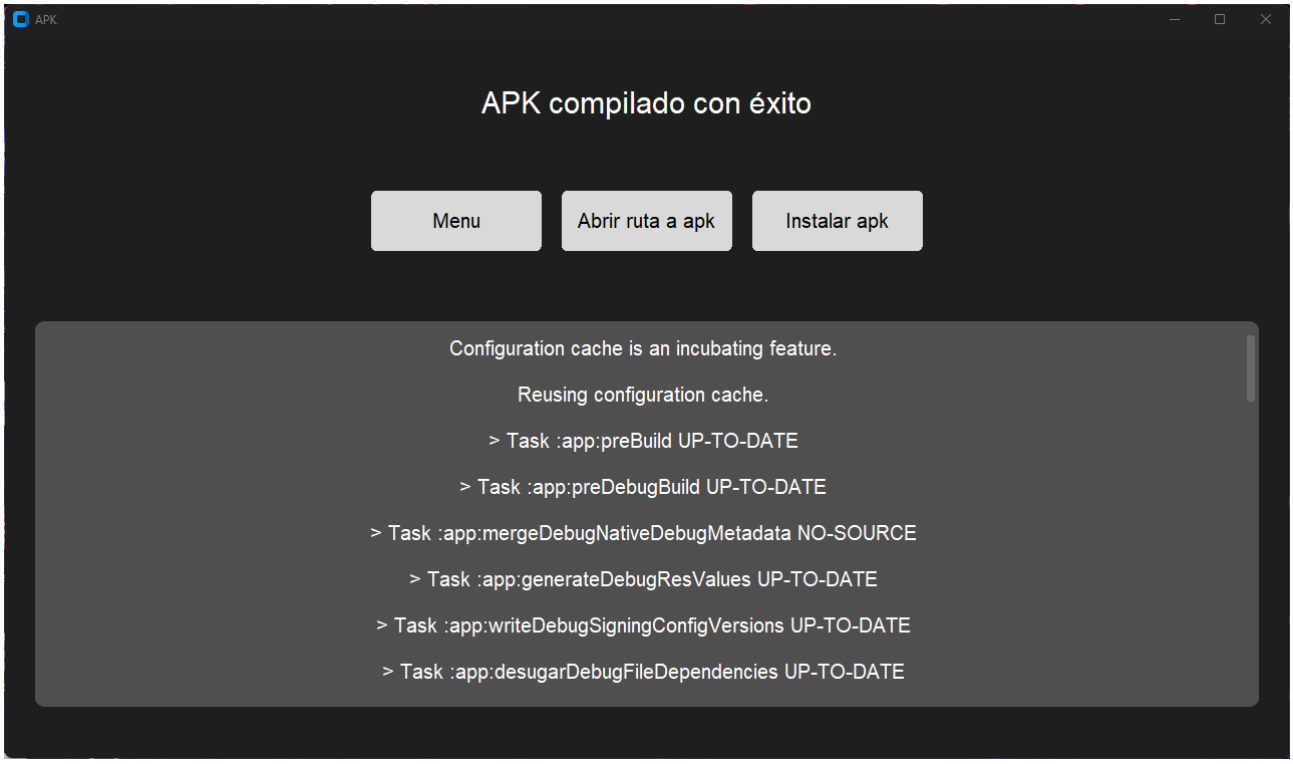

Figura B.8: Resultado de compilación correcta de apk

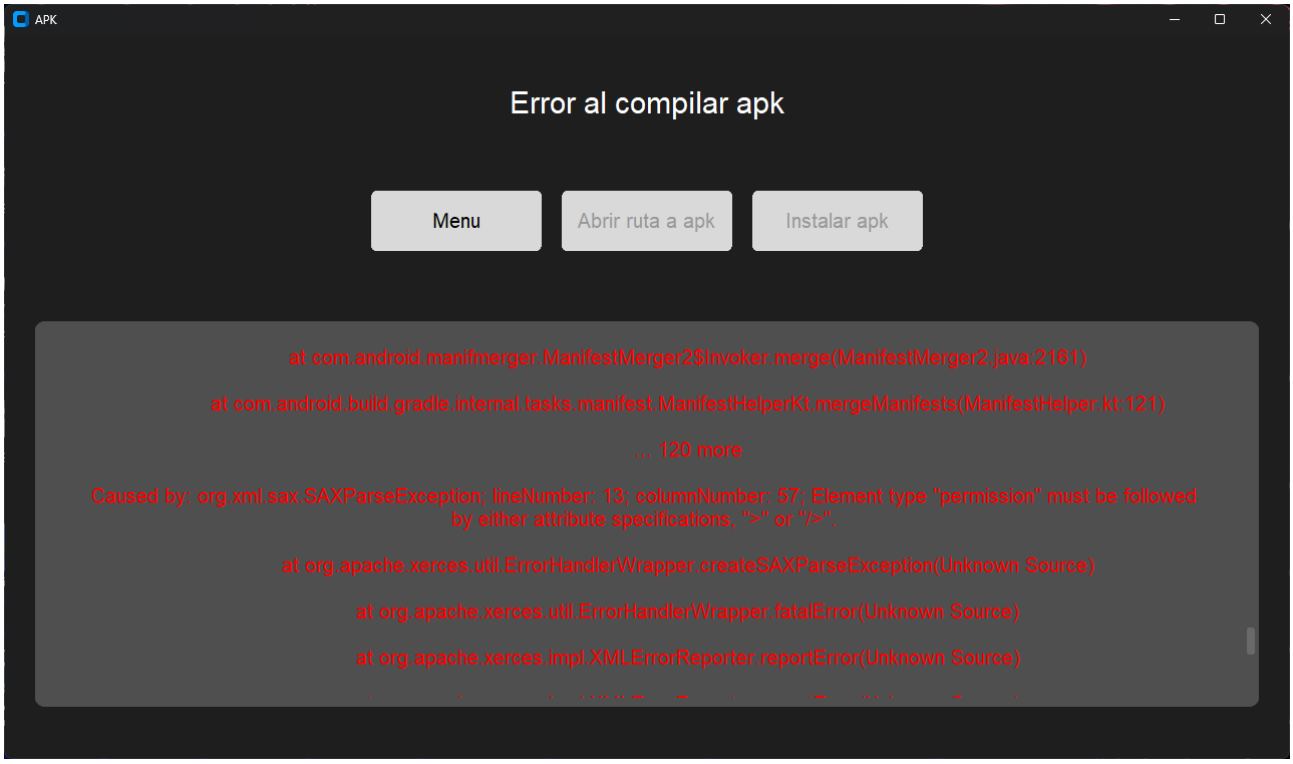

Figura B.9: Resultado de compilación de apk erronea

Cambiar protection level

Este botón nos permite probar que ocurre si intentamos cambiar el protection level de un permiso, en este caso, ya podemos acceder a la columna “Protection level” y debemos elegir uno de estos y un

permiso al que intentaremos cambiar el protection level.

Figura B.10: Vista de modificación del Android Manifest para cambiar el protection level

Asignar permiso normal a un grupo

En este caso, veremos una lista con los permisos normales y otra con los grupos, de nuevo debemos seleccionar uno de cada y al pulsar siguiente se compilará el apk que, de nuevo, podremos instalar en un dispositivo conectado o emulador.

Asignar permiso signature a un grupo

Como antes, veremos una lista con los permisos signature y otra con los grupos, debemos seleccionar uno de cada y al pulsar siguiente se compilará el apk que no dará errores.

B.3.3. Permisos del dispositivo

En esta sección, podemos conectar un dispositivo Android con depuración USB activada y ver sus permisos y grupos de permisos mediante un comando de adb.

B.3.4. Aplicaciones firmadas

Otra funcionalidad importante de la app es hacer pruebas con apps firmadas, para ello tenemos este apartado en el cual tenemos 3 opciones, las 2 primeras se encuentran en la parte superior donde

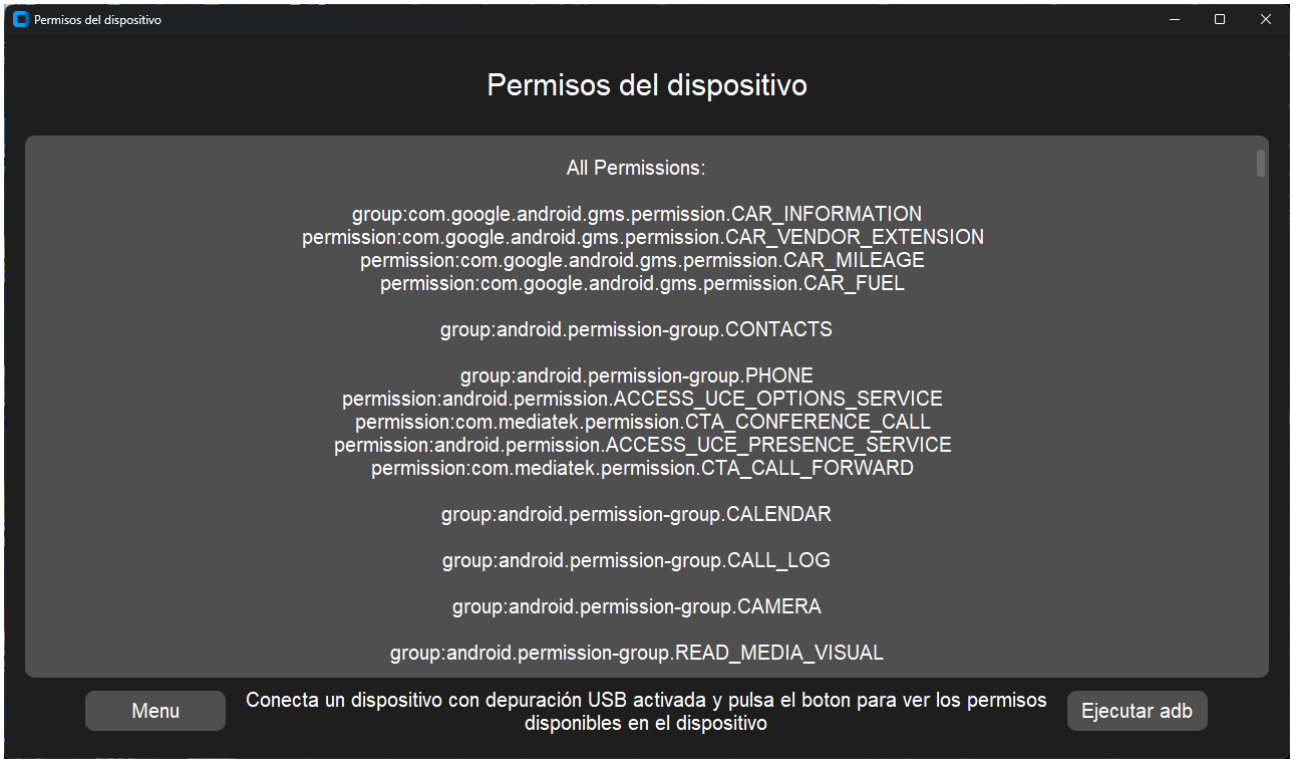

Figura B.11: Vista de permisos del dispositivo

vemos que podemos “Instalar apps firmadas dangerous” o “Instalar apps firmadas signature”.

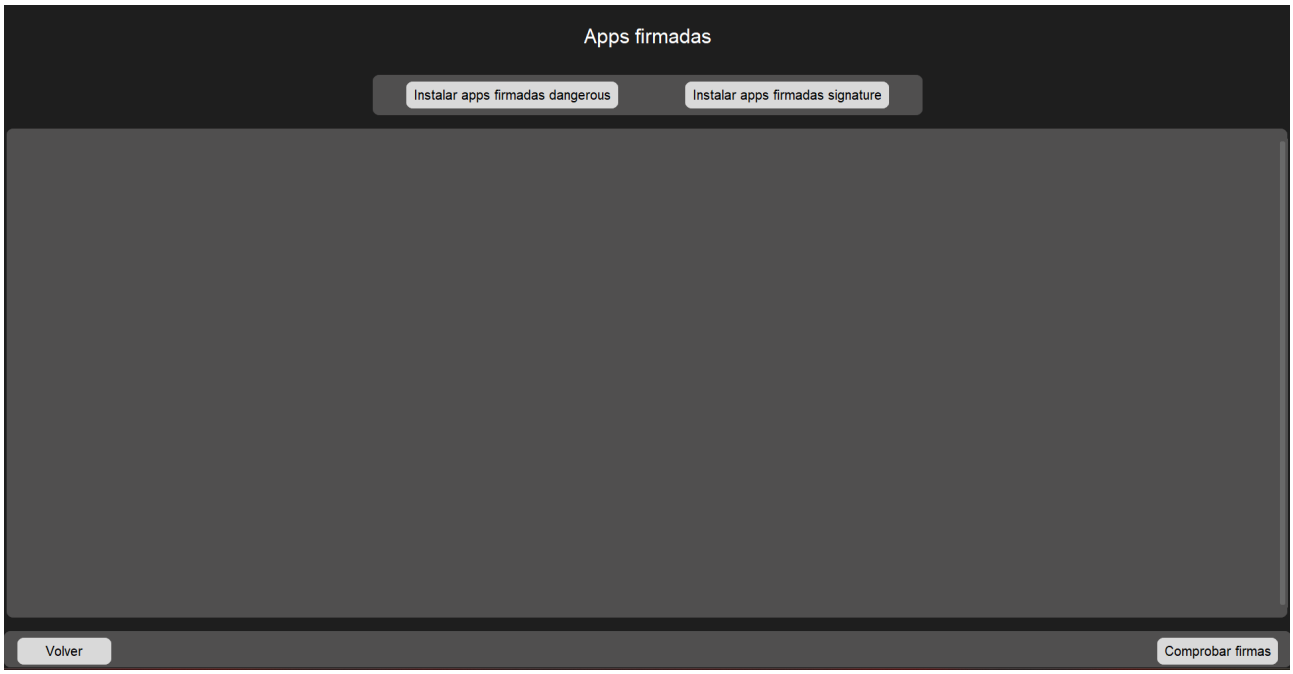

Figura B.12: Vista del menu de pruebas con apps firmadas.

La primera opción instala dos aplicaciones que estética y funcionalmente son iguales a la instalada en el apartado de Grupos y permisos por defecto, pero una de ellas solo tiene 2 permisos declarados, uno dangerous y otro signature. La segunda app solo declara un permiso que es el mismo que el

permiso dangerous que tiene la otra app que se instala y ambas están firmadas con el mismo certificado.

La segunda opción instala dos aplicaciones que estética y funcionalmente son iguales a la instalada en el apartado de Grupos y permisos por defecto, pero una de ellas solo tiene 2 permisos declarados, uno dangerous y otro signature que es el mismo que tiene la otra app que se instala y ambas están firmadas con el mismo certificado.

Finalmente tenemos una opción que nos dará información del certificado usado para firmar las 3 apps que usamos en esta sección de forma que podemos ver que es el mismo certificado para las 3.